Toward Efficient Clustering of Time Series Automotive Signal Data

A Thesis Presented to

The Faculty of the Computer Science Department

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Computer Science

by

Student Name:
Kyle Robert Crockett

Advisor:
Dr. Michael Soltys

September 2021

© 2021

Kyle Robert Crockett

ALL RIGHTS RESERVED

APPROVED FOR MS IN COMPUTER SCIENCE

Advisor: Dr. Michael Soltys **Date**

Dr. Reza Abdolee **Date**

Dr. Jason Isaacs **Date**

APPROVED FOR THE UNIVERSITY

Dr. Jill Leafstedt **Date**

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Title of Item

3 to 5 keywords or phrases to describe the item

Author(s) Name (Print)

Author(s) Signature Date

This is a permitted, modified version of the Non-exclusive Distribution License from MIT Libraries and the University of Kansas.

Toward Efficient Clustering of Time Series Automotive Signal Data

Kyle Robert Crockett

September 7, 2021

Abstract

As the demands on automotive communications systems have increased, so have the amount of data and the complexity. Connected vehicles pose a unique challenge due to the fact that they have become mobile embedded networks that directly impact the physical well being of people in and around the vehicle. Reverse engineering of automotive communications has become an area of interest for researchers hoping to develop supervisory security systems that are able to learn normal operation characteristics and detect anomalies. This paper focuses on time series analysis and first surveys existing methods, proposes a new method of sample rate reduction, and finally analyzes the real-time processing feasibility of each method. This paper also provides a counter argument to the widely held assumptions

that DTW is both necessary or even useful to automotive time series analysis.

Contents

List of Figures

1 Introduction

We are currently living in an age of unfathomable complexity. Whereas it was once possible to design customs systems for every use it is now highly desirable to employ systems that can adapt and optimize for their specific use case. With the growth of the Internet of things (IoT), the connected automobile has grown with it. The connected car poses a unique security risk since it is a combination of both information and physical systems. With the connected car it is possible for not only the owner's information to be stolen but also physical harm can be inflicted upon them. Because of the large risks associated with connected cars it is of great interest to the automotive industry to cost effectively monitor and detect any malicious incursion.

The available information streaming from a modern vehicle's sensors has increased several orders of magnitude from what was available just a few years prior. Whit a focus on clustering, this thesis provides insight into a viable method of sample rate reduction for use with automotive time series sensor data. Many of the most widely used tools for the analysis of time series data have quadratic time and space complexity [1][2][3][4][5]. Most of the prior research has been focused on dimension reduction by generalized optimization or approximation of the algorithms without much focus on the end use case [6][7]. This Thesis provides a new method of sample rate reduction by analyzing the full resolution signal first and then removing data samples while maintaining enough information to successfully perform clustering.

In the last 10 years the field of automotive data analysis and reverse engineering has gained a lot of attention. The need for automotive cybersecurity research has been demonstrated by security researches and met with interest from the academic community [8][9][10]. Most of the prior work has been centered around the reverse engineering of the controller area network bus or classifying signals [11][12][10]. Many of the methods used employ Dynamic Time Warping to perform their comparisons [11]. There have been many efforts to optimize Dynamic Time Warping but they have remained limited to modifying the algorithm choosing a subset of data to analyze [**?**][6][13][13]. The optimizations and windowing have been unfortunately discovered to be of limited use [5][14]. In the world of time series analysis however, Dynamic Time Warping is only one of many techniques currently employed. UC Riverside has developed several techniques such as Symbolic Aggregate approXimation, Shapelets, and MPdist [15][16][17]. There have been some efforts to employ clustering on automotive data but they have been relatively few in number and performed with labeled data to cluster attributes like driving styles [18][19].

1.1 Contributions

This thesis refutes a widespread assumption within the automotive research community as to the importance of dynamic time warping in signal clustering. This thesis provides evidence that Euclidean distance is not only faster to perform but just as accurate as dynamic time warping in the vast majority of situations.

The concept of the time series *Self Distance* for measuring signal degradation at various sample rates is introduced in this Thesis. It is proposed as a relative measure and alternative to more complex methods of sample rate optimization.

This thesis argues that due to the longer sample times required for effective capture of signal data, the definition of real-time or online analysis allows for a surprisingly large amount of analysis time. Measurements are first made on a one-to-one comparison to determine if the industry favored DTW algorithm can process a series of signals in less time than it took to capture them [20]. The experiments were run on a Nvidia Jetson Nano to simulate the type of embeddable hardware used in automotive applications. DTW is not likely to be able to meet this criterion in real use, but it is proposed that DTW can still be considered real-time if it is able to be calculated faster than the data is received. This for example, would be 300% of the sample time for an 8 hour driving day and a full 24 hours in which to process the signals.

2 Background

2.1 Cybersecurity and Connected Vehicles

2.1.1 Evolution of Automotive Electronics

In the early days of the automobile the electrical systems were remarkably simple, and every single component added its own dedicated wiring. Most of the early uses of electrical systems were used to run lighting and functions such as the starter motor and in some cases power windows. As technology progressed the complexity of the electrical system did as well. Soon because of ever tightening emissions requirements, computerization took hold of key functions. An example of this shift towards computer control is demonstrated by systems such as electronic fuel injection (EFI).

By the mid 1980s computers were now being fully integrated within the vehicle performing many different tasks the obvious next step was to have these systems log and report data for diagnostics purposes [21]. In 1987 BMW brought to market the first car with diagnostics capabilities [21]. Meanwhile, car manufacturers were searching for a solution to an ever-increasing bulk of wiring within the vehicles. The growth in volume of wiring within vehicles was not only an economic drawback because of the cost of the wiring itself but also due to the added mass caused a decline in fuel efficiency [21].

2.1.2 Intra-Vehicle Communication Systems

In 1983 the major automotive component supplier Bosch began an internal project to find a solution for the industries wiring problem. The solution that Bosch devised was to be known as Controller Area Network [21]. Controller Area Network allowed for communications between hundreds of nodes on a single twisted-pair of wire. During this time, many proprietary solutions were developed by each manufacturer such as or Diagnosis Bus and I/K Bus by BMW in 1987 and 1991 respectively [21]. The following year, in 1992 Daimler introduced the first production vehicle incorporating controller area network [21]. The CAN bus ultimately won out over all of the proprietary solutions due to its open standard. The controller area network bus facilitated complex interactions between components within the car. Command and control signals could be sent to enable or disable door locks and control lighting. In addition to command-and-control signals, nodes were able to give status reports such as whether a light bulb was out or the position of a switch. Analog signals were also transmitted across the bus in the form of temperature sensors or engine speed data. The early diagnostic systems were relatively simple in nature consisting of minimum or maximum threshold values to toggle an alarm. In modern vehicles the diagnostic systems are far more complex and use many statistical techniques to predict long term problems before they result in the failure of a component. In the years since then, many other solutions have been developed for intra-vehicle communication

but CAN remains the most widely used today.

2.1.3 Telematics, V2V and V2I

As high-speed cellular network coverage has become more ubiquitous, manufacturers have been able to make use of telematics units that are connected to the internet and continually transmit vehicle sensor data in real-time. At the present time, a wide variety of vehicles can be purchased with real-time information streamed directly to and from the vehicle. The data to be consumed by the vehicles occupants includes real-time traffic data, music and weather reports. Vehicles will send information to manufacturers such as onboard diagnostics data. This diagnostics data is then used by manufacturers to recommend maintenance before they become more costly repairs.

Still only in its infant stages, the technology of vehicle to vehicle and vehicle to infrastructure communication is the industry solution to many of the challenges faced by autonomous driving. Vehicle to vehicle communication would enable vehicles to give positional data to one another to avoid collisions and a vehicle could transmit road hazard information, such as objects obstructing the road, to the vehicles behind it [22]. Vehicle to infrastructure communication would assist vehicles in localization by referencing the positions of infrastructure nodes relative to the vehicles position [22]. Another use of vehicles infrastructure would be traffic lights telling vehicles their status ahead of time which would reduce and autonomous vehicles reliance on camera data.

2.1.4 Cyber Threats and Risks

For all the benefits that we have discussed there is a dark side which must be considered. Every mode of communication enabled within a vehicle and from a vehicle to the outside world is a potential path for exploit by a bad actor. Because of the ever increasing complexity of on-board systems, the automotive industry has enacted stringent protocols for ensuring the safety and reliability of their products. Up until recently the internal networks within a vehicle were only accessible by directly and physically accessing the network cables. Generally speaking it would be extremely difficult for a threat to compromise a moving vehicle in this manner. Security researchers, Charlie Miller and Chris Valasek demonstrated the vulnerability of the internal vehicle buses in a technical white paper published in 2014 [8]. While the pair did go on to publish other papers related to remote exploits, most of their work demonstrating safety critical vulnerabilities required hands-on access to the vehicles. In the years since Miller and Valasek's research it has become increasingly common to find vehicles with permanent internet connectivity and this trend is expected to continue [23].

A modern connected car is vulnerable from theft of the data contained within such as GPS locations and credentials for subscription services as well as the vulnerability of safety critical systems that ensure the well-being of its occupants. Just as diagnostic systems were implemented to predict the need for repair or replacement of mechanical components within the vehicle

there is now a need for supervisory systems that will predict or detect a potential exploitation. The main use of having such on-board systems is as a last line of defense in the event of cyber attack. Security best practices such as rotating security certificates and enforcing least privilege are among the first lines of defense in securing the data centers and vehicles against attack. If those countermeasures fail it is not acceptable to a allow a threat to go undetected especially in a cyber physical system carrying humans traveling at a high rate of speed.

2.1.5 Fleet Management

It is highly desirable for both manufacturers and owners of large fleets to make use of telematics to maintain their vehicles. It is currently common for fleet maintenance technicians to be required to spend countless hours performing diagnosis of failed vehicles. By only detecting the need for repair after catastrophic failure, additional downtime is incurred. By utilizing telemetry to predict potential failures it is possible to perform the repairs during scheduled maintenance, thus reducing downtime. If an on-board diagnostics system detects an error or potential cyber threat it is possible for the proper teams to be notified within the organization.

2.2 Time Series Data Analysis

2.2.1 Classifying VS Clustering

In the fields of data science and machine learning both classifying and clustering are use to make sense of data. Classifying is when unknown data is labeled with predefined groups. Classifying is useful when the data is well understood and it is possible to list all of potential labels for the data. Clustering is grouping unknown or unlabeled data by similarities. Clustering is most useful when trying to make sense of unknown or unlabeled data. By grouping the data by its similarities it is then possible to attempt to reverse engineer or suss out the labels and meaning of the data.

2.2.2 Basic Methods for Time Series Analysis

One of the simplest methods of analyzing to time series data sets is through Euclidean distance. Euclidean distance measures the difference in value between each data point on a one-to-one mapping. A visualization of this mapping is shown in Figure 1. It can be seen that both time series segments are similar however not exactly the same. Euclidean distance has been demonstrated to not be very effective with certain data sets [24][**?**][13].

k Nearest Neighbors (KNN) can be used for time series classification and has been shown to perform as well as Dynamic Time Warping (DTW) with some data sets [24]. The KNN algorithm works by comparing a data point to a specified k number of previously classified neighboring data points. The

data point in question will be labeled with the predominant neighboring point's class. For time series, the KNN algorithm is performed for every point in the data set and then the series data will be classified with the most closely correlated label. Both Euclidean distance and DTW are specific implementations of one nearest neighbor (1-NN) [24]. k-means is similar to KNN but is used in time series clustering by taking the moving average of each data series in a group and then clustering each series' moving average with that of the most similar other moving averages [2]. The k-means clustering algorithm works on data that has been minimally prepossessed, unlike KNN which requires previously classified sample data to compare the unclassified data series to [2].

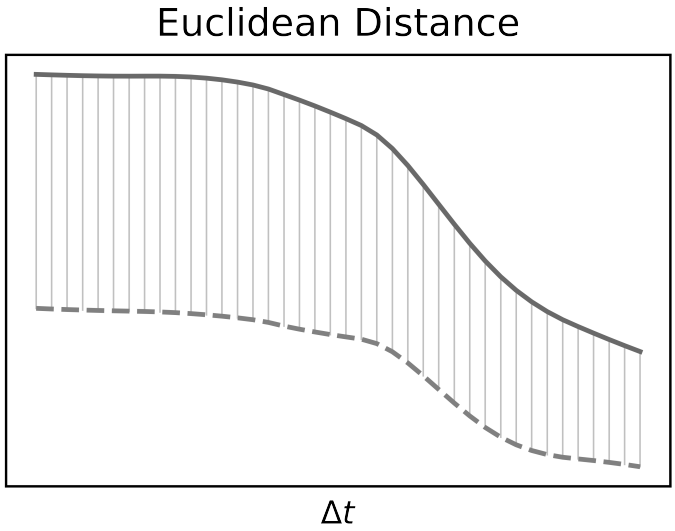

Figure 1: Euclidean distance of two time series

2.2.3 Dynamic Time Warping

Basic DTW: Dynamic Time Warping (DTW) attempts to solve the issues present when using Euclidean distance [25]. DTW will allow for warping in the time series along a cost matrix to try to find the best fit. This warping is demonstrated in Figure 2. The algorithm attempts to take the two similar time series and explain the differences in offset. Figure 3 illustrates the warping path taken across the cost matrix. The path is defined by the dark black region traversing the matrix. The varying cost of each element is illustrated by the shaded grey area with longer distances, or higher cost, being darker. The warping path in Figure 3 shows a warping bias towards the shorter (lighter) regions of the matrix. The cost matrix in Figure 3 has been constrained by the Itakura parallelogram where the warping path is allowed a maximum slope defined by the constraints of the parallelogram [26]. This is one of the many attempts to optimize and speed up DTW execution. The well known algorithm that computes the similarity of two strings known as the Levenshtein Distance is an implementation of DTW. The Levenshtein Distance is an invaluable tool used in search recommendations and spell checking [27].

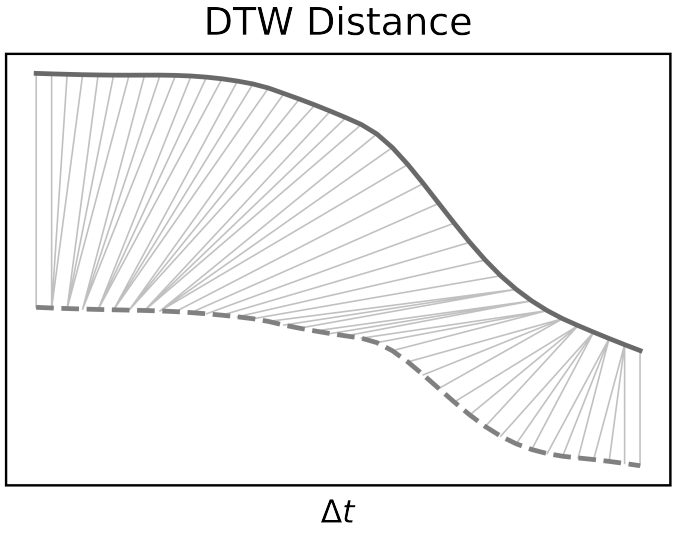

Figure 2: DTW distance of two time series

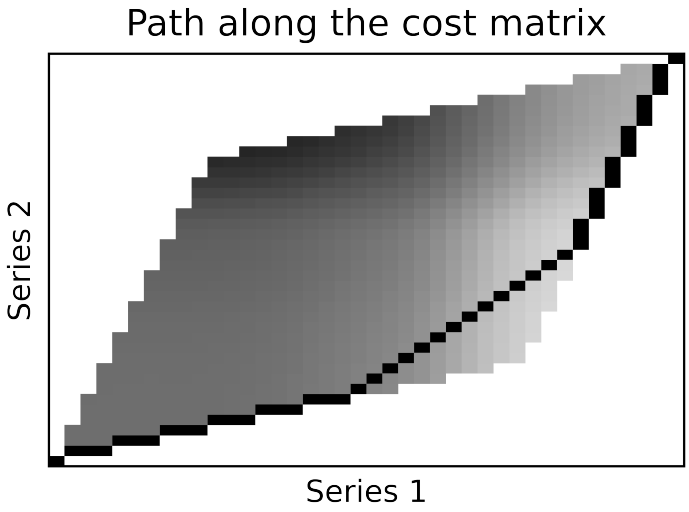

Figure 3: Cost Matrix Diagram

Derivative DTW: One issue with DTW is the tendency for certain features in the data to lead to pathological warping around a singularity [**?**]. One solution proposed by Keogh et al. is through the use of what they call Derivative Dynamic Time Warping (DDTW) [**?**]. In DDTW the DTW is performed on the derivative of the data. The use of the derivatives has good performance on reducing the number of singularities and pathological warping. DDTW is compared by Keogh et al. against some more common DTW constraining methods like Windowing, Slope Weighting and Step Patterns. The process of Windowing takes a subset of the data and performs DTW on it. To address the complete data set the window is then slid along the data with DTW being performed at each step. Windowing is able to limit the effect of singularities on data but not eliminate their presence all together [**?**]. Slope Weighting applies a penalty of increasing weight as the warping path diverges further from the diagonal on the cost matrix [**?**]. The weights can be tuned to increase or decrease the bias towards the diagonal path. Step Patterns attempt to control singularities and pathological warping by creating an absolute outer bounds to the cost matrix warping path. This is often accomplished by assigning infinite cost to elements outside of the Step Pattern. Many different Step Pattern geometries have been devised but come at the peril of choosing the correct one for the data [**?**]. Other attempts at solving the pathological warping problem have been developed such as EventDTW proposed by Jaing et al.[28]. EventDTW attempts to limit warping by matching the data trends (upward and downward slopes)

to one another in a pre-processing step.

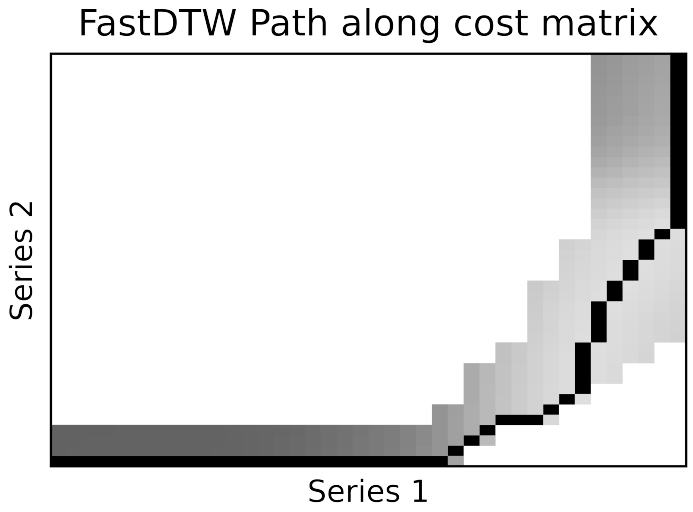

Figure 4: FastDTW Diagram

FastDTW: Because of the utility of DTW it has been applied quite often in research involving time series data. A major drawback to utilizing DTW however, is its large time and space complexity of $O(N^2)$. This complexity limits the maximum size of any data series to a few thousand data points [6]. It should then be no surprise that there has been much effort directed at increasing the efficiency of the DTW algorithm. FastDTW is one of the most widely implemented instances of these efforts with over one thousand citations [5]. The fast data algorithm works by reducing the resolution of the initial cost matrix to determine an approximate warping path and then gradually increasing resolution within the previously defined approximate warping path [6]. Figure 4 illustrates the initial low resolution warping path

shown by the grey shaded region of the cost matrix. As with Figure 3, the shaded area in Figure 4 corresponds to shorter distance (lower cost) as lighter in color. This region is then the bounds for the final pass of the full resolution DTW shown in Figure 4 by the black path along the cost matrix. In spite of its wide use, FastDTW is not without its critics. In a paper published in 2020, Wu et al. argues that the approximation provided by the FastDTW algorithm and it's repetitious nature is not as useful and is ultimately slower than just performing full DTW one time [5]. They argue that for all of the times that FastDTW was used in research, it would have been better to just use standard DTW and that the approximation provided by FastDTW is not worth the minimal time savings [5].

2.2.4 Shapelets and Motifs

Shapelets: As a solution the drawback of KNN requiring a full analysis of the time series data a new primitive called the "Shapelet" was devised. Time series Shapelets are defined as "subsequences which are in some sense maximally representative of a class" [16]. When using the nearest neighbors approach to time series classification it is only possible to determine how the data should be classified and will not provide any further information about why the data should belong to a particular class [16]. By using KNN it is not possible to create smarter or more highly optimized ways of extracting meaning from data. The first method for identifying Shapelets for a given class is using a brute force method. Given a number of time series of a given

class, each series is broken into sub-series and each sub series is compared along a sliding window to the whole data series. The distance between each Shapelet candidate and each whole time series are used to determine how useful the Shapelet candidate is at representing the class as a whole [16].

The large amount of research devoted to the efficient discovery of Shapelets in time series is testimony to their utility.

The python *tslearn* [29] library is an alternative to *pyts* [30] that implements many of the advances in time series shapelets. One algorithm of note is Learning Time-Series Shapelets Josif by Graboka et al. [31] that proposes a novel method of Shapelet discovery capable of outperforming all previous methods.

The "Fast Shapelets" algorithm proposed by Rakthanmanon and Keogh attempts to make use of symbolic representation to identify the most likely candidates and prune away any regions where no Shapelet candidates are likely to be found [32].

Motifs: The next data series primitive of interest is known as a Motif. A motif is defined as "pairs of individual time series, or subsequences of a longer time series, which are very similar to each other" [33]. A Motif differs from a Shapelet in both complexity and frequency of occurrence. One or more different Shapelts must only appear once to classify a data series and are generally used on shorter data series. Motifs are more complex subregions of a large data series and must appear multiple times. Also further simplifying the

difference, it can be stated that a Motif may contain Shapelets, but a Shapelet cannot contain any number of Motifs. The process of Motif discovery works by selecting a subset of a data series and then ordering all other regions by their distance from that test subset. Then the best-so-far matches are returned as the top Motif candidates [33].

2.2.5 Matrix Profile Distance (MPdist)

Matrix Profile Distance attempts to provide a solution to a fundamental weakness in both the Euclidean and DTW time series measures. When attempting to measure the similarity of time series data with the Eclidean or DTW measure, not only is all of the data used in the measurement but also the order in which it arrives [17]. Leveraging the lessons learned from the Shapelet, namely the fact that the correct sub-sequence of data can be used to provide insight into the greater time series as a whole, MPdist attempts to cluster signals based on similarities without the limitations of Euclidian or DTW measures [17]. The power of MPdist is that it is capable of clustering data based on the number of similar sub-sequences regardless of their order.

2.2.6 Symbolic Aggregate approXimation (SAX)

The Symbolic Aggregate approXimation (SAX) algorithm attempts to improve the performance of other analysis methods by creating a high level, reduced dimension symbolic representation of a data series [15]. SAX differs from other symbolic representation schemes because it makes use of Piece-

wise Aggregate Approximation (PAA) representation as a interim step. PAA provides a time tested means of reducing the sample rate of time series data. PAA works by segmenting the source data with a sliding window and then taking the mean value in each window [15]. The values generated by PAA are then mapped on the alphabet. The size of the alphabet is of concern because if it is too small then the resolution will be to low to properly represent the data. Too large of an alphabet however, will lead to diminishing returns since the computational complexity increases as well. SAX is shown to be most effective with an alphabet size between 5 and 8 letters [15]. Then, just as in written language the alphabets are combined into words that represent the time series data. The words are subsets of the time series data that are representative of specific features or defining characteristics. By transforming time series data into a symbolic format it opens the door for many new toolsets such as natural language processing that were incompatible with time series data in its raw form.

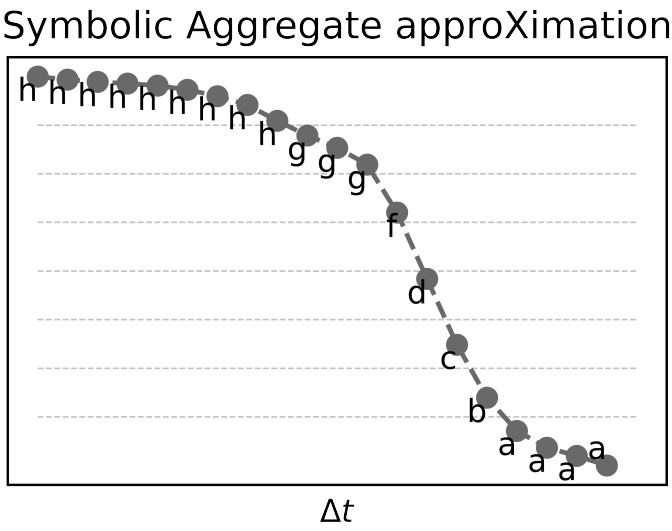

Figure 5: SAX Diagram

2.2.7 Online versus Offline Analysis

It is of importance to discuss the benefits and drawbacks of online and offline analysis techniques. The use case for off-line analysis is when data has already been collected and there is time available. Offline analysis has the benefit of being able to query the data set in its entirety, that is there is no future data that needs to be accounted for since all data is present. Another benefit of offline analysis is that since all of the data is static it is possible to optimize hardware and algorithms for the specific task. The advantage of offline analysis is the ability to perform very complex and accurate analysis. This contrast with online analysis in a number of ways. When analyzing data online the stream is often unknown and must be dealt with and analyzed in

a timely fashion. This time constraint puts strict limitations on the types and complexity of analysis that can be performed. Another difficulty with online analysis is that it is often performed in less than ideal circumstances such as on mobile hardware or other power constrained devices. A major advantage to online analysis is that the data collection and analysis phases are combined and the results of the analysis are delivered far quicker.

2.2.8 Sample Rate Reduction

The easiest way to reduce the compute time of any algorithm is the reduce the number of elements for which it must be run. In other words, we must seek to process the minimum number of data points in a time series that will still maintain enough information to accomplish our goal.

A method for maintaining the maximum information with the minimum number of data points was first introduced by Douglas and Peucker that would vary the sample rate of a time series based on the localized information content [34]. A drawback to Douglas and Peucker's method is that one must now contend with a variable sample rate and also store the Δt information of each data point.

2.2.9 Time Series Research in Automotive

In the world of automotive security research, the internal vehicle communications busses are a large area of interest. The most commonly found in modern vehicles and thus the most commonly researched topic is the Con-

troller Area Network bus, commonly called the CAN Bus for short. Research on the CAN bus almost exclusively centers around the extraction of the data from the individual data frames. While the canned protocol is standardized, the messages contained within each data frame are not. The standard CAN message fame is outlined in Figure 6. For the sake or brevity, only the most critical bytes of the CAN frame will be explained here. The ID field contains the Message Identifier that is used by other nodes on the bus to determine values contained within the data field [35]. The data field size is is determined by the Data Length Code (DLC) and can be anywhere between 1 and 8 bytes in length [35]. The signal data contained within the data field is often aligned to the byte boundaries, for example signal A is in bytes 0-1 and signals B and C are in bytes 2 and 3 respectively. but this is not always the case. Manufacturers are allowed to define the message content and bit offsets as they see fit.

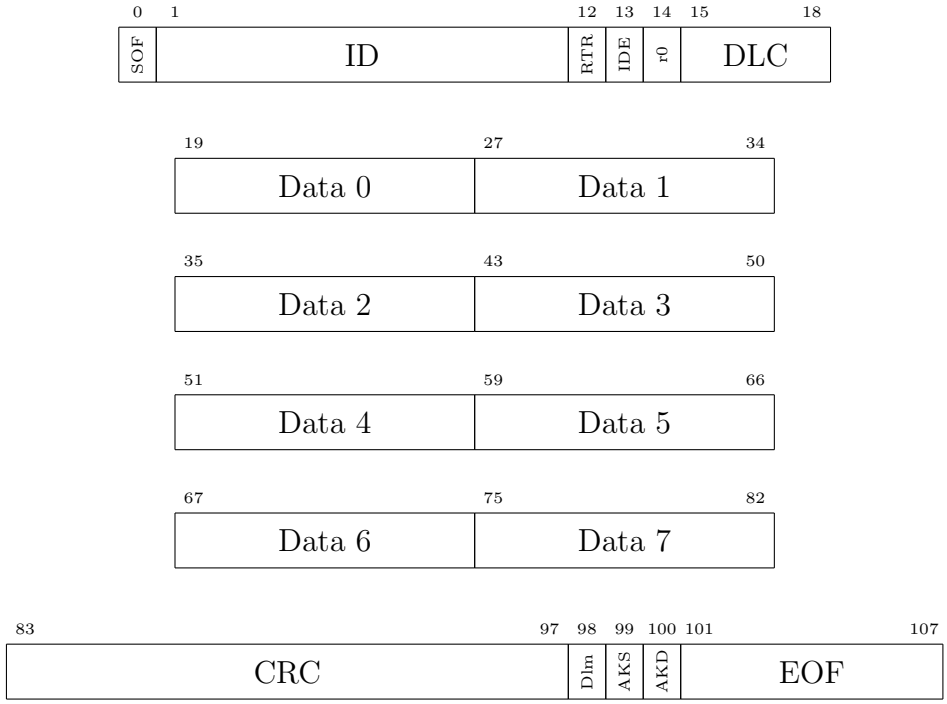

Figure 6: CAN Bit Field Diagram

The "Reverse Engineering of Automotive Data Frames" or READ algorithm works on the principle that the frequency of bit flips is directly related to their position within a numerical bit field [12]. By looking at a histogram of the frequency of bit flips the signal bit boundaries will be visible with the most flips occurring at the least significant bit (LSB) and a gradual decrease in bit flips up to the most significant bit (MSB). What is interesting to note is that this correlation is only present in time series data. Information such as actuator positions or other Boolean data, while represented in the bit fields,

will not be easily detected by READ. In expanding on the READ algorithm the work of Pesé et al. also takes into account externally available data to use as a reference for comparison to the extracted READ signals [9]. The first set of external reference signals comes from an On-board Diagnostics interface (OBD). The OBD interface is a standardized protocol for obtaining basic data used in diagnosing vehicle faults. OBD provides access to information such as error codes and faults as well as basic parameters like engine speed, vehicle speed, and coolant temperature [9]. The second set of reference parameters is from an Inertial Measurement Unit (IMU) from within a mobile phone [9]. The IMU provides information on acceleration and rotation in the X, Y and Z axis. The reference data is not only useful for comparing directly to the extracted signals but can also give insight into the classification of signals with similarities between two other signals. The automotive signal classification research efforts described thus far have all been focused on off-line analysis, that is, analyzing log files generated by a vehicle after the fact. By analyzing the logs off-line there is a benefit of being able to use more powerful hardware at the cost of receiving timely results. The work of de Hoog at al. focuses on a solution to the online problem by proposing the use of an LSTM neural network to generate the expected reference signals and compare them with the extracted signals using FastDTW [36][5]. The work presented attempts to ultimately solve a similar automotive data reverse engineering problem proposed by other researchers but offers the use of clustering as a first step towards the goal of classification. To keep a nar-

row scope the work presented applies only to the first step (clustering) in this process. The reason for focusing on clustering before classification is to be able to rule out data that is drastically different and not applicable to a particular label. Through the process of clustering the number of candidates for a given label is reduced.

3 Implementation

The goals of this thesis are to be able to effectively reduce the number of samples in a time series data set without sacrificing the ability to effectively cluster the data and explore other options counter to the assumption that dynamic time warping is useful in clustering automotive data. The ultimate goal in the reduction in data set size is to enable faster processing without modification of existing algorithms or to find algorithms with faster execution times that are comparable or superior to dynamic time warping.

3.1 Methodology

With the aim being to implement a real-time clustering algorithm comparable to MPdist it is useful to first get a baseline for what MPdist is capable of doing when given a large, complete data set. A nonprofit called the Matrix Profile Foundation [37] was formed to implement the work of Dr. Eamon Keogh et. al. the body of research known as the matrix profile includes the previous discussed shapelets, motifs and other time series toolsets developed at UC Riverside. The Matrix Profile Foundation implementations of these tools are in Python and while easy to use they are not the fastest implementations. In order to approach real-time processing a leaner and more efficient language must be utilized. Google's go language chosen for this task, because go is a modern language with simple readable syntax that compiles efficiently and has runtime efficiency orders of magnitude better than Python while not as

fast as C/C++. In the spirit of the theme of implementing a real-time system that can be used in the real world, all of the algorithms will be benchmarked on an Nvidia Jetson nano embedded computing device. The Jetson nano was chosen because it is one of the higher performing embedded computing platforms that are still compact enough to be installed within a development vehicle in the hypothetical real-world use case. Because the signals to be analyzed were generated through simulation the Jetson nano will not be installed in a vehicle and will only be used as a way to demonstrate efficiency on resource constrained platforms.

3.2 Data Set Generation

The data set used was generated with IPG Automotive GmbH. CarMaker simulation software. CarMaker allows for the execution of complex simulations that allow for the logging of parameters at a far higher rate than would be possible in real life. Four hours of driving in the real world can be simulated in approximately 10 minutes. The software also allows for a variation in vehicle types and has parameters for physical attributes such as vehicle mass and vehicle power. There is the ability to simulate different driving styles including slow reaction times and very aggressive driving styles. It was important to be able to generate a wide variety of data from various situations in order to be able to test out a solution that is generalized enough to be applicable to real-world scenarios. By varying small attributes such as cruising speed and driving style it was possible to create a vastly different data

sample even when using the same vehicle and driving course. The dataset consists of 1.5TB of data from more than 20 different roads and 12 vehicles. The driving style attributes were varied from leisurely to a highly aggressive driving style.

3.3 Time Series Sample Rate Reduction

Critique of current methods: One of the largest issues that arises when dealing with analysis of time series data is that many of the algorithms have extremely large time and space complexities. DTW for example, has a complexity of $O(n^2)$ for both time and space. The previously discussed FastDTW attempts to initially reduce the sample rate of the data and then increase resolution as the DTW warping path is established [6]. The efficacy of the approximated approach of FastDTW while disputed by Wu et. al [5] has one obvious hole in both Salvador's and Wu's assumptions about resolution and compute complexity reduction. The first assumption relates to the need to use the maximum possible resolution of the time series. It is not necessary to use the maximum possible resolution to get useful information when clustering automotive sensor data. In an extreme example, when comparing the decreasing resolution progressions between Wheel Speed in Figure 7 and Brake Force in Figure 8 it is quite apparent that the two signals will be discernibly different at any resolution. The question then is, what is the minimum viable resolution for two similar signals? This leads to the second assumption, that you must use the highest resolution required by the two sig-

nals to be compared. It shouldn't be assumed that the original sample rate of the time series is what is required to accurately perform clustering. For the purposes of clustering it will be shown that only the minimum resolution of the signal of interest is required to adequately perform clustering.

FFT and Frequency analysis: When attempting to reduce the sample rate of a time series the obvious choice is to attempt using Fourier Transforms and frequency spectral analysis. Automotive sensor data presents a unique challenge in that the signal frequency content to sample length are disproportionate. That is to say there is not a lot of frequency information given a dataset. Often much of the useful data is marked with large periods of little to no change such as when a vehicle is sitting at a stoplight or when cruising on a highway. The vehicle speed may increase or decrease by 5 miles an hour but the frequency of this oscillation is so slow that it is almost impossible to detect. Regardless, the information contained within a given signal is still subject to the limitations of the Nyquist Rate shown in Equation 1. Given a continuous function $x(t)$ sampled at a rate of f_s Hertz, the Fourier transform of any frequency $X(f)$ that is greater than or equal to $\frac{1}{2}f_s$ will be zero:

$$X(f) = 0 \quad \forall \quad |f| \geq \frac{1}{2}f_s \tag{1}$$

Because the usual methods for determining frequency content and thus minimum sample rate are not useful in this application another method must

be determined. Figure 7 shows that quite a bit of information can be removed from the signal without much obvious distortion. On the other hand, Figure 8 demonstrates that almost the maximum resolution is necessary before all information is lost. What is interesting to note is that Figure 7 contains information throughout the entirety of the data set while Figure 8 only contains small amounts of information at very specific points. While Figure 7's data set sample rate can be reduced evenly throughout it might still be possible to remove much of the useless data in Figure 8's set by removing the large sections of zero value data.

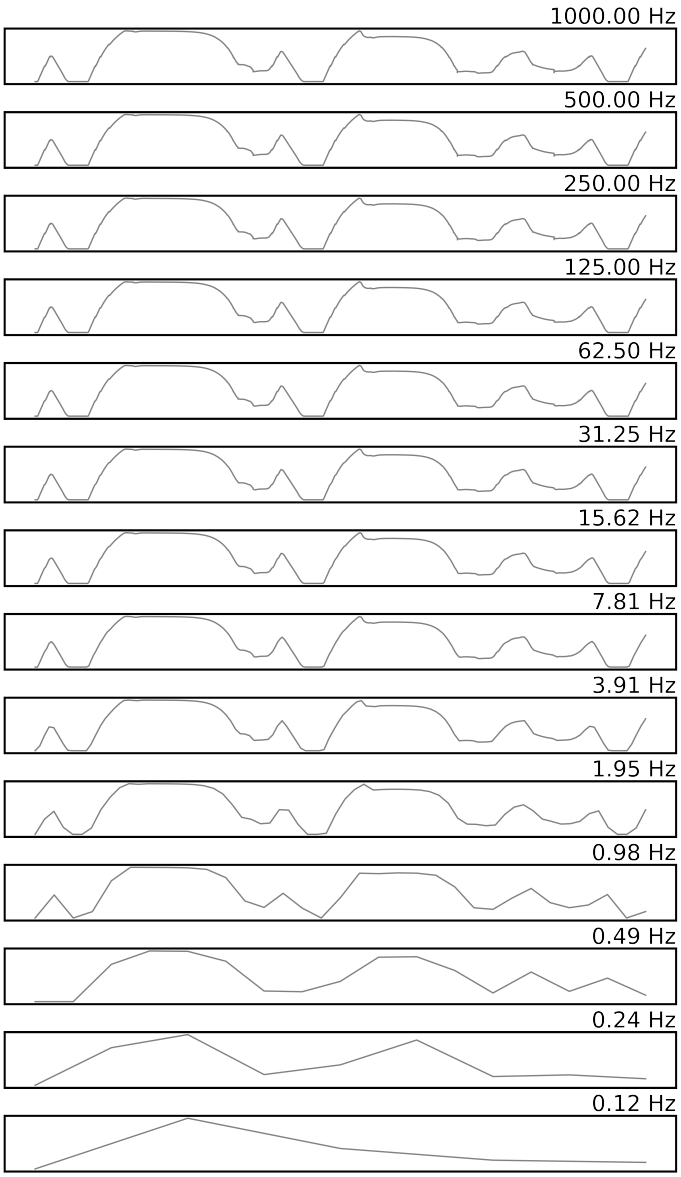

Figure 7: Wheel Speed Data at Decreasing Sample Rate

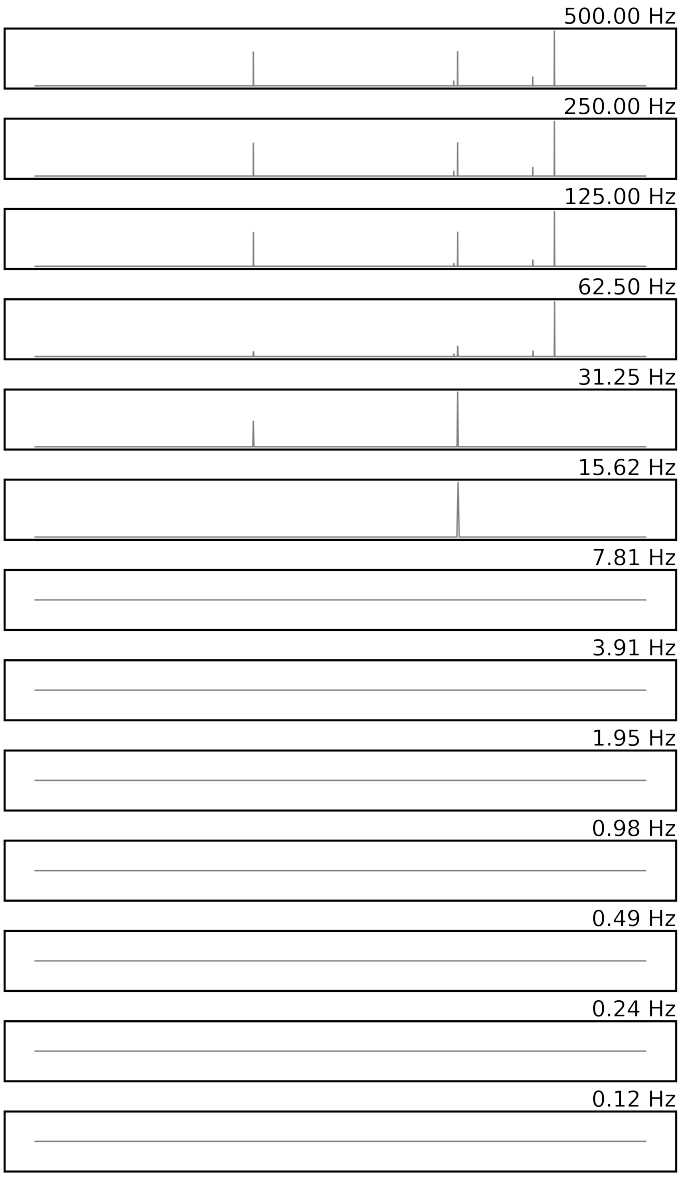

Figure 8: Brake Force Data at Decreasing Sample Rate

Self Distance: The concept of time series *Self Distance* is designed to give a sense of how much data can be removed before a time series loses most of its meaning. Self Distance is intended to be utilized as a preprocessing step, before further analysis between different signals, that will optimize a data set's sample rate to the lower resolution, while still yielding the same performance as the original sample rate. When implementing Self-Distance, a benefit of measuring a signal against itself is that there will be no time shift and so Euclidean distance is acceptable and removes the need for the quadratic complexity of Dynamic Time Warping. The previous statement refers to DTW as a step within the determination of Self-Distance only and not any further comparison of different signals. Self Distance has the benefit of having $mO(n)$ time complexity, where n is the length of the data set and m is the number of iterations required to reach the desired de-resolution percentage, $d_{\%max}$. Equation 2 defines m as the maximum number of iterations where d_m is less than d_{max}. The percentage, $d_{\%max}$ is selected by the user based on the performance required for the application and is somewhat arbitrary. The lowered resolution signal's Euclidean distance is measured against the full resolution signal and the distance is observed. In Equation 3 the Full Self Distance is defined as the sum of the Euclidean distances from a straight line along the x axis at $y = 0$.

$$m := \lfloor d_{\%} \rfloor = d_{\%max} \tag{2}$$

$$d_{full} = \sum_{t=0}^{n} |S_t| \tag{3}$$

Equation 4 defines the Self Distance at iteration m as the absolute value of the euclidean distance between time series S and it's lower resolution version, M.

$$d_m = \sum_{t=0}^{n} |S_t - M_t| \tag{4}$$

Equation 5 defines the distance percentage, $d_{\%}$, as the full resolution distance, d_{full}, over the distance at the current iteration, d_m.

$$d_{\%} = \frac{d_{full}}{d_m} \tag{5}$$

Figure 9 demonstrates that the Euclidean and DTW Self Distances are comparable for a given resolution. Due to DTW's ability to warp it does demonstrate pathological warping and thus the greater distance given a resolution but the trends are similar. While the example shown in Figure 9 demonstrates an exponential curve which is to be expected given that the reduction in resolution also follows an exponential curve of 2^n, many of the

signals do not follow such a curve. One such example is that shown in Figure 8. As can be seen by the almost immediate loss of information the distance increases and tops out almost immediately.

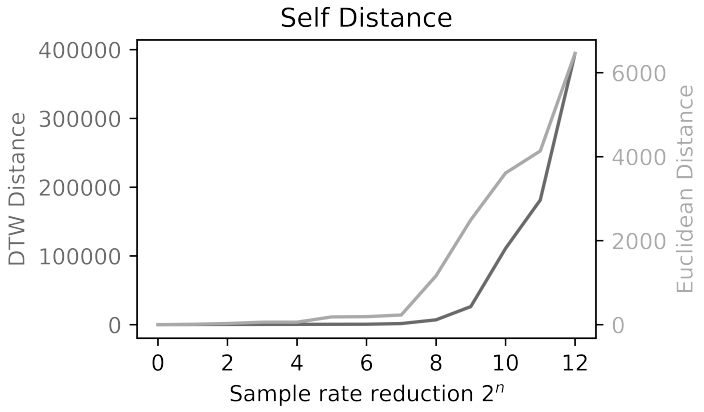

Figure 9: DTW on Original Signal at Decreasing Sample Rate

The effect of the same analysis performed in Figure 9 is shown in Figure 10 and Figure 11. The analysis was however, performed on a sine wave with varying cycles. The Nyquist rate is illustrated by a vertical black bar. It is interesting to note that on a pure sine wave the Euclidean distance oscillates around a fairly linear increase in distance as the resolution decreases. The wild oscillations of the Euclidean distance are in stark contrast to the smooth exponential curve of DTW.

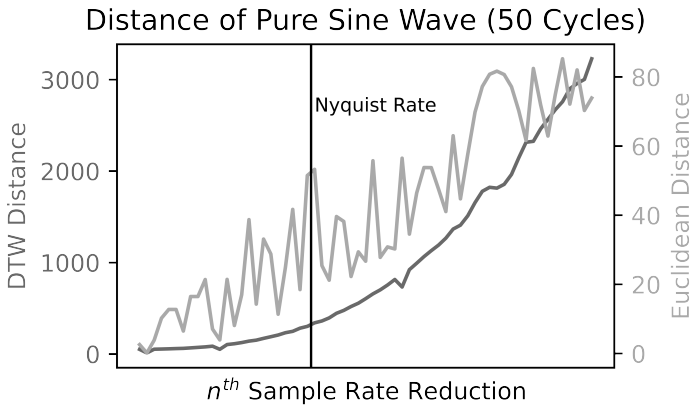

Figure 10: 50 Cycles of Sine Wave at Decreasing Sample Rate

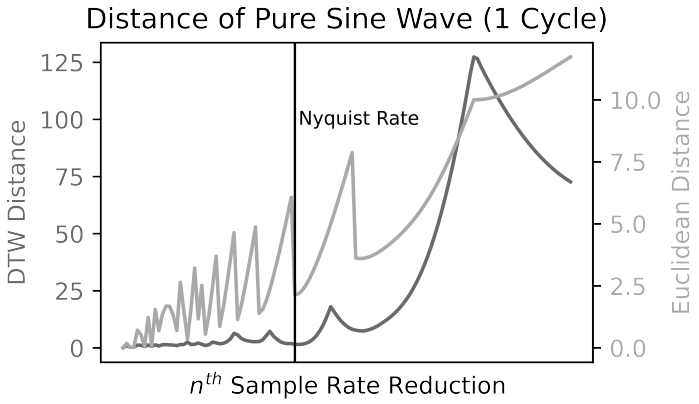

Figure 11: Single Cycle of Sine Wave at Decreasing Sample Rate

Ramer–Douglas–Peucker Algorithm: Another method is to use the Ramer–Douglas–Peucker (RDP) Algorithm for reducing the sample rate of the signals in question. Figure 12 demonstrates just how much of the original time series profile can be preserved using RDP as samples are removed. RDP

works by first drawing a straight line between the first and last points. Next, the point furthest from the line is found and a new line is drawn from the first point to the furthest point. Any points along this new line that fall within a defined distance, ϵ, will be removed. The process then repeats recursively using the unprocessed portion of the time series [34].

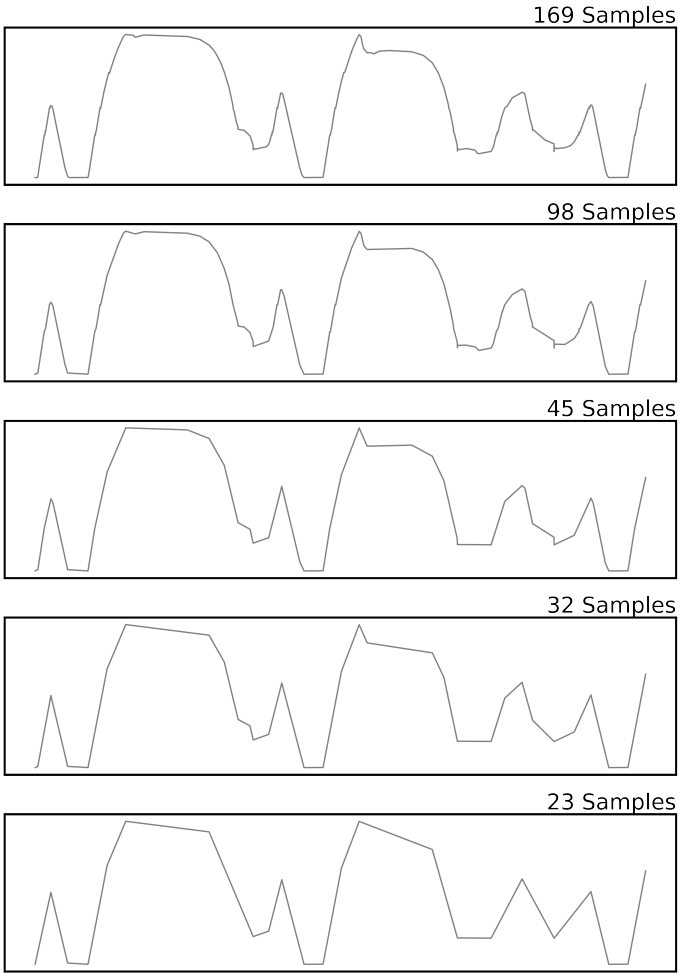

Figure 12: Ramer–Douglas–Peucker Algorithm

Initial Findings: When running preliminary exploration it was unsurprising to find that the time required to compare the DTW distance of one wheel speed sensor to the other 762 signals was reduced quadratically as shown in Figure 13. But what was surprising was the results of the DTW distance measurements. The DTW distances of the reduced resolution samples were lower but this was a product of less elements being summed. When four different resolutions were compared to one another they all show the same approximate ratios between one another. Using the DTW distance in this fashion does not work as an absolute measure but it will work in a relative measure such as clustering.

```
Primary: 2x
Primary ran in 2656.1244 seconds

Primary: 16x
Primary ran in 22.7270 seconds

Reduction: 64x
Reduction ran in 0.6780 seconds

Reduction: 512x
Reduction ran in 0.0967 seconds
```

Figure 13: DTW Comparison Run Times

After reading so many papers focused on the use of dynamic time warping and optimizing its use in automotive applications it is apparent from this basic exercise that the need for high resolution, highly optimized implementations is dubious at best. Because most of the signals available on a vehicle move at such a slow rate the need for extremely high resolution sampling may not be as important as others have implied. Impulse (Sparse) signals are some of the few that will require a higher resolution mainly due to their higher frequency content.

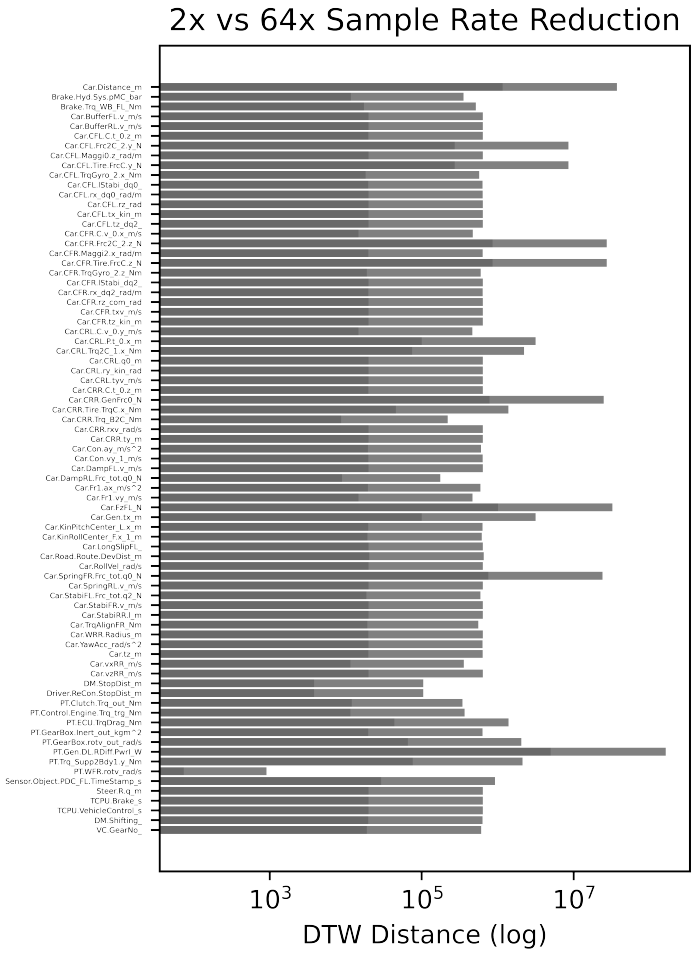

Figure 14: DTW on Original Signal at Decreasing Sample Rate

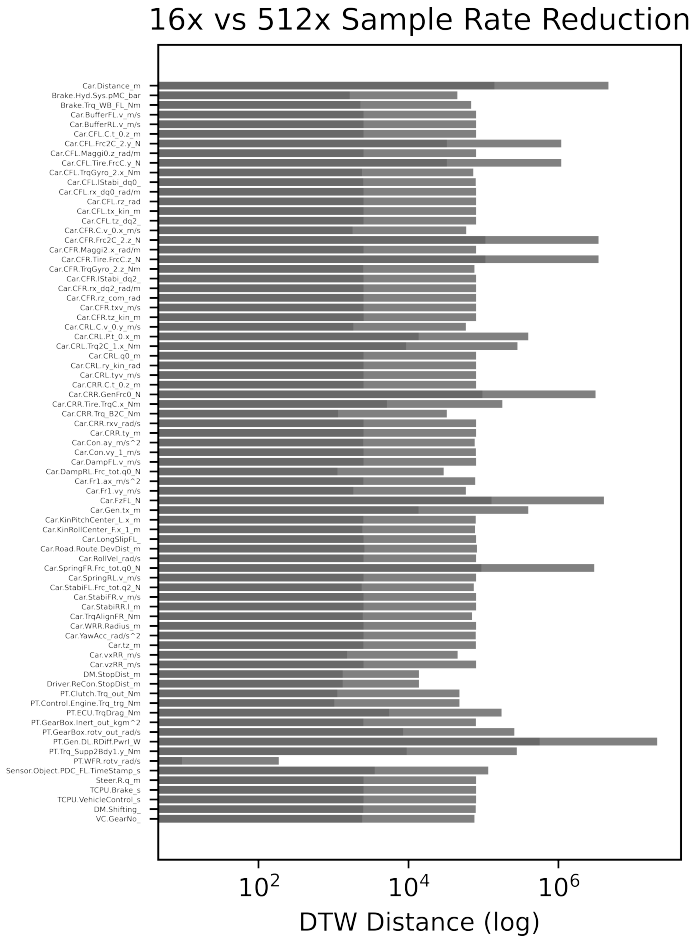

Figure 15: DTW on Original Signal at Decreasing Sample Rate

Discovered Signal Types: Upon analysis automotive sensor signal data can be divided within one of two basic types. The first type will be called meandering time series data. This meandering data is provided by sources such as engine speed, vehicle speed, and torque requests. There is a constant flow of continuous values and the entirety of the data set must be considered.

The second type of data is what will be known as impulse data. Impulse data can be provided by sources such as binary switch position, or by very high speed continuous values like that found in steering wheel position. The reason that signals like steering wheel position are included in the impulse category is because the vast majority of time the steering wheel is either roughly centered or making very minor adjustments. It is only when encountering a corner that the rate of rotation of the steering wheel increases rapidly and then moves back to a centered position.

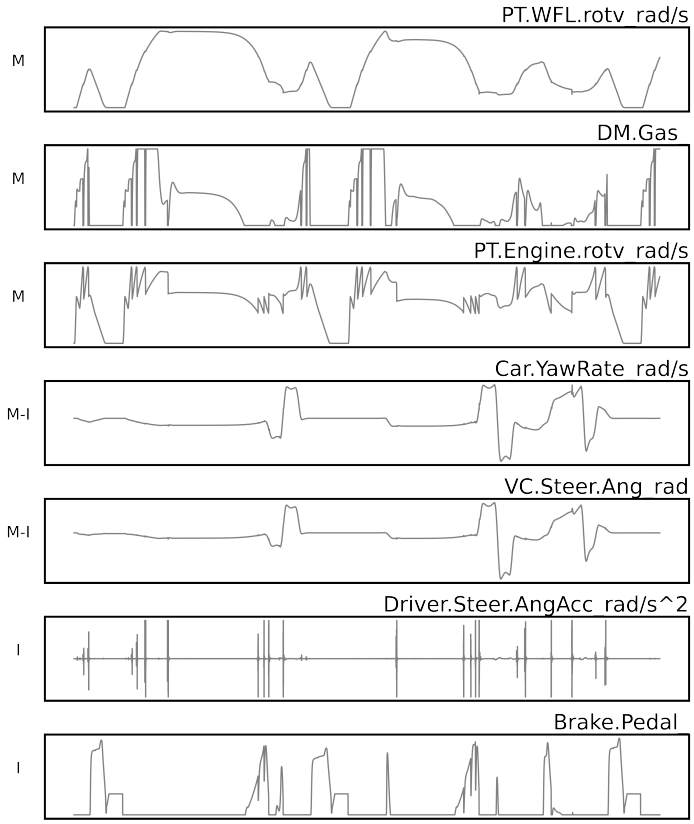

Figure 16: Meandering vs. Impulse Signal Types

By observing the signals in Figure 16 we can see that the top three signals (`PT.WFL.rotv_rad/s, DM.Gas, PT.Engine.rotv_rad/s`) are considered to be of the meandering type because they do not consistently settle at a central value. The next two, (`Car.YawRate_rad/s, VC.Steer.Ang_rad`) are an in-between case that can be treated as either meandering or impulse. Finally the last two signals (`Driver.Steer.AngAcc_rad/s^2, Brake.Pedal`) are treated as impulse signals because they settle on a singular or very narrow

range of values for a large percentage of the time.

3.4 Impulse (Sparse) Time Series

In the case of Impulse or Sparse time series data it can be assumed that the regions with constant values will fall into one of two categories when measured against other signals. In the first case, the second signal will also exhibit identical stable characteristics and thus the distance will be 0 or extremely small. In contrast to other methods of analyzing Sparse time series such as AWarp developed by Mueen et al.[38] it is not necessary warp the Impulse events much beyond their position in time since most related automotive signals will appear in approximately the same time as one another. In the second case, the second signal will not follow the first and the distance will be large. By using only the dynamic regions and discarding the stable regions within an Impulse signal adequate correlation for use in clustering can be obtained. After each sub-region is extracted from the larger data set it is then treated as continuous or what we will call meandering data.

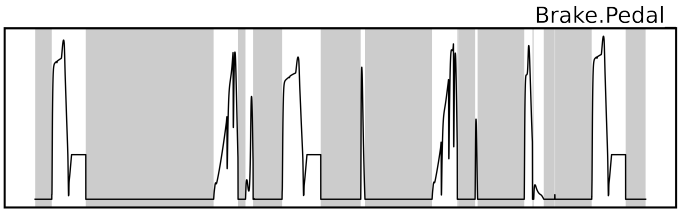

Figure 17: Impulse Masking

One place where RDP shines is in the case of sparse data. RDP is able to remove all points that fall within a specified distance, σ, of two other points. Figure 18 shows a prake pedal position signal. In this figure the large regions of inactivity, where the driver is not applying any brake force, would normally still contain data points. The remaining RDP data points are denoted by the "X"s in Figure 18 and it can be observed that the regions where no brake is applied are in fact devoid of points. The distance between points in the RDP processed signal are shown in Figure 19 and show how there are large spikes of removed data points that correspond to the regions of no applied braking.

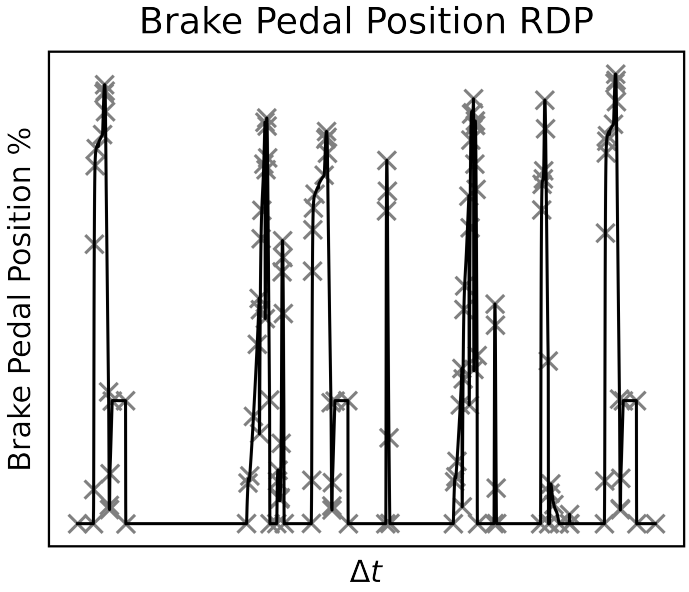

Figure 18: RDP Points

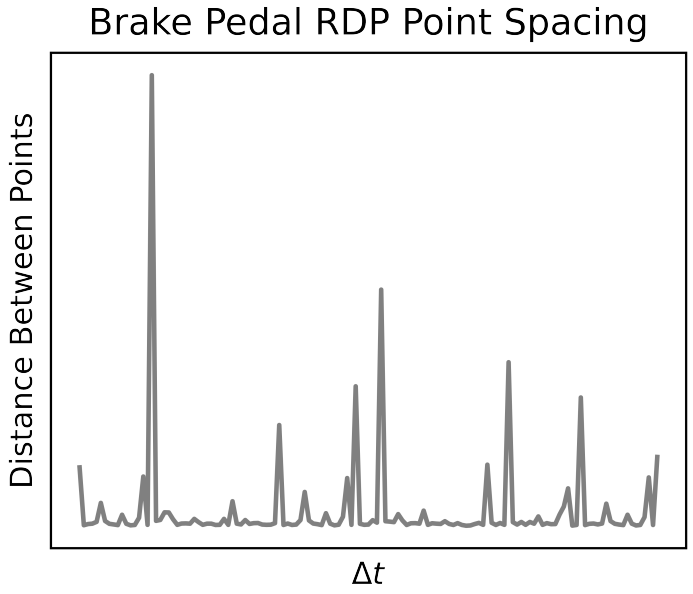

Figure 19: RDP Point Spacing

Figure 17 illustrates the masking concept on the Brake Position signal. The common time series definition is given in Equation 6 where T is the index set based on time, Y the time series data, and t is an individual time point. Equation 7 describes the logic behind masking these regions. For a time series Y any continuous sub-series, X_{mask}, of Y greater than the minimum length l_{min} where the sum, Σ, of the absolute value of each element is less than a given threshold value, y_{th} will be masked off from further analysis.

$$Y = (Y_t : t \in T) \tag{6}$$

$$X_{mask} \in Y : \sum |X_{mask}| \leq y_{th} \tag{7}$$

3.5 Meandering Time Series

The meandering data is either defined as an entire time series that does not contain any static regions subject to being masked or as the individual sub-regions that are the product of masking a larger time series. The data is then reduced in resolution and measured against the original full resolution signal. The resolution is continually reduced until the Euclidean self distance exceeds a specified value. At this point all other signals are reduced to the same resolution as our test signal.

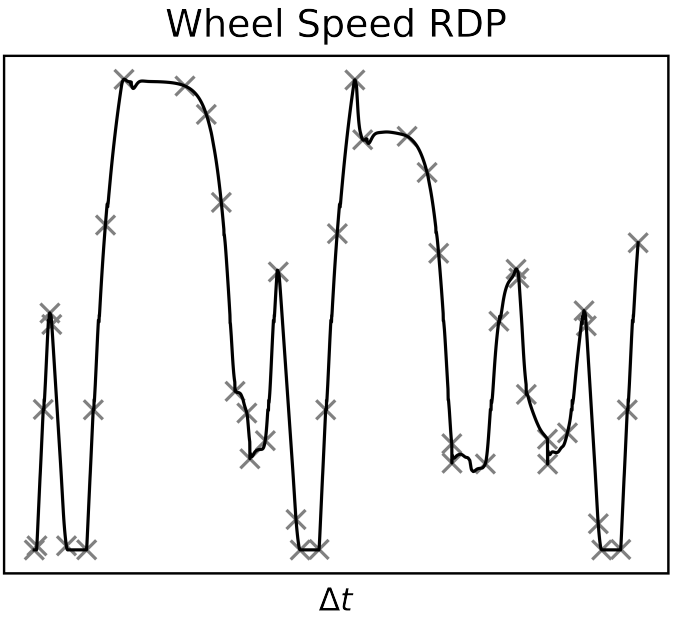

Figure 20: RDP Points

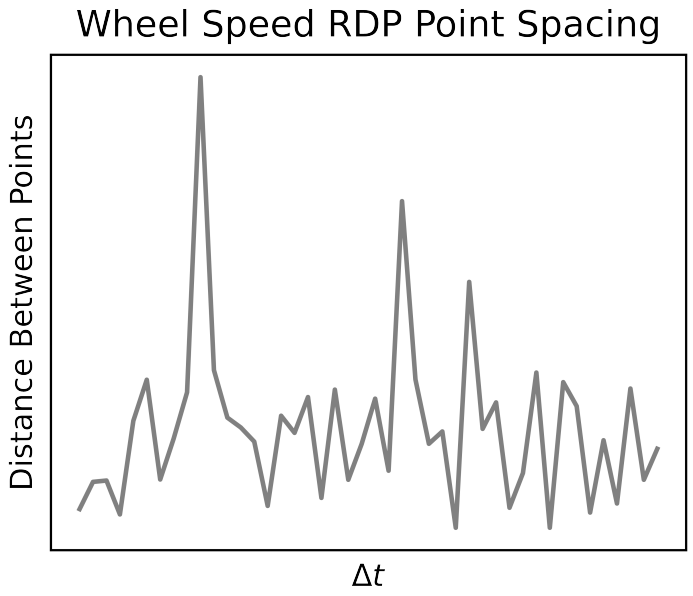

Figure 21: RDP Point Spacing

In contrast to Figure 18 and Figure 19, Figure 20 shows the points selected by the RDP algorithm along a meandering signal type and that they are more evenly spaced when compared to that of the impulse data. Figure 21 backs up this observation with the lack of isolated massive spikes.

3.6 Algorithms

The segmentation algorithm, first introduced in Figure 17, responsible for removing large areas devoid of information is described in Algorithm 1. Algorithm 1 is not revolutionary by any means but it is new to the processing of impulse automotive signals. The first step shown in line 8 of Algorithm 1, is

processing the incoming signal data is to mask the regions of the signals that are uninteresting or otherwise devoid of useful data. The next step shown in line 5 of Algorithm 1 describes the algorithm used to segment the regions of interest.

Algorithm 1 Segmentation and Masking

Input:

Time series T of length n
Minimum mask length, l
Mask threshold value, v

Output:

Array of ≥ 1 time series, *data* returned as answer.

```
1:  q ⇐ 0
2:  N ⇐ n − l
3:  i ⇐ l
4:  while i ≤ N do
5:      if Sum of T[N − l] : T[N] > v then
6:          Append T[N − l] : T[N] to data[q]
7:          i ⇐ i + l
8:      else if Sum of T[N − l] : T[N] ≤ v then
9:          while Sum of T[N − l] : T[N] ≤ v do
10:             i ⇐ i + l
11:         q ⇐ q + 1
```

After the regions of interest are segmented then the minimum resolution that maintains most of the information is determined. Since Euclidean distance requires a one to one mapping of points in each series, it is necessary to use linear interpolation on the reduced resolution signal in order to have an accurate distance measurement. Algorithm 2 puts a lower limit on the minimum possible resolution, m, to avoid excess processing of low information signals. Maximum self-distance, m is a tunable parameter and must be

selected by the user.

Algorithm 2 Sample Rate Reduction

Input:

Time series T of length n
Minimum allowable resolution, m
Maximum self-distance, d

Output:

Value $minRes$ returned as answer.

1: $i \Leftarrow 1$
2: $Distance \Leftarrow 0$
3: **while** $Distance \leq d$ AND $\frac{n}{i} \geq m$ **do**
4: $\quad tempSeries \Leftarrow$ every $i-th$ value of T
5: $\quad tempSeries \Leftarrow tempSeries$ linearly interpolated to n values
6: $\quad Distance \Leftarrow$ Euclidean distance between T and $tempSeries$
7: $\quad i \Leftarrow 2i$

As stated previously, the time complexity of Algorithm 2 is $mO(n)$ with n the length of the time series and m the number of iterations required to minimize the resolution. A comparable algorithm, Symbolic Aggregate approXimation (SAX) has a time complexity of $O(n)$ but lacks any way to self scale it's parameters. SAX alphabet and word length must be hand chosen for best results. Self Distance by comparison, only requires a maximum distance parameter to be set and the algorithm will automatically adjust based on the individual data set. The Ramer-Douglas-Peucker algorithm (RDP), while having promise, is not a competitor to Self Distance because of it's $O(n^2)$ time complexity. Self Distance demonstrates it's strengths against similar algorithms by enabling an auto tuning scheme and keeping time complexity to a minimum.

3.7 Viability

A stated aim of this thesis has been to determine viable of automotive signals specifically using onboard embedded hardware. The author has an aversion to using Python due to its potential for bloat and slow running algorithms. The initial intent was to utilize Golang for runtime reduction and only use Python for exploration of an algorithms properties. Python is a perfect choice for this type of exploration due to its simple syntax and wide array of libraries that make quick work of any specific task no matter how niche. It was however, quickly found that as with most software projects technical debt accumulates and is hard to pay off. Due to the large amount of work that had already been done in Python it made sense to try to find a way to speed up certain operations in the language. A well-known solution for this is Cython [39]. Cython is a C optimizing compiler for Python that facilitates the execution of Python compatible modules at compiled C speeds.

Figure 22 demonstrates the speed up in the Raymer Douglas figure algorithm. The first runtime is a Python native implementation [40] utilizing the Python3 interpreter and the second runtime is a Cython optimized implementation [41]. The data being processed represents a sample time of 33.6 seconds and 763 individual signals. It is obvious that the issue lies with being able to perform DTW on a full signal set in real time.

```
Python RDP ran in 13.3503 seconds
Cython RDP ran in 0.0020 seconds
```

Figure 22: Python vs Cython RDP Run Times

Figure 23 also demonstrates a slightly less impressive, however still substantial speed improvement with DTW [42][30]. Cython can be compiled with the arm64 architecture and is thus a valid option for an automotive embedded system.

```
Python DTW ran in 162.3225 seconds
Cython DTW ran in 60.5227 seconds
```

Figure 23: Python vs Cython DTW Run Times

3.8 DTW Sample Rate

The minimum viable sample rate for performing DTW will be measured against the self distance to determine a fast method for reducing the number of data points requited to perform DTW. It should be noted that the standard for what will be considered acceptable performance of DTW is somewhat arbitrary. A threshold of 90% performance of full-scale DTW will be used.

3.9 Scoring

Since, when clustering we only care about relative distances and not absolute distances or labels the top 5% of results from each of the experiments will be scored against the top 5% of full resolution DTW and Euclidean distances. In the case of the dataset currently being used, the top 5% of results is approximately 33 different signals to be compared. The scoring will also only include whether the signals appear on the list not their position, so two algorithms may have a 100% match but not rank all of the top 33 signals in exactly the same order.

4 Conclusion and Future Work

4.1 Conclusion

One quite important discovery made by sheer happenstance during the course of this thesis is a refutation of an implicit assumption made by many of the researchers working in the field of automotive signal reverse engineering. The vast majority of researchers are focusing on real-time high speed methods when in actuality the information content within automotive signals is so slow that such large time is required to gather the information it makes little sense to pursue a real-time reverse engineering capability. The true value of this data is in optimizing storage of as large amount of sample data as possible for later analysis. This analysis may be done onboard a vehicle in the case of a supervisory cybersecurity appliance or off-board but the fact still remains the same that an extremely large amount of sampling time is required to be useful for clustering.

4.1.1 Euclidean Distance for Automotive Time Series Clustering

For the vast majority of clustering applications performed in this thesis dynamic time warping was almost indecipherable from Euclidean distance. When performing the rankings between the Euclidean distance top 5% and dynamic time warping top 5% the matches were greater than 90% similarity on almost all occurrences. A few outliers existed such as very sparse data,

for example a brake light switch that is a Boolean value and only activated for a few moments at a time. However it is interesting to note that of these outliers the top two or three matches were always consistent and the lower ranking matches can be explained by DTW's need to force fit data through extreme warping. It is important to note that the dynamic time warping performed on this data was not constrained anyway, that is to say the warping path was a full classic DTW cost matrix. Figure 24 demonstrates that as the sample length becomes longer the results of Euclidean and dynamic time warping distances will converge.

Data Length (s)	DTW Top 5% Matches (33 Signals)
≤ 60	63.6%
> 60 - ≤ 180	90.9%
> 180	100%

Figure 24: Euclidean Distance and DTW

4.1.2 Self Distance and DTW

Self distance proved useful in reducing the sample rate of time series data. What is interesting to note is that meandering time series are able to be reduced in resolution with a gradual increase in self distance. Impulse time series data however, will tend to explode or increase distance exponentially. This is naturally due to the sparsity of information within the signal. The remedy for this has been proposed as the segmentation of the signals that

then removes the large gaps in information. Looking at Figure 25 we can see that up until approximately 10% self-distance, the full resolution and reduced resolution signals are identical to the DTW algorithm. At around 20% self-distance the top five matches drops to 72.7%. The experimental limit for self distance and DTW accuracy is approximately 12%, when the matches are above 95% accuracy.

Self-distance %	DTW Top 5% Matches (33 Signals)
0	100%
5	100%
10	97.0%
20	72.7%

Figure 25: Self-distance and DTW

4.1.3 RDP for Clustering

The results of using RDP for dynamic time warping and Euclidean distance optimization were unimpressive. Since RDP is using an optimized subset of the time series, it did demonstrate a more optimized way of reducing data points and keeping self distance to a minimum. Feeding RDP into the dynamic time warping algorithm did not provide any more useful information than simply using Euclidean distance.

4.1.4 Real-time Implementation

The Euclidean and dynamic time warping operations performed on the data were benchmarked against the simulated real-world sample time to determine if it would be possible to perform this analysis in real time. It is assumed that the processing will be done on dedicated hardware that has no other computational overhead. The benchmarks are only measuring execution time. Data retrieval and storage times are neglected. Figure 26 shows the results of these calculations performed on Nvidia Jetson Nano development board. The real-time percent metric is based on 100% being equal sample time and processing time, where 125% for example would be 25% longer processing than sample time. As demonstrated in Figure 26, both Euclidean distance and RDP are viable options for an embedded system such as the Jetson Nano. Dynamic time warping comes close to real-time feasibility but just isn't quite there. An argument could be made for the fact that a vehicle is not driving 24 hours a day and so the duty cycle of drive time versus processing time is skewed in favor of processing time, while the vehicle is sitting in a parking lot for example.

Operation %	Percent Real-time Speed
Euclidean	7.2%
RDP	20.4%
Cython (10% Self-distance) DTW	104%

Figure 26: Self-distance and DTW

4.2 Future Work

Future work in this field should be focused on the efficient collection and storage of large amounts of time series data. Ideally, this data would be used in an application such as a Variational Auto-encoder that would then be fed real-time data. The use of the VAE would be of great benefit in a number of applications. One obvious application would be anomaly detection that would be capable of predicting sensor failure or malicious actors. A second valuable use would be in the form of a predictive observer system, this would assist in anticipating hard braking or other safety critical events. A third use would be in the ability to generate highly accurate simulated sensor data model after real-world conditions. The fourth and most applicable to furthering the work of this thesis is the clustering and labeling of signals based on their similarity to a reference signal supplied by a VAE. The flowchart in Figure 27 illustrates the function of such a system. The input would require a few labeled signals to be fed into the VAE. The VAE will then generate a larger number of labeled signals. Each unknown signal will then be measured

against the VAE generated ones and if a match is found then a label will be applied. This method will allow for the bootstrapping of signal identification since as each real signal is identified it can then be feed into the VAE to generate further labeled reference signals.

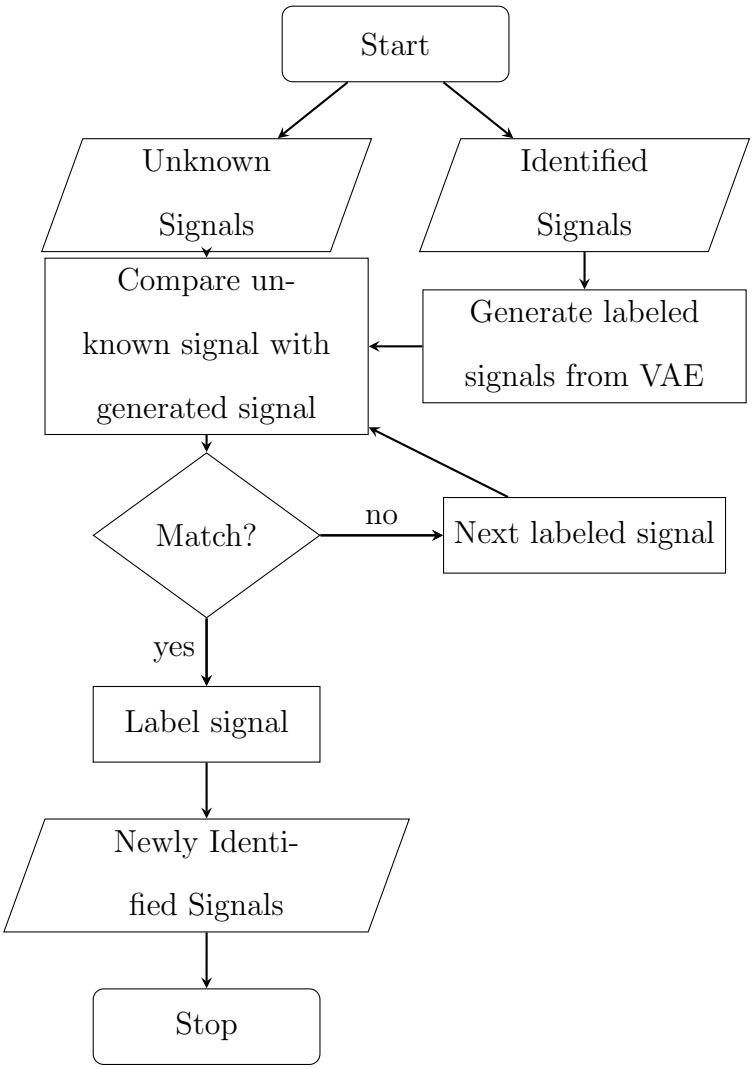

Figure 27: Future Work - Classifying and Clustering with VAE

Currently, the simulation data such as that created by IPG is generated using complex and intricate models that emulate the input and output of every individual system on a vehicle. By capturing enough data from a fleet of vehicles it would be possible to create a VAE that would be able to predict the sensor states for any given condition without having to explicitly tune every individual parameter on a model.

References

[1] Patrick Schäfer. Scalable time series classification. *Data Mining and Knowledge Discovery*, 30(5):1273–1298, 9 2016.

[2] Pjotr Roelofsen. *Time-series clustering*. PhD thesis, Vrije Universiteit Amsterdam, 2018.

[3] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

[4] Shima Imani and Eamonn Keogh. Matrix Profile XIX: Time series semantic motifs: A new primitive for finding higher-level structure in time series. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2019-Novem:329–338, 2019.

[5] Renjie Wu and Eamonn J. Keogh. FastDTW is approximate and Generally Slower than the Algorithm it Approximates. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 3 2020.

[6] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 10 2007.

[7] Jesin Zakaria, Abdullah Mueen, Eamonn Keogh, and Neal Young. Accelerating the discovery of unsupervised-shapelets. *Data Mining and Knowledge Discovery*, 30(1):243–281, 2016.

[8] Charlie Miller and Chris Valasek. Adventures in Automotive Networks and Control Units. *Technical White Paper*, page 99, 2014.

[9] Mirco Marchetti and Dario Stabili. READ: Reverse engineering of automotive data frames. *IEEE Transactions on Information Forensics and Security*, 14(4), 2019.

[10] Mert D. Pesé, Troy Stacer, C. Andrés Campos, Eric Newberry, Dongyao Chen, and Kang G. Shin. LibreCan: Automated CAN message translator. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 2283–2300, 2019.

[11] Jens de Hoog, Nick Castermans, Siegfried Mercelis, and Peter Hellinckx. Online Reverse Engineering of CAN Data. pages 776–785. 2020.

[12] Mirco Marchetti and Dario Stabili. READ: Reverse Engineering of Automotive Data Frames. *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, 14(4), 2019.

[13] Hoang Anh Dau, Diego Furtado Silva, François Petitjean, Germain Forestier, Anthony Bagnall, Abdullah Mueen, and Eamonn Keogh. Optimizing dynamic time warping's window width for time series data min-

ing applications. *Data Mining and Knowledge Discovery*, 32(4):1074–1120, 7 2018.

[14] Jessica Lin, Eamonn Keogh, and Wagner Truppel. Clustering of streaming time series is meaningless. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03*, page 56, New York, New York, USA, 2003. ACM Press.

[15] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 8 2007.

[16] Lexiang Ye and Eamonn Keogh. Time series shapelets: A new primitive for data mining. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–955, 2009.

[17] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, Amirali Darvishzadeh, and Eamonn Keogh. An ultra-fast time series distance measure to allow data mining in more complex real-world deployments. *Data Mining and Knowledge Discovery*, 34(4):1104–1135, 7 2020.

[18] Julia Hartung, Gabriele Gühring, Valentin Licht, and Alexander Warta. Comparing multidimensional sensor data from vehicle fleets with methods of sequential data mining. *SN Applied Sciences*, 2(4):1–13, 2020.

[19] Umberto Fugiglando, Emanuele Massaro, Paolo Santi, Sebastiano Milardo, Kacem Abida, Rainer Stahlmann, Florian Netter, and Carlo Ratti. Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):737–748, 2019.

[20] Chotirat Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. *Third Workshop on Mining Temporal and Sequential Data*, pages 22–25, 2004.

[21] BMW Munich Kirsten Matheus and BMW Munich Thomas Königseder. *Automotive Ethernet*. Cambridge University Press, Cambridge, 2 edition, 2017.

[22] Jurgen Ronald K, editor. *V2V/V2I Communications for Improved Road Safety and Efficiency*. SAE International, Warrendale, PA, 8 2012.

[23] Cisco: 2020 CISO Benchmark Report. *Computer Fraud and Security*, 2020(3):4, 3 2020.

[24] Anthony Bagnall and Jason Lines. An Experimental Evaluation of Nearest Neighbour Time Series Classification. 6 2014.

[25] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 2 1978.

[26] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 2 1975.

[27] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707 – 710, 1966.

[28] Yihang Jiang, Yuankai Qi, Will Ke Wang, Brinnae Bent, Robert Avram, Jeffrey Olgin, and Jessilyn Dunn. EventDTW: An improved dynamic time warping algorithm for aligning biomedical signals of nonuniform sampling frequencies. *Sensors (Switzerland)*, 20(9), 2020.

[29] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar Woods, and Eli. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[30] Johann Faouzi and Hicham Janati. pyts: A Python Package for Time Series Classification. *Journal of Machine Learning Research*, 21(46):1–6, 2020.

[31] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 392–401. Association for Computing Machinery, 2014.

[32] Thanawin Rakthanmanon and Eamonn Keogh. Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676, Philadelphia, PA, 5 2013. Society for Industrial and Applied Mathematics.

[33] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact Discovery of Time Series Motifs. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, volume 1, pages 473–484, Philadelphia, PA, 4 2009. Society for Industrial and Applied Mathematics.

[34] DAVID H DOUGLAS and THOMAS K PEUCKER. ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 12 1973.

[35] Robert Bosch. CAN Specification Version 2.0. *Rober Bousch GmbH, Postfach*, 300240:72, 1991.

[36] Jens de Hoog, Toon Bogaerts, Wim Casteels, Siegfried Mercelis, and Peter Hellinckx. Online reverse engineering of CAN data. *Internet of Things*, 11:100232, 2020.

[37] Andrew Van Benschoten, Austin Ouyang, Francisco Bischoff, and Tyler Marrs. MPA: a novel cross-language API for time series analysis. *Journal of Open Source Software*, 5(49):2179, 5 2020.

[38] Abdullah Mueen, Nikan Chavoshi, Noor Abu-El-Rub, Hossein Hamooni, and Amanda Minnich. AWarp: Fast warping distance for sparse time series. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 350–359, 2017.

[39] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The Best of Both Worlds. *Computing in Science and Engineering*, 13(2):31–39, 3 2011.

[40] Fabian Hirschmann. rdp - pypi.org, 2016.

[41] Ran Bi. crdp - pypi.org, 2019.

[42] Wannes Meert, Kilian Hendrickx, and Toon Van Craenendonck. wannesm/dtaidistance, 8 2020.