Autonomous Vehicle Path Planning with Remote Sensing Data

Aaron J. Dalton

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Geography

Laurence W. Carstensen, Ph.D., Committee Chair
James B. Campbell, Ph.D.
Alfred L. Wicks, Ph. D.

December 2nd, 2008
Blacksburg, Virginia

Keywords: LIDAR, Friction Surface, Accumulated Surface, Least-cost path, Autonomous Ground Vehicle

Autonomous Vehicle Path Planning with Remote Sensing Data

Aaron J. Dalton

Abstract

Long range path planning for an autonomous ground vehicle with minimal a-priori data is still very much an open problem. Previous research has demonstrated that least cost paths generated from aerial LIDAR and GIS data could play a role in automatically determining suitable routes over otherwise unknown terrain. However, most of this research has been theoretical. Consequently, there is very little literature dealing with the effectiveness of these techniques in plotting paths for an actual autonomous vehicle. This research aims to develop an algorithm for using aerial LIDAR and imagery to plan paths for a full size autonomous car. Methods of identifying obstacles and potential roadways from the aerial LIDAR and imagery are reviewed. A scheme for integrating the path planning algorithms into the autonomous vehicle's existing systems was developed and eight paths were generated and driven by the autonomous vehicle. The paths were then analyzed for their drivability and the model itself was validated against the vehicle measurements of slope and position. The methods described were found to be suitable for generating paths both on and off road.

Acknowledgements

I would first like to thank my committee: To my advisor Dr. Carstensen, who taught my first Geography course, introduced me to the Urban Challenge, and whose thoughts, comments, and advice have been invaluable over the course of writing this thesis. Thanks to his guidance, my experience as a Geography graduate student at Virginia Tech has been one of the most rewarding experiences of my life. To Dr. Campbell whose door was always open and who was always willing to listen to any idea I wanted to run by him. And to Dr. Wicks who helped assemble and lead the brightest group of individuals I've have ever worked with and who believed a geographer could play an integral role in an autonomous vehicle project. To you all: your experience, wisdom, and candid input have been a crucial part of my development as a Geography student and as a person.

Special thanks go out to my friend and Urban Challenge teammate Patrick Currier who was always willing to discuss my research and who agreed to serve as my safety driver the morning after he flew back from Arizona. Also, thanks to Arvind Bhuta who came out on a cold October morning to help me test. Thanks as well to all my DARPA Urban Challenge professors and teammates, especially Dr. Charlie Reinholtz, for not only accepting "the odd man out" but also letting me be in charge of a crucial component of the vehicle and having full faith and confidence in my ability to pull it off.

Finally, I like to give special recognition to my parents, who encouraged me, supported me, and nurtured me. The let me reach for the stars and caught me when I fell short. This thesis is for them. I love you both.

Attribution

Several colleagues aided in the writing and research in this dissertation. A brief description of their background and their contributions are included here.

Dr. Laurence W. Carstensen - Ph.D. (Department of Geography, Virginia Tech) is the primary Advisor and Committee Chair. Dr. Carstensen originally proposed the thesis idea, and was invaluable in the editing of this research.

Dr. Alfred L. Wicks - Ph.D. (Department of Mechanical Engineering, Virginia Tech) is a member of my committee and the main faculty advisor for the team that developed the autonomous vehicle used in this research. He also facilitated access to the equipment used in this thesis.

Patrick Currier – Graduate Student (Department of Mechanical Engineering, Virginia Tech) served as my safety driver for this research and aided in data collection and analysis.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

The state of the art of autonomous vehicles has advanced significantly over the past ten years, due in no small part to the exposure and incentives offered by DARPA's Grand Challenge and Urban Challenge competitions. These challenges have shown that autonomous vehicles can obey traffic laws, avoid static obstacles, and interact with moving vehicles. But while modern sensors and software can classify situations within a few hundred meters of the vehicle, most autonomous systems still need external data to give them a wider spatial perspective. For example, the drivable roadways and paths used in both challenges, with few exceptions, were well surveyed prior to the competition. Most long range autonomous ground vehicle implementations have followed the same basic assumption that accurate and properly formatted information on all usable routes exists for the area in which the vehicle operates. This has led to reliance on data gathered specifically for the autonomous vehicle and has limited autonomous vehicle operation to known predetermined routes.

One way to overcome these limitations would be to use some type of aerial remote sensing data such as aerial imagery or LIDAR to determine potential drivable paths in a semi-automated fashion. Such a solution would have several benefits over alternative methods; Aerial remote sensing data is widely used and is well understood. Data acquisition is a routine procedure and aerial remote sensing data is either commercially available or can be captured through commercial services. Furthermore, in a military application aerial reconnaissance is a common precursor to a ground mission. Additionally software and algorithms exist to process and store aerial remote sensing data. Most of the previous research on vehicle path planning has used aerial LIDAR data. Aerial LIDAR is especially suited to path planning for several reasons. First, aerial LIDAR data can have a very high spatial resolution and vertical accuracy. Furthermore, because aerial LIDAR records the distance to physical objects, important path

planning features such potential obstacles, terrain slope, and terrain roughness can be derived from aerial LIDAR data.

In 2005, as part of the first DARPA Grand Challenge, Chris Stahl wrote a thesis titled "Accumulated Surfaces & Least-Cost Paths: GIS Modeling for Autonomous Ground Vehicle (AGV) Navigation" that explored the used of GIS data and techniques to derive paths for an autonomous ground vehicle. While the model described in Stahl (2005) appeared to create valid paths, he was not able to validate any of his results on an autonomous vehicle.

Later as part of the second DARPA Grand Challenge, Om Poudel wrote a thesis expanding on this work and described a novel method for using airborne LIDAR data to calculate dynamically the most suitable path for driving an autonomous vehicle using accumulated surfaces. Suitability was determined by several factors including the path length, presence of obstacles, terrain slope, and overall terrain roughness. Poudel (2007) also concluded that paths generated by the described method behaved in the appropriate manner and were significantly more suitable for driving than standard Euclidean paths.

While both Stahl (2005) and Poudel (2007) demonstrated that reasonable paths can be generated using GIS techniques, there is still an open question as to whether the same algorithm could be successfully implemented to guide an autonomous vehicle. There are additional implementation details that must be developed in order to turn the grid based path produced by a GIS algorithm into a form that an autonomous vehicle can follow. Furthermore, this method must be compatible with other operational aspects of the autonomous vehicle including obstacle avoidance.

1.2 Problem Statement

Previous research in Poudel (2007) and Stahl (2005) conceptualized a method of using remote sensing data to determine a suitable driving path for an autonomous vehicle. The purpose of this research is to operationalize the concept of an autonomous vehicle path planning algorithm using remote sensing data and methods described in Poudel (2007) and Stahl (2005) for an actual autonomous vehicle and answer the following questions:

- Can the methods described in previous research be implemented to determine a route for driving an autonomous vehicle?
- What would a practical implementation of these methods look like?
- What are issues encountered when implementing these methods?

For this particular experiment "Odin", Virginia Tech's 2007 entry into the DARPA Urban Challenge will be used as the autonomous vehicle platform (Reinholtz 2007). Odin is a converted 2005 Hybrid Ford Escape that has custom software programs that allow it to drive in a predetermined road network, avoid obstacles, and obey simple traffic laws. It has a high precision GPS navigation system and several laser scanners which it uses to detect obstacles. One of the goals of this project was to implement a path planning algorithm for the autonomous vehicle while still maintaining much of the existing functionality for obstacle avoidance and short range path planning.

This thesis consists of four chapters. Chapter one contains the statement of purpose and introduces some important concepts. Chapter two consists of an overview of the previous research, a presentation of relevant literature, as well as a review of other operational autonomous vehicle path planning systems. Chapter three is a discussion of the process used to operationalize the algorithm, data gathering and analysis technique, results and conclusion. Chapter four will discuss potential for further research.

1.3 LIDAR

LIDAR, or **LI**ght **D**etection **A**nd **R**anging, is an active remote sensing system that uses laser pulses to measure distances to objects by measuring the time of flight of a laser pulse. LADAR (**LA**ser **D**etection **A**nd **R**anging) is sometimes used as an alternate term and both acronyms can be used interchangeably. LIDAR originally was developed as a simple ranging tool, but it has gained wider acceptance as a multi-use sensor. One of the earliest uses of LIDAR was on the Apollo 15, 16, and 17 lunar missions as lunar topography was measured using a laser altimeter on the orbital lunar modules (Bender, et al. 1973). As the technology has matured, LIDAR has found a use in many different fields. Aerial LIDAR systems specifically have

been used to gather bathymetry data, determine topography, and measure forest parameters such as canopy height and leaf area.

All LIDAR systems function on the principal of laser ranging. A laser pulse is sent at a specific (sensor relative) angle and the time of flight is measured and used to determine a distance. Simultaneously, attitude and positional data are captured using a high accuracy GPS system (usually Differential or DGPS). Both systems, the LIDAR scanner and the position and orientation system, are time synchronized. The range and angle data capture by the LIDAR system are then coordinated with the position and orientation measurements to geo-reference the laser returns (Wehr and Lohr 1999). Using direct measurements of position and orientation in geo-referencing, instead of correlating data to known reference points, is known as direct geo-referencing. In this paper I will mention two different types of LIDAR, aerial and ground based. While this paper will primarily focus on the processing of aerial LIDAR, both systems operate in a similar fashion.

While there are many similarities between ground based and aerial LIDAR, there are some significant differences. First, ground based LIDAR generally has a limited range of tens of meters to hundreds of meters. This limitation is usually a result of a wide field of view coupled with a low angular resolution. While ground based LIDAR can sense returns up to several hundred meters, these returns aren't useful due to the large distance between scans at that range. Additionally, ground based LIDAR systems usually rely on multiple planes to increase spatial resolution over the typical push broom methods of aerial LIDAR.

Chapter 2: Literature Review

2.1 Introduction

Events such as the two DARPA Grand Challenges and the DARPA Urban Challenge have demonstrated the operational viability of autonomous ground vehicles. Still one criticism of these challenges is that DARPA provided carefully prepared datasets for use in path planning. These datasets would not be practical in an operational setting. In such situations it would be useful to have a method that uses commonly available, multipurpose geospatial data to plot drivable routes rather than surveying or digitizing roadways and potential drivable routes specifically for autonomous vehicle operation. Additionally such a system would allow for easy dynamic re-planning along automatically identified routes when a planned route proved unfeasible.

While there is significant literature on basic path planning and on path planning for autonomous vehicles, there is very little literature that deals with the practical issues and difficulties involved in developing a path planning system for an autonomous vehicle using remotely sensed data. Additionally there is a dearth of experimental results of implemented systems. This chapter will present research on path planning in both the GIS and autonomous vehicle realms, outline some of the previous results by other researchers, give an overview of some of the common techniques and algorithms used in the field, and finally, give some background on the particular autonomous platform. In the process, this chapter will demonstrate the necessity for more research into traditional geospatial approaches to the autonomous vehicle path planning problem

2.2 Path Planning in GIS

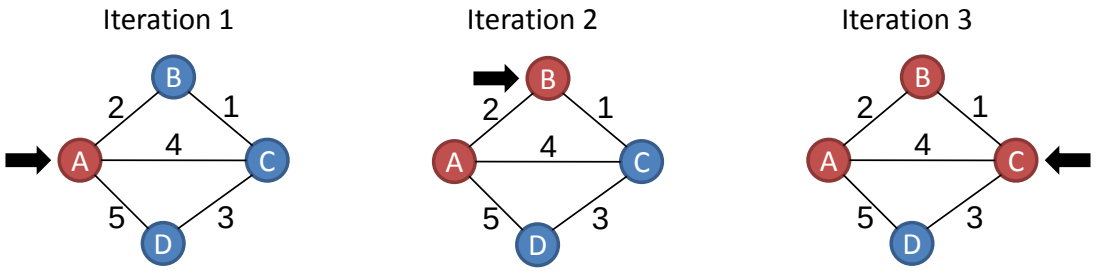
Path planning is defined as determining a route based on and expressed in terms of an abstract world model. Such algorithms usually try to optimize a specific parameter such as a generic path cost or path length and are classified based on the particular world model used.

Two general approaches to path planning have seen widespread adoption in the GIS community: a network or graph based approach and a grid based approach.

2.2.1 Network Based Path Planning

The first approach, sometimes referred to as route planning, finds the shortest path using a network or directed weighted graph as a world model. Such approaches are useful when the solution set for a path planning algorithm must be chosen from a fixed set of predefined paths. Furthermore, with network based analysis, no paths outside of the initial fixed set can be considered. The most recognizable application for this kind of modeling is the selection of the 'best' route in a network of roads. This application seems only natural; graphs have served as a representation for roadways since Euler first pondered on a tour of the bridges of Königsberg in 1736(Hopkins and Wilson 2004). The basic model for this application treats road intersections as graph nodes and roads as graph edges. Edge weights are usually assigned based on physical properties of the road such as distance or travel time.

The algorithms for solving this type of network-based path planning were developed in the late 1950's (Dijkstra 1959 and Moore 1959) and have been integrated into GIS software since the mid 1980's (Lupien, Moreland and Dangermond 1987). The basic algorithm was first outlined in Dijkstra's 1959 paper and is appropriately referred to as Dijkstra's algorithm. The algorithm itself is described in Figure 2.1. It is a greedy algorithm, meaning it uses local optimization at each stage in order to reach a global optimum. In the general case Dijkstra's algorithm computes the minimum total path cost from a source node to all other nodes in a non-negative edge weighted graph (a network) and assigns an identifier to the previous node in the shortest path to each vertex. This previous node value can be used to reconstruct the least cost path from any node back to the source. Dijkstra's algorithm can also be used to solve the directed graph shortest path problem. In the case in which a least cost path between two given vertices is required, one vertex is chosen as a source vertex and the algorithm terminates when the destination vertex is reached.



N	$D_A(N)$	$P_A(N)$
A	0	--
B	2	A
C	4	A
D	5	A

N	$D_A(N)$	$P_A(N)$
A	0	--
B	2	A
C	2+1	B
D	5	A

N	$D_A(N)$	$P_A(N)$
A	0	--
B	2	A
C	2+1	B
D	5	A

$D_A(N)$ - Shortest known distance from node A to a given node N
$P_A(N)$ - Previous node in the least cost path from node A to given node N

1. Create an array of shortest known distances and previous nodes
2. Set the current node (➡) to the given source node
3. Mark the current node as visited (●)
4. Update the array of distances and the and the previous nodes based on the nodes reachable from the current node (➡)
5. If there are unvisited nodes, set the current node to the unvisited node with the smallest distance value and repeat from step 3

created by author

Figure 2.1: Dijkstra's Algorithm

Dijkstra's algorithm is also very important to path planning because it serves as a basis for several algorithms that are commonly used in both GIS and robotics. The A* (pronounced A-star) algorithm (which will be described in the next section) and many other heuristic search algorithms are derivatives of Dijkstra's algorithm. Dijkstra's algorithm or its derivatives are also widely used as the basis for many robotic path planning algorithms (Dolgov 2008, Reinholtz 2008, and Bacha 2008). It is also important to note that Dijkstra's algorithm has a grid based analog, described in greater detail in a later section, that is widely used to derive least cost path solutions using a grid based model.

Dijkstra's algorithm provably finds the shortest cost path from one point to another without any additional information about the graph or edge weights. However for real world path planning, Dijkstra's algorithm can be computationally inefficient because a global structure often exists that can be used to further optimize the searching algorithm. Specifically, there

may exist a simple function that gives a lower bound estimation of the graph distance between a given node and the destination node. For example, in a road network where the edge weights represent road lengths and nodes represent road intersections, the minimum possible distance between any node and the destination node is the Euclidean distance between the two corresponding intersections. The A* algorithm, first presented in Hart, Nilsson and Raphael (1968), uses this minimal heuristic measurement to improve path finding efficiency over Dijkstra's algorithm. An example of a minimum heuristic is given in **Error! Reference source not found.**.

Formally, the minimal heuristic is a simple measure that is guaranteed to be less than or equal to the shortest graph distance between two given nodes. In most path planning algorithms the Euclidean distance is used as minimal heuristic between two nodes because it is simple and easy to calculate, but a minimal heuristic can be any function that meets the previous criteria. The speed up is largely based on the quality of the heuristic; the better the minimum heuristic estimates the shortest graph distance, the larger the speedup over standard Dijkstra's algorithm.

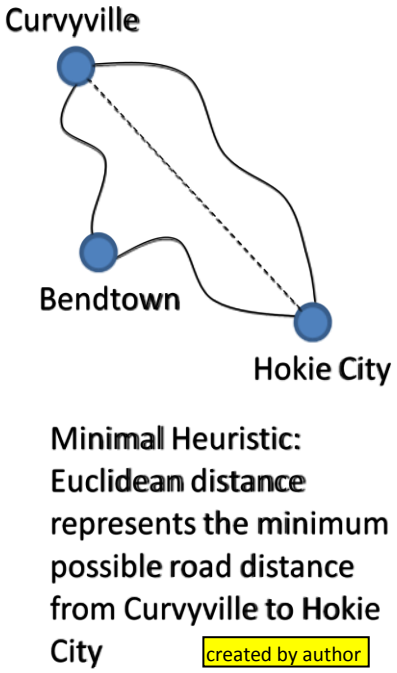


Figure 2.2: Example of a minimal heuristic

The A* algorithm uses two main strategies to increase efficiency. First, the A* algorithm uses the path distance from the source plus the given minimal heuristic measure to the destination to determine the search order of potential nodes. This strategy works because this value represents the minimum possible path distance through the given node and also generally serves as a good estimation of total path length. Second, it uses the heuristic to prune the search tree of unnecessary branches by comparing the minimum possible path distance (again the path distance from the source plus the minimal heuristic) through the given node with previous computed path lengths.

The A* algorithm is generally preferred over

Dijkstra's algorithm to solve path finding problems because of its increased efficiency. Still there are a few drawbacks: The A* algorithm doesn't calculate the shortest path from a source to every other node, just to a given destination node. Additionally, the speedup is largely dependent upon the heuristic used, which can change based on the underlying cost function. Therefore the A* algorithm is primarily useful in specific applications, such as in-vehicle route planning, where the cost equation is well defined and suitable heuristics can be determined ahead of time. In general GIS cost modeling the A* algorithm has not seen widespread use.

Network based path planning, as implemented in modern GIS, was outlined in detail in Lupien, Moreland and Dangermond (1987). In this paper, the authors present a framework for using GIS data to parameterize and solve network problem in GIS systems. They also outline how standard GIS conventions can be effectively extended to create network datasets and to solve network based problems. They also discuss solutions to common problems that occur in GIS analysis, such as road based resource allocation and travel time estimation.

2.3 Grid Based Path Planning

Network based path planning is very useful when there exists a fixed set of paths to chose from. But for a general case with unstructured terrain and off road driving, a grid based path planning technique is more appropriate. Such algorithms represent the terrain as a discrete grid where each cell has a cost to move through. The least cost path in a gridded algorithm is the collection of cells such that the cost of moving from the center of a source cell to the center of a destination cell is minimized.

2.3.1 Node/Link Representation

Most grid based path planning algorithms are related forms of the network based algorithms outlined in the previous section. In order to apply the network based algorithm to the raster data, a node/link representation is used. In this representation, the gridded data set forms a lattice in which each cell center represents a node with edges to its direct neighbors. Edge weights are assigned based on cost values of the two adjacent cells and the travel direction as well as other potential directional weights. Using this method GIS implementations

can apply network algorithms such as A* and Dijkstra's Algorithm to solve path planning algorithms on the raster datasets.

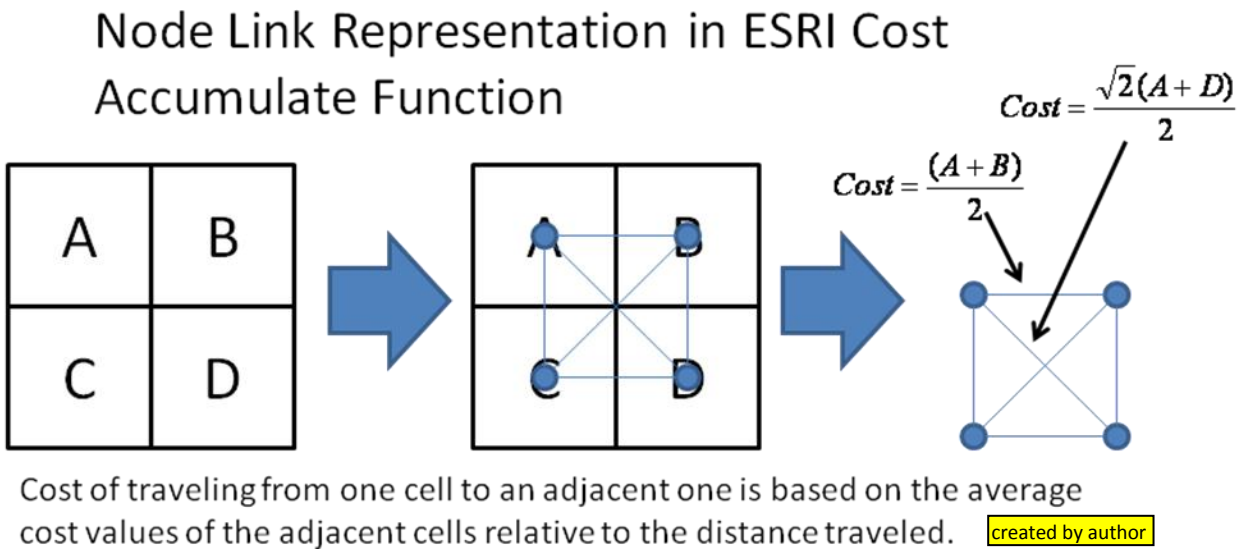
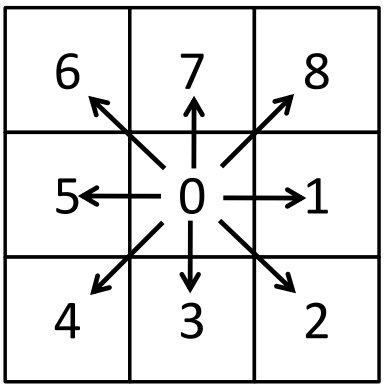


Figure 2.3: Node Link Representation

2.3.2 Accumulated Cost Surface

Most GIS systems that implement a grid based path planning algorithms include an accumulated cost surface method. This method uses a cost layer that represents the relative difficulty of moving through a particular cell and then finds the path from a source cell to a destination cell (through cell centers) that has the least total cost. This is accomplished by creating a layer, called an accumulated cost surface, in which each cell is accumulated cost of the shortest path from that cell to the given source cell. Simultaneously, a layer, called a backtrack layer, is generated in which each cell represents the direction to the previous cell in the shortest path. Using this backtrack layer a least cost path can be traced from any cell back to the source cell.

created by author



Backtrack Direction as specified in ArcGIS

Figure 2.4: Backtrack direction values

From a conceptual point of view, the accumulated cost surface and backtrack layer are generated using a modified Dijkstra's algorithm on the node/link grid

representation of the cost layer. In relation to Figure 2.1 (Dijkstra's algorithm) the accumulated surface stores the list of shortest distance values while the backtrack layer stores a value that represents the next neighbor along the least cost path. Since each cell location is only connected to its neighbors in the node/link representation, the backtrack value can be represented as an integer in the range of one to eight that specifies the specific neighbor cell along the least cost path. The exact values used in the ArcGIS implementation are given in Figure 2.4(ESRI 2007).

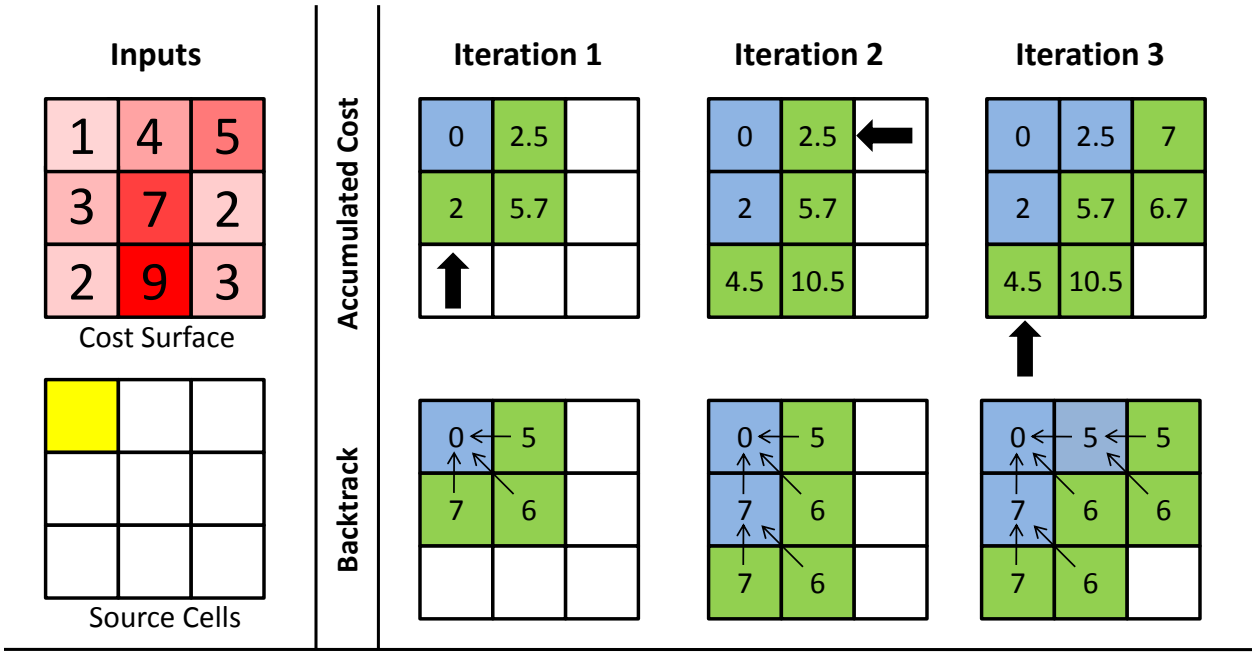


Figure 2.5: Illustration of cost accumulation algorithm

An illustration of the full algorithm for generating the accumulated cost surface and backtrack layer is included in Figure 2.5. The inputs for this algorithm are a positive value cost layer and a binary layer indicating source points. During the update step of algorithm a new accumulated cost and backtrack value is calculated for all the non-visited neighbors of the current cell. During this update step the cost of traveling to a neighboring cell is calculated

using the edge weights of the node/link representation. That cost is added to the accumulated cost value for the current cell to get potential accumulated cost for the neighbor. If there is no accumulated cost set for that neighbor or the current accumulated cost value is less than a previously calculated accumulated cost value, the accumulated cost is updated and the backtrack layer is set to the value that represents the direction from the neighboring cell to the current cell.

Accumulated cost surface based path planning has several advantages that make it very useful for GIS applications. The primary advantage is that the raster cost layer can be easily derived from other raster based GIS layers using simple map algebra. Raster cost parameters can also be easily derived from pre-existing gridded data sources, such as gridded Digital Elevation Models (DEMs), remote sensing imagery, or derived sources such as land cover classifications. An example of using elevation and vector data to generate a cost map is given in Figure 2.6.

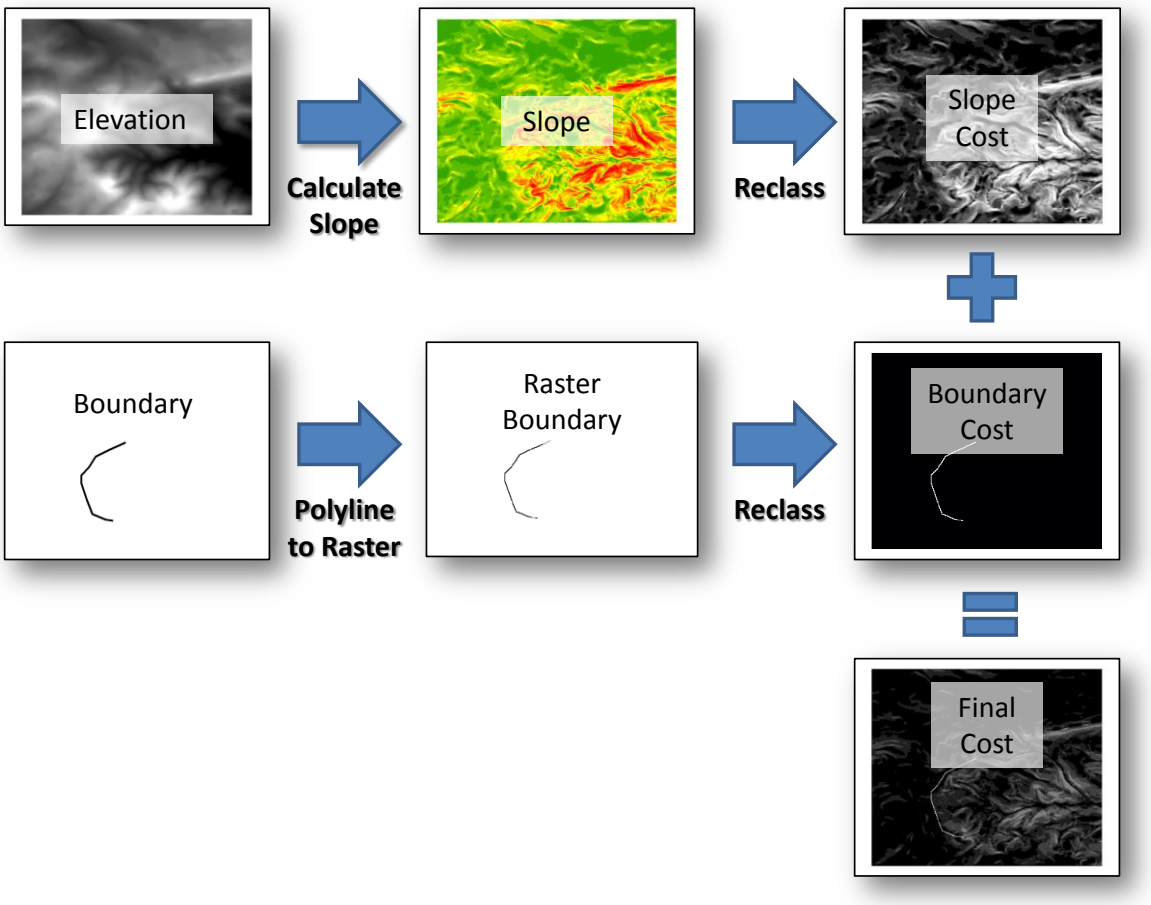


Figure 2.6: Example of cost surface generation

There are several drawbacks to the standard accumulated cost surface model. First, there is inherent distance based error that arises due to cellular motion being restricted to 45 degree increments. While there are methods to correct for these errors, the implementation is complex and is typically not used in most GIS implementations. Second, because the accumulated cost surface model uses Dijkstra's algorithm, it is relatively slow for path planning when compared to similar A* algorithms.

2.4 Path Planning in Autonomous Vehicle Navigation

Path planning in autonomous vehicle navigation is a diverse field that encompasses a broad range of techniques and strategies. The term 'path planning' in robotics can be used to describe everything from determining the flight-path of robotic helicopters to planning the movement of a robotic arm on an assembly line. Therefore, the solutions for solving a path planning problem vary widely based on the capabilities of the vehicle, the operating environment, and the parameters of the path planning problem itself. In order to simplify the discussion, this paper is going to focus on path planning for a ground based autonomous vehicle and will give a broad and practical overview of the key concepts.

Path planning for autonomous ground vehicles is usually implemented as a sub-problem of motion planning. Motion planning is defined as the process of determining a continuous motion from one given position to another through a configuration space while avoiding obstacles. A configuration space is generally defined as the set of all possible positions for the robot (Latombe 1991, LaValle 2006). This approach to path planning ensures that kinematic constraints are considered and that the resulting paths will be drivable. Also the resulting paths can be defined in terms of vehicle states, such as steering angle and speed, further simplifying integration of motion planning into autonomous driving.

Since the configuration space is potentially infinite, discretizing the configuration space is an important part of solving the motion planning problem. There are two basic approaches described in LaValle (2006). One method involves creating a path by decomposing a mathematical representation of the configuration space. Algorithms based on this approach will always give a solution if one exists, however they are usually not computationally feasible for

real-time path planning with many obstacles (LaValle 2006). The second method involves sampling the configuration space and then generating a path along these sampled road-marks. While different sampling schemes are common in the literature, in practice autonomous ground vehicles usually use a latticed or grid based representation of the configuration's space which is then searched using the previously described search methods (A* and Dijikstra's algorithms).

Most general motion planning algorithms are theoretical solutions to a well defined problem. Specifically most of these algorithms assume a complete representation of the world. The reality of path planning in a real world problem is much more complex. A real autonomous vehicle must integrate a-priori information with live sensor data while facing hard constraints on processing time. The next section examines the DARPA Urban Challenge as a comprehensive autonomous vehicle problem and details two different solutions to motion planning that were used successfully in the challenge.

2.4.1 The DARPA Urban Challenge

A description of The DARPA Urban Challenge is included for three reasons. First, the DARPA Urban Challenge represents a fairly comprehensive autonomous ground vehicle application, encompassing many of the difficult problems involved in autonomous vehicle path planning. The descriptions of the algorithms and systems used in the DARPA Urban Challenge form a body of literature that describes state-of-the-art solutions to this comprehensive problem. Second, the autonomous vehicle platform that is used in this research was originally developed for the Urban Challenge and therefore a certain familiarity with the competition itself will be useful in understanding the system as a whole. Third, this research is a direct result of previous research done for two DARPA challenges

In brief, the DARPA Urban Challenge was a competition to build an autonomous ground vehicle that could navigate through an urban environment and interact with traffic while obeying California traffic laws. Held in Victorville, California the competition consisted of the National Qualifying Event and the Final Event for qualifying vehicles.

The DARPA Urban Challenge required that an autonomous vehicle follow an ordered list of checkpoints in a road network. A simplified representation of the road network was supplied as a-priori information in the form of a Route Network Definition File (RNDF). This file contained information about roads, road intersections, and unstructured zones. The RNDF represented roads as a collection of lanes with the associated metadata such as the number of lanes, the travel direction, and the lane width. Each lane was represented as a series of waypoints marking the lane centerline. Some of these way points were marked as checkpoints and others marked as entrance and exit points that connected the lane to the rest of the road network. Missions were specified using a Mission Definition File (MDF) that listed checkpoints that had to be visited in order. The Mission Definition File also contains information on route speed limits.

While the Urban Challenge rules stated that waypoints for lanes and zones could be inaccurate, in practice the waypoints for each lane were surveyed(Institute of Navigation 2008). The waypoints therefore represented fixed reference points as to the position of a road on the map. There was a significant amount of data gathering and surveying that needed to be done on the course in order for the vehicles to operate.

2.4.2 Path Planning in the Urban Challenge

There are two implementations of practical autonomous vehicle path planning algorithms that will be discussed. One implementation is the path planning method used in Team VictorTango's Odin. Most of the information on Odin was derived from the DARPA Technical Report (Reinholtz 2007), an article written by the team for the Journal of field Robotics (Bacha 2008), and firsthand knowledge. The other implementation will be the motion planning algorithm used in Stanford Racing's entry, Stanley, and presented in Dolgov, et al. (2008).

2.4.3 Odin

Named after the Norse god of knowledge, Odin is a 2005 Hybrid Ford Escape that has been modified for autonomous operation. The following is a simplified description which can be used as a starting point for understanding the implementation outlined in this research. For example, this experiment does not require the vehicle to follow any traffic behaviors, so I've

ignored that part of the implementation. A full description of Odin is given in Reinholtz (2007) and Bacha (2008).

In terms of hardware, Odin is equipped with several sensors that are critical for autonomous operation. For positioning, Odin uses a Novatel Propak LB+ which provides a filtered INS and GPS solution. In the best case this system provides localization accuracy within 10 cm. The vehicle uses several ground based LIDAR sensors, three four-plane laser rangefinders and four single-plane rangefinders, for obstacle detection and avoidance. Complex onboard motion planning systems keep the vehicle away from sensed obstacles while also being able to follow the given road path.

Path planning on Odin was a two-stage process. The simplest level of path planning took the checkpoints specified in the MDF and used a graph representation of the road network (given in the RNDF) to determine the shortest path of travel along the road network using the A* algorithm. Edge weights were assigned using a model of estimated travel time. This travel time model was derived using each road segment's speed limit and length and was augmented using estimated time values for specific events such as stopping intersections, performing a U-turn, and entering a zone. Once the shortest route along the road network was determined, a splined representation of that route was turned into lane boundaries and used as an input into the more complex motion planning system.

Motion planning on Odin was similar to many of the previous grid based path planning algorithms in that it used an underlying grid based cost map. This cost map was configurable, but the final configuration used was grid with a square cell size of 0.2 m extending 30 meters in front of the vehicle and 15 meters to the sides and to the rear. The values of the cost map were derived from a combination of obstacles detected using the vehicle's on-board sensors and the lane boundaries, which were used to create a virtual boundary around the current lane of travel. All costs were represented as 8-bit values for which the cost decreased based on the distance from the obstacle, thus creating a soft boundary around potential obstacles. Other costs such as distance from the road centerline and time were also used based on the situation.

Once the cost map was created, a trajectory search was performed to find a given set of goal configurations based on a configuration space consisting of the vehicles desired position

and orientation. This search was performed using a standard A* search algorithm on a regularly sampled configuration space where graph edges represented the predicted results of a finite set of future actions. In this case these predictions represented the results of driving at a certain speed with a certain turning rate for a specific amount of time. To increase efficiency these predictions were pre-computed using a model of the vehicle dynamics.

This approach was fairly robust and flexible in that allowed collision-free motion planning even with varying vehicle behaviors. In fact this motion planner was the basis for both the road and zone driving. However because this method used a grid based cost map and the vehicles state was discredited on cell centers, driving smoothness was largely dependent upon grid size.

2.4.4 Path Planning in Stanley

Stanford's entry into the DARPA Urban challenge, Stanley, used a novel and complex motion planning scheme that combined several different motion planning ideas into one algorithm. At its core, Stanley's motion planning algorithm is a lattice based A*-like search algorithm. However, Dolgov, et al.(2008) modified the standard A* algorithm to allow for a continuous position (using a vector field based method), thereby eliminating some discrete error. A potential field based on a Voronoi diagram of the drivable area was used to assign travel cost. After the original path was found, a non-linear optimization was performed to smooth and condition the resulting path.

Dolgov et al. (2008)used two methods to increase search efficiency. First, special A* heuristics were implemented to prune the search tree and second, analytical methods were used to extend the search tree. In terms of search heuristics Dolgov (2008) used linear programming (another analytical path finding method) to determine a shortest possible path through the obstacle grid and used that as a minimal heuristic. Also he pre-computed paths without obstacles and used the resulting paths a minimal heuristics in the real-time motion planning algorithm. The Stanford Paper also used analytical motion planning algorithms, specifically the Reed-Schlep motion planning algorithm, to quickly extend the search tree in areas without obstacle.

This algorithm represents a complex motion planning algorithm that includes many ideas from both discrete and analytical motion planning strategies. Additionally, the hybrid of the A* algorithms with vector field path planning is useful for speeding up algorithms that do not use the standard node/link gridded model (such as the algorithm used in this research) to trace back the least cost path. This algorithm is also an example of a path planning algorithm that uses path smoothing in order to condition the path for drivability. Finally, many of the path processing ideas used in this research, such as the vector field based path tracing and the smoothing step, were inspired by methods used in this algorithm.

2.4.5 Critique of the DARPA Challenges

One critique of the DARPA Challenges is that they relied upon carefully surveyed waypoint data. Basic waypoint following behavior served as the basis for almost every action any vehicle performed in all of the challenges; most of the interesting behaviors were treated as exceptions to waypoint following. As Norm Whitaker , the DARPA program director for the Urban Challenge, commented about the first two challenges in an interview for the Institute of Navigation: “Eighty percent of the time, they are following waypoints. It’s the other twenty percent that jumps up and bites you” (Institute of Navigation 2008). While the Urban Challenge was much more dynamic than the first two Grand Challenges, several of the teams still relied on the waypoints given in the RNDF for basic navigation. In practice however this reliance upon surveyed data limits the efficacy of autonomous vehicle technology in hazardous, hostile, or simply unknown environments. While waypoints can be digitized manually from remote sensing data, research into automated methods is important in order to decrease deployment time.

2.5 Previous Research at Virginia Tech

As part of the first and second DARPA Grand Challenge teams at Virginia Tech, two theses were written on automated path planning using remote sensing and GIS data (Stahl 2005, Poudel 2007). While the procedures outlined in both theses were never used to drive an

autonomous vehicle, they both contain interesting methods and identify significant issues involved in implementing a remote sensing based path finding algorithm.

Stahl (2005) used one foot contour data taken from surveys around the Virginia Tech campus along with road line data to create paths using different factor weights. He then performed a variability analysis on the resulting paths to examine the impact that different cost model exponentiations had on the resulting path both with and without road factors. The work was done using a combination of IDRISI Kilimanjaro for the actual path planning and ArcMap for the path analysis.

Stahl (2005) introduced several interesting concepts including a method of incorporating GIS road data into the path finding process. Stahl (2005) also has a brief discussion on how an A* algorithm could be applied Still the method described in Stahl (2005) used GIS road data and surveyed elevation as the primary inputs, both of which require either the availability of a-priori data or manual production. In the conclusion, Stahl (2005) also discussed the need for a smoothing algorithm to smooth the paths produced by his method.

Poudel (2007) developed a procedure that used the first return from aerial LIDAR data to identify obstacles such as buildings and trees and incorporate that information into a slope based cost model. While the actual cost model presented in Poudel (2006) was much simpler than the one presented in Stahl (2005), the obstacle identification procedures resulted in paths that had a much higher confidence of being navigable than those that used the bare earth elevation method presented in Stahl (2005). Still, unlike Stahl (2005) the method outlined in Poudel (2007) didn't take into account road data, though the paths generated through this method do tend to follow roads due to the cost model used and the accuracy of the data itself. However the paths do not tend to track the centerline of the roadways very well. Additionally, the paths generated are jagged and nearly un-drivable because of sharp bends and oscillations.

2.6 Summary

In summary, there is significant literature on the general path planning problem in the realms of GIS and robotics. However, most of the research in the former is conceptual and most of the research in the latter uses very structured, preplanned datasets to do the

underlying route planning. The purpose of this research was to bridge the two fields and develop a long range (i.e. beyond the range of the vehicles sensors) path planning algorithm that used remote sensing data to automatically generate paths that will be driven by an autonomous vehicle. In doing so, this experiment would also validate some of the methods developed by previous researchers and serve as a starting point for further research into remote sensing based autonomous vehicle path planning.

*Chapter 3: Autonomous Vehicle Path Planning with Remote Sensing Data**

*This chapter is a manuscript in preparation for submission to the 2009 Meeting of the American Association of Geographers

Aaron Dalton, Bill Carstensen
Department of Geography, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061

Abstract: Long range path planning for an autonomous ground vehicle with minimal a-priori data is still very much an open problem. Previous research has demonstrated that least cost paths generated from aerial LIDAR and GIS data could play a role in automatically determining suitable routes over otherwise unknown terrain. However, most of this research has been theoretical. Consequently, there is very little literature dealing with the effectiveness of these techniques in plotting paths for an actual autonomous vehicle. This research aims to develop an algorithm for using aerial LIDAR and imagery to plan paths for a full size autonomous car. Methods of identifying obstacles and potential roadways from the aerial LIDAR and imagery are reviewed. A scheme for integrating the path planning algorithms into the autonomous vehicle's existing systems was developed and eight paths were generated and driven by the autonomous vehicle. The paths were then analyzed for their drivability and the model itself was validated against the vehicle measurements of slope and position. The methods described were found to be suitable for generating paths both on and off road.

3.1 Introduction

The state of the art of autonomous vehicles has advanced significantly over the past ten years, due in no small part to the exposure and incentives offered by DARPA's Grand Challenge and Urban Challenge competitions. These challenges have shown that autonomous vehicles can obey traffic laws, avoid static obstacles, and interact with moving vehicles. But while modern sensors and software can classify situations within a few hundred meters of the vehicle, most autonomous systems still need external data to give them a wider spatial perspective. For

example, the drivable roadways and paths used in both challenges, with few exceptions, were well surveyed prior to the competition. Most autonomous ground vehicle implementations have followed the same basic assumption that accurate and properly formatted information on usable routes exists for the area in which the vehicle operates. This has led to a reliance on data gathered specifically for the autonomous vehicle and has limited autonomous vehicle operation to known predetermined routes.

One way to overcome these limitations would be to use remote sensing data such as aerial imagery and aerial LIDAR to determine potential drivable paths in a semi-automated fashion. Such a solution would have several benefits over alternative methods. Aerial remote sensing data is widely used and is well understood. Data acquisition is a routine procedure and aerial remote sensing data is either commercially available or can be captured through commercial services. Additionally, software and algorithms exist to process and store aerial remote sensing data. Previous research on vehicle path planning has focused on the use of aerial LIDAR data (Poudel 2007). Aerial LIDAR is especially suited to path planning for several reasons. First, aerial LIDAR data can have a very high spatial resolution and vertical accuracy. Furthermore, it records the distance to physical objects thus important path planning features such potential obstacles, terrain slope, and terrain roughness can be derived from aerial LIDAR data.

While previous research (Poudel 2007, Stahl 2005) has demonstrated that methods using only remote sensing and GIS data generate reasonable paths, there is still an open question as to whether similar methods could be successfully implemented to execute a drive by an autonomous vehicle. Also, there are additional implementation details that must be considered in order to turn a GIS path into a form that an autonomous vehicle can follow. Furthermore, any method developed to solve this problem must be compatible with the other operational systems of the autonomous platform including obstacle avoidance.

3.1.1 Problem Statement

The purpose of this research was to implement an autonomous vehicle path planning algorithm using aerial LIDAR and remote sensing imagery and then drive the computed pathways with an autonomous vehicle. A successful method would help to answer the following questions:

- Can the methods described in previous research (Stahl 2005, Poudel 2007) be implemented to determine a route for driving an autonomous vehicle?
- What would a practical implementation of these methods look like?
- What are issues encountered when implementing these methods on an autonomous vehicle in conjunction with other systems?

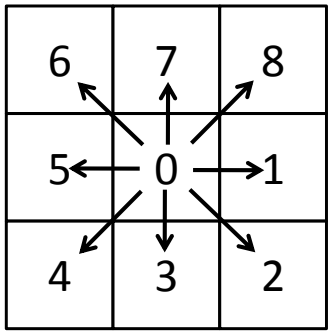
3.2 Previous Work

The idea of using GIS and remote sensing techniques to aid in autonomous vehicle path planning has existed for several decades. Possibly the first paper to discuss the subject was Zimmerman (1985) who discussed GIS requirements for autonomous vehicle operation. In the literature, GIS implementations have served as repositories of geographic information for autonomous vehicle navigation (Li, Zheng and Cheng 2004)(Srinivasan, Jonathan and Chandrasekhar 2004) and as an interface to support autonomous vehicle operations (Meguro, et al. 2005). GIS databases have also been incorporated into dynamic world model implementations for robotic and autonomous vehicles, an application with a direct relationship to autonomous vehicle path planning (Kent 2007).

Several autonomous vehicle applications have used remote sensing or GIS technology. The most widespread adoption has been in the field of agricultural vehicle automation. The agriculture industry has a long history of using GIS and remote sensing technology in its management practices, so it's no surprise that research into agricultural automation should incorporate these technologies as well. Several papers have discussed the potential of GIS and remote sensing relative to path planning for agricultural purposes (Blackmore, et al. 2004, Xiangjian and Ganh 2007). Blackmore, et al. (2004) described the system requirements for a fleet of autonomous tractors and discussed the integration of GIS and remote sensing into a

centralized management system that would control the vehicles and serve as a repository of data gathered by the fleet. Xiangjian and Ganh (2007) developed a Navigational Geographic information system that combined a GIS system with a path searching algorithm to control an autonomous tractor. Their system was capable of planning paths between given waypoints in an open field and used GIS software to digitize the path points and display a preview of the planned path. They did not use any GIS or remote sensing data to plan their paths however. Other applications that have incorporated a GIS into autonomous vehicle planning include security (Meguro, et al. 2005). Meguro used GIS software to control a fleet of autonomous security vehicles and to view data from the entire autonomous system. There is also a history of using GIS and remote sensing technology in the DARPA Grand and Urban Challenges that were the original impetus for this research.

3.2.1 Accumulated Cost Surface



Backtrack Direction as specified in ArcGIS

Figure 3.1: Backtrack direction

Most GIS implementations that incorporate grid based path planning algorithms include an accumulated cost surface method. This method uses a cost layer that represents the relative difficulty of moving through a particular cell and then finds the path from a source cell to a destination cell (through cell centers) that has the least total cost. This is accomplished by creating a layer, called an accumulated cost surface, in which each cell holds the accumulated cost of the shortest path from that cell to the given source cell. Simultaneously, a layer, called a backtrack layer, is generated in which each cell represents the direction to the previous cell in the shortest path. Using this backtrack layer a path can be traced from any cell back to the source cell.

Most grid based path planning algorithms are related forms of network based algorithms such as Dijkstra's and A*. In order to apply a network based algorithm to raster data, a node/link representation is used. In this representation, the gridded data set forms a lattice in which each cell center represents a node with edges to its direct neighbors. Edge

weights are assigned based on cost values of the two adjacent cells and the travel direction as well as other potential directional weights. Using this method GIS implementations can apply network algorithms such as A* and Dijkstra's Algorithm to solve path planning algorithms on raster datasets.

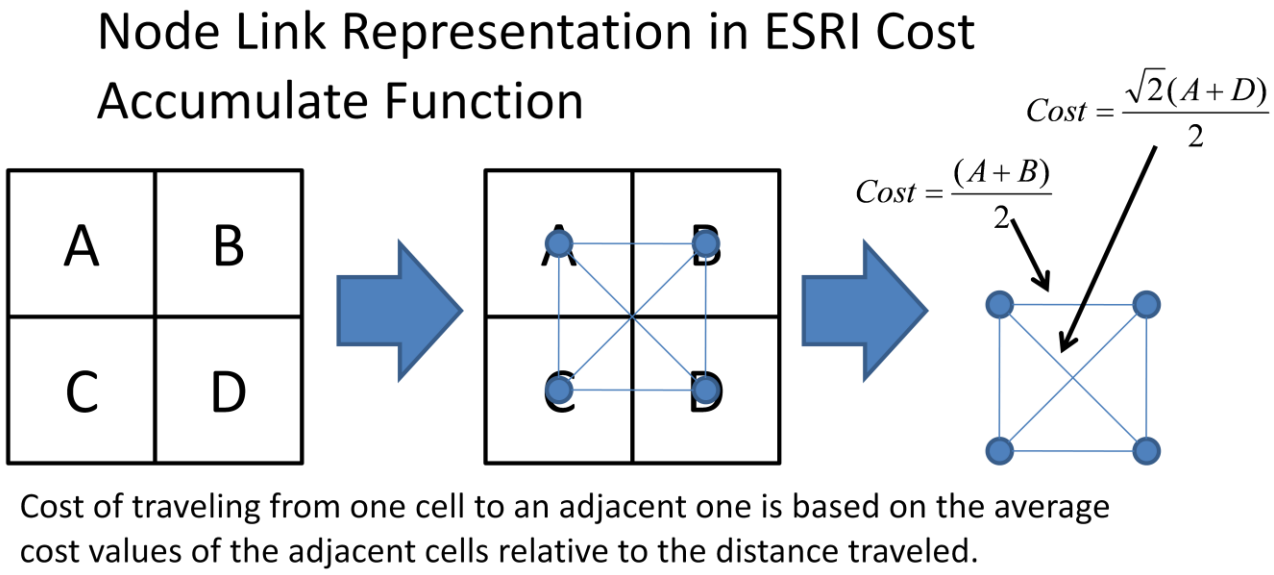
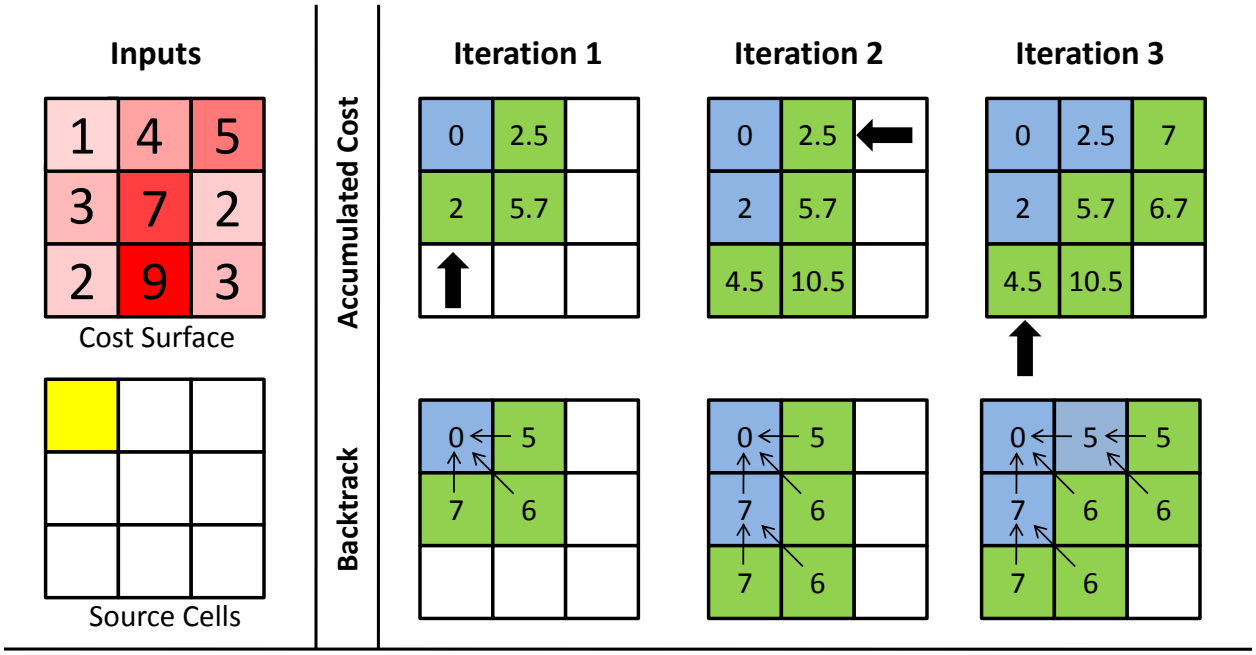


Figure 3.2: Illustration of node/link representation

From a conceptual point of view, the cost accumulation step uses a modified Dijkstra's algorithm on the node/link grid representation of the given raster layer as the path search method. Since each cell location is only connected to its immediate neighbors in the node/link representation, the backtrack value can be represented as an integer in the range of one to eight that specifies the specific neighbor cell along the least cost path. The exact values used in the ArcGIS implementation are given in Figure 3.1(ArcGIS 2008).

An illustration of the full algorithm for generating the accumulated cost surface and the backtrack layer is included in Figure 3.3. The inputs to this algorithm are a positive value cost layer and a binary layer indicating source points. During the update step of the algorithm a new accumulated cost and backtrack value is calculated for all the non-visited neighbors of the current cell. During this update step the cost of traveling to a neighboring cell is calculated using the edge weights of the node/link representation. That cost is added to the accumulated cost value for the current cell to get potential accumulated cost for the neighbor. If there is no accumulated cost set for that neighbor or the current accumulated cost value is less than a

previously calculated accumulated cost value, the accumulated cost is updated and the backtrack layer is set to the value that represents the direction from the neighboring cell to the current cell.



1. Create an accumulated cost and backtrack layer at the same size as the cost surface layer
2. Mark the source cells as visited ( ). Initialize the backtrack and accumulated cost layer to zero at the location of the source cells. Update the backtrack and accumulated cost of possible neighbors ( ).
3. Select the neighbor ( ) with the least accumulated cost as the current cell ( ).
4. Mark the current cell as visited ( ). Update the backtrack and accumulated cost layers based on cells reachable from the current cell. Update list of possible neighbors ( ).
5. While there are still cells not visited repeat from step 3.

Figure 3.3: Illustration of cost accumulation algorithm

Accumulated cost surface based path planning has several advantages that make it very useful for GIS applications. The primary advantage is that the raster cost layer can be easily derived from other raster based GIS layers using simple map algebra. Raster cost parameters can also be easily derived from pre-existing gridded data sources, such as gridded Digital Elevation Models (DEMs), remote sensing imagery, or derived sources such as land cover classifications. An example of using elevation and vector data to generate a cost map is given in Figure 3.4.

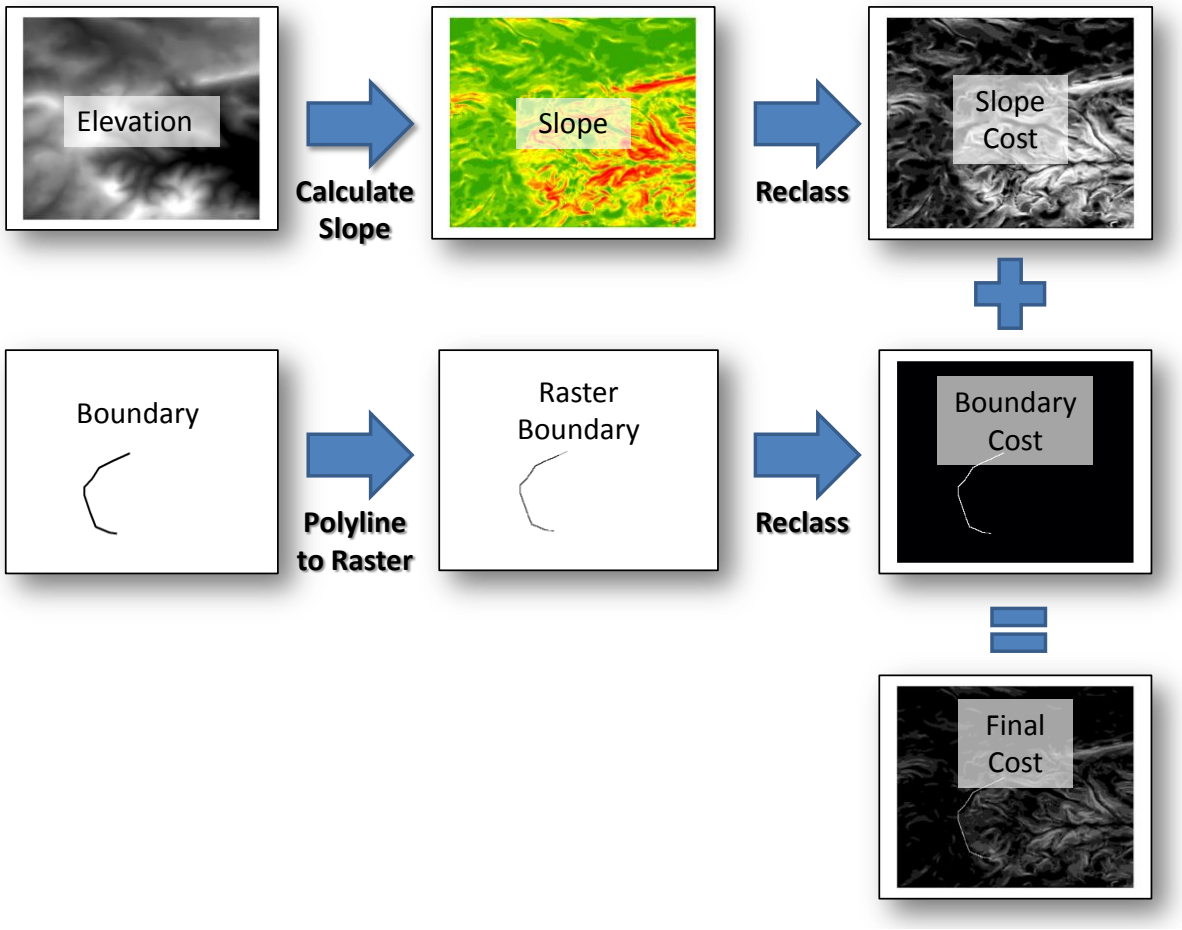


Figure 3.4: Example of cost surface generation

However, there are several drawbacks to the standard accumulated cost surface model. First, there is inherent path distance error that arises from the restriction of to 45 degree increments. While there are methods to correct for these errors, the implementation is complex and is typically not used in most GIS applications. Second, because the accumulated cost surface model uses Dijkstra's algorithm, it is relatively slow for path planning when compared to similar A* algorithms.

3.3 GIS and Remote Sensing in the DARPA Challenges

The DARPA challenges are unique in the field of autonomous robotics as they form a set of standardized problems that were investigated by several dozen research groups, each with vastly different approaches. The DARPA challenges form a benchmark by which different autonomous vehicle methods can be compared. Over the course of the two DARPA Grand

Challenges and the DARPA Urban Challenge, there have been several teams that have used a combination of remote sensing and GIS techniques to aid in path planning.

Toth (2006) described the mapping support system developed at Ohio State University and used in two DARPA Grand Challenges by Team TerraMax and Desert Buckeye respectively. This mapping system served two purposes. First, there was an interactive component that visualized the given DARPA waypoints along with DOQQ imagery, SRTM topography, and vehicle dynamics. Using this system, the given waypoints could be analyzed and new waypoints added or the corridor width changed manually to increase the drivability of the given route. Second, there was a real time mapping system that used the onboard GIS data along with real time sensor input to plan a path if the given DARPA waypoints were too sparse or if the vehicle moved a long distance from the predetermined path due to obstacle avoidance. In what would become a common theme however, these systems were rarely used because the given DARPA waypoints were unexpectedly dense.

Another implementation, briefly outlined in Goldberg (2004), was remarkable because GIS software was used in almost every aspect of vehicle planning. Goldberg (2004) hinted at a system which used both raster and vector path planning methods to improve the given route information, developed a GIS workflow for manual digitization, and made use of geo-databases for backend storage. Still, these methods were never formally described and the team did not finish the DARPA Urban Challenge.

3.4 LIDAR Processing

The LIDAR processing routines used in this experiment are largely an extension of methods used in Poudel (2007). Poudel described a process for deriving a viable cost surface for use in autonomous path planning from aerial LIDAR. Poudel first identified obstacles from gridded first-return LIDAR data using standard morphological operations and assigned them an arbitrarily high travel cost. The method of separating obstacles from the LIDAR data was largely based on the results of Weidner and Forstner (1995) and Morgan and Habib (2000) dealing with the separation of above ground objects from the ground surface using aerial LIDAR data. After likely obstacles were indentified, Poudel then used the LIDAR data to calculate slope values and

assign costs to different value classes. A final cost surface was then derived by combining the obstacle and slope costs.

Extracting features and surface information from raw LIDAR point cloud data is a very difficult problem, so most algorithms convert the LIDAR point cloud to a grid based Digital Surface Model (DSM) Poudel (2006) used triangulation with linear interpolation to derive a DSM. Current research shows that for a sufficiently dense LIDAR dataset relative to grid size there is little difference among interpolation methods (Goncalves 2006).

3.4.1 Odin - The Autonomous Platform

Odin, Virginia Tech's entry into the DARPA Urban Challenge, served as the autonomous vehicle platform for this experiment (Figure 3.5). Named after the Norse god of knowledge, Odin is a 2005 Hybrid Ford Escape that has been modified for autonomous operation. The following is a simplified description to be used as a starting point for understanding the path-finding implementation outlined in this paper. As this experiment does not require the vehicle to follow any traffic behaviors, this part of the original implementation is not described. A full description of Odin is given in Reinholtz (2007) and Bacha (2008).

Figure 3.5: Odin – Virginia Tech's entry into the DARPA Urban Challenge

Odin is equipped with several sensors that work together to enable autonomous driving. For positioning, Odin uses a Novatel Propak LB+ which provides a filtered Inertial Navigation System and Global Positioning System solution. In the best case the localization accuracy is within 10 cm. The vehicle uses several ground based LIDAR sensors, three four-plane laser rangefinders and four single-plane rangefinders, for obstacle detection and avoidance. A complex onboard motion planning system helps Odin avoid obstacles while following a predetermined path.

The fundamental problem in the DARPA Urban Challenge was to drive to specific waypoints in a predetermined road network while avoiding obstacles and obeying traffic laws. The road network was defined by a Route Network Definition File (RNDF) with the target waypoints were defined by a Mission definition file (MDF). In general, the first step in autonomous operation was to determine a global route through the road network that most efficiently visited the points specified in the MDF. Once this global route was determined, a two dimensional representation of the road surface was converted into boundaries for a small cost surface. Obstacles detected with real-time sensors onboard the vehicle were then added to the cost surface and a motion planning algorithm was run to calculate vehicle commands needed to reach a certain target point and velocity. The vehicle then moved and the process was repeated until all the specified waypoints were visited.

In order to perform this experiment and have Odin drive the-generated paths, the output of the path planning algorithms was converted to an RNDF and an MDF. The RNDF specified a simple one-lane virtual road with a constant lane width corresponding to the average lane width of a two-lane road as reported by the Virginia Department of Transportation (12 feet). This virtual road represented a path from the vehicle's current position to the chosen destination. The MDF specified the last point in the RNDF as the target point. Conceptually, the path finding algorithm simply defined a fixed width corridor through which the vehicle operated. Within that corridor the vehicle's normal obstacle avoidance could guide the vehicle past obstacles that might not have been evident in the remote sensing data either because of scale of representation or changes since the LIDAR data were flown in 2005.

This implementation had several benefits. First, it was simple and required no modification of existing vehicle systems. The path finding algorithm could then be tested independently from the rest of the vehicle's software. No additional considerations were needed to deal with obstacles close to the predicted path as the vehicle's onboard systems would avoid them.

The obstacle avoidance routines also presented some problems however. The obstacle detection algorithms were tuned for operation on roadways and not configured for off-road use. Small features such as overhanging branches, tall undergrowth, and large potholes were seen as obstacles by Odin and occasionally impeded vehicle progress. Additionally, large obstacles that took up the entire virtual lane were un-navigable because the vehicle is programmed not to go beyond the boundaries of its virtual road, and the virtual road was too narrow to allow the on-board systems to by-pass such obstacles. In these cases, the vehicle would stop awaiting further input from the human driver.

3.5 Path Planning

In general, a good path for an autonomous vehicle would be exactly the same as the path a human driver would choose. This means following roads whenever possible carefully weighing tradeoffs between driving distance, difficulty and obstacle avoidance. Additionally, because the path is described by waypoints, normal sampling constraints must be met; waypoints must be sufficiently dense. Finally, when developing paths for an autonomous vehicle, mechanical constraints of that vehicle must be considered. Valid paths are required to be smooth (i.e. the rate of directional change is to be relatively continuous) and that the path must not exceed a maximum curvature defined by the vehicle's turning radius. In this experiment, normal accumulated cost surface models were used to accomplish the first task. Such models are good at weighing multiple factors in determining a path. To achieve a sufficiently smooth and dense path, a vector field based model was used to generate the path from the accumulated surface. Lastly, a smoothing algorithm was used to produce a path that the met the requirements of path curvature.

3.5.1 Cost Surface Model

There were three components of the cost surface model used in this experiment. One was a base cost layer that was derived from weighted slope classes. This component was derived from aerial LIDAR data for the study area. The other component was a "likely road" layer that was used to reduce slope cost based on proximity to a road centerline. This layer was derived from true color aerial photography. In the final model the presence of a road reduced the base cost by a factor based on the density of likely road cells in that area. Finally, obstacles were identified and assigned a barrier cost which was then added to the combined base cost and road factor layer.

It should be stressed that none of these methods is meant to be particularly rigorous, but rather to assure that a GIS solution for autonomous vehicle navigation is possible. Nor should it be assumed that these methods are universally applicable to every study area. Further research is necessary to develop rigorous methods that can be applied to a broad variety of areas.

3.5.2 Study Site, Data and Data Issues

Several factors were important in site selection for this project. First, were safety and traffic considerations. The study area needed to have relatively little traffic so that testing could be conducted safely. The study area also should be able to be easily controlled with very little through traffic. The study area also needed to have a mixture of sloped and flat areas to test the influence of slope cost on the path model. Both high quality area photography and LIDAR data needed to be readily available for the study area. The study area used in Poudel (2007) and partially used in Stahl (2005), known as plantation road on the Virginia Tech Campus, met all of these requirements.

Plantation road is a cluster of laboratories and buildings bordered by a small wooded area and surrounded by agricultural fields. There is only one entrance in and out of the area. The main road is paved with two gravel roads branching off on both sides and leading up the hill towards a wooded area. One road travels into the woods and over a hill to labs on the

other side. An image of the study area, with an overlay of all driven routes is presented in Figure 3.12.

Figure 3.6: Exaggerated 3D view of study area

This experiment used true color aerial photography publicly available through the Town of Blacksburg and LIDAR data obtained from the Center for Geospatial Information Technology at Virginia Tech and used with permission from the Town of Blacksburg. Both the aerial photography and the LIDAR data were acquired in 2005. The aerial photography is nominally 3 inch resolution and the LIDAR has a resolution of 3 ft along the scan line and 7 ft between scan lines. The aerial LIDAR used in this model appeared to have significant data omissions. Large swathes of returns were missing from both forested areas and very flat, highly reflective surfaces such as building roofs and roadways. Examples are given in Figure 3.7. These data omissions are a result of post processing done on the LIDAR data and can cause problems when the LIDAR points are gridded. Preferably, raw geo-referenced LIDAR data would be used in this process.

Figure 3.7: LIDAR omissions on building an in wooded area

The first step in LIDAR processing was to grid the data using a Inverse Distance Weighted interpolation method with a 2.5 distance weighted power, a variable radius, a minimum of 4 target points, and a maximum radius distance of 5 meters. The small radius, low minimum number of target points and high distance weighting minimized elevation errors caused by distant target points in the gridding process.

3.5.3 Derivation of Base Cost

The base cost was derived based on a similar method presented in Poudel (2007). A slope calculation was performed on the gridded LIDAR data and the resulting slope values were then separated into several classes. The lowest class was then assigned a base cost of one and each subsequent cost class was assigned a cost twice as much as the previous class. The actual cost values for the slope ranges are illustrated in Table 3.1.

Table 3.1: Slope cost values

Slope Range (degrees)	Cost Value
0.0 - 3.294	1
3.294 – 6.537	2
6.537 – 12.966	4
12.966 – 27.994	8
27.994 – 90.00	16

Rather than the exponentiated slope cost proposed in Stahl (2005) (with an upper bound being quadratic), this experiment uses an exponential cost base on classified values. Therefore the base cost is largely dependent on the number of classes used to segment the slope and the breaks used in slope classification, making the base cost algorithm highly tunable. The cost values for different numbers of slope classes are illustrated in Figure 3.8. These relative weights are less than the values proposed in Poudel (2007), to allow for a higher influence of reduced road costs.

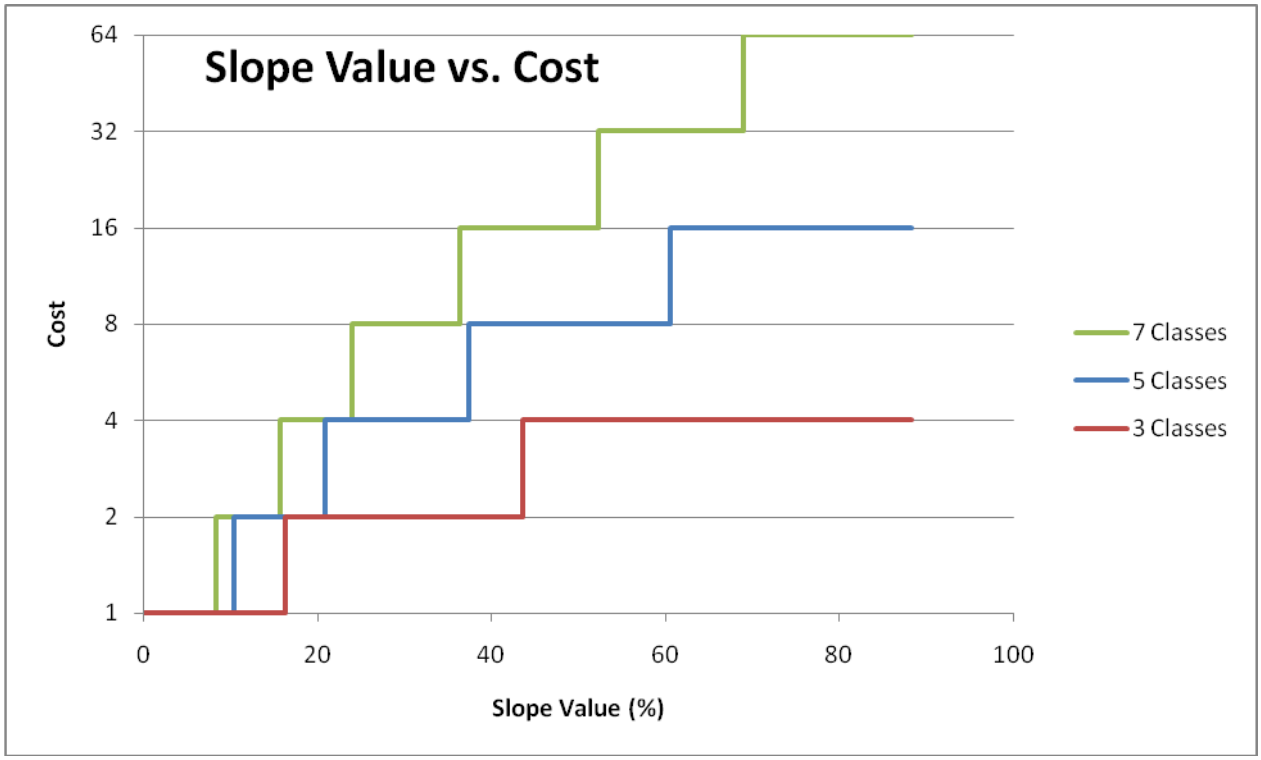


Figure 3.8: Slope value vs. number of classes

3.5.4 Road Identification

The first step in the road identification procedure was to select a small number of sample road points. Based on experimentation, fifteen to twenty road points are usually sufficient to get good results. Once these road points were selected, an iso-cluster routine was run on the high resolution imagery to produce 20 classes (a number determined by experimentation). A maximum likelihood classification was then performed on the imagery to produce a classified image. The sample points were then used on the classified image in a voting scheme to select the best road class. In this scheme, each sample point was given one vote for the underlying class at that sample point. The class with the most 'votes' was then selected as the most likely road class. Finally, this road class was used in a class probability analysis. The results of this probability analysis were then reclassified so that all pixels with a probability greater than or equal to 50% were identified as road pixels and those less that 50% were identified as non-road. Then the results were re-sampled using majority re-sampling scheme to the resolution of the cost map.

The previous procedure resulted in a binary image of potential road pixels at the same resolution as the cost layer. This result is similar to what was used in Stahl (2005), but it is still insufficient. He had used an interpolated vector layer that represented the road centerline while the results of the procedure above represented the entire likely road area. Ideally road centerlines would have the highest value. To accomplish this, a 5x5 mean filter was run on the resulting image. The filter size was chosen to be slightly larger than the virtual lane size (at 1m resolution a 5x5 rectangle covers 16.4 feet in both directs verses 9.8 feet for a 3x3 filter). The reasoning was that the road pixel value should be a low estimate of percentage of road area within a lane centered at that pixel. The mean filter produced an image where each cell had a value from 0 to 1, representing the average value of pixels in a 5x5 window. Thus pixels in the center of the road (which were surrounded by likely road pixels) were given a higher value than pixels on the road edge. This filter also helped smooth out errors caused by misclassification due to shadows or other imagery issues. Finally the resulting road fuzzy logic image (R) was rescaled using the equation $Cost = \frac{1}{4R-1}$ and then multiplied against the based cost. This rescaling assured that pixels with a value of 1 (a likely road center pixel) would be one fifth the

base travel cost of a non-road pixel with the same slope class. Such a scaling enhances the probability that road cells will be used to create the output GIS path.

There are several issues with this classification scheme. First, this method is only usable in areas where there's a significant spectral difference between roads and non-road areas as existed in the study area. Next, small spectral differences such as shadows caused several road areas to be misclassified. More significantly, other manmade features such as buildings were often misclassified as road pixels, as illustrated in Figure 3.9. Clearly a more elaborate classification scheme could have produced a better likely roads layer, however, in practice the obstacle identification scheme described in the next section were sufficient to counteract the effects of this misclassification.

Figure 3.9: Building misclassified as road

3.5.5 Obstacle Identification

Potential obstacle costs were also calculated based on methods in Poudel (2007). Obstacles were identified by first deriving a Minimum Height Surface (MHS) from the Digital Surface Model (DSM) using repeated application of a minimum morphological filter. The MHS is then subtracted from the DSM and if resulting value is greater than a certain threshold, the pixel is classified as an obstacle and assigned an arbitrarily high obstacle cost. This step removed the buildings effectively from the likely roads layer.

3.5.6 Final Cost Surface

The final cost surface was derived by multiplying the base cost by the road cost factor and then adding the obstacle cost. This scheme ensured that barriers always had a very high cost and that a road pixel had a value relative to its road confidence and slope cost.

3.6 Path Derivation

Given a source pixel and the cost surface described above, an accumulated surface and backtrack layer were calculated using standard accumulated cost surface algorithms. In a standard accumulation cost surface path planning model, a raster based path would be generated by tracing a path of cells from a source cell to the destination using the backtrack layer. Then the vector path would be generated by connecting the center points of each cell in the path. This method tends to produce very jagged paths, as angles between line segments that make up the path are limited to forty-five degree increments. Additionally the points that define the path are unevenly spaced due to the diagonal path segments whose length is 1.414 times that of axial path segments.

In order to overcome these drawbacks in this experiment, a vector field path tracing method was used. In this method, the backtrack layer is converted to a vector field in which each cell was represented as a vector with a magnitude equal to the cell size, pointing in the direction of the next cell in the least cost path. The destination cell is then set to a vector with a velocity of 0. This vector field therefore has only one sink at the destination cell. In order to trace a path, a point is placed at the source and the position change is calculated using a bilinear interpolation between the closest four vectors. This process is repeated until the position change falls below a certain threshold. An illustration of this method is given in Figure 3.10.

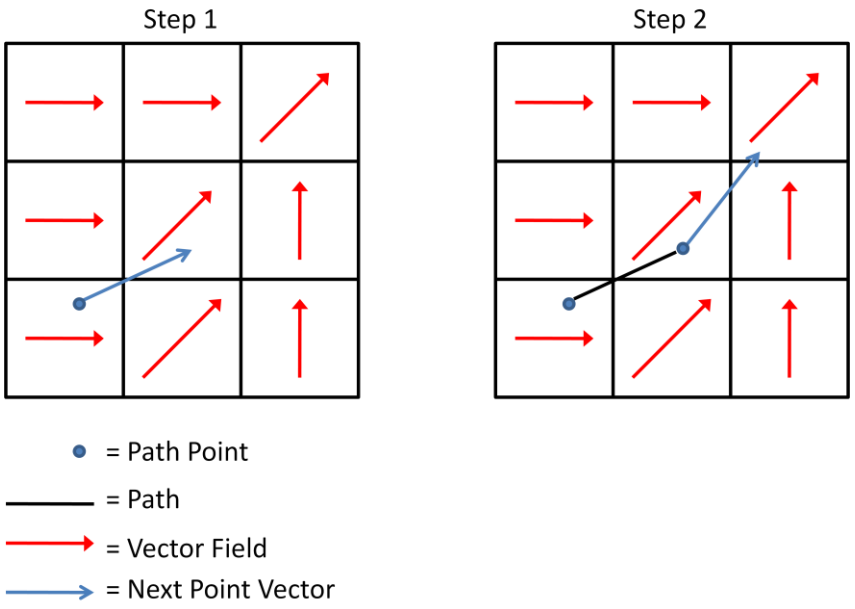


Figure 3.10: Vector field path planning

One issue with this method is that at the destination point, the path can become erratic because of overshooting the destination cell. This error can be corrected by a filter of the angle made by the last three points. If the angle formed by the last three points is greater than a given threshold than the last point is removed and the filter is rerun until no more points are removed.

3.7 Path Smoothing

The last step in the process of generating the path is to run a smoothing algorithm from the resulting points. In fact Stahl (2005) wrote about the need for a smoothing algorithm in his conclusion as a requirement for a practical implementation. An ideal smoothing algorithm should preserve even point spacing and the general shape of the path itself. The Douglas-Peucker algorithm, the most common smoothing algorithm in GIS, affects both point spacing and overall curve geometry. The Wang algorithm (Wang, 1999), another popular smoothing algorithm, preserves general curve geometry but still affects point spacing. Therefore, the Snakes line smoothing algorithm was chosen for this experiment (Berghardt 2005).

Snakes is a point displacement algorithm, meaning it moves points rather than removes them. It preserves the general shape of the path, while also smoothing both the line itself and the curvature of the line. The code used for performing the snakes smoothing was an altered version of the code used in GRASS GIS (GRASS GIS 2008) and the amount of smoothing was controlled by the number of iterations of the algorithm. In this case twenty iterations were performed to derive a smooth path. The results of the smoothing algorithm on an unsmoothed path are shown in Figure 3.11.

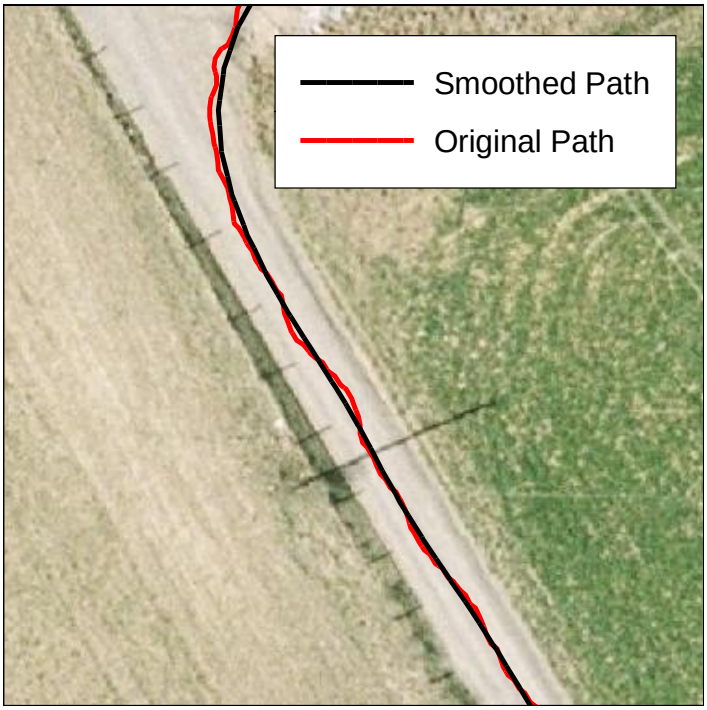


Figure 3.11: Smoothed vs. Original Path

3.8 Results

3.8.1 Experimental Methodology

For the actual experiment, eight paths were planned and driven. Each path was chosen to test certain aspects of the implementation including obstacle avoidance, rough terrain, wooded area, and off-road path planning. The vehicle was driven to a general starting area as laid out in the test plan. The current vehicle position was then used as the starting point for each path. The endpoint was chosen in the field during the test based on the test plan. Telemetry data, including position and orientation, were captured from the onboard GPS/IMU system continuously at a rate of 50 Hz for the entire test. Additionally, software logs were kept from all of the onboard components that control autonomous driving. These logs were later time synchronized to the GPS/IMU logs and used to parse out the individual autonomous runs.

Video was captured from a roof mounted camera located on the drivers' side for six of the seven paths (the first path driven was a dry run of testing procedures). The generated paths for all the runs are mapped in Figure 3.12. The data presented in this report represents telemetry logged from the beginning to the end of each autonomous run.

Figure 3.12: Map of all runs

For the eighth run, there was a fence that was not detected by the aerial LIDAR data and therefore was not recognized as an obstacle. In order to test off-road path planning, this obstacle was added manually to this cost layer as seen in Figure 3.13. Additionally, because we

were testing in an un-mowed field, which would have appeared as an obstacle to the vehicle's onboard systems, onboard obstacle detection was disabled for the length of this run.

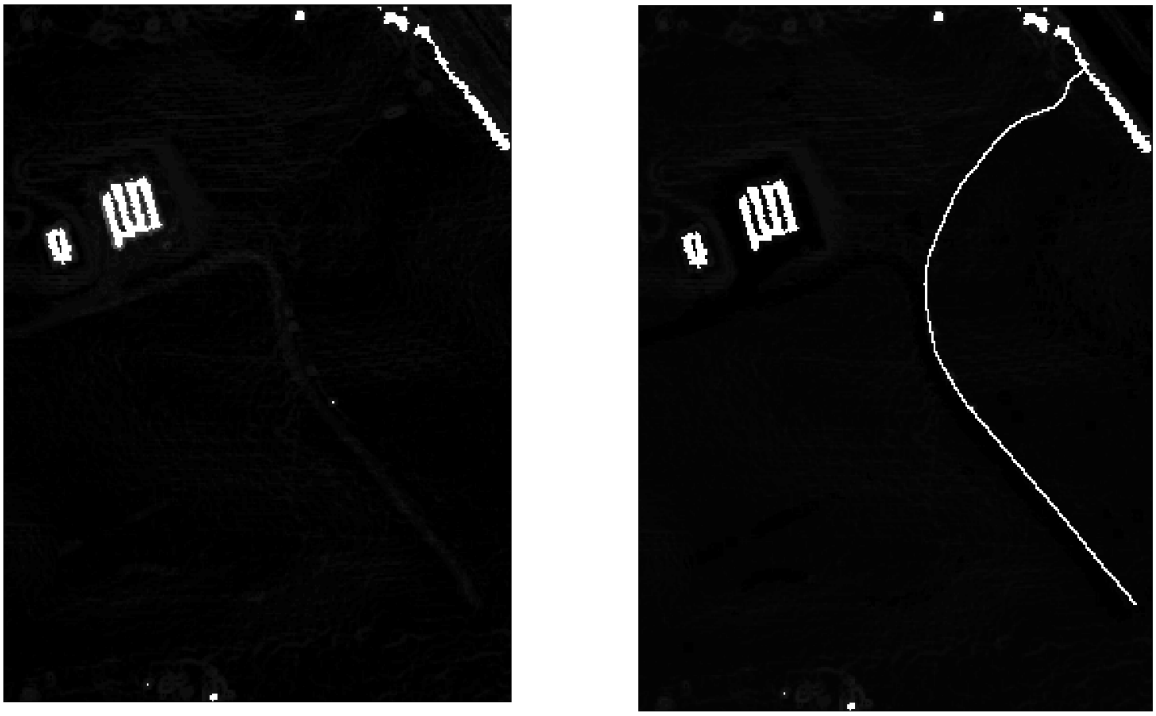


Figure 3.13: Fence manually inserted into cost layer

3.8.2 General Impressions

The following are descriptions of each run and general impressions taken from test notes and video analysis.

Run 1

Illustrated in Figure 3.14, this was a dry-run for testing procedures. It was a simple run down the main paved street in our test area. There was a slight planned motion to the right hand side of the road, probably caused by the parking lot affecting the value of the road factor on that side of the road. This anomaly is highlighted as 'feature 1' in Figure 3.14 and also appears in when the run was repeated in the opposite direction in run 2.

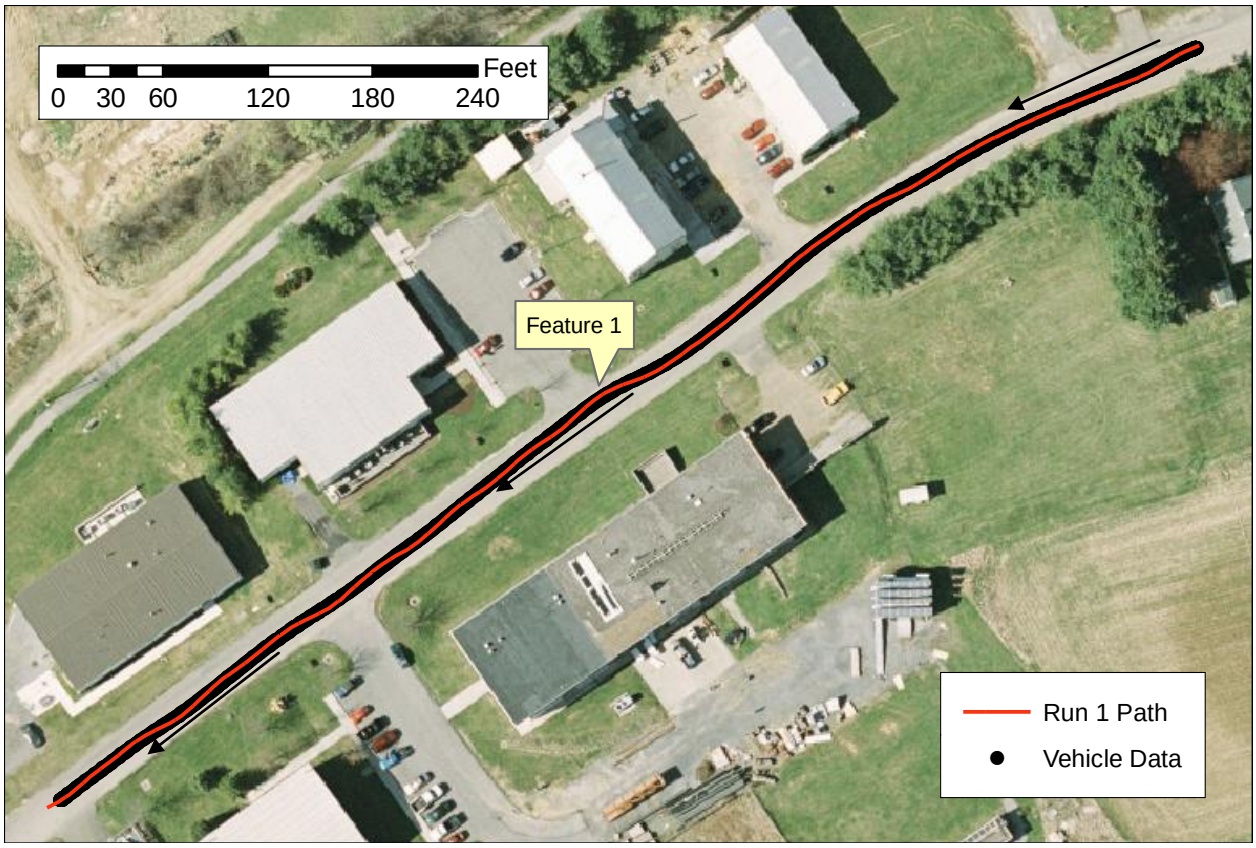


Figure 3.14: Map of Run 1

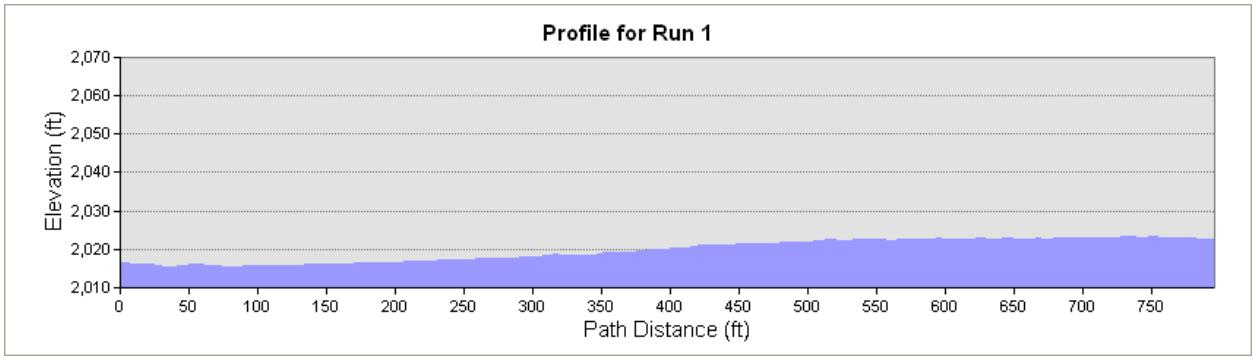


Figure 3.15: Profile for Run 1

Run 2

Illustrated in Figure 3.16, this run was a repeat of the last run in the opposite direction. Again this run was supposed to drive down the main paved road. The curve towards the parking lot was also present and can be seen as 'feature 1' in the figure and from the video image in Figure 3.18. Finally, a similar anomaly was seen at the end of the run when the vehicle ran off the road. This was also caused by an anomaly in the road layer caused by a combination

of gravel and the building on the left hand side and overhanging trees on the right. This is illustrated as 'feature 2' in the map and from the video image in Figure 3.19.

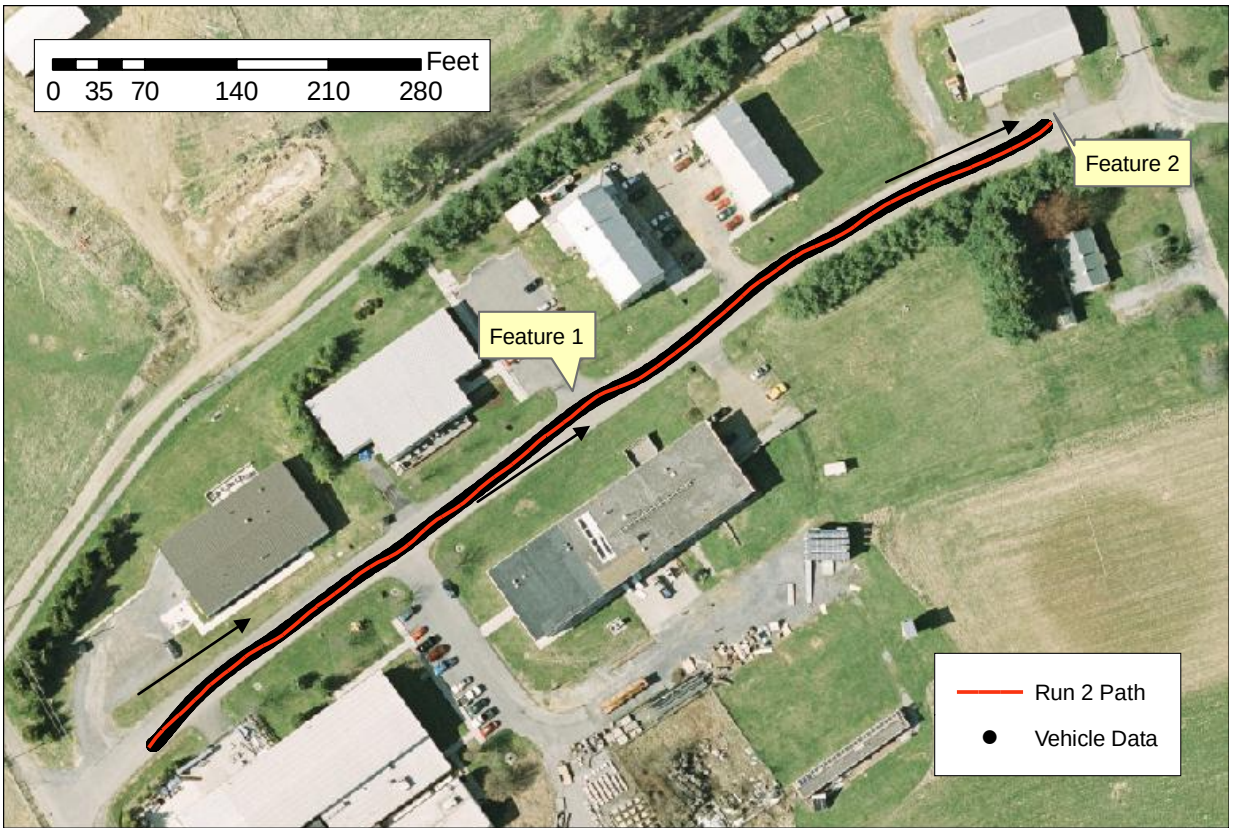


Figure 3.16: Map for Run 2

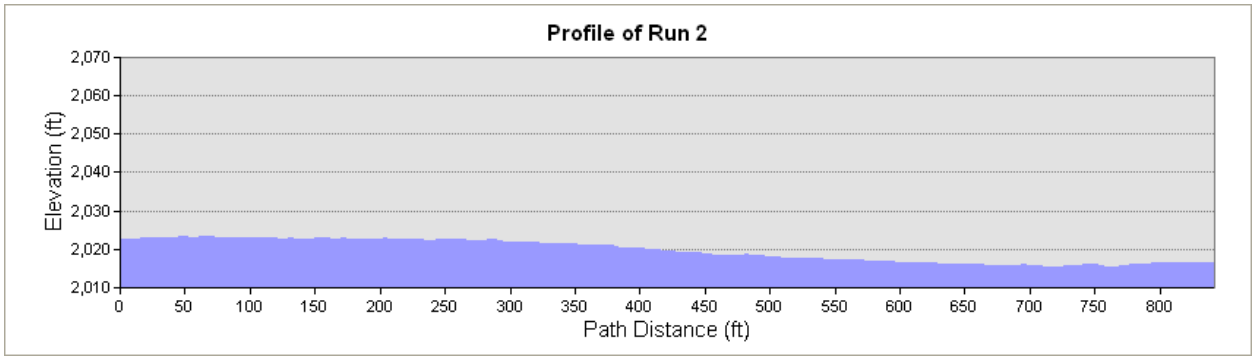


Figure 3.17: Profile for Run 2

Figure 3.18: Vehicle tracking towards side of road

Figure 3.19: Vehicle in gravel

Run 3

Figure 3.20: Map of Run 3

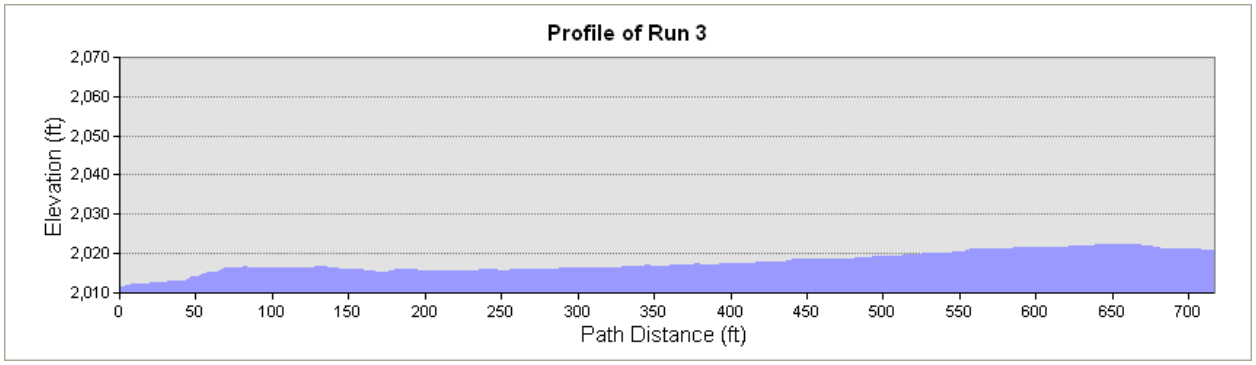


Figure 3.21: Profile for Run 3

This run, illustrated in

Figure 3.20 was a more complex version of run 2 that started and ended in two different parking lots. While the same features were seen in this route as well, the vehicle mostly tracked towards the center of the road and entered the parking lot without incident.

Run 4

This run was the same as run 3, only in the opposite direction (Figure 3.22). Again this was done in order to demonstrate repeatability of the algorithm. Like run 3, the vehicle left the

first parking lot and entered the second without incident. It should be noted that after the route was generated for this run, the vehicle had to be turned because it was facing slightly in the wrong direction. It should therefore be noted that a drawback of the algorithm is that it doesn't take into account the direction that the vehicle is pointing when the path is generated.

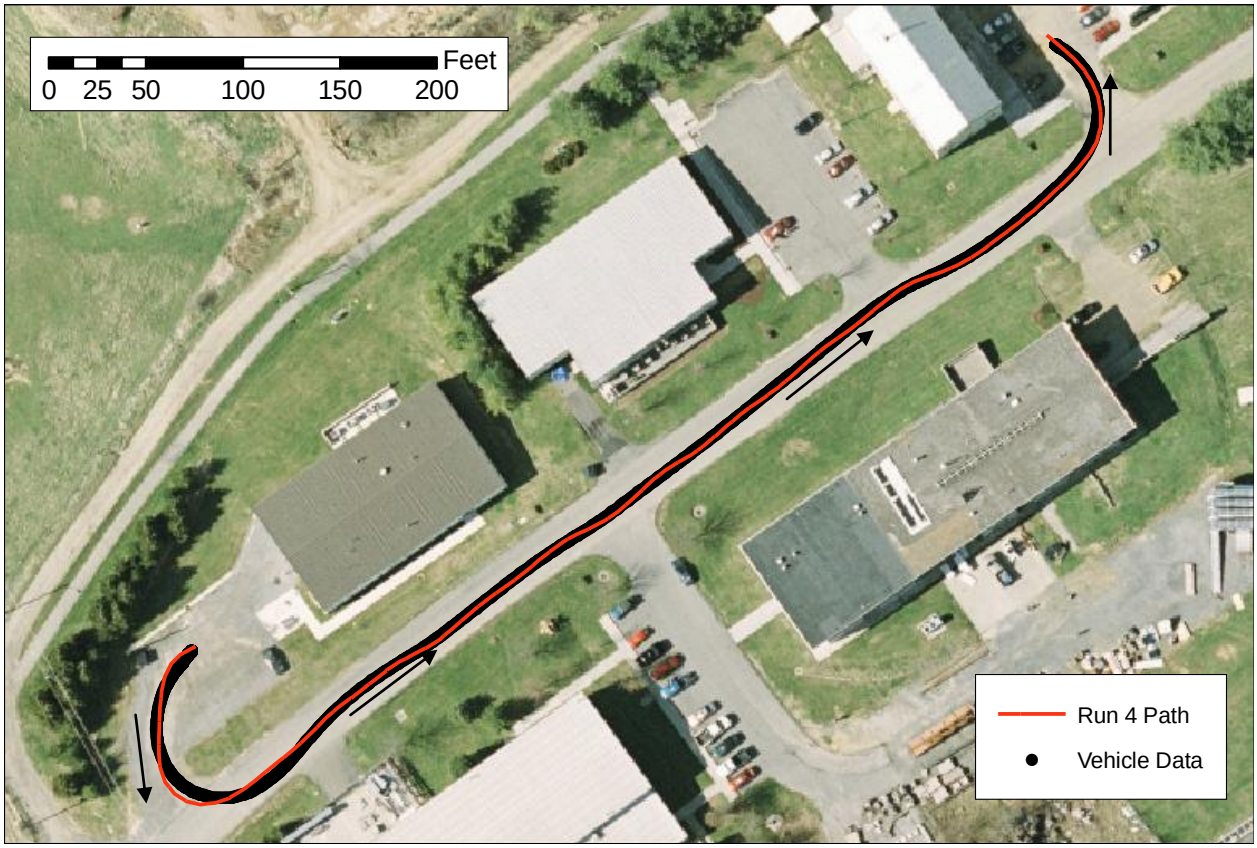


Figure 3.22: Map of Run 4

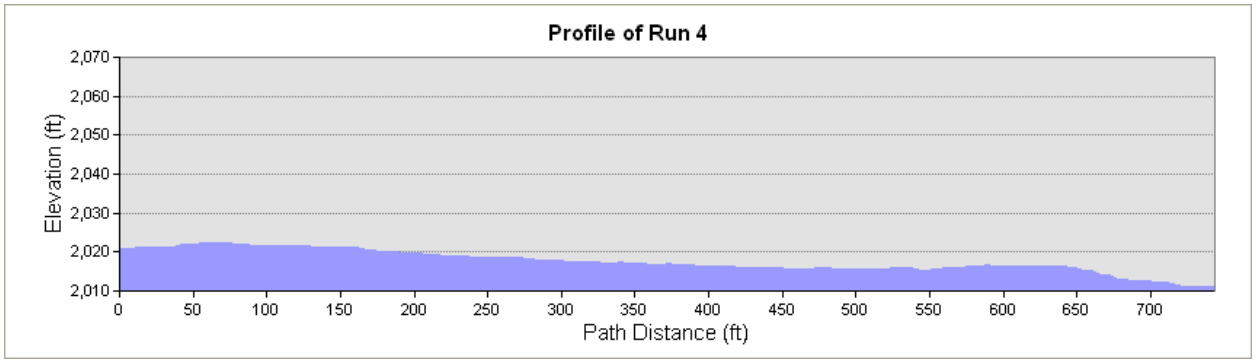


Figure 3.23: Profile of Run 4

Run 5

This run was performed to test vehicle operation along rough roads and through obstacles. The road in the very bottom of Figure 3.24 was full of potholes and dips while the area labeled as 'the junkyard' was a collection of building materials lying on the side of the road. While there was a path through this obstacle course, some of the objects were smaller than would be expected to be detected by LIDAR. The main goal of this route was to see if the vehicle could operate on the rugged road and avoid most of the obstacles with the automatically generated route.

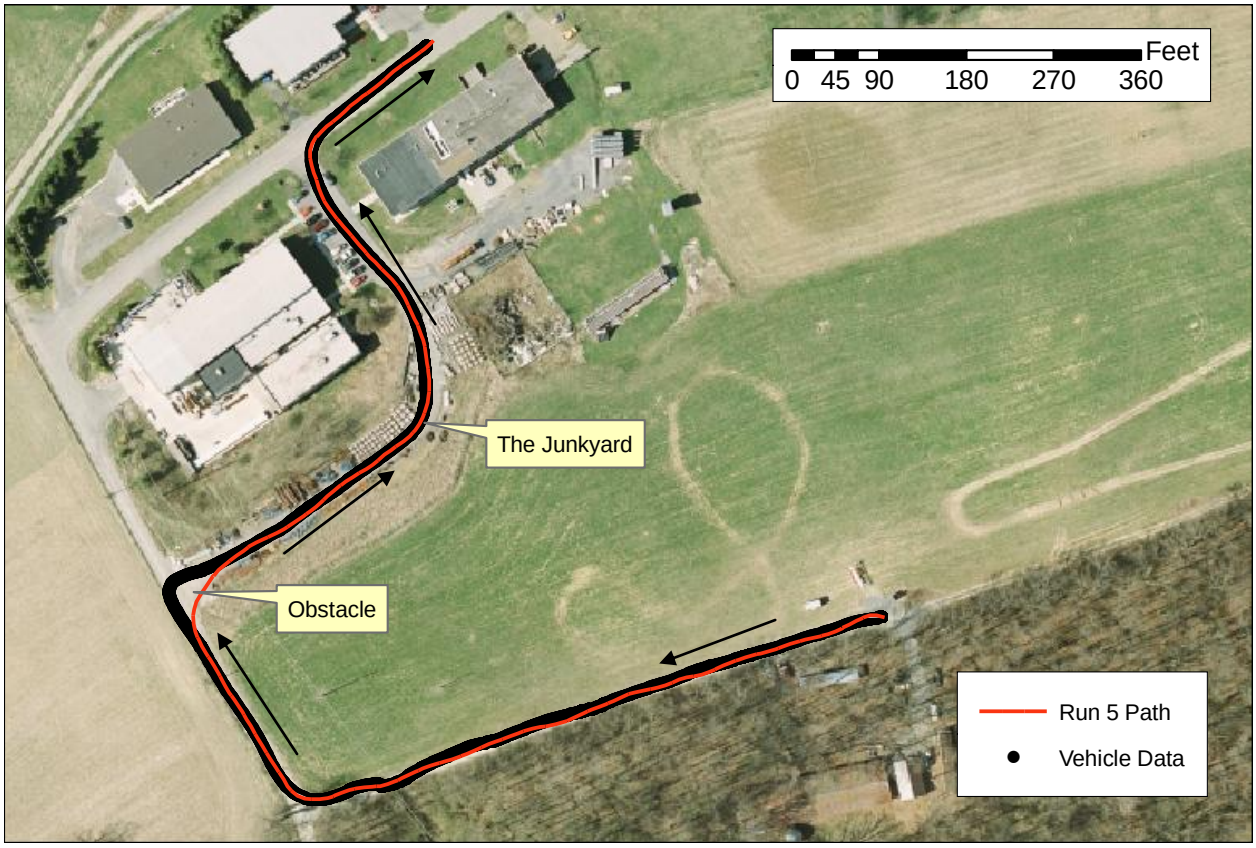


Figure 3.24: Map of Run 5

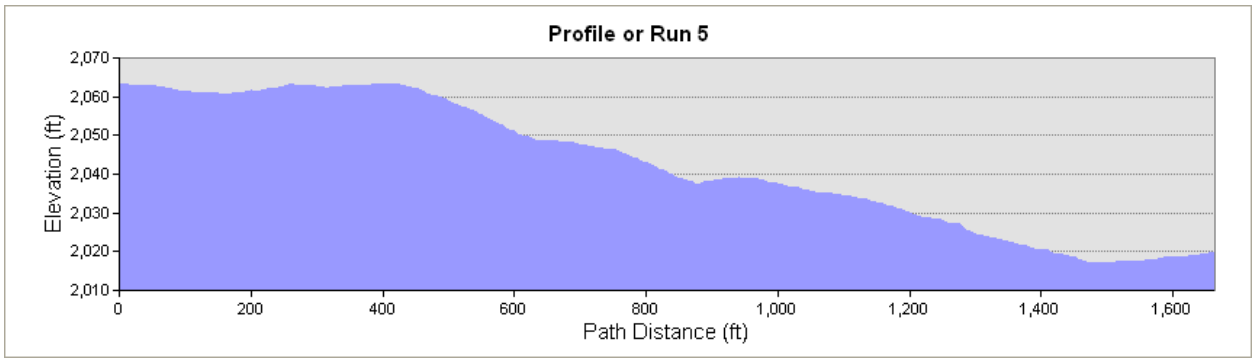


Figure 3.25: Profile of Run 5

The vehicle had to be taken out of autonomous mode three times during this run. The first time was due to the potholes in the road being identified as obstacles by the onboard obstacle avoidance system. While the path that was generated was valid, the vehicle refused to drive as it was stopped by the onboard obstacle avoidance system. The second time the vehicle was taken out of autonomous mode was because a pile of rubble which was not in the original data was directly in the middle of the planned path. Again the obstacle avoidance software kept the vehicle from moving, but this time the path itself was the issue. While the vehicle may have been able to navigate around the obstacle if the lane width had been wider, the width for this experiment intentionally set very small as a safety precaution to ensure that the vehicle stayed on some of the narrower roads in the test area. The pile of rubble can be seen in the right side of the screen in Figure 3.26. Finally the vehicle was taken out of autonomous mode a third time, when it failed to recognize an obstacle in 'the junkyard' that was slightly in the path, causing a collision danger. . Still the path that was planned appeared valid and most of the issues during this run were either caused by the age of the data or by the onboard obstacle avoidance systems.

Figure 3.26: Pile of rubble in path

Run 6

This run (Figure 3.27) was planned through a shaded and wooded area. This path was made specifically to gauge the impact of potential LIDAR gridding errors due to returns from overhanging branches and potential imagery misclassification due to shadows. The vehicle drove this path without incident and the path appeared to be valid for the entire length of the run.

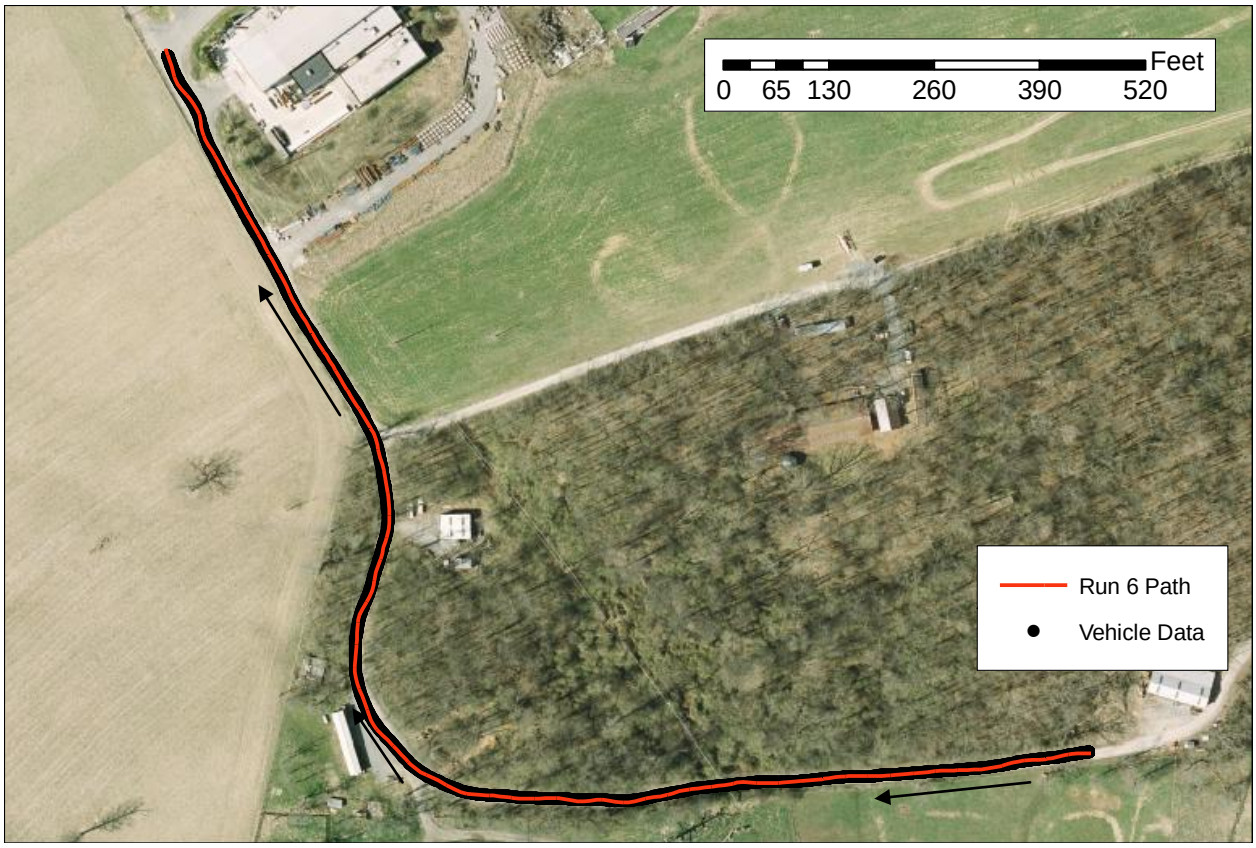


Figure 3.27: Map for Run 6

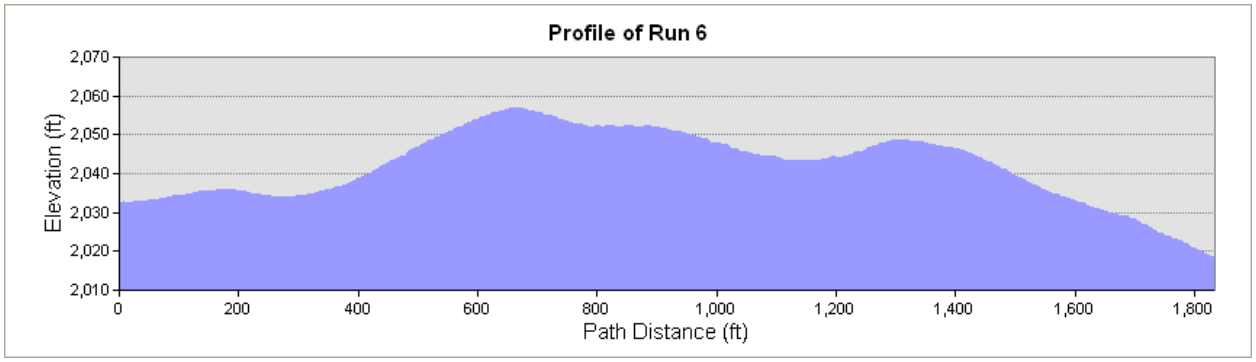


Figure 3.28: Profile of Run 6

Run 7

This run (Figure 3.29) was designed to confirm the influence of the road layer and to the ability to discern the shortest path from multiple possible road paths. There was one unexpected curve in this path where the vehicle swerved to avoid a pile of dirt that was in the datasets but not in the actual roadway. This swerve can be seen labeled as 'Feature 1' in Figure 3.29. In terms of elevation change, this path started at the top of a hill and descended

downward until reaching the main road at the bottom. The road on the far right of the figure had several potholes, and the vehicle tended to drift to the right side of the path, sometimes ending up driving through the grass on the edge of the road.

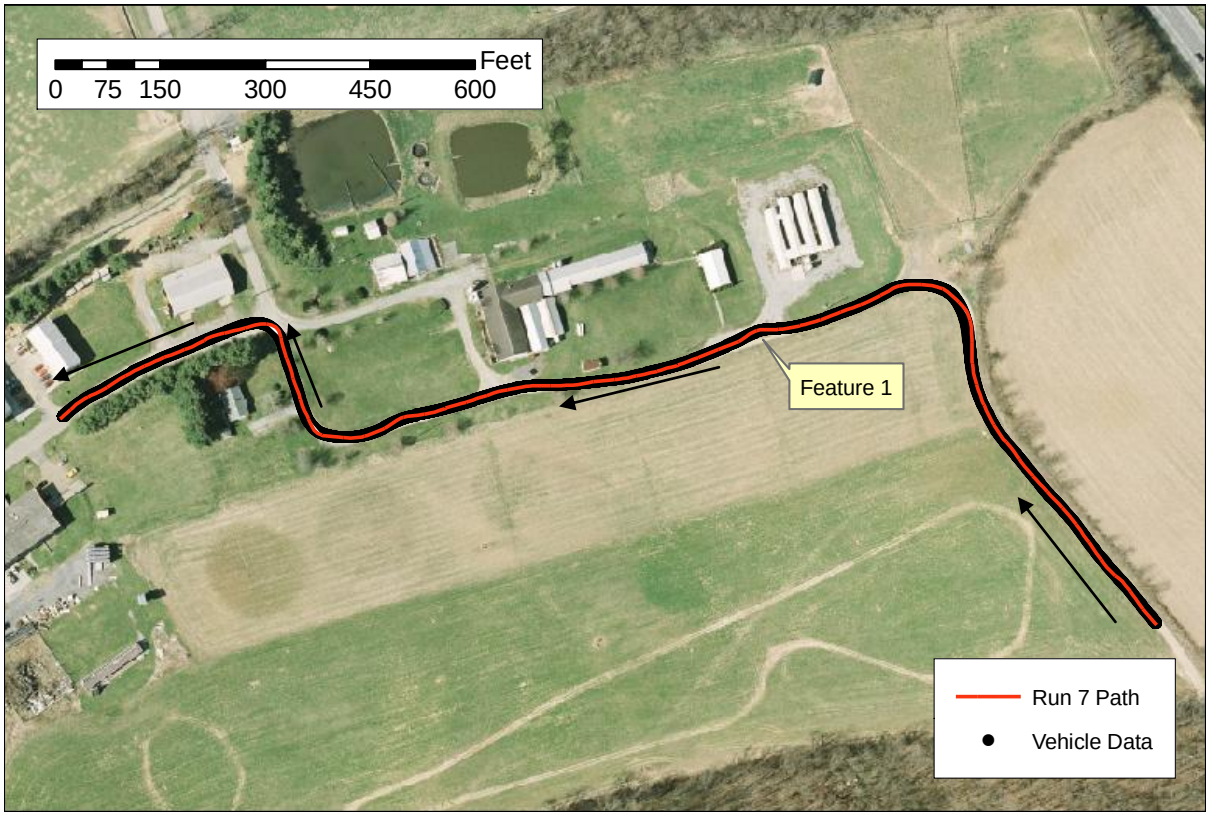


Figure 3.29: Map of Run 7

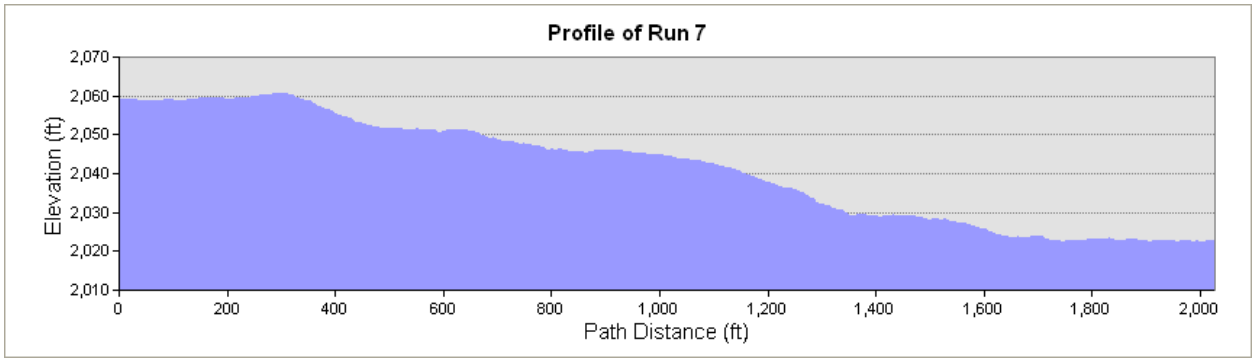


Figure 3.30: Profile of Run 7

Run 8

Figure 3.31: Map of Run 8

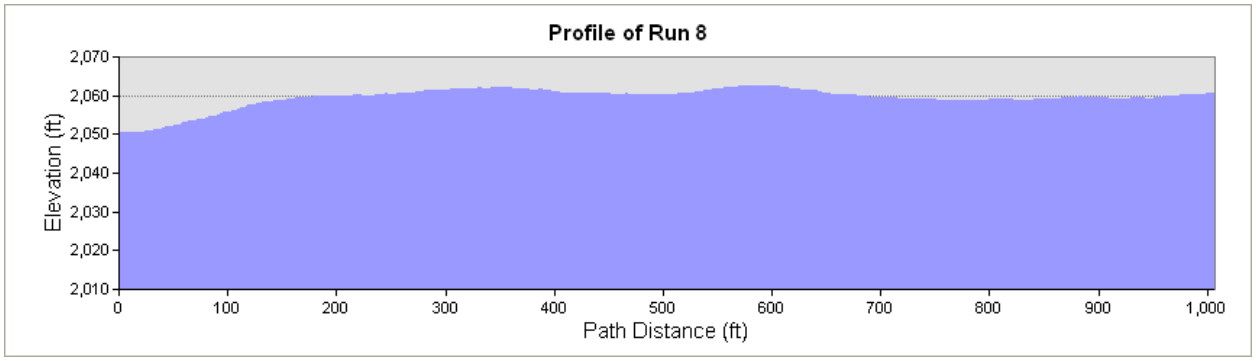


Figure 3.32: Profile for Run 8

This run was designed to test off-road driving. As stated earlier an artificial barrier had to be created for a fence that did not appear in the LIDAR data. Additionally, because of the high grass, the vehicle's obstacle avoidance system had to be disabled in order to drive off road. The vehicle also had to be taken out of autonomous mode when driving around the fence, because obstacle avoidance wasn't on to keep the vehicle away from the fence edge. Despite these drawbacks, the path chosen was reasonable, both on and off the road.

3.8.3 Data Analysis

Path Drivability

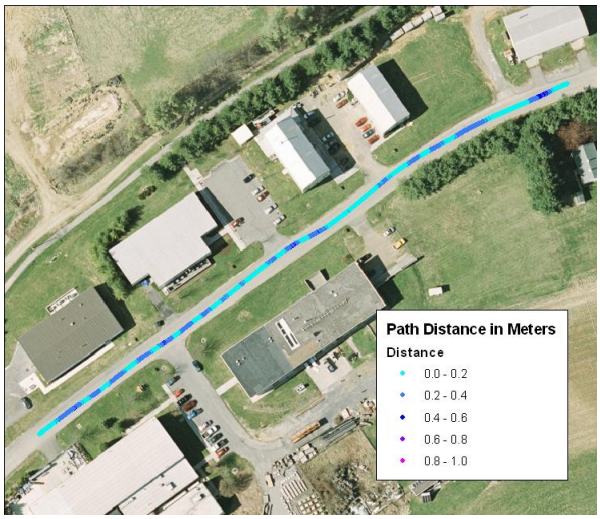


Figure 3.33: Path Distance for Run 1

Figure 3.34: Path Distance for Run 2

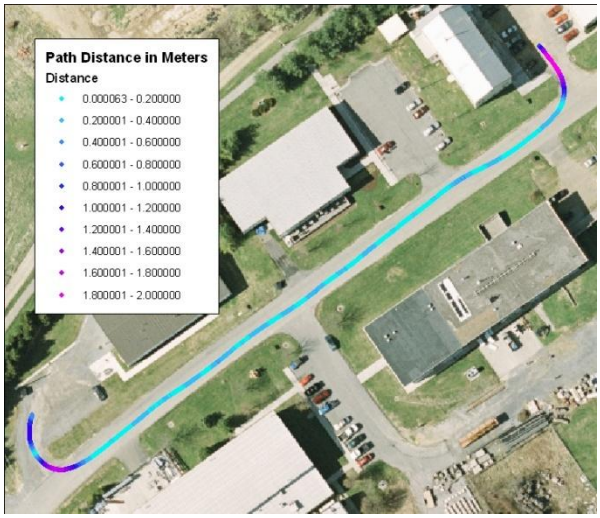


Figure 3.35: Path Distance for Run 3

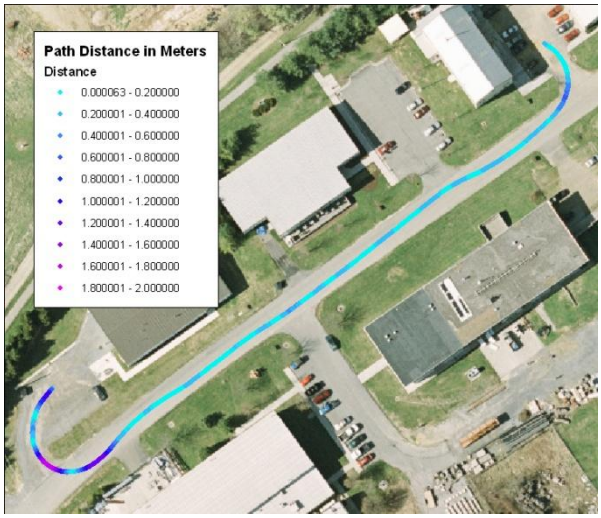


Figure 3.36: Path Distance for Run 4

A good objective measure of path drivability is distance between the driven paths and the planned paths. Figure 3.33 through Figure 3.40 represent the results from comparing the vehicle data during a particular run to the path planned for that run.

On average, the vehicle drove all of the paths within one or two meters. The exceptions came on sharp curves as seen in Runs three and four where the vehicle's own motion control determined a better path, and places where the vehicle had to be taken out of autonomous

mode due to unexpected or unforeseen obstacles such as the rubble blockage in run four and

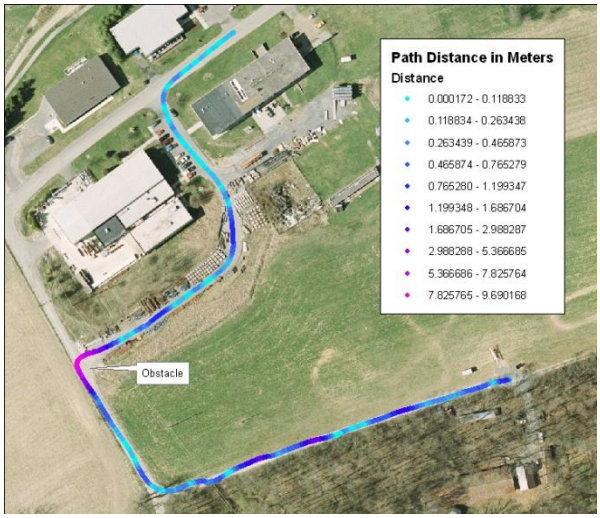


Figure 3.37: Path Distance for Run 5

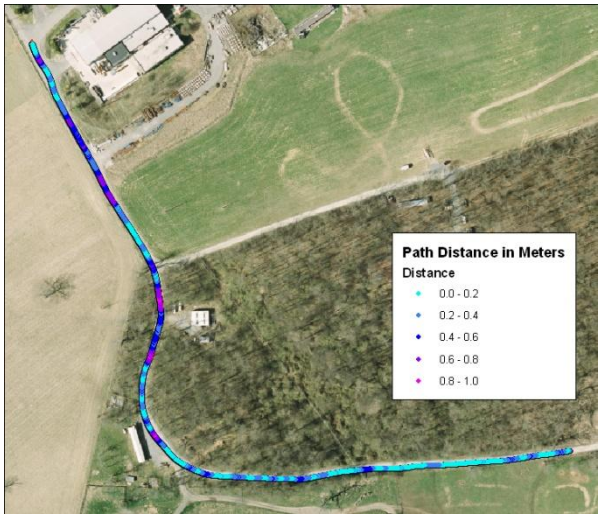


Figure 3.38: Path Distance for Run 6

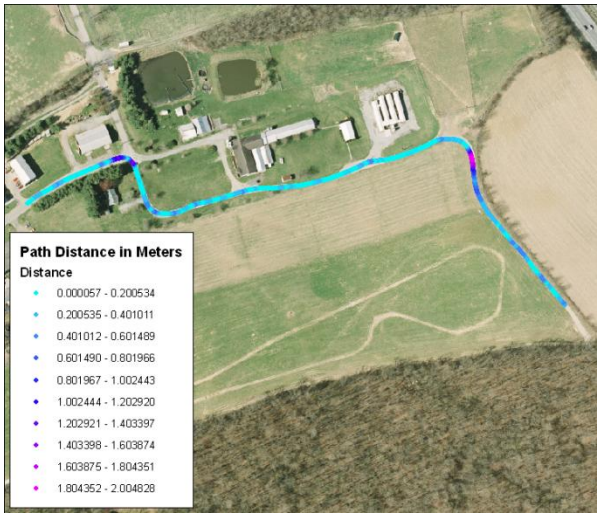


Figure 3.39: Path Distance for Run 7

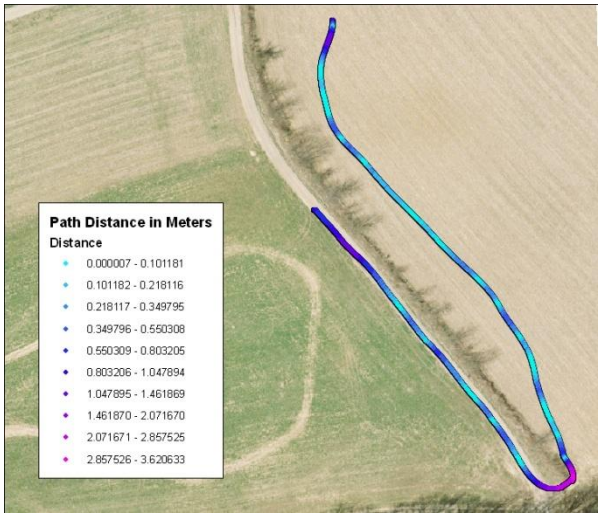


Figure 3.40: Path Distance for Run 8

the fence in run eight.

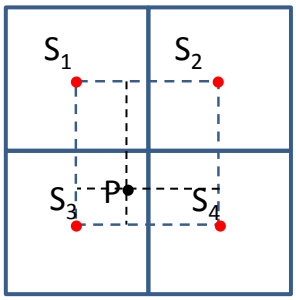
In general the presence or absence of very large path deviations (greater than two meters) can be viewed as a relative measure of path suitability. If a particular path was unsuitable, due either to un-drivable terrain or undetected obstacles, the path distance would be very high as a safety driver would be forced to drive around the bad area. Most of the paths have a maximum difference of two to three meters and a mean path distance of less than a meter. In fact, most of the path differences were likely a result of the sub-optimal kinematic characteristics of the generated path, rather than a result of the path going through an un-

drivable area. This hypothesis is supported by the observation that most of these errors occur around tight corners (especially in runs three and four). In this case the data supports the contention that the method described successfully planned suitable paths for the autonomous vehicle.

Slope Analysis

One method of measuring the accuracy of the LIDAR data in predicting travel slope is to compare the vehicle measured slope values against the predicted slope values based on the LIDAR data. In this analysis two different slope measurements were correlated. Specifically slope values were calculated using the onboard IMU measurement of pitch and roll and then compared to slope values interpolated the gridded LIDAR slope at the same GPS location as the IMU measurement. Runs six, seven, and eight were chosen because of the wide range of potential slopes available in these runs.

For each run used in the analysis, the pitch and roll data recorded from the combined IMU/GPS system was used to calculate a slope value for every sample point. Along with the pitch and roll data, the IMU/GPS system also generates a position solution. In post-processing that position is used to interpolate a slope value from the four surrounding cells in the gridded slope dataset. In this case, gridded slope values used were the same values that were used in generating the cost model as mentioned in Section 3.5.1. An illustration of the interpolation method is described in Figure 3.41.



Slope value at point P is assigned via a bilinear interpolation of slope values S_1, S_2, S_3, and S_4 at the center points of the slope grid.

Figure 3.41: Illustration of slope interpolation

While the slope measurements derived from the IMU were relatively smooth, the slope values interpolated from the gridded dataset had a significant amount of noise. Therefore a second analysis was performed where both the IMU and LIDAR derived slope values were averaged on one second intervals. These per-second averages helped to smooth the noise in the gridded dataset and reduce the effect of spatial autocorrelation on the dataset as a whole. Additionally

subsets of the data were chosen in areas that were free of overhanging branches and the slope values were compared in these small subsets. For each run and run subset a correlation coefficient was calculated between the IMU derived slope and LIDAR derived slope. A hypothesis test was then performed to calculate the significance of the derived correlation. Finally, the linear model was examined for any systematic irregularities.

Often, the results of statistical analysis on special data can be skewed by spatial autocorrelation. Spatial autocorrelation can inflate measures of statistical correlation and thus over-emphasize the correlation significance. In order to detect significant autocorrelation, a Durbin-Watson test for autocorrelation was performed on each pair that was being tested. If the autocorrelation was significant (within a 95% significance threshold) the Cochrane-Orcutt Procedure was performed on the dataset to correct the effects of autocorrelation on the resulting correlation tests. All results presented have either passed the Durbin-Watson test or have been corrected for autocorrelation using the Cochrane-Orcutt Procedure.

Slope Analysis Results

Run six was a long continuous run through a forested area. In such a situation, LIDAR returns from overhanging branches can affect the gridded Inverse Distance Weighted (IDW) surface, and by extension the resulting slope layer. For the entire run there was a small but significant correlation between the IMU measured slope values and the interpolated model slope values ($r = 0.0425948$, $p = 0.000334$). Additionally, there is also a significant correlation between the averaged slope values($r = 0.2044$, $p = 0.015417$).

The impact of the gaps in the LIDAR data can clearly be seen in the peaks in Figure 3.42. These peaks are caused by the IDW algorithm interpolating between sparse data points in the underling elevation layer, leading to a large slope nearer to the road. In each instance the cost surface algorithm planned a path around the high slope areas, but due to small errors in driving precision the actual path driven was through the high slope areas. If there had actually been an obstacle, the vehicles systems would have detected and avoided it.

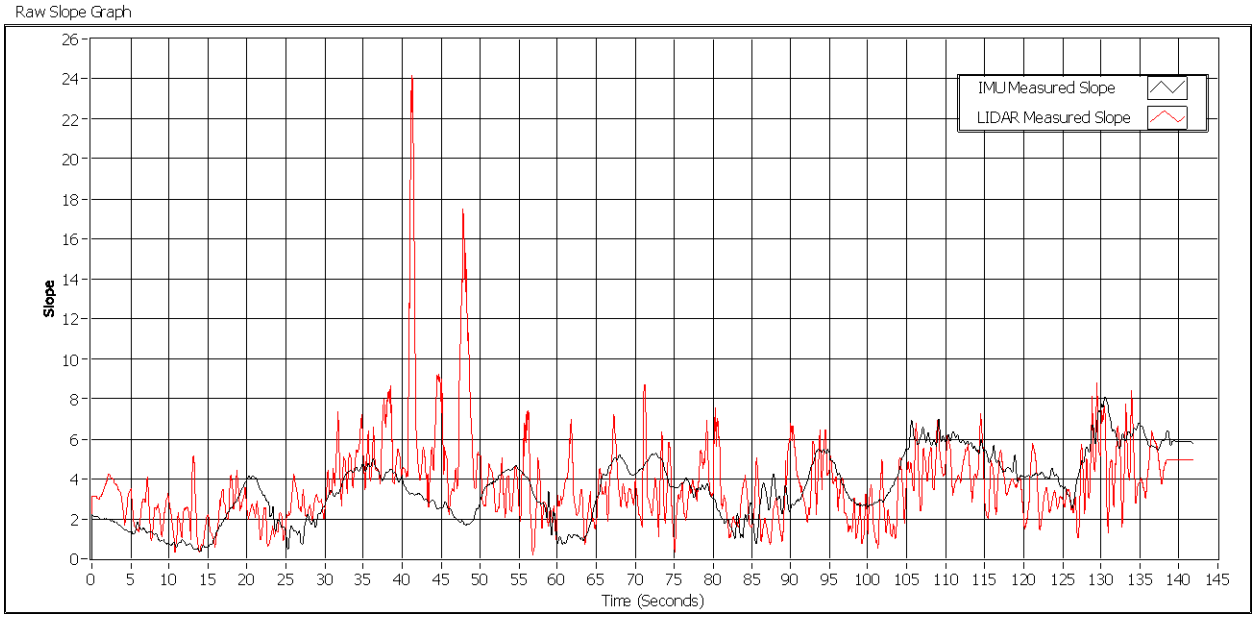


Figure 3.42: Slope plot for Run 6

The best fit line on run six with the average slope per second data slope is $y = 0.366012x + 2.434278$, far from the idealized $y = x$ that was expected.

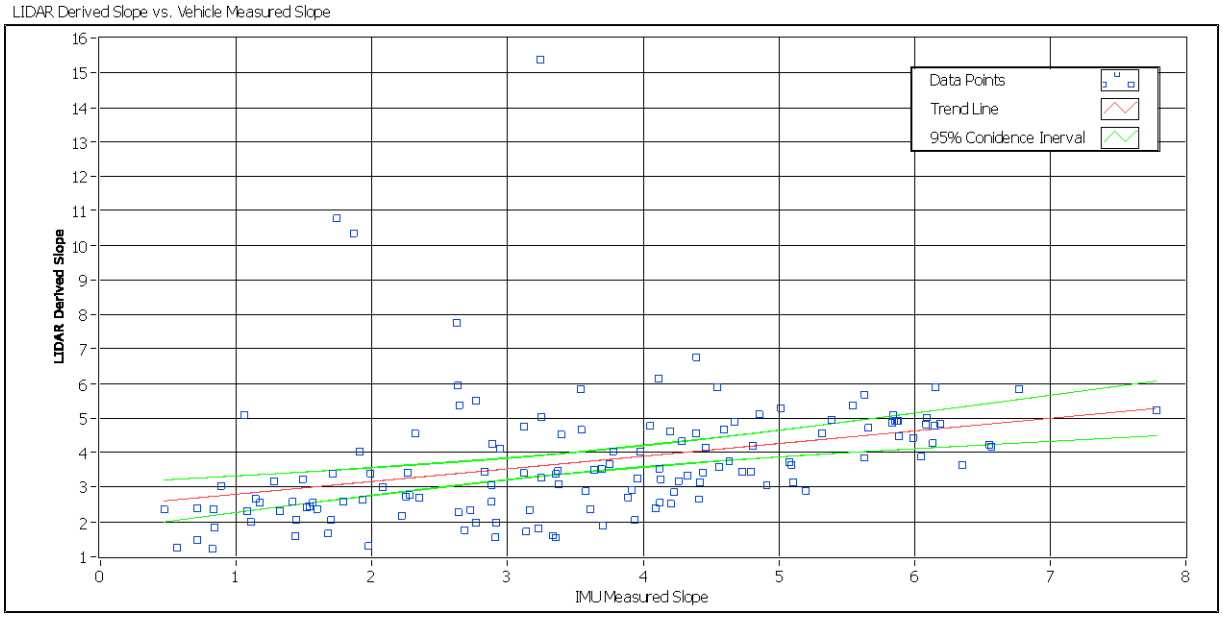


Figure 3.43: Scatter plot & trend line of average slope per second values of Run 6

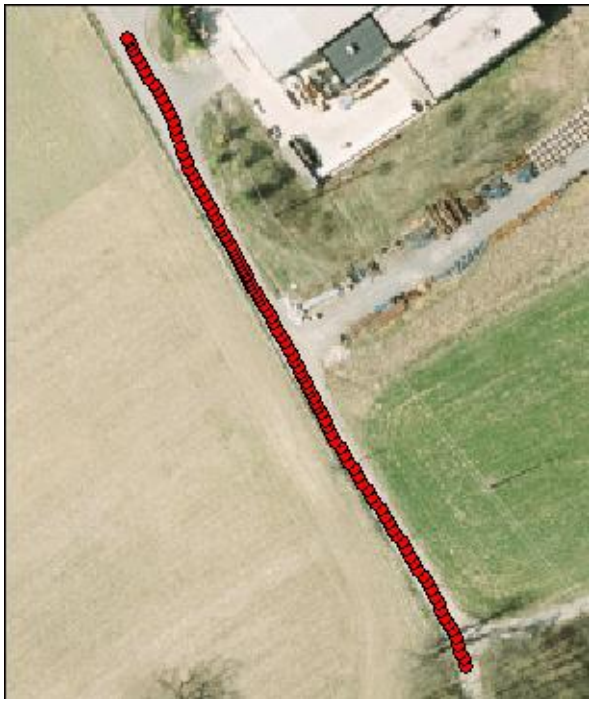


Figure 3.44: Run 6 subset map

Figure 3.45: Run 8 subset Map

Figure 3.46: Run 7 subset map

Using the subset of run six (mapped in Figure 3.44) a positive correlation was obtained for the raw data ($r = 0.0777729$, $p = 0.000364$). Similarly, a very significant correlation was obtained for

the average slope per second data (r = 0.842111, p = 2.767342E-12). The linear fit model yielded the equation $y = 0.748355x + 0.082577$, which was closer to the expected ideal.

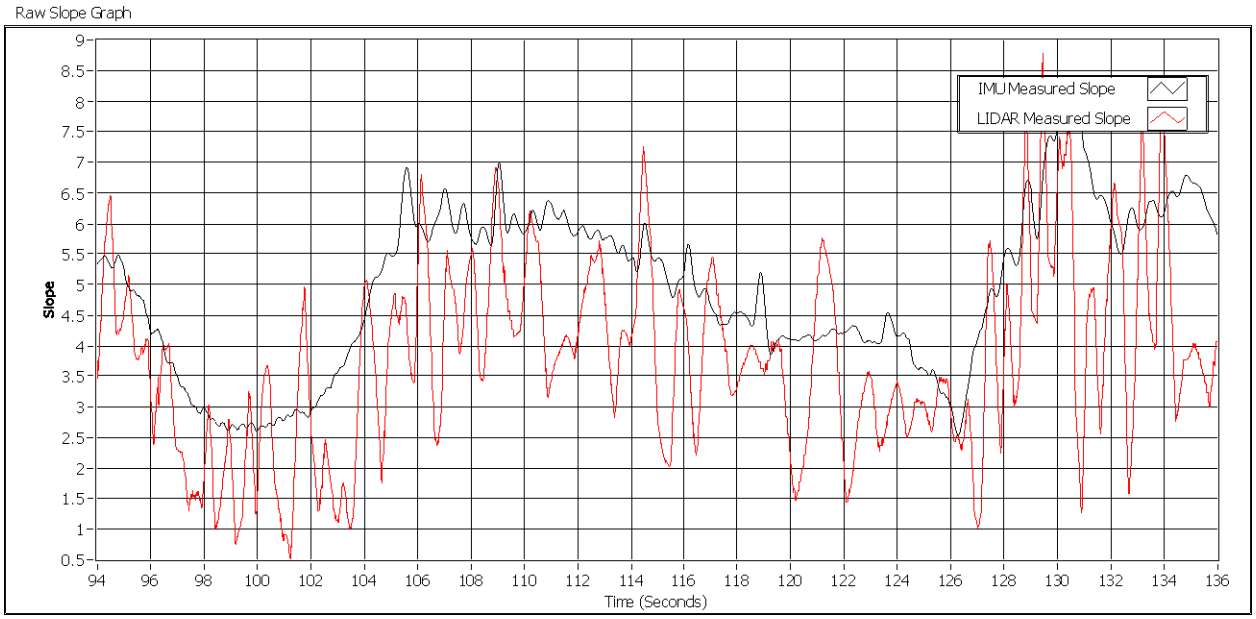


Figure 3.47: Slope plots for subset of Run 6

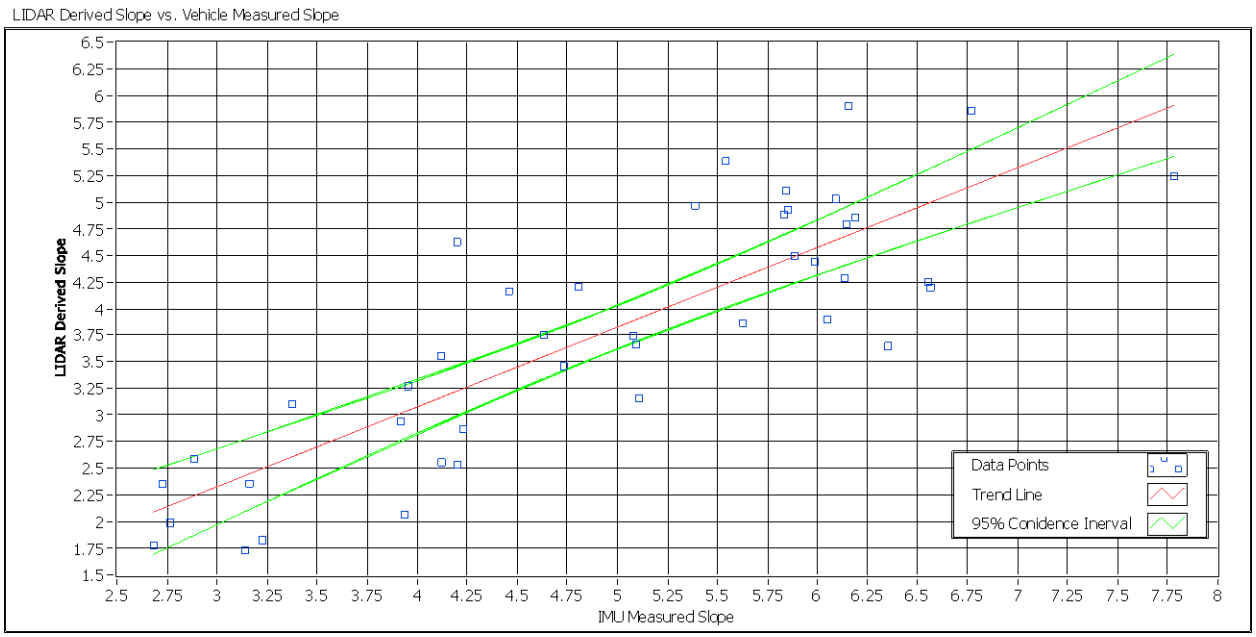


Figure 3.48: Scatter plot & trend line of averaged slope over a subset of Run 6

Run seven traveled along the edge of a field from the top of a hill to the bottom. For the entire run the correlation between the IMU measured slope values and the interpolated model slope values was not significant; however the correlation between the averaged slope values was significant (r = 0.456657, p = 3.873228E-10). There was also a significant correlation

between the averaged slope values in the chosen subset (r = 0.515473, p < 4.066194E-8). There were several instances during run seven when the vehicle was stationary while it attempted to plan a path around what appeared to be obstacles. These intervals were not included in the subset analysis.

Run eight was an off-road path and the vehicle had to be stopped in the middle of the run so as to not run into an obstacle. When serial autocorrelation is taken into account there was not a significant correlation between the raw slope values. However, there was a significant positive correlation (r = 0.217677, p <0.010049) between the averaged slope values. The correlation in this subset was also statistically significant (r= 0.715316, p < 4.865372E-8) and the final linear regression equation was $y = 0.907522x + 0.925727$.

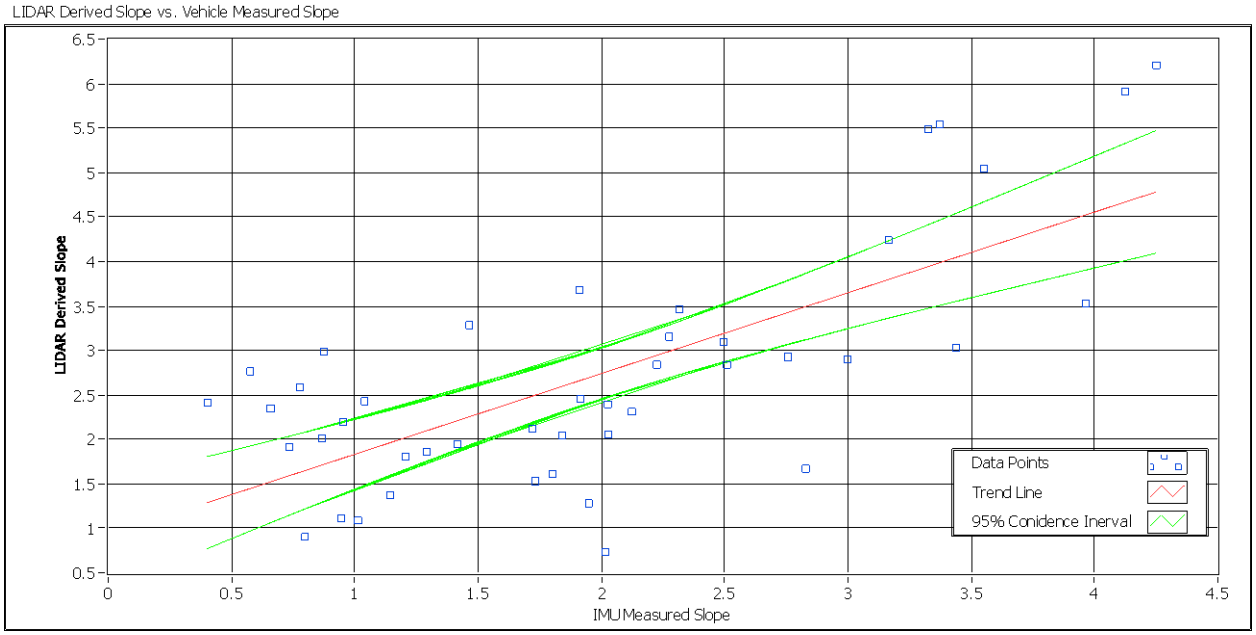


Figure 3.49: Scatter plot & trend line of averaged slope over a subset of Run 8

A summary of all of the correlation coefficients and the results of the correlation tests is presented in Table 3.2. In all cases there was a significant positive correlation between the IMU measured slope and the slope used in the model when the slope values were averaged over an interval of one second. Additionally, run six displayed a significant correlation between the unprocessed slope values over the entire data set and runs six and eight had significant correlation between the unprocessed slope values for the chosen subsets. While the linear best fit model did not approach the ideal value there was a significant positive correlation between the averaged data sets.

Table 3.2: Slope correlation significance tests

Dataset	r[1]	r^{sig}[2]	Reject Null Hypothesis [3]	p[4]
Run 6 (raw)	0.0425605	0.0234304	Yes	0.000369
Run 6 (average)	0.20192	0.166583	Yes	0.017139
Run 6 (subset)	0.839211	0.284519	Yes	< 1E-12
Run 7 (raw)	0.011584	0.0211478	No	0.283040
Run 7 (average)	0.456657	0.150575	Yes	3.873228E-10
Run 7 (subset)	0.515473	0.196551	Yes	4.066194E-8
Run 8 (raw)	0.00437429	0.0233537	No	0.713571
Run 8 (average)	0.217677	0.166583	Yes	0.010049
Run 8 (subset)	0.715316	0.297315	Yes	4.865372E-8

While there was never a time when the vehicle was in any danger due to excessive slope, the slope measurements did impact the path driven by the autonomous vehicle. In the off-road section, the slope measurements were the sole determining factor for the shape of the curve driven by the vehicle. Additionally, slope often determined the side of the road favored by the vehicle.

The IDW surface derived slope measurements displayed a significant amount of variance, likely caused by gaps in the LIDAR data. Gaps however, don't explain the large amount of noise present in areas where there was a significant density of LIDAR points. This noise is probably due to several factors including underlying data noise, varying elevations and elevation points within a grid cell, and interactions with the gridding algorithm.

3.9 Conclusion

[1] Pearson product-moment correlation coefficient
[2] 95% significance level correlation threshold
[3] Null hypothesis of $r = 0$
[4] Probability of wrongly rejecting the null hypothesis

In general this experiment achieved all of the main objectives. The algorithm described in this paper was able to produce reasonable paths for an autonomous vehicle and the vehicle was able to drive these paths. Still there are a few issues that need to be discussed.

One reason the vehicle could not drive part of two of the generated paths autonomously was deficiencies in the vehicle's own on-board obstacle detection. Most of these issues were related to the way the obstacle classification routines were designed. Odin was originally developed to run on a flat road in an urban environment so tall grass and large potholes were not considered "trafficable" by the vehicle. These problems are part of the autonomous vehicle platform and are therefore outside the scope of this research.

Another reason that the vehicle could not drive all the generated paths was because of a large obstacle (a pile of rubble) that filled the entire lane. Since the vehicle could not plan a path around the rubble, it was stuck and had to be restarted past the obstacle. Giving the vehicle the ability to dynamically insert obstacles to the GIS database and re-plan paths would get around this particular issue.

Finally, some obstacles such as the fence bordering the field in run eight weren't reflected in the cost map. This is evidence that research is needed in more advanced algorithms to identify potential obstacles and potential roads. More universal obstacle and road detection is also needed before these methods can be applied to more varied terrain or on a larger scale.

Despite these drawbacks, this research has shown that the concept of an autonomous vehicle driven solely by remote sensing data is possible. This experiment has demonstrated the feasibility of one such algorithm that accomplishes this task.

Bibliography

Bacha, Andrew, et al. "Odin: Team VictorTango's entry in the DARPA Urban Challenge." *Journal of Field Robotics* 25, no. 8 (August 2008): 467-492.

Bender, P. L., et al. "The Lunar Laser Ranging Experiment: Accurate ranges have given a large improvement in the lunar orbit and new selenophysical information." *Science* 182 (1973): 229-238.

Blackmore, B. S., S. Fountas, L. Tang, and H. Have. "Systems requirements for a small autonomous tractor." *Agricultural Engineering International: the CIGR Journal of Scientific Research and Development.*, 2004: 1-13.

Dijkstra, E.W. "A note on two problems in connexion with graphs." *Numerische Mathematik* 1 (1959): 269-271.

Dolgov, Dmitri, Sebastian Thrun, Michael Montemerlo, and James Diebel. "Practical Search Techniques in Path Planning for Autonomous Driving." *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08).* AAAI, 2008.

ESRI. "ArcGIS 9.2 Desktop Help." Software Manual, 2007.

Goncalves, G. "Analysis of interpolation errors in urban digital surface models created from Lidar data." *7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences.* 2006. 160-168.

Grass GIS. *Grass GIS Subversion Server.* November 1, 2007. http://svn.osgeo.org/grass/grass/trunk/vector/v.generalize/smoothing.c (accessed September 2008).

Hart, PE, NJ Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *Systems Science and Cybernetics, IEEE Transactions on* 4 (1968): 100-107.

Hopkins, Brian, and Robin Wilson. "The Truth about Konigsberg." *The College Mathematics Journal* 35, no. 3 (May 2004): 198-207.

Institute of Navigation. "Behind the Schenes at DARPA's Urban Challenge." *Institute of Navigation Newsletter*, Winter 2008: 9-11.

Kent, Daniel A. "Storing and Predicting Dynamic Attributes in a World Model Knowledge Store." PhD Thesis, Mechanical Engineering , University of Florida, 2007.

Latombe, J.C. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

LaValle, S.M. *Planning Algorithms.* Cambridge University Press, 2006.

Li, Qing, Nanning Zheng, and Hong Cheng. "Springrobot: A Prototype Autonomous Vehicle and Its Algorithms for Lane Detection." *IEEE Transaction on Intellegent Transportation Systems* 5, no. 4 (December 2004): 300-308.

Lupien, Anthony W., William H. Moreland, and Jack Dangermond. "Network Analysis in Geographic Information Systems." *Photogrammetric Engineering and Remote Sensing* 53 (1987): 1417-1421.

Meguro, J., et al. "Creating spatial temporal database by autonomous mobile surveillance systems." *Proceedings of the 2005 IEEE International Workshop on Safety, Security and Rescue Robotics.* 2005. 143-150.

Moore, E.F. "The shortest path through a maze." *Proceedings of an International Synposium on the Theory of Switching.* Harvard: Harvard University Press, 1959. 285-292.

Poudel, O.P. "Identification of barriers and least cost paths for autonomous vehicle navigation using airborne LIDAR data." Virginia Polytechnic Institute and State University, 2007.

Reinholtz, Charles. "DARPA Urban Challenge Technical Paper." Technical Report, Virginia Tech, 2007.

Srinivasan, T., J.B.S. Jonathan, and A. Chandrasekhar. "Mechanism for an Intelligent Neural network based Driving system." *E-Tech 2004*, 2004: 53-60.

Stahl, Chris. "Accumulated Surfaces & Least-Cost Paths: GIS Modeling for Autonomous Ground Vehicle (AGV) Navigation." Masters Thesis, Department of Geography, Virginia Tech, Blacksburg, VA, 2005.

Toth, Charles K., Eva Paska, Qi Chen, Yongjie Zhu, and Keith Redmill. "Mapping Support for the OSU DARPA Grand Challenge Vehicle." *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference.* Toronto, Canada: IEEE, 2006. 1580-1585.

Wehr, Aloysius, and Uwe Lohr. "Airborne laser scanning-an introduction and overview." *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (1999): 68-82.

Xiangjian, M., and L. Ganh. "Integrating GIS and GPS to realise autonomous navigation of farm machinery." *New Zealand Journal of Agricultural Research* 50 (2007): 807-812.

Zimmerman, Bruce B., Richard L. Rosenthal, and Jerry M. Breen. *Geographic Information Systems in Robotic Vehicles.* The Defense Technical Information Center Research Report, 1986.