Spring 2012

Vehicle path verification using wireless sensor networks

Gerry Wayne Howser

Department:

VEHICLE PATH VERIFICATION USING WIRELESS SENSOR NETWORKS

by

GERRY WAYNE HOWSER

A THESIS

Presented to the Faculty of the Graduate School of

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2012

Approved by

Dr. Sriram Chellappan, Advisor
Dr. Sanjay Madria
Dr. Zhaozheng Yin

ABSTRACT

Path Verification is a problem where a verifier would like to determine how closely a vehicle actually traversed a path that it claims to have traversed. This problem has critical significances in terms of vehicle mobility. Mobile nodes can be patrols officers or cab drivers, while respective verifiers can be police dispatchers or cab operators. In this paper, we design a sensor network assisted technique for vehicle path verification. In our design, a number of static wireless sensors placed in road segments will serve as witnesses and certify vehicles as they move. Post movement, these witness certificates will be utilized by the verifier to derive the actual path of a suspect vehicle. The challenge now is how to compare a Claimed Path as reported by the vehicle and the Actual Path derived from witness certificates. In this paper, we design a simple, yet effective technique for comparing similarity between two vehicle paths. Our technique extends from Continuous Dynamic Time Warping, which involves constructing a universal manifold from the two paths and then finding the geodesic on the resulting polygonal surface (shortest path along the surface) which is a diagonal from the origin of the surface to the terminal point. This distance is analogous to the Fréchet distance and yields a good measure of the similarity between two paths. Using simulations and real experiments, we demonstrate the performance of our technique from the perspective of detecting false paths claims from correct ones. We also design light-weight cryptographic techniques to prevent vehicle masquerading and certificate forging attacks.

A proof of concept experiment was conducted on the streets of Rolla, Missouri. A sensor grid was established on a small section of Rolla and a vehicle with a transmitter was driven through the grid many times. The analysis of the data yielded results consistent with the expected ones.

ACKNOWLEDGMENT

I would like to acknowledge the guidance of my advisor, Dr. Sriram Chellappan and the members of my advisory committee Dr. Sanjay Madria and Dr. Zhaozheng Yin. I am indebted to them for their help in bringing my work to a successful conclusion.

I would like to thank my instructors in Computer Science for their patience in dealing with someone with a strong industry background and a weak theoretical computation background. Their willingness to help me fill in the gaps in my knowledge has amazed me at times. I would especially like to thank Dr. Maggie Ching, Dr. Ralph Wilkerson, Dr. Ali Hurson, and Dr. Bruce McMillin.

I would also like to thank my instructors and mentors from my studies in Physics at the University of Missouri-Rolla for giving me a solid grounding in Physics and Mathematics. I would especially like to thank Dr. Jerry Peacher, Dr. Don Sparlin, Dr. William Snow, and Dr. Ralph Alexander.

I am especially indebted to my fellow students, Komal Patel Chintala and Bandar Kabous. It was our class project that lead to this work.

I would like to thank my family for all of the support over the years, especially my mother for convincing me very early on that nothing you learn is ever wasted. I thank my sons for supporting their father returning to school. I would like to thank my grandchildren, and those children that adopted me as their grandfather, for inspiring me to continue my studies.

Lastly, I would like to thank my friends who stood beside me in spirit when the work was daunting.

DEDICATION

I dedicate this work to the late Dr. Anthony Pineco who inspired in me a life-long love of classic mathematical problems and to my wife Patricia Berens for urging me to complete some unfinished business. I would never have done this without both of you.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

List of Algorithms

NOMENCLATURE

Symbol	Description
$\ominus$	Negative Minkowski sum
$\oplus$	Positive Minkowski sum
A_i	Sensing area of sensor i
al_i	Single location on the actual path
ap_i	Single location, time pair detected by the WSN
at_i	Single time on actual path
cl_i	Single location on the claimed path
cp_i	Single location, time pair claimed by the mobile
ct_i	Single time on claimed path
F_o	Sensor Field
R_i	Sensing range of sensor s_i
s_i	Sensor i
AP	Actual path
BS	Base station sensor node
CDTW	Continuous Dynamic Time Warping
CP	Claimed path
Geodesic	Shortest distance between two points on a surface
VANET	Vehicular Ad Hoc Network
WSN	Wireless Sensor Network

1. INTRODUCTION

Vehicular Networking is a topic that is receiving significant attention in the government, industry and academia. A number of organizations today are investing in Vehicular Ad Hoc Networks (VANETs) to leverage wireless networking support to improve state-of-the-art in road transportation. The US Federal Communications Commission (FCC) has allocated 75MHz of spectrum in the 5.9GHz band for Dedicated Short Range Communications, a set of protocols and standards for short to medium-range wireless communication for automotive use. Some recent VANET efforts are the USDOT's Vehicle Infrastructure Integration (VII), which is a cooperative initiative between USDOT and automobile manufacturers, focusing on feasibility of deploying communications systems for safety and efficiency of road transportation systems [28]. The ERTICO partnership is a multi-sector partnership pursuing development and deployment of Intelligent Transport Systems across Europe [26]. Apart from these efforts, a variety of VANET test-beds have been set up in academia like DRIVE-IN at Carnegie Mellon [24], CarTel at MIT [25], C-VET and CarTorrent at UCLA [7, 19], DOME testbed at UMASS-Amherst [27] among others. With the advent of vehicular networking today, a number of applications previously not possible are now becoming realities. Instances include content sharing among vehicles, real-time congestion detection, traffic re-routing to improve efficiency, emergency vehicle preemption etc.

1.1. PROBLEM ADDRESSED AND SIGNIFICANCE

In this thesis, we address a new problem in the realm of vehicular networking: *Given a path claimed to have been taken by a Vehicle V, how can a Verifier S determine whether or not Vehicle V actually took the path it claims to have taken.*

The problem addressed is practical and significant. According to Sergeant Letha Young at the Missouri S&T Police Department, a central problem for police vehicle dispatchers is to verify movements of patrol cars between the time they leave the precinct and when they return. Critical police services like quick response, patrolling high crime zones and operational efficiency are all related to path verification of patrol

cars. Path verification could prove useful as unbiased evidence of the exact location and path of an officer or a police vehicle. Apart from law enforcement, a variety of other commercial businesses like delivery companies, cab companies, trucking service operators also face the problem of verifying the paths of vehicles on roads mainly for operational efficiency reasons.

The most straightforward approach to detect false path claims is to have a mechanism to track each vehicle as it moves. For instance, a GPS receiver can be placed in each vehicle and it can report in real time the location of every vehicle. Unfortunately, this technique incurs significant extra cost per vehicle, and also significant communication overhead when location updates are sent frequently. Secondly, and more significantly, we are only interested in *verifying* whether or not a claimed path was actually taken. In other words, we would like to simply get a *Yes/ No* answer to whether or not a claimed path and an actual path are similar. Blindly tracking and reporting each and every location of the vehicle is clearly an overkill in such a scenario. Even if a number of actual location updates are provided, manually comparing them point by point is too time consuming and impractical. Also, a host of hardware attacks are possible with GPS receivers that can induce false location claims and are quite hard to detect [29, 31, 30].

Another technique that is commonly used in some police precincts is to perform image processing on cameras located in patrol cars, and use the time and date on these cameras to verify mobility claims. Naturally, this technique is also very time consuming, incurs significant overhead and also can be easily circumvented by modifying camera settings. Fundamentally the basic limitation in such techniques is that the mechanism to retrieve the actual path in fact comes from the suspect itself (i.e., the GPS Receiver or the Police Camera is always physically with the suspect). There is hence no independent authority that can validate claims. To the best of our knowledge, a practical, effective and efficient solution to the problem of verifying path claims of a vehicle is not yet there.

1.2. PROPOSED TECHNIQUE

In this thesis, we design a wireless sensor network assisted technique for vehicle path verification. In our technique, a number of static wireless sensor motes will

be deployed at selected positions in roads like traffic lights, road intersections and highway ramps. In fact, cameras deployed at many urban intersections today are themselves instances of static sensors. Vehicles moving in roads will periodically issue broadcast messages. Sensors within the communicating range of a vehicle will serve as witnesses for the vehicle, and provide it with a certificate authenticating the vehicle's time and location. Consider a Vehicle V that has traversed a path and reports a Claimed Path (CP) to the verifier. A Claimed Path is a set of $\langle location, time \rangle$ pairs, denoted as $\langle \{cl_1, cl_2, cl_3, \ldots, cl_n\}, \{ct_1, ct_2, ct_3, \ldots, ct_n\} \rangle$, where cl_i is the location claimed to have been visited by the vehicle at time ct_i. From the certificates provided by the witness sensors, the Verifier will determine an Actual Path (AP) for Vehicle V. An Actual Path is a set of $\langle location, time \rangle$ pairs, denoted as $\langle \{al_1, al_2, al_3, \ldots, al_m\}, \{at_1, at_2, at_3, \ldots, at_m\} \rangle$, where al_i is the location actually visited by the vehicle at time at_i.

In this thesis, we design an algorithm which when given a Claimed Path and an Actual Path, outputs the Fréchet distance between them, which in effect measures the similarity between the two paths. A high Fréchet distance indicates significant deviation between the paths, while a low distance indicates significant similarity. Our algorithm design is partly inspired by recent results in the domain of curve matching. In particular, our algorithm extends the Continuous Dynamic Time Warping (CDTW) technique in [13] for matching two curves. The CTDW technique works by constructing a universal manifold from two given curves and then finding the geodesic on the resulting polygonal surface (shortest path along the surface) which is a diagonal from the origin of the surface to the terminal point. While the basic CDTW technique compares two continuous curves, we propose extensions to the basic technique in order to compare two vehicular paths to determine an estimate of the Fréchet distance between them. We also design simple and light-weight cryptographic techniques that can defend against attacks that spoof vehicles and the forgery of witness certificates.

The proposed scheme is evaluated using extensive simulations, and using real experiments conducted with a network of TelosB sensor motes (see Figure 4.6) deployed within a small segment in the city of Rolla. Our data demonstrates that the proposed technique is practical, deployable in urban traffic networks, and

the Fréchet distance computed from our technique is highly sensitive to deviations between claimed and actual paths. We point out that the techniques proposed in this thesis are not restricted to vehicular applications. They also have applicability in environments like battlefields that are deployed with static wireless sensor networks where actors in the network are mobile and assist static sensors while performing the mission. In such cases, verifying the mobility of actors may be important, and our techniques can directly apply to verification of actor mobility in such wireless sensor and actor networks also.

2. RELATED WORK

Clearly one of the main difficulties when deciding how closely an agent followed a given path, or how well the agent reported the actual path is defining what is meant by "close". Many different metrics were considered, many of which were useful to a certain degree. An additional difficulty was the computational complexity of any algorithm that determines the similarity between two paths composed of polygonal chains.

From a mobility perspective, much of the current work in Wireless Sensor Networks (WSN) is focused on the detection of intruders [1, 4, 14]. If the mobile node is a vehicle, verification of the location and path of the node becomes of interest in our communities. Consider a police officer driving a patrol car on a beat. In this situation we propose a very simple question: how can the police dispatcher or police captain know if the officer actually patrolled where he or she claims to have patrolled and how can this be verified? More generally, how can we know if a vehicle has closely followed the assigned path?

In tracking, the goal of the sensor network is to pro-actively detect where the intruder is likely to be in space and time. The problem we are addressing is reactive in the sense that we are interested in whether or not a claimed path was taken by the suspect vehicle by examining the data post movement. To the best of our knowledge, such a problem is unique and not yet addressed. Within the realm of WSNs, there have been a number of recent efforts arguing for their deployment to assist in the solution of transportation engineering problems. In [22], wireless sensor networks have been used for vehicle theft detection. The core idea is to allow the sensors in the vehicles that are parked within the same parking area first form a sensor network, then monitor and identify possible vehicle thefts by detecting unauthorized vehicle movement. In [21], the feasibility of WSNs being used for a number of transportation applications like road safety, traffic control, intelligent traffic management systems are discussed along with the ensuing challenges from the perspective of data aggregation and information processing. In [11], wireless sensor networks of multiple modalities have been used to address the vehicle classification problem, which has a number of

applications in civilian and military scenarios.

Our research in this thesis is partly inspired by the documented successes of the research discussed above in using wireless sensor network to address problems in transportation engineering. With the rich advances in wireless sensing, processing and communication technologies, coupled with advanced protocols to optimize their performance and low cost, we envisage WSNs to be widely deployed in emerging traffic infrastructures. As pointed out earlier, with the ubiquitous deployment of cameras on roads and traffic lights that can communicate, such a vision is not too far off. We propose a scheme to use an existing sensor network to verify the claimed path of a moving node. The core ideas of deploying a wireless sensor network are widely accepted. They are easy to deploy, well understood, quickly form ***ad hoc*** networks, perform robust routing, and are adaptable to a number of tasks including tracking, monitoring, surveillance, and other related tasks. Using WSNs for multiple tasks is already well understood. Our solution does not present a significant additional load on a WSN deployed for other tasks.

In the presence of an existing WSN, there is one straight forward solution. We allow the WSN to track the vehicle as it traverses the network. The WSN then reports back to a Base Station (BS) which sensors detected the vehicle. This solution presents challenges. First, we do not need to suspect and track all vehicles as typically only a few are untrusted. Tracking multiple trusted vehicles and reporting back to the BS consumes large amounts of energy and exposes the location of the sensors unnecessarily. Second, predicting the movements of the vehicle and triangulation in a sparse WSN is difficult and unnecessary when we are mainly interested in verifying a previous movement. Our chief interest is in completed movements, not current locations. Third, it is unlikely that any vehicle will visit the desired points exactly or leave the established roadway.

Many different measurements have been defined for 'closeness' [13, 5] such as the Hausdorff distance [2, 6, 16]. For our purposes, the most useful measure is the Fréchet distance.

2.1. FRÉCHET DISTANCE

The Fréchet distance comes from the classical problem of walking a dog through

a park(see Figure 2.1). The man visits the same points in the park but the dog will typically range from side to side closely following the man's path but not exactly. Does the dog follow the man's path closely each day or does it range further on some days than on others? How can we measure that 'closeness'? An intuitive way to understand the meaning of the Fréchet distance is to consider two sets of linked line segments, α and β. If the man walks on curve α and the dog walks on curve β, we can define the Fréchet distance as the maximum length of a leash between the man and dog. Formally, let $d(p, q)$ denote the Euclidean distance between two points p and q on a plane. The Fréchet distance between α and β at points p and q is given by [12]:

$$F(\alpha, \beta) = min_{f:[0,1]\to\alpha, g:[0,1]\to\beta} \left[max \left[d(f(t), g(t)) \right] \right]. \tag{2.1}$$

where f and g are continuous non-decreasing functions defining the positions of the man and the dog on the curves α and β at every instant t. If the dog is on an elastic leash, the length of the leash relates very closely to how similar the two paths are at any given point. This length, the Fréchet distance, is a way to measure the matching between two curves or even two surfaces [5]. Matching curves can be done in polynomial time with reasonable results. If one curve has n samples and the other has m samples, the match can be done in $O(mn \log mn)$ time [13].

The difficulties lie mainly in two areas; defining good measures for the results, like many sensor problems, is application dependent. Acceptably close on a battlefield may not be the same as acceptably close for a police car patrolling a highway.

The other area of difficulty is more problematic. Matching two curves and finding the distance between them is sensitive to the sampling technique employed. The techniques developed depend upon translational invariance and temporal invariance. However, two paths are not the same if one is translated a distance from the other. We must also determine if the untrusted agent followed the path in the correct sequence of points. These are two things which traditional curve matching usually ignores. We must solve the difficult problem of forcing translational

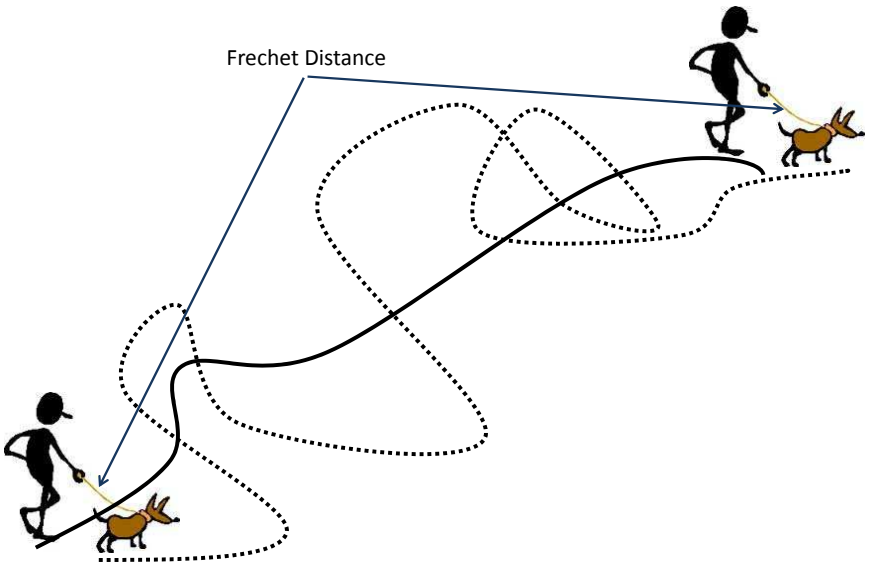

Figure 2.1. The Fréchet distance

variance, ordering, and time sequencing upon curve matching.

2.2. CDTW

This thesis will use the Continuous Dynamic Time Warping(CDTW) technique of Efrat *et al* [13]. The strengths and weaknesses of this technique will be used to compute measures which will show how closely the untrusted agent follows the directed path. The concept of forcing the agent path to start and end at the appropriate way points, what we call a 'force', is introduced to use the translational invariance of CDTW to reveal the underlying nature of the agent's deviations from the assigned path. Two typical types of variance from the path of way points are studied. A shift where the agent shadows the way points exactly but at a safe distance and a spike where the agent follows the way points but quickly leaves and returns to visit an unauthorized point for other purposes.

3. METHODOLOGY

In this section we present our method to produce secure witness location certificates. We show a method to use these certificates to generate the actual path for comparision to the claimed path. We then present an algorithm to generate the CDTW version of the Fréchet distance.

3.1. SYSTEM FRAMEWORK

We now present our sensor network based framework for vehicle path verification. In our system, a number of wireless sensor motes will be placed on selected locations in roads. Likely locations can include road intersections, traffic signals, highway ramps etc. Such sensors can include commercial off the shelf motes like MicaZ and TelosB motes that have demonstrated communication ranges in excess of 100 ft, are cheap, tiny and lend themselves for quick deployment. The locations of sensors are fixed, and energy issues are not considered since they can easily be powered when deployed statically in roads. All sensors are assumed to be trusted, and there is some loose time synchronization mechanism in place among all sensors in the network.

In our system, each vehicle to be verified will be equipped with a wireless transmitter, that will broadcast a message periodically as it moves. The transmitter in each vehicle will communicate with sensors deployed on roads using an authentication protocol. Sensors receiving the broadcast will validate locations of the vehicle as its moves, and giving each vehicle an unforgeable witness certificate. Post completion of the path, each vehicle will forward its claimed path and witness certificates from all sensors to the verifier, from which the actual path is derived. The verifier will then compute the Fréchet Distance between the paths. If the computed Fréchet Distance is small, the vehicle's claims are accepted, or rejected otherwise. In the following, we illustrate our technique via two protocols: Authentication Protocol to Generate Witness Certificates and Actual Path; and Protocol to Compute the Fréchet Distance between two paths.

3.2. AUTHENTICATION PROTOCOL TO GENERATE WITNESS CERTIFICATES AND ACTUAL PATH

Protocol 1 presents our authentication protocol. Just before a Vehicle V starts its path, the Verifier S will assign a unique random seed R_V which is secret to V. As it moves, V will periodically broadcast Message $M = \{V, t_V, \mathcal{H}(V, R_V, t_V)\}$, where $\mathcal{H}$ is a Hash Function known to all parties. Denoting t_V and t_u are the current times in Vehicle V and Sensor u respectively, every Sensor u that receives M will first check if the difference between t_V in M and t_u in u is less than a system configured ϵ. If not, Sensor u will simply discard M. Otherwise, sensor u will reply back with an authenticated witness certificate $N_u = \{u, t_V, t_u, \mathcal{H}(u, t_u, \mathcal{H}(V, R_V, t_V), E_u)\}$, where E_u is a key secret to Sensor u. The witness certificate is stored by V, and the process repeats for every sensor in the path of V. At the conclusion of the path, V will forward its Claimed Path and all witness certificates to the verifier upon request.

The Verifier S will receive the Claimed Path, CP, and witness certificates from V, where:

$$CP = \langle \{cl_1, cl_2, cl_3, \ldots, cl_n\}, \{ct_1, ct_2, ct_3, \ldots, ct_n\} \rangle.$$

Consider a certificate from Sensor u, which is

$$N_u = \{u, t_V, t_u, \mathcal{H}(u, t_u, \mathcal{H}(V, R_V, t_V), E_u)\}.$$

The verifier will extract the t_V and t_u from N_u, and using, R_V and E_u (known to the verifier), it will independently compute $\mathcal{H}(u, t_u, \mathcal{H}(V, R_V, t_V), E_u)$. If this does not match what was found in N_u, the verifier rejects the certificate and reports V as "Failing Path Verification". Otherwise, the location of sensor u and t_u are added to the Actual Path of V. The process repeats for all certificates and the Actual Path is generated. The Claimed Path and Actual Paths are now ready for comparison.

Discussions: The protocol prevents masquerading of Vehicle V. When a Vehicle $\bar{V}$ attempts to masquerade as V, it will not have the correct Seed of V. When the Verifier gets the Witness Certificate from $\bar{V}$, Step 24 of Protocol 1 will Fail, and

Protocol 1 Protocol to Authenticate Certificates and Actual Path

1: **Verifier**: S; **Vehicle**: V; **Unique Seed**: R_V; **Hash Function**: $\mathcal{H}$
2: t_V = current_time in Vehicle V
3: t_u = current_time in Sensor u
4: **Pre-configured acceptable time difference:** ϵ
5: **for** Every Location in Path of V **do**
6: $V \rightarrow *$: M
7: $M = \{V, t_V, \mathcal{H}(V, R_V, t_V)\}$
8: **for** Every Sensor u Receiving M **do**
9: **if** $|t_V - t_u| < \epsilon$ **then**
10: $u \rightarrow V$: N_u
11: $N_u = \{u, t_V, t_u, \mathcal{H}(u, t_u, \mathcal{H}(V, R_V, t_V), E_u)\}$
12: **else**
13: Discard M
14: **end if**
15: **end for**
16: Store N_u for all sensors
17: **end for**
18: $V \rightarrow S$: Set of Witness Certificates for all Sensors
19: $V \rightarrow S$: CP $\langle \{cl_1, cl_2, cl_3, \ldots, cl_n\}, \{ct_1, ct_2, ct_3, \ldots, ct_n\} \rangle$
20: Following Steps Executed by S
21: **for** Each Entry in Witness Certificate **do**
22: Extract $X = \mathcal{H}(u, t_u, \mathcal{H}(V, R_V, t_V), E_u)$ from N_u
23: Compute $Y = \mathcal{H}(u, t_u, \mathcal{H}(V, R_V, t_V), E_u)$
24: **if** $X = Y$ **then**
25: Add Location of u and t_u in N_u to Actual Path of V
26: **else**
27: Report V as "Failing Path Verification"
28: **end if**
29: **end for**

the masquerading is detected. A Vehicle $\bar{V}$ also cannot create or forge certificates for a Sensor u. If $\bar{V}$ attempts to modify the time or the id of the sensor generating a witness message, again Step 24 will detect this and hence reject $\bar{V}$. Note that if Vehicle $\bar{V}$ incorrectly reports its time during its broadcast to the sensor network, the check in Step 9 will fail and $\bar{V}$ will not get any valid certificate. Note that it may happen that Vehicle $\bar{V}$ simply does not broadcast its location for a long time. The verifier can easily detect this behavior by seeing significant time gaps in witness certificates and can hence reject such claims easily, provided the density of sensors in

the network is sufficient.

At the conclusion of Protocol 1, the verifier will either have a valid Actual Path for Vehicle V for similarity comparison with the Claimed Path, or it will reject the Claim of V. Since authentication of the temporal (i.e., time) aspect of a Vehicle's path is already accomplished in Protocol 1 (via Steps 9 and 24), we ignore the temporal aspect of Claimed and Actual Paths for following discussions. In other words, the Claimed Path and Actual Path for similarity comparison from here will only be denoted by $\langle\{cl_1, cl_2, cl_3, \ldots, cl_n\}\rangle$, $\langle\{al_1, al_2, al_3, \ldots, al_m\}\rangle$ respectively

3.3. PROTOCOL TO COMPUTE FRÉCHET DISTANCE BETWEEN TWO PATHS

We now discuss our technique to compare two vehicular paths and determine the Fréchet distance between them. First, we present some challenges in the problem, followed by a technique based on Continuous Dynamic Time Warping to resolve the challenges. The detailed protocol and modifications to the Continuous Dynamic Time Warping technique are presented subsequently.

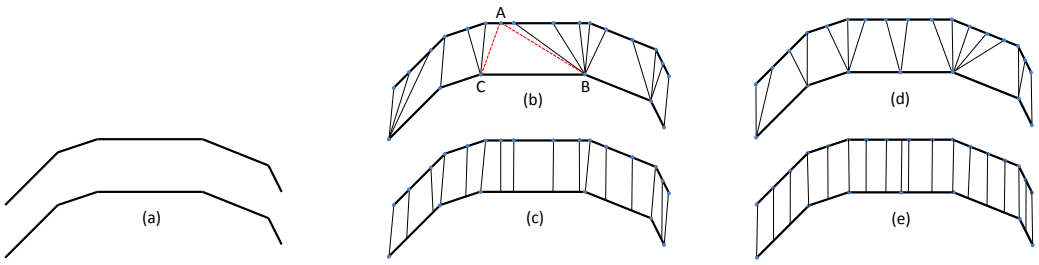

Figure 3.1. Two path segments (a). Samplings in (b) and (d) are bad. Samplings in (c) and (d) are good.

Fréchet distance and Sampling Issues: One of the classical problems of mathematics is that of defining how closely two curves match. Clearly, our problem of determining how closely a claimed path is truly followed can be analyzed in this way. However, the challenge here is how to compare two curves, and measure the distance between them? A metric for this is called the Fréchet distance, and is the

maximum distance between two curves. However, measuring the distance between two points is trivial, but the same is not true for two curves [2] [3]. Note that in our problem, the two curves (or rather vehicular paths) we are comparing are not continuous, and the sampling rate between the Claimed and Actual Paths are clearly not the same. Very likely, the sampling rate of the Claimed Paths (say from a GPS device in the vehicle) is much more that the sampling rate of Actual Paths (which are derived from witness sensors).

Unfortunately, the maximum distance between two curves is wildly sensitive to the sampling method used. Figure 3.1 illustrates the problem where there are two curves in Figure 3.1 (a) that we would like to compare, and various samplings of the two curves are shown in Figure 3.1 (b) to Figure 3.1 (d). As we can see, there are any number of ways to match the sampled points on two curves. Even if the known number of points is the same for both curves, there is no guarantee that matching a point $p(i)$ on one curve with a point $q(i)$ on the other curve (for the same i) will give correct results. Furthermore, even after such comparisons verifying the result in the presence of sampling errors is difficult. Taking the mean of all the distances between matching points looks promising. However, if the same number of points are placed on both curves, but at different distances along the curve, the results will still be wrong. In our problem we have no control over the data points on the claimed or the actual path in either number or placement. If the sampling is done wrong, virtually any large value is possible for the Fréchet distance, and the resulting verification check will be rendered useless. Controlling the sampling rate is hence critical to derive right measures of the distance between two curves.

Continuous Dynamic Time Warping: Continuous Dynamic Time Warping (CDTW) is a technique that resolves the sampling challenge [13]. The basic CDTW consists of constructing a universal manifold from the two paths and then finding the geodesic on the resulting polygonal surface (shortest path along the surface) which is a diagonal from the origin of the surface to the terminal point. This distance is analogous to the Fréchet distance and is a very good measure of the similarity between the two curves.

If we construct a virtual manifold, we can use this manifold to control the sampling of the two paths. Once we have used the segments of the two paths to form

polygonal patches to make up the manifold, we can place additional points, called Steiner points, along the edges to control the curve sampling for us. The accuracy of CDTW can be controlled by the placement of these Steiner points. The more Steiner points the more accurately we can calculate the geodesic and the closer that value will be to the Fréchet distance. Steiner points are placed to provide more granularity to the geodesic and to force it to more closely approach the exact geodesic. This allows us to sample the curves as finely as required and match the two samples intelligently as in Figure 3.1, (c) and (e). Without this method, we could sample the curves as in Figure 3.1 (b) or (d). In the case of Figure 3.1 (b) or (d), it is easy to see that we could get a very large and erroneous distance such as distance of segment AB rather than a more reasonable one such as distance of segment AC. Construction of the manifold is possible in linear time but finding the geodesic along a set of polygonal patches or plates is computationally difficult. Note here that since, we are extracting positions from witness sensors in proximity of a vehicle, there is clearly an inherent error in position estimation. In our technique we aim to compute approximate values for the Fréchet Distance hence significantly saving computational complexity. So in our technique, while a Fréchet Distance of zero is unlikely when two paths are practically the same, a reasonable small value is accepted as both two paths being actually the same.

Constructing the Manifold: A manifold is constructed deriving from the technique as in [13] using the set of way points as one polygonal chain and the agent locations as determined from the sensor network for the other chain. Consider two points, cp_i and ap_j, from which we construct a patch, P_{ij}, by the Minkowski negative sum of $CP \ominus AP$, as in Figure 3.2. We define the standard Minkowski Sum, $\oplus$, and negative Minkowski Sum, $\ominus$, as

$$CP \oplus AP = \{cp_i + ap_j | cp_i \in CP, ap_j \in AP\} \tag{3.1}$$

$$CP \ominus AP = CP \oplus (-AP). \tag{3.2}$$

The distance between points (i, j) and $(i + 1, j + 1)$ is minimized when the two segments are isometric and parallel. Furthermore, the patch may degenerate to a line or even a single point. If we construct a full set of patches from CP and AP, the

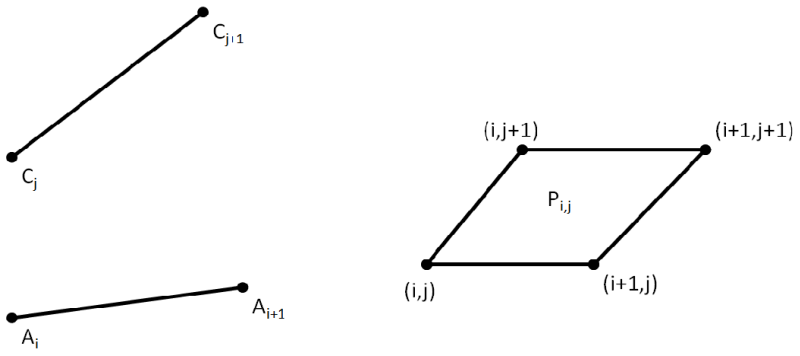

Figure 3.2. Two path segments and the patch formed from them

resulting diagonal distance from $(0,0) \rightarrow (n,m)$ is the DTW version of the Fréchet distance.

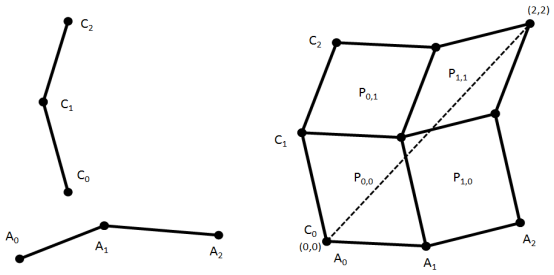

Figure 3.3. Two path segments and a manifold formed from them

Once the manifold is constructed, Steiner points are placed at equal intervals on all the edges of each patch. This is done to force the calculated geodesic to more closely approximate the actual geodesic. The construction of the manifold and the placement of the Steiner points can be done in linear time, $O(mn)$.

While there are algorithms to exactly find the geodesic on a closed polygonal surface [20], a wireless sensor network cannot achieve exact accuracy and using CDTW can provide the desired level of accuracy much quicker than the exact algorithms. Dijkstra's method [10] is used to find the shortest path on the manifold from the

origin to the furthest diagonal endpoint in approximately $O(mn \lg mn)$ [10, 9, 32]. This distance is the Fréchet distance. The Fréchet distance as determined by CDTW is then normalized by dividing by the total number of samples on the two curves. This is done to remove any variance caused by the number of Steiner points used. This also has the side effect of reducing the Fréchet distance to a more manageable number for the user without losing any of its usefulness. For most applications, the normalized Fréchet Distance will be much more useful in determining if the vehicle has indeed followed the path.

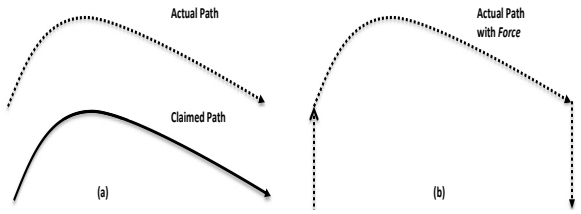

Figure 3.4. Claimed and Actual Paths (a), and Resulting Actual Path with Force Option (b)

Force Option to address Translational Invariance: Note that a feature of the CTDW technique is that two curves that are translationally invariant (like in Figures 3.1 (a) and (c)) will yield a Fréchet distance of zero in the basic technique [13]. However, two vehicle paths that are translationally invariant are still not be the same, and must hence yield a high value for the Fréchet Distance for them to be considered dissimilar. The issue of translational invariance is resolved using a simple refinement. Prior to computing the Fréchet Distance between the Claimed and Actual Paths, we introduce an option called "Force" wherein the end points of the Actual Path (i.e., origin and destination) are forced to be the end points of the Claimed Path (i.e., origin and destination). Figure 3.4 shows a simple illustration. As we see despite both paths being translationally invariant, the "Force" option when implemented changes the shape of the Actual Path and hence the computed Fréchet Distance will realistically capture the scenario of both paths being dissimilar, which cannot be accomplished with existing CDTW technique.

The entire sequence of steps is illustrated in Protocol 2.

Protocol 2 Curve Match Protocol to Compute Fréchet Distance

Curve Match

procedure INPUT BOTH PATHS
    Read m locations in Claimed Path
    Read n locations in Actual Path
    Force start and end points of Actual Path to be Claimed Paths
    Force end point $ap_n = cp_m$
end procedure
procedure CONSTRUCT MANIFOLD (M,N)
    Construct empty 2 PL manifold as an m by n array
    while more claimed points **do**
        while more agent points **do**
            Construct patch $P_{i,j}$
        end while
    end while
end procedure
procedure EXPAND ARRAY
    Place Steiner points along "horizontal" segments
    Place Steiner points along "vertical" segments
    Place Steiner points in the interior of each patch
end procedure
procedure FRÉCHET DISTANCE ACROSS THE MANIFOLD
    while unprocessed patches **do**
        Patch-wise Dijkstra to corner closest to origin
        for each point, including Steiner points **do**
            run Dijkstra Shortest Path Algorithm
            Add to point.distance
        end for
    end while
end procedure
procedure OUTPUT
    Compute path difference
    Fréchet distance from Patch-wise Dijkstra ▷ This is point.distance(m, n)
    Compute mean Fréchet Distance per sample
end procedure

4. SIMULATION AND EXPERIMENTAL DESIGN

Our study was divided into two parts: simulation and experimental "proof of concept." Before we could design actual experiments, simulation was required to prove the algorithms could detect the expected spoofing actions of a vehicle.

4.1. SIMULATION BASED STUDY

A simulator was developed and a number of possible path deviations were simulated and studied. The simulator was designed to be open-ended to allow for future capabilities.

4.1.1. Simulation of a Wireless Sensor Network. The simulator was developed to generate reasonable sample data(see Protocol 3). To be useful, the simulator had to be simple, quick, and generate random accurate data. The difficulties in generating the data by hand were too daunting.

Condition 4.1.1. *The sensors are deployed at random over a planar field F_0, with a sensing area of A_0*[1].

Condition 4.1.2. *It is desirable to be able to generate sensors with a common sensing range as well as sensors with varying ranges. A user option was designed into the simulator to allow sensors to be generated with constant ranges or with sensing ranges that varied at random over a predefined set of sensing ranges.*

Condition 4.1.3. *Each sensor must be placed at random within the sensor field.*

Condition 4.1.4. *No two sensors can be at the exact same location.*

4.1.2. Overview. The problem of creating sensor data by hand to test the algorithms proved to be very difficult. The solution was to create a very simple simulator to produce a set of sensor reports based upon user selected parameters, the dimension of the sensor field F_0, the number of sensors to be placed at random, the minimum sensing range of the sensors, an optional amount by which the sensing range may be increased for random varying ranges, and a set of points the agent visits.

[1]For future work, we will include an option to allow the sensors to be deployed based on real road topologies to represent urban traffic networks.

Protocol 3 Sensor Field Simulator

Simulator

```
1: Get user selections from global variables in source
2: procedure GENERATE SENSOR FIELD
3:     while more sensors to place do
4:         Generate unique random sensor ID
5:         Generate random x and y between (0, 0) and max (x, y)
6:         if a sensor has already been placed at (x, y) then
7:             Reject this location and try again
8:         end if
9:         if user selected for random ranges then
10:            Generate sensing range between 0 and max range
11:        else
12:            set sensing range to max range
13:        end if
14:        Add sensor to linked list of sensors
15:    end while
16: end procedure
17: procedure CREATE AGENT PATH
18:    Load agent points from a text file
19:    if user selected to add additional points then
20:        for each segment of the agent path do
21:            Add a random number of points on the segment
22:        end for
23:    end if
24:    timestamp ← 0
25:    while there are agent points to process do
26:        Add 1 to the timestamp to simulate time passing
27:        for each sensor do
28:            if agent point within current sensor range then
29:                Create event (sensor ID, timestamp, and distance)
30:                Add event to list
31:            end if
32:        end for
33:        Move to next agent point
34:    end while
35: end procedure
36: procedure OUTPUT DATA FOR CURVE MATCH
37:    Shuffle the sensor list
38:    Shuffle the event list
39:    Write the sensor list to a text file
40:    Write the event list to a text file
41:    Display field statistics
42: end procedure
```

4.1.3. User Options. The simulator has a number of options that can be selected by the user. These options are controlled by global booleans in the source. This approach was taken to relieve the user from being forced to provide console input for each run. Only the most important options are given below.

debug Used to control the production of extra console messages that are intended for debugging the code.

reload Used to suppress the generation of randomly placed sensors with random maximum ranges. This is intended to allow the same sensor field to be used for multiple runs of the simulator. This was very useful in examining the operation of the Curve Matching algorithm.

number of sensors Used to specify how many sensors to generate and place at random in the sensor field.

range Used to select the range of the sensors. The user can select a common range or ask for random ranges.

field size Used to specify the maximum x and y size of the sensor field. The details of how the simulator works can be determined from the algorithm, but an overview will be given here.

4.1.4. Brief program description. The user selected number of sensors are placed at random (x, y) coordinates within the sensor field with a random sensor ID. If a sensor is to be placed on top of another sensor, the random location is discarded and a new random location generated. To preserve the Poisson distribution properties, no allowance is made for the sensor location other than that it be within the sensor field. The range of the sensor is set to the user selected minimum range. If the user has selected an amount by which the range can vary, a random range from zero to the varying amount is added to the range. The sensor is then added to the list of sensors. This process continues until all the sensors have been created and placed.

A simple text file is then read for the (x, y) locations which the agent is to visit. These may, or may not, be the same as the way points the agent has been assigned to visit. The simulator has the ability to generate additional points at random on the line segments connecting the points read from the text file. This is done to simulate the agent traveling at an inconstant rate as would be expected in a real situation. Each point is then assigned a time one 'second' after the previous point.

The set of agent points generated by the simulator from the user input plus the randomly generated additional points is then processed in time sequence. For each point the sensor field is swept for the sensors that should sense the agent at that point. For each such sensor a report is generated with the sensor ID, time stamp, and range to the agent. These reports are held in a linked list for future processing.

When all of the agent points have been processed and all of the sensor reports generated, the simulator generates two text files for the Curve Match program. The first file contains the sensor information; i.e., the sensor ID, x, y coordinates, and sensing range. The file is generated in random order to simulate the expected lack of order in a real situation. The second file is the file of sensor reports generated by the simulator. This file contains one record for each report. The record consists of sensor ID, time stamp, and range to target. This file is also produced in random order to more closely simulate the expected real output of a sensor field.

The simulator is constructed in such a way that data about the k-coverage of the simulated sensor field could be easily calculated and displayed for the user. In practice, this turned out to be very useful.

We now analyze the performance of our proposed curve matching technique using simulations. The idea is to understand the sensitivity (or the trend) of the Fréchet distance computed to various practical deviations between a Claimed and Actual Path. Note that sensitivity for more complex path deviations can be analyzed by integrating sensitivity of simple deviations. For all simulations below, we consider the Actual Path to be a straight line of length 15 units.

Sensitivity of Fréchet distance to Translationally Invariant Paths: In Figure 4.1, we study the sensitivity of Fréchet distance to Claimed Paths that are translationally invariant to the Actual Path. By translationally invariant, we refer to paths that are exactly similar in shape, length and direction to the Actual Path, but are offset by a certain length. The X-axis in Figure 4.1 denotes the offset length the Claimed and Actual Path and the Y-axis is the Fréchet distance between them. As we can see with increasing offset length, the Fréchet distance computed by our protocol increases demonstrating increasing dissimilarity between the paths.

Sensitivity of Fréchet distance to Reverse Paths: In Figure 4.2, we study the sensitivity of Fréchet distance to two paths that all both similar in all respects,

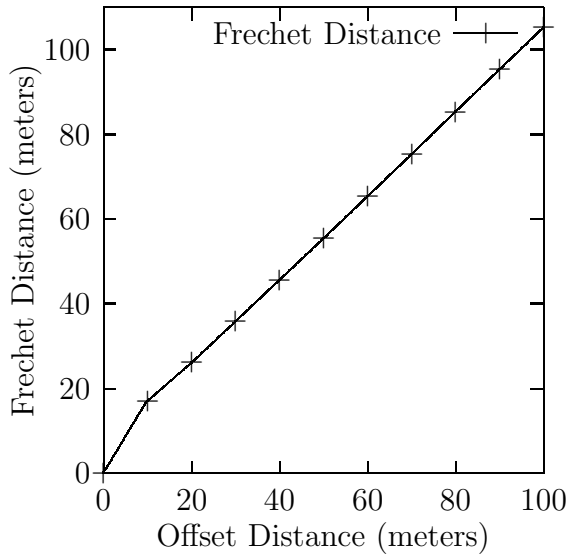

Figure 4.1. Shifted Path and Fréchet Distance

except for one being the exact reverse of the other in directionality. As we can see when the path length increases, the Fréchet distance increases, since reverse directionality with increasing path lengths mean two paths becoming increasingly dissimilar.

Sensitivity of Fréchet distance to Paths Spikes: Finally, in Figure 4.3, we study the sensitivity of Fréchet distance to two paths that overlap each other except there being a spike in one of paths compared to the other (similar to Figure 4.4 (d)). One of the paths is a straight line length 15 units, and other path is similar to this one but for the spike. The X-axis in Figure 4.3 denotes the (to and fro) length of the spike, while the Y-axis is the Fréchet distance. Clearly, increasing spike lengths means more dissimilar paths and hence increases in Fréchet distance.

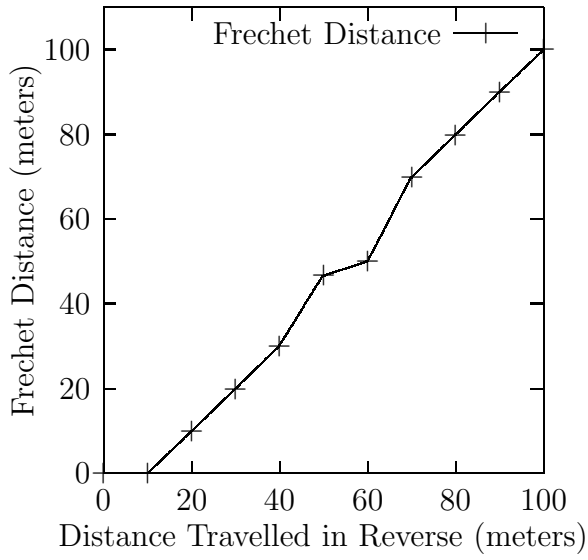

Figure 4.2. Following the Path in Reverse And Fréchet Distance

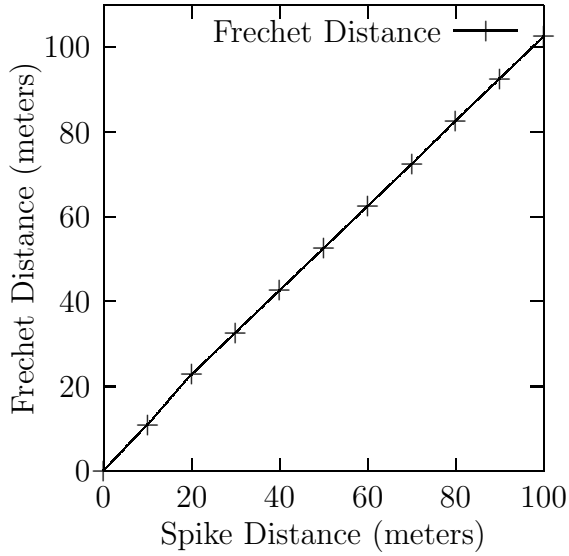

Figure 4.3. Spike Path and Fréchet Distance

4.2. EXPERIMENTAL DESIGN

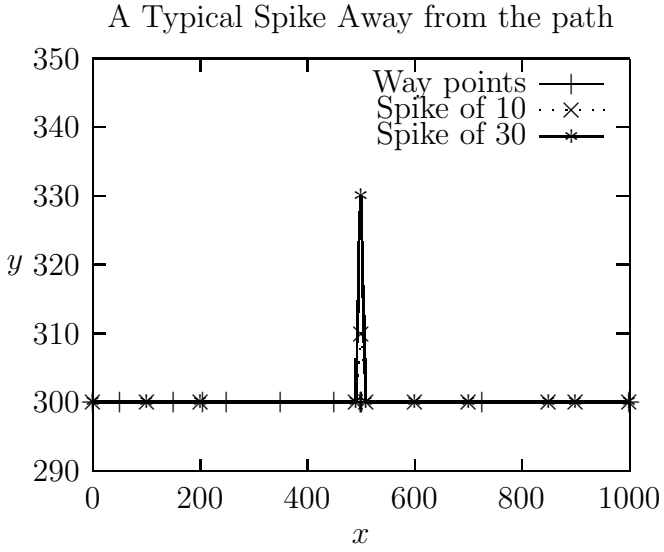

Figure 4.4. A Typical Spike Path

We now report results from real experiments conducted with a wireless sensor network and a single car in the city of Rolla (Missouri, USA). A section in Rolla was chosen for experiments as seen in Figure 4.5, where the circles denote locations where sensor motes were placed. The motes (20 in number) were TelosB motes whose details are shown in Figure 4.6. A laptop with a TelosB mote attached to transmitted and received packets from static motes. The transmitter mote was connected to a Laptop, and was placed in a Toyota Prius Car and driven around the experimental site. The average speed of the car was 15mph. For our experiments, we did not observe any packet drops during two way communication between the static motes and the mote in the car. The responses (i.e., witness certificates) from static motes was recorded in the Laptop to derive the Actual Path and compared with the Claimed Path post completion to compute the Fréchet Distance between them. For simplicity, a Cartesian Coordinate system was used for locations. However, using a coordinate system like GPS is straightforward as well.

Note that for proof of concept, only one car was used. As such, the car and sensor motes did not execute the Authentication Protocol in Protocol 1 for experiments conducted. We point out that the feasibility of hash functions in sensor networks has been well studied in works like [8, 18, 23], and as such, we did not emphasize it in our

Figure 4.5. Google Map of our Experimental Site in Rolla. Sensor locations are indicated with Circles

TelosB Specs	
Processor	TI MSP430 16bit Microcontroller
Program flash	48kB
RAM	10kB
Clock Speed	8MHz
Baud Rate	250Kbaud

Figure 4.6. Specifications of TelosB motes

experiments. In our experiments, as the car moves, the laptop recorded the witness certificates from sensors in its path that consisted of the sensor's location and current time, which became the Actual Path. Implementing and experimenting with the complete technique (including Authentication) with multiple cars is a part of future work. Also, due to space limitations, results from only selected (but generalizable)

test cases are reported here. The conclusions we derive here did hold for many other cases not reported here. For all the following discussions, a grid layout of the actual experimental site is used for illustration purposes.

4.3. RESULTS

Test Case 1: Figure 4.7 shows a test where the car was driven through four blocks from West to East on 4^{th} Street. When the Claimed Path was correctly reported the Fréchet Distance was zero. However, for two incorrectly reported Claimed Paths (dotted paths in Figure 4.7, the Fréchet Distance was large. Thus, paths reverse of the Actual Path, and Paths offset from the Actual Path are easily detected as fake.

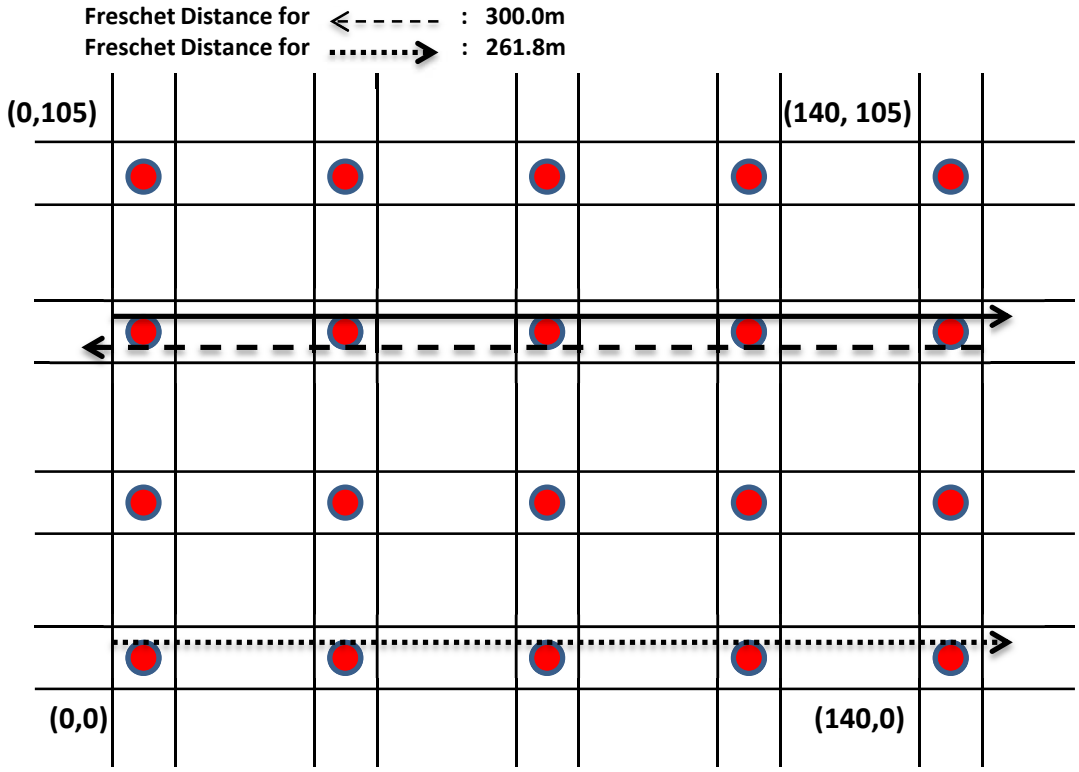

Figure 4.7. Detecting Reverses and Offsets

Test Case 2: The second experiment involved an Actual Path on 2^{nd} Street four blocks West to East. This time also, the Fréchet Distance between the reverse path and a path that deviates from the Actual Path with a spike are easily detected

as fake as seen in Figure 4.8). Note that the path exhibiting the spike is in the same direction as the Actual Path. If such a path were in reverse direction, the Fréchet Distance would be even larger

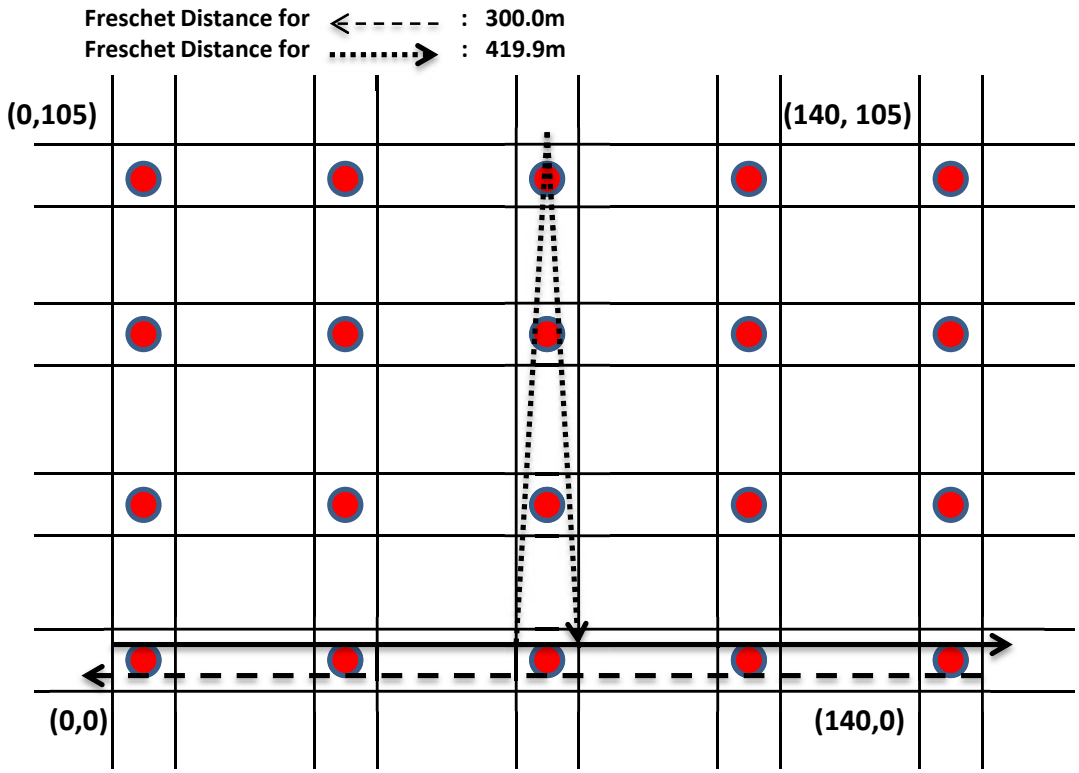

Figure 4.8. Detecting Reverses and Spikes

Test Case 3: For this case, the Actual Path consisted of traversing all 19 blocks in the experimental site in the trajectory shown in Figure 4.9. We consider a Claimed Path that still traverses 19 blocks, but reports as skipping the intersections of Elm and 4^{th} Street, and Elm and 3^{rd} Street. The corresponding Fréchet Distance is quite high, and hence is detected as fake.

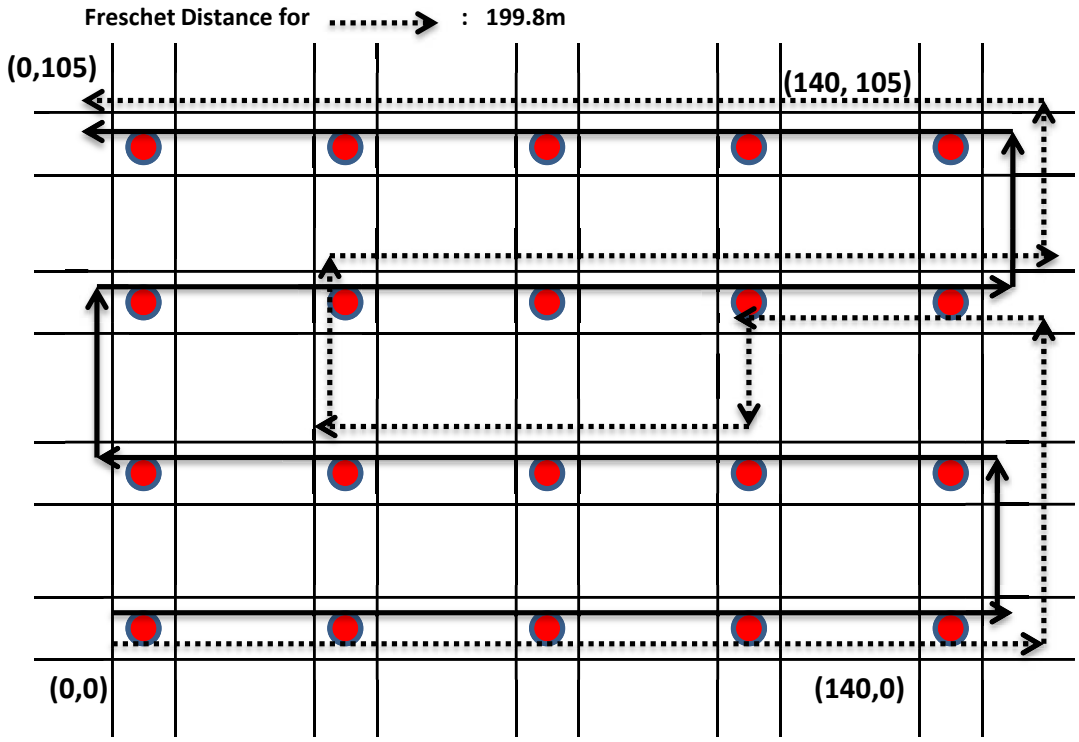

Figure 4.9. Detecting Complex Deviations

5. DISCUSSION AND FUTURE WORK

We have presented a method to verify the matching of a vehicle's actual path as reported by a wireless sensor network with its claimed path as reported by the driver. The difficulty lies in determining the exact situation if the vehicle actually deviated from the claimed path. There are a number of intuitive solutions such as GPS, but GPS is expensive and there are known hardware attacks to allow the positions to be wrongly reported. On the battlefield, one would expect GPS to be spoofed or jammed [15] [31] [30]. At the very best, GPS data requires time and expense to analyze. What is wanted is a "*yes/no*" answer or a simple to understand metric. The number of possible metrics is quite large with many metrics examined being found to be unsuitable for the problem at hand.

5.1. UNSUITABLE METRICS EXAMINED

Euclidean distance from a detection point to the claimed path. The first metric we examined was to find the Euclidean distance from each detection point to the claimed path. This presented a number of difficulties. The distance was sensitive to the choice of claimed path segment, or even worse which claimed point, to measure the deviation. It became obvious very early on that the only possible choice was to determine the perpendicular distance from the detection point to the claimed path segment that yielded the smallest distance. The choice appeared to involve either a judgment (human intervention) or an exhaustive search of all possible segments and this search would need to be repeated for each detection point. In a dense sensor network this would be computationally prohibitive.

The second issue with using the Euclidean distance from each detection point to the claimed path arises from sampling methods. The simple sum of the distances was wildly sensitive to the sampling method as was the mean of the distances zx(see Figure 3.1). It was easy to determine that a clever sampling of the detected points could yield any desired mean Euclidean distance.

Path length comparison between Actual Path and Claimed Path Another naive approach would be to simply compare the length of the Actual and

Claimed paths. The vehicle would clearly trace a longer actual path if it left the path and then returned; however, in our experiment we clearly showed a number of possible exploits that the vehicle could use to disguise a path variation. Taken alone, the difference in path lengths is not a useful measurement. It does give some additional useful information in specific cases such as when the node retraces a small portion of the path.

5.2. CONTINUOUS DYNAMIC TIME WARPING

CDTW was developed as an effective way to compare two curves and determining a matching metric. All of the issues we have dealt with in comparing two paths arise in curve comparison. The same questions of how closely two paths match, what kinds of metrics are useful, and what exactly is met by close also arise in curve matching. The main differences between the two problems arise in how the sampling is done and how to deal with the problems of invariance.

The two different techniques of Dynamic Time Warping, Continuous and simple DTW, both approach the problem by first constructing a chain from the sampled points. This lends itself much better to our problem as we defined the claimed path in terms of a number of points the vehicle was asked to visit in a specific order. The actual path the node followed was determined by constructing a chain from a number of sensor location reports. It turns out that comparing two chains is much simpler than comparing two smooth curves, indeed the most effective way to compare two curves using CDTW is to sample the curves and use the samples to construct chains. This sampling can be made as accurately as desired given sufficient computing power.

The issue of invariance is more problematic but simple to solve. When comparing two curves, the goal is to recognize the similarities regardless of the presentation of the curves. CDTW is able to provide good results regardless of the orientation of the curves (translational invariance). If this same technique were used in path matching, a path that follows the claimed path at a constant distance (see Figure 4.1) would return a prefect match. This would defeat our purpose.

Another type of invariance that creates issues is directional invariance. Two curves can match exactly even if one is drawn in the opposite direction from the other. In fact, when matching curves it is critical that the curves not have any

directional sense at all. This would defeat our purposes. When matching paths, it is critical that we take direction into account. We wish to enforce a directional sense on the vehicle so that it is known to have visited the claimed points in the claimed order. A direct consequence of directional sense is the notion of time. Given the times and locations a node is detected leads directly to the ability to match direction and time between two paths. However, it should be noted that the sense of time is often relative (point a before point b) due to problems in synchronizing clocks in sensor networks.

We resolved these issues by imposing boundary conditions on the actual path. If the vehicle is not detected within a reasonable distance of the claimed starting point, the paths are forced to match by pre-pending claimed starting point to the actual path. If the vehicle is not detected at the claimed path ending point, the claimed path ending point is appended to the actual path. In simple terms, the two paths are forced to begin and end at the same location and time. This forces a change to the shape of the actual path which the protocol easily detects. In addition to removing translational invariance, the boundary conditions enforce direction and time sense on the curve matching.

5.3. FUTURE WORK

During the experimental phase of this work, some future extensions became clear. There are other possible problem behaviors that will be considered in a future enhancement of the procedures. For example, how can we determine if a gap in detection is unreasonable? If the agent simply turns off the mobile node for a period of time we must be able to determine if the agent should have been detected by a sensor node. To determine the reasonable allowable gaps in sensing, we need to know the mean free path of a mobile node through a sensor field.

5.3.1. Mean Free Path Through a Sparse Sensor Network. An important measure of the effectiveness of a sensor network is the distance a mobile node can move through the network without being detected[17]. This problem is directly analogous to the problem of the mean free path of a molecule through an ideal gas.

Condition 5.3.1. *The sensors are deployed at random over a planar field F_0, with*

a sensing area of A_0.

While it is not strictly necessary that the sensors all have a common sensing range, it makes the calculations much simpler. Therefore, we assume N sensors each with a sensing range R deployed uniformly at random over the sensing field.

Condition 5.3.2. *Each sensor senses in a circular pattern with an area of* A_i *and a radius of R.*

Problem 1. Calculate the Mean Free Path
Given a sensor field F_0 with an area A_0 sensed by N sensors $(s_i : i = 1, 2, 3, \ldots N)$ with sensor s_i having a sensing area A_i, randomly and independently deployed throughout F_0, compute the probability $P(k)$ that a mobile sensor X randomly crossing A_0 is detected by at least k sensors assuming X is traveling in a straight line.

Definition 1 (Sensing Area)**.** *The sensing area* A_i *of sensor* s_i *is the effective target thickness* $E(T_i)$ *and defined as a disk of radius* $r_i = \frac{E(T_i)}{2}$ *which is identical to the range of the sensor* s_i.

Theorem 1. *Mean Free Path until First Detection*
The Mean Free Path[17],$E(\sigma)$ *of a mobile agent node moving in a straight line through the sensing field* F_0 *is given by:* $E(\sigma) = \frac{A_0}{NE(T)}$.

Proof. A target X can move a distance σ undetected if the trajectory does not intercept the sensing area of any sensor. When all sensors have the identical sensing areas, the target is undetected if, and only if, its trajectory does not come within a distance of $\frac{E(T)}{2}$ of any sensor. This is equivalent to considering the target node to have an effective thickness of *E(T)* and the sensors to be points with an effective sensing range of zero. When the mobile node moves a distance of σ_X, it sweeps out an area of size, $F(\sigma_X) = E(T)\sigma_X + f$ where f is some residual area due to the geometry of the sensor field not having a straight perimeter normal to the trajectory. Therefore, the probability that the mobile node X travels a distance $\sigma \geq \sigma_X$ is equal to the probability that no sensor is located within $F(\sigma_X)$. Given that the sensors are randomly and equally deployed, the number of sensors within $F(\sigma_X)$ is given by a homogeneous Poisson point process of density $\rho = \frac{N}{A_0}$. Therefore:

$$P(k) = \frac{(\rho F)^k}{k!} e^{-\rho F}. \tag{5.1}$$

This equation holds true as $F_0 \to \infty$ if the density of sensors, ρ, remains constant. The probability that the free path of the target X is σ_X is the same as the probability that no sensors exist within an area of size $F(\sigma_X)$:

$$P(\sigma \geq \sigma_X) = P(N_{F(\sigma_X)} = 0). \tag{5.2}$$

$$e^{-\rho F(\sigma)} = e^{-\rho(E(T)\sigma + f)}. \tag{5.3}$$

The random variable σ is a non-negative continuous variable and has an expected value given by:

$$E(\sigma) = \int_0^Q P(\sigma \geq \sigma_X) d\sigma_X. \tag{5.4}$$

$$E(\sigma) = \frac{e^{-\rho f}}{\rho E(T)}(1 - e^{-\rho E(T) Q}). \tag{5.5}$$

where Q is the maximum possible path length through the sensor field along the trajectory of X. If the residual area f is small enough such that $e^{-\rho f} \approx 1$ and Q is long enough such that $e^{-\rho E(T) Q} \approx 0$,

$$E(\sigma) \approx \frac{1}{\rho E(T)} = \frac{A_0}{N E(T)}. \tag{5.6}$$

□

5.3.2. Mean Distance to Detection and Coverage Simulator. The basic simulator was then modified to produce Mean Distance to Detection information. This is done by generating a sensor field in the same manner as the first simulator. In order to simplify the calculations, a single sensing range is chosen for all sensors. An agent path is then chosen with a random x along the y-axis and sent into the sensor field along a straight line with a random angle to the y-axis. The distance to first detection is recorded. Data are recorded for each run and analyzed to determine how well the simulation agrees with the theoretical values. Additionally, the coverage data for each run are also recorded and compared to the theoretical values.

5.3.3. Publication. A condensed version of this work, with extensions in the field of vehicular tracking, is planned for future publication. This work will include the experiments discussed in this thesis as well as attempts to detect tampering with the mobile node such as turning the node on and off to avoid detection of path deviations. The security certificate exchange presented here will be implemented on the mobile node and sensor network. Other extensions will surely present themselves during the completion of the proposed paper.

6. CONCLUSIONS

In this thesis, we study the issue of path verification of mobile vehicles. The problem has practical significance to a number of organizations including police precincts, cab operators, trucking companies etc. Our proposed solution leverages wireless sensor network support to authenticate vehicle movements. Then we designed a protocol to compute the Fréchet Distance between a claimed path reported by a vehicle and an actual path derived from certificates provided by sensors. Extensive simulations and real experiments conducted in Rolla validate our proposed techniques.

BIBLIOGRAPHY

[1] T. Abdelzaher, B. Blum, Q. Cao, D. Evans, J. George, S. George, T. He, L. Luo, S. Son, R. Stoleru, J. Stankovic, and A. Wood. Envirotrack: An environmental programming model for tracking applications in distributed sensor networks. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, Tokyo, March 2004.

[2] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3):251–265, 1995.

[3] H. Alt and M. Godau. Computing the frechet distance between two polygonal curves. In *International Journal of Computational Geometry Applications*, volume 5, pages 75–91, 1995.

[4] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proceedings of International symposium on Information processing in sensor networks (IPSN)*, Los Angeles, 2003.

[5] K. Buchin and M. a. W. Y. Buchin. Exact algorithms for partial curve matching via the fréchet distnce. *Procedings 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 645–654, unknown 2009.

[6] D. Cardoze and L. Schulman. Pattern matching for spatial point sets. In *Proceedings of the 39th IEEE Symposium Foundations of Computer Science*, pages 156–165, 1998.

[7] M. Cesana, L. Fratta, M. Gerla, E. Giordano, and G. Pau. C-vet the ucla campus vehicular testbed: Integration of vanet and mesh networks. In *Wireless Conference (EW), 2010 European*, pages 689–695. IEEE, 2010.

[8] Y. Chen, I. Lin, C. Lei, and Y. Liao. Broadcast authentication in sensor networks using compressed bloom filters. *Distributed Computing in Sensor Systems*, pages 99–111, 2008.

[9] T. Cormen, C. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Third ed.)*. MIT Press, 2009.

[10] E. W. Dijksta. A note on two problems in connection with graphs. *Numerische Mathematick*, pages 269–271, 1959.

[11] M. Duarte and Y. Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.

[12] Efrat, Guibas, S. Har-Peled, Mitchell, and Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete & Computational Geometry*, 28:535–569, 2002. 10.1007/s00454-002-2886-1.

[13] A. Efrat, Q. Fan, and S. Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision*, 27:203–216, 2007. 10.1007/s10851-006-0647-0.

[14] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *In Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom)*, Philadelphia, 2004.

[15] T. Humphreys, B. Ledvina, M. Psiaki, B. OHanlon, and P. Kintner Jr. Assessing the spoofing threat: development of a portable gps civilian spoofer. In *ION GNSS*, 2008.

[16] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 457–465. Society for Industrial and Applied Mathematics, 1999.

[17] L. Lazos, R. Poovendran, and J. A. Titcey. Probabilistic detection of mobile targets in heterogeneous sensor networks. *IPSN 2007, 6th International Symposium on Information Processing in Sensro Networks*, pages 519 – 528, April 2007.

[18] H. Lee, Y. Choi, and H. Kim. Implementation of tinyhash based on hash algorithm for sensor network. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 10, pages 135–139, 2005.

[19] K. Lee, S. Lee, R. Cheung, U. Lee, and M. Gerla. First experience with cartorrent in a real vehicular ad hoc network testbed. In *2007 Mobile Networking for Vehicular Environments*, pages 109–114. IEEE, 2007.

[20] J. Mitchell, D. Mount, and C. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16:647, 1987.

[21] M. Nekovee. Sensor networks on the road: the promises and challenges of vehicular ad hoc networks and grids. In *Workshop on Ubiquitous Computing and e-Research*, page 24, 2005.

[22] H. Song, S. Zhu, and G. Cao. Svats: A sensor-network-based vehicle anti-theft system. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 2128–2136. IEEE, 2008.

[23] P. Trakadas, T. Zahariadis, H. Leligou, S. Voliotis, and K. Papadopoulos. Analyzing energy and time overhead of security mechanisms in wireless sensor networks. In *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*, pages 137–140. IEEE, 2008.

[24] unknown:comuportugal.org. http://drive-in.cmuportugal.org/.

[25] unknown:csail. http://cartel.csail.mit.edu/doku.php.

[26] unknown:ertico.com. http://www.ertico.com/.

[27] unknown:umass.edu. http://prisms.cs.umass.edu/dome/.

[28] unknown:vehicle infrastructure.org. http://www.vehicle-infrastructure.org/.

[29] J. Villasenor. Ensuring hardware cybersecurity. In *Issues in Technology Innovation*. Center for Technology Innovation at Brookings, 2011.

[30] J. Warner and R. Johnston. Gps spoofing countermeasures. *Homeland Security Journal*, 2003.

[31] C. Wullems. A spoofing detection method for civilian l1 gps and the e1-b galileo safety of life service. *IEEE Transactions on Aerospace and Electronic Systems*, 2011.

[32] J. Xiao and F. Lu. An improvement of the shortest path algorithm based on dijkstra algorithm. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 2, pages 383–385. IEEE, 2010.

CURRICULUM VITAE

Name Gerry Wayne Howser
Date of Birth April 24, 1952
Place of Birth Jefferson City, Missouri, USA
Current Residence Rolla, Missouri, USA
Citizenship United States of America

Education & Professional Development

BS in Physics University of Missouri-Rolla, Rolla, Missouri, May 1974
PhD candidate, Physics University of Missouri-Rolla, Rolla, Missouri, 1974-1977
MS, Computer Science Missouri University of Science & Technology, expected: May 2012
PhD, Computer Science Missouri University of Science & Technology, expected: uncertain

Industry Classes and Seminars Taught:

1998 APECS/Discovery National Users Association seminar, "How **Not** to Put a UNISYS Mainframe on the INTERNET"
1999-2000 FORE ATM Fundamentals
1999-2000 FORE Core Technologies Certification
1999-2000 FORE Edge Technologies Certification
2000-2002 ATM Essentials
2000-2002 ATM Internetworking
2000-2002 ATM Troubleshooting

Teaching Experience

1975 Graduate Teaching Assistant(GTA), University of Missouri-Rolla Physics Department
1981-1998 Adjunct Faculty, Computer Science Department, Lincoln University, Jefferson City, Missouri
2002-2010 Career and Technical Education Instructor, Kalamazoo Regional Educational Service Agency, Kalamazoo, Michigan
2007 Instructor, Computer Science Department, Kalamazoo Valley Community College(KVCC), Kalamazoo, Michigan
2012 GTA, Missouri University of Science & Technology, Computer Science Department

Professional and Academic Publications

Gerry Howser and Bruce McMillin, Vulnerabilities and Security Models of Manufacturer/Automobile Systems, *Sixth IFIP WG 11.10 Conference on Critical Infrastructure Protection*, (August 2012)

Professional Certifications

Marconi Certified Instructor Marconi Systems
FORE Certified LAN Engineer Marconi Systems

FORE Certified CORE Engineer Marconi Systems
FORE Certified Edge Engineer Marconi Systems
ATM Certified Engineer Global Knowledge
CompTIA Network+ Certification CompTIA
Career and Technical Education Certified Instructor State of Michigan (2002-2011)

Professional Affiliations and Civic Leadership

2010-2012 Association for Computing Machinery(ACM)
2011-2012 Institute of Electrical and Electronic Engineers(IEEE), Computer Society
2002-2010 Van Buren (Michigan) Vo-Tech Center Advisory Committee
2002-2010 Kalamazoo Valley Community College (Michigan) IT Advisory Committee
2009 Michigan Academic Alignment Panel
1990-1992 Advisory Board for the Missouri Department of Elementary and Secondary Education (Mathematics and Sciences)
1990 Original Investigator on the grant that founded the Missouri Research and Education Network (MORENET)
1992-1993 Secretary, Governing Board of the Missouri Research and Education Network
1990-1998 Governing Board of the Missouri Research and Education Network
1990-1998 MORENET Technical Advisory Committee
1984-1997 Boy Scouts of America, Silver Beaver Award 1990