NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Geometrical Road Segmentation and Clustering

DIMITRIOS - NIKOLAOS - KONSTANTAKIS

Supervisors : *Ioannis Emiris,* Professor, National and Kapodistrian University of Athens and *Ioannis Chamodrakas*, Member of the Instructional Lab Personnel (EDIP), National and Kapodistrian University of Athens.

ATHENS

APRIL 2018

BSc THESIS

Geometrical Road Segmentation and Clustering

Dimitrios – N - Konstantakis

S.N.: 1115201300079

SUPERVISORS: **Ioannis Emiris, Professor, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens.**
Ioannis Chamodrakas, Member of the Instructional Lab Personnel (EDIP), Department of Informatics and Telecommunications, National and Kapodistrian University of Athens.

ABSTRACT

Region-based analysis is fundamental and crucial in many geospatial-related applications and research themes, such as traffic analysis, human mobility study and urban planning.The current thesis examines various methods for road segmentation and structures that can identify the similarities and the morphology of the generated segments. To achieve these tasks, a research study was conducted for the detection of possible ways that can lead to a prosperous division. Compared to previous studies that focus on segmenting the roads trajectories, in this research the segmentation of roads is supported by tracking the junctions and the variation of curvature among the roads. Data structures, such as hash tables or sets of objects were implemented in order to parcel segments out. The basic criteria of road comparison are stirred up by the application of locality-sensitive hashing and cluster analysis. Moreover, in the process of segments alignment by translation and rotation, we designed a set of proposed methods that examine the deviation in their morphology. Finally, a number of experiments, that retrieve the segments by dividing the roads and determine the suitable heuristic for each classification, was conducted. We compared the findings from our experiments and we concluded that the best results for high-performance roads were achieved when segmentation by junctions was applied. For low-performance or link roads, the curvature heuristic was the one that offered the best results.

SUBJECT AREA: Computational geometry, Nearest neighbor search, clustering

KEYWORDS: road segmentation, morphology, junction, curvature, locality-sensitive hashing, cluster analysis, heuristics

SPECIAL THANKS

For the realization of this current thesis, we would like to thank our supervisors, professor Ioannis Emiris and Dr. Ioannis Chamodrakas for their cooperation and valuable guidance for its completion.

CONTENTS

1. INTRODUCTION

The primary goal of research in computational geometry is to develop efficient algorithms and data structures for solving problems stated in terms of basic geometrical objects, such as points, line segments or polygonal curves. Some of these problems seem so simple that they were not regarded as problems at all until the advent of computers.

In many geospatial-related applications, such as trip planning, urban planning/urban computing and traffic analysis, an urban area is often segmented into sub-regions for in-depth analysis or complexity reduction. In a Geographical Information System (GIS), there are two major models to represent spatial data: vector-based model and raster-based model. Vector-based model uses geometric primitives such as points, lines and polygons to represent spatial objects referenced by Cartesian coordinates, while raster-based model quantizes an area into small discrete grid-cells (cuboids for the 3D spatial objects) indexing all the spatial objects. For example, the vector model of a road network stores a road segment as a polyline (polygone for a circuit road), where a polyline consists of a sequence of shape points, represented by coordinates. The representation of roads as sets of points that form polygonal curves, is the basic concept of this thesis.

Generally, road segmentation is proven a useful approach to represent and compare map information with various applications, such as traffic analysis and anomaly detection. What is the most interesting about this thesis is that we can divide roads into geometrical objects by implementing diverse segmentation rules. We propose two segmentation methods, that either check the existence of intersections or the variation of curvature among the roads. In addition, by introducing a formula based on the morphology and properties of roads, we are able to retrieve uniform and consistent segments. The experimental setting and evaluation of the road segmentation problem and the nearest neighbor search were focused on roads that originate from the map of big cites provided by Open Street Map. The comparison of the results that each segmentation method provides, through its experiments, is based on the amount and the attributes of the produced segments.

Open Street Map is a collaborative project to create a free editable map of the world. that uses a topological data structure, with two core *elements* (also known as data primitives). In this thesis, we take advantage of the basic data primitives, nodes and ways. Specifically, n*odes* are defined as points with a geographic position, stored as coordinates (pairs of a latitude and a longitude) according to *WGS-84*. Outside of their usage in ways, they represent map features without a size. Also, we can describe *ways as* ordered lists of nodes, standing for a poly-line, or a polygon if they form a closed loop. They are commonly used for representing linear features such as streets, and areas.

In order to complete this study, we conduct various experiments retrieving the input map data with the utilization of *OSMParser,* a collection of Java classes allowing to parse raw OSM XML files. Usage of this tool is accomplished by declaring and instantiating a parser object that produces a structure with the map data primitives. The input data offer information about the geodetic coordinates of each location. Typically, Open Street Map records points with their latitude and longitude (x/y coordinates). We can overtake the absence of elevation (height) by implementing a

client-side application that claims elevation data for each location and by constructing a method that converts geodetic coordinates to their respective Cartesian. Both geodetic and Cartesian coordinates have advantages and disadvantages depending on the specific applications. For instance, Cartesian coordinates are more powerful for precisely measuring distances, whereas it requires intensive computation when performing topological analysis. At the other end of the spectrum, the utilization of geodetic coordinates may cause deviation between the results. To surpass this limitation, it is inevitable to develop an application that retrieves elevation data for every map position. The Google Maps Elevation API provides elevation data for all locations on the surface of the earth. Accessing the Google Maps Elevation API is accomplished through an HTTP interface, with requests constructed as a URL string, consisted of a free API key and the geodetic coordinates (latitude / longitude) that identify the locations or path vertices.

In a first phase, by applying various segmentation rules, we can divide roads into multiple segments. We have to point out that we focus on clustering of the road segments that originate from a topological analysis and their associations, i.e. by inspecting the existence of intersections. Contrary to our thesis, other studies, up to the present, dwell on division and clustering of road trajectories. In a second phase, we study data grouping and proximity search algorithms by executing processes based on techniques, such as locality-sensitive hashing and cluster analysis.

Hashing is one of the popular solutions for approximate nearest neighbor search. In general, hashing is an approach of transforming the data item to a low-dimensional representation, or equivalently a short code consisting of a sequence of bits. The application of hashing to approximate nearest neighbor search includes two ways: indexing data items using hash tables that is formed by storing the items with the same code in a hash bucket, and approximating the distance using the one computed with short codes. Locality-sensitive hashing is defined as a technique for grouping objects into 'buckets' based on some distance metric. Objects that are close to each other under the chosen metric are mapped to the same bucket with high probability. As each road segment is defined by sets of Cartesian coordinates, the metric that is applied, is the Euclidean Distance.

Clustering is the unsupervised classification of patterns (observations, data items or feature vectors) into groups clusters. The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. However clustering is a difficult problem combinatorially and differences in assumptions and contexts in different communities has made the transfer of useful generic concepts and methodologies slow to occur. Overall, we can define clustering as the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups. Specifically, the similarity of two objects described by polygonal curves, is measured by the computation of their discrete Fréchet distance. In addition, we present two alignment heuristics for matching two polygonal curves with respect to the Fréchet distance. The problem of alignment two polygonal chains under translation and rotation to minimize their distance has been studied using various proposed methods.

2. BACKGROUND AND RELATED WORK

As we mentioned in the Introduction, road segmentation is proven a useful approach to represent and compare map information with various applications, such as traffic analysis and anomaly detection. By applying various segmentation rules, we can divide roads into segments and achieve significant clustering. In the following paragraphs of this subsection, we will refer to previous works, which have used the segmentation of geometrical objects and clustering, in a number of research fields.

To begin with, we would like to highlight the contribution of a unpublished research conducted by our professors, Ioannis Emiris and Ioannis Chamodrakas. Their study is focused on clustering of geometrical objects and methods to evaluate their performance. The idea of implementing road segmentation by topological features and road associations originates from their research. The organization of the whole plan and the conduction of the experiments are performed by their guidance and consultation.

Gonzalez et al. [11] proposes a novel road network partition approach based on the road hierarchy. Specifically, the road networks are first divided into areas by high level roads, then the partition process is recursively performed for each area. The partition process is implemented by finding the strongly connected components after the removal of the intersection nodes connected to high level roads as well as the terminals of high level road segments themselves.

In this paper [13], an image-processing-based approach to segment urban areas into regions by road networks is reported. Each segmented region is bounded by the high-level road segments, covering some neighborhoods and low-level streets. Typically, road segments are classified into different levels (e.g., highways and expressways are usually high-level roads), providing the research with a more natural and semantic segmentation of urban spaces than the grid-based partition method. It is shown that through simple morphological operators, an urban road network can be efficiently segmented into regions. In addition, a case study in trajectory mining is presented in order to demonstrate the usability of the proposed segmentation method, that implements dilation. The purpose of the dilation operation is to remove the unnecessary details for map segmentation, avoiding the small connected areas induced by these unnecessary details such as bridges and lanes.

Gaffney et al. [14] proposes trajectory clustering algorithm, which mainly focuses on grouping similar trajectories as a whole. They model a set of trajectories using a regression mixture model and use EM (Expectation-Maximization) algorithm to determine the cluster memberships. However, due to the slow convergence of EM algorithm, applying this algorithm for our problem, online clustering requiring an instant response, is not appropriate. In our study, we focus on a clustering algorithm that utilizes road segments described by polygonal curves.

3. WEB APPLICATION FOR ELEVATION DATA

3.1 Description

For the retrieval of the necessary elevation information, a web application was developed. This application behaves as a client that connects to a server and sends HTTP requests. In order to generate the requests, the application needs to parse the geodetic coordinates of each location from a *xml* file. The format of the HTTP request is described in the next section. The web application was developed as a JAVA class that contains methods that perform the functionality described in the following section.

3.2 Request format

In order to access the Google Maps Elevation API, we send requests constructed as a URL string, using latitude/longitude coordinates to identify the locations. Latitude and longitude coordinate strings are defined using numerals within a comma-separated text string and must correspond to a valid location on the face of the earth. Latitudes can take any value between -90 and 90 while longitude values can take any value between -180 and 180. If an invalid latitude or longitude value is specified, the request will be rejected as a bad request. Apart from the values of the geodetic coordinates, the request requires a free API key, provided by Google Maps Elevation API.

3.3 Functionality

The functions of the web application are presented at the following figure.

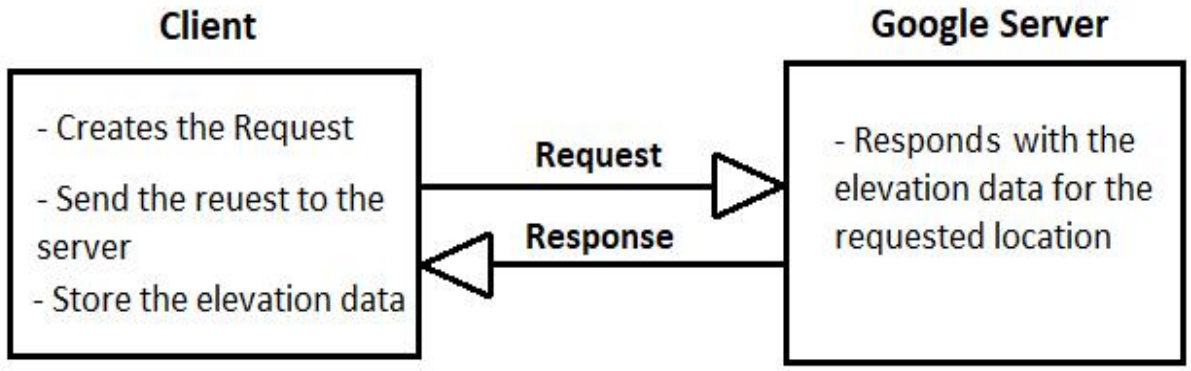


Image 1 : The functionality of the web application

4. COORDINATE SYSTEM

As it was mentioned in the Introduction, Open Street Map uses the *WGS-84* coordinate system, as do most GPS units. A coordinate system is a system which uses one or more numbers, or *coordinates*, to uniquely determine the position of the points or other geometric elements on a manifold such as Euclidean space. Coordinates systems are often used to specify the position of a point, but they may also be used to specify the position of more complex figures such as lines, planes, circles or spheres.

4.1 Geographic - Geodetic coordinate system

A geographic coordinate system is a coordinate system used in geography that enables every location on Earth to be specified by a set of numbers, called geodetic coordinates. In geodetic coordinates, the Earth's surface is approximated by an ellipsoid, and locations near the surface are described in terms of latitude, longitude and height. The Geodetic system uses polar coordinates defined as follows:

Definition of Latitude 4.1.1 Angle north and south of the equator. Positive angles are in the northern hemisphere, and negative angles are in the southern hemisphere. The range of angles is -90 degrees (-π/2 radians) to +90 degrees (+π/2 radians). Points on the equator have a latitude of zero.

Definition of Longitude 4.1.2 Angle east and west of the Prime Meridian. The Prime Meridian is a north-south line that passes through Greenwich, United Kingdom. Positive longitudes are to the east of the Prime Meridian, and negative angles are to the west. The range of angles is -180 degrees (-π radians) to +180 degrees (+π).

Definition of Height 4.1.3 Also called altitude or elevation, this represents the height above the Earth ellipsoid, measured in meters. The Earth ellipsoid is a mathematical surface defined by a semi-major axis and a semi-minor axis. The most common values for these two parameters are defined by the World Geodetic Standard 1984 (WGS-84). The WGS-84 ellipsoid is intended to correspond to mean sea level, although in practice the actual mean sea level varies around the world due to ocean currents, Coriolis effects, and local variations in Earth's gravitational field. A Geodetic height of zero therefore roughly corresponds to sea level, with positive values increasing away from the Earth's center.

4.2 Definition of Earth-Centered, Earth-Fixed (ECEF) Coordinates

In order to analyze the properties of the ECEF coordinate system, we need to point out the definition of the Cartesian coordinate system. Generally, a Cartesian coordinate system specifies the position of any point in three-dimensional space by three Cartesian coordinates. Choosing this system for a three-dimensional space means choosing an ordered triplet of lines (axes) that are pair-wise perpendicular, have a single unit of length for all three axes and have an orientation for each axis.

ECEF is a right-handed Cartesian coordinate system with the origin at the Earth's center, and that is fixed with respect to the Earth. The three axis are defined as:

X: Passes through the equator at the Prime Meridian (latitude = 0, longitude = 0).

Y: Passes through the equator 90 degrees east of the Prime Meridian (latitude = 0, longitude = 90 degrees).

Z: Passes through the North Pole (latitude = 90 degrees, longitude = any value).

Conversion from Geodetic to ECEF

As it was mentioned in the Introduction, we need to convert the geodetic coordinates to the respective Cartesian coordinates (ECEF). An algorithm invented by Olson can accomplish this conversion. The Olson's algorithm is considered relatively straight forward, involving only a few calculations. [17]

Definition 4.2.1 Let ***lat*** be the latitude and ***lon*** be the longitude of a given location measured in radians, and let ***alt*** be the requested elevation. The produced Cartesian coordinates are x,y and z.

1. $N = earthRadius \Big/ \sqrt{\left(1 - e^2 \cdot \sin^2\left(lat\right)\right)}$
2. $x = \left(N + alt\right) \cdot \cos\left(lat\right) \cdot \cos\left(lon\right)$
3. $y = \left(N + alt\right) \cdot \cos\left(lat\right) \cdot \sin\left(lon\right)$
4. $z = \left(N \cdot \left(1 - e^2\right) + alt\right) \cdot \sin\left(lat\right)$

Accuracy

Olson's algorithm is computationally cheap, and unlike some solutions requires no special treatment at the poles or equator. It is also extremely accurate. In his original paper Olson ran a large number of points, regularly sampled in latitude and longitude, and computed the 3D error in meters for each. For the latitude and longitude measurements, the maximum error for any point was 4.44×10^{-16} radians. At the Earth's surface, this corresponds to a maximum position error of 2.8×10^{-9} meters. The maximum elevation error for any point was 4.47×10^{-8} meters. These errors are smaller than the wavelength of visible light, which is accurate indeed.

4.3 Conclusion

Both of these coordinate systems have advantages and disadvantages. The Geodetic system is used for navigation, mapping and GPS applications, and its three components can be intuitively interpreted as representing north/south, east/west, and up/down movements respectively. The ECEF system, however, is more convenient for calculations involving Euclidean geometry and rotation matrices. It is therefore frequently necessary to convert back and forth between the two systems.

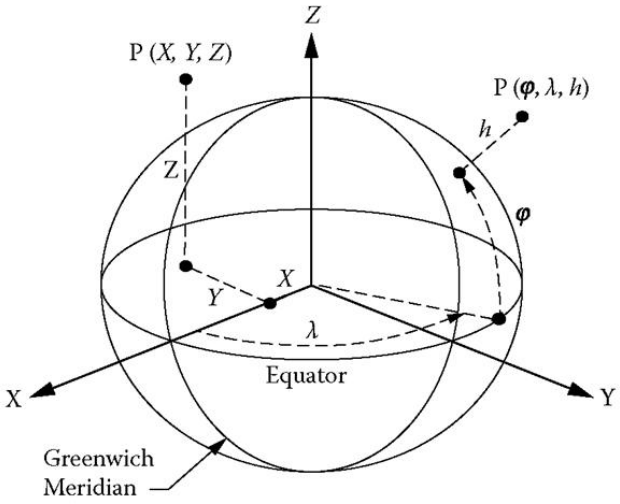


Image 2 : Geodetic Coordinates : latitude(φ), longitude(λ), height(h)

5. HIGHWAYS

A *highway* is any public or private road or other public way on land. It is used for major roads, but also includes other public roads and public tracks. A highway can be distinguished between three basic categories, high-performance,low-performance and link roads. The criteria used to classify the highways are based on the properties and the morphology of roads.

5.1 High-performance roads

Motorway: A controlled-access highway or motorway is a type of highway which has been designed for high-speed vehicular traffic, with all traffic flow and ingress/egress regulated.

Trunk: A trunk road, trunk highway, or strategic road is a major road, usually connecting two or more cities, ports, airports and other places, which is the recommended route for long-distance and freight traffic.

Primary: Primary roads form the major routes between the major urban centers. It is considered as a major highway linking large towns.

Secondary: A secondary road is a highway which is not part of a major route, but nevertheless forming a link in the national route network.

Tertiary: Outside urban areas, tertiary roads are those with low to moderate traffic which link smaller settlements such as villages or hamlets.

5.2 Low-performance roads

Residential: A residential road is a street or road generally used for local traffic within settlement. These roads serve as an access to housing, without function of connecting settlements. Often lined with housing.

Unclassified: Unclassified roads serve for interconnection of small settlements. Unclassified roads have lower importance in the road network than tertiary roads, and are not residential streets or agricultural tracks.

Service: A service road is a local road that runs parallel to a motorway or interstate highway and that provides access to the property bordering it. This is also commonly used for access to parking, driveways, and alleys.

5.3 Link Roads

A link road is considered as a transport infrastructure road that links two conurbations or other major road transport facilities, often added because of increasing road traffic. A link leads to/from a highway (high or low performance) from/to a highway that belongs to the same or a lower class. For example, a motorway-link road may lead from a motorway to a primary road.

6. ROAD SEGMENTATION

Road Segmentation is the process of partitioning a road into multiple segments. The goal of segmentation is to simplify the task of data grouping and change the content of a road into something that is more meaningful and easier to analyze or compare. Generally, road segmentation means to divide roads into parts, or segments that can be defined as polygonal curves in the three-dimensional space. The result of road segmentation is a set of segments that collectively cover the entire highway. Each segment consists of two or more nodes. Each node can be described as a point with Cartesian coordinates $(x, y, z) \in R^3$.

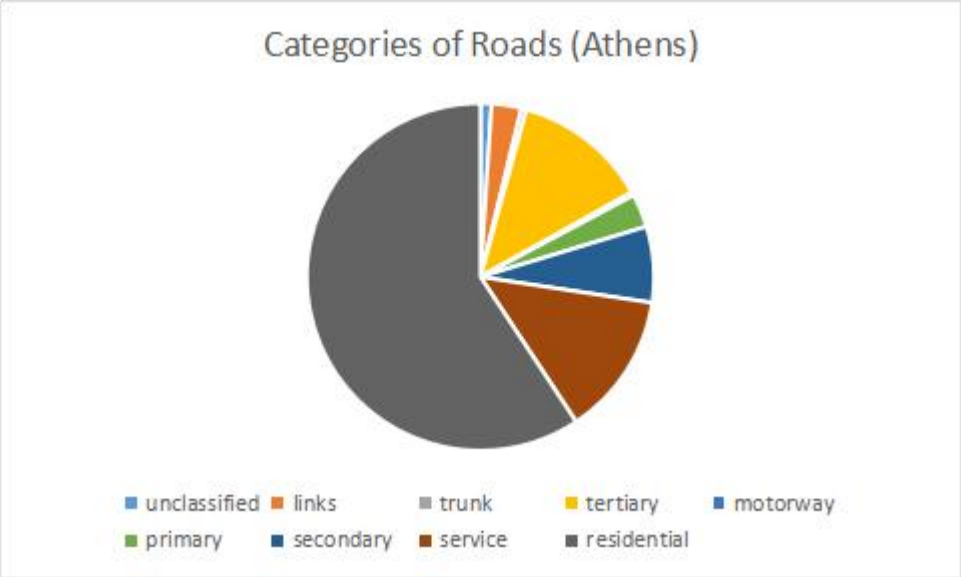


Image 3 : Categories of Roads

In this thesis, we propose two different methods of road segmentation, *segmentation by junction* and *segmentation by curvature*. Both vary in their implementation and are applied for different categories of highways. The criteria used for the selection of the most suitable method for each road classification are mostly depended on the general attributes and morphology of roads.

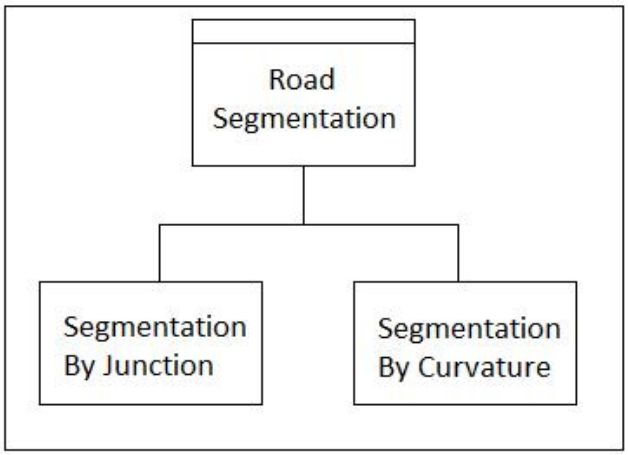


Image 4 : Road Segmentation Analysis

6.1 SEGMENTATION BY JUNCTION

Segmentation by Junction is one of the simplest and most widely used type of road segmentation. Many applications perform segmentation by junction because it is considered as an efficient way to produce the road segments by checking only the existence of intersections among the roads. Especially, in populous and industrial cities with complicated roads network, segmentation by junction can be applied and may lead to a prosperous and much higher level of division of the roads. The nodes that constitutes the produced segments do not display many meaningful variations regarding their content. This fact helps the conduction of the research and prompts better analysis of the results.

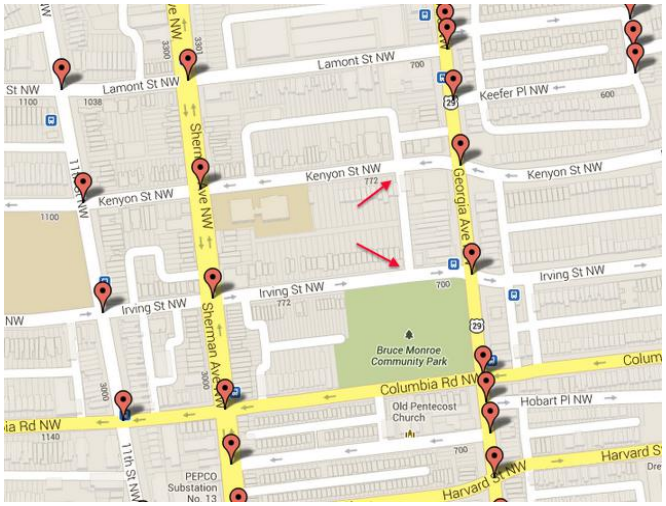


Image 5 : Junctions in a city

Generally, segmentation by junction is applied for roads that are considered as high-performance. Given a highway as input, the developed application, responsible for the segmentation of roads, iterates through each of its nodes and checks if there is an intersection. An intersection (or junction) is occurred when a node is part of two or more roads simultaneously. Our methodology is based on the rule *"After the validation of an intersection a segment is created".*

By applying the above rule, a problem arises. There is some great possibility that the generated segments may include insignificant amount of nodes. This problem could lead to the construction of plentiful segments and the lack of the system's capability to manage the numerous objects. In order to regulate the number of the segments, it is urgent avoiding the generation of small roads that include two or less nodes. In addition, it is inevitable that some highways may not include any intersections. To face this possibility, we invent a formula that takes into account the number of the nodes and the estimated length of the current segment that is under creation.

6.1.1 FORMULA

The following section demonstrates a mathematical equation that is used for regulating the number of the created segments. It involves the application of an upper bound concerning the number of points that a segment may have and its approximate maximum length. It was invented through experimental processes and

various sets of data. Overall, it improves significantly the methods that generate the results.

This mathematical equation contains two conditions that must be fulfilled simultaneously in order to construct a new segment. It is applied as an additional rule to the respective condition used to determine when a segment is generated (junction existence or curvature variation).

Formula Definition: Let P be the set of points and λ be the length of the under construction segment. Also, let *H* be the set of points of the highway which is divided to the segments, μ the mean length of all the highways that belong to the same category and ρ a minor positive number (e.x. ρ = 8). The formula is:

$$|P| > \left(|H|/\left(|H|/\rho + 1\right)\right) \quad and \quad \left(\lambda > \mu\right)$$

6.1.2 Segmentation by Junction - Algorithms

In this paper a set of *road segmentation methods* are proposed, that are based on the segmentation by junction. The proposed methods are independent from the highway content, so these methods can be used in the segmentation stage of verity applications. Also, the fact that the developed methods include multiple independent steps, results in capability of altering each step separately to optimize the overall performance for specific application.

There are three implementations of the *road segmentation method by junction.* The basic difference between the proposed methods is that each one of them alternates in a unique way the rules that must be validated in order to generate a new segment.

6.1.2.1 Algorithm 1 - Segmentation only by Junction

In this section, a segmentation algorithm is presented that is based on the model of intersections among the roads. The proposed algorithm has five main steps. To distinguish between the other two algorithms, in the proposed segmentation method, a segment is generated when a junction is detected. There is no other limitation regarding the conditions that affects the construction of a new segment.

Description: Given a highway as an argument, we want to generate a number of segments. The highway contains a set of Points $P = \{p_1, p_2, \ldots, p_n\}$ in R^3. Each point p_i is defined by the Cartesian coordinates that were generated by the respective geodetic. Every created segment includes a list of segment points S, populated by points of the set P.

Algorithm

1. Define a list of segment points S. If the number of points for the highway is equal to zero, exit.

2. Iterate through the list of points $P = \{p_1, p_2, \ldots, p_n\}$. For each iteration, store the current point p_i.
3. Add the current point p_i to the set S.
4. If there is a junction that includes the current point p_i, create a new segment using the set S. After the successful generation of the segment, clear S and initialize it by adding point p_i. Skip to the next iteration.
5. If p_i is the last point $(i = n)$, create a new segment using S and exit.

6.1.2.2 Algorithm 2 - Segmentation by Junction - Previous Point Inspection

In this section, a second segmentation algorithm is presented that is based on the model of junctions among the highways combined with an additional inspection. The proposed algorithm has six main steps. To differentiate between this proposed method and the one described in the previous section, we proceed to the following acceptance: the junctions are chosen when a segment point belongs to two or more highways and at the same time, the previous segment point must not be part of another intersection. This limitation is used to reduce the quantity of generated segments consisted of insufficient number of points.

Description : As described in the algorithm for the first proposed method, a highway is given as an argument. The purpose is to generate a number of segments. A highway is defined as a polygonal curve that contains a set of Points $P = \{p_1, p_2, \ldots, p_n\}$ in R^3. Each point p_i is defined by the Cartesian coordinates that were generated by the respective geodetic. Every created segment includes a list of segment points S, populated by points of the set P. Also, a Boolean variable named *junction-found* containing the information that the previous point belongs or not to an intersection.

Algorithm

1. Define a list of segment points S. If the number of points for the highway is equal to zero, exit.
2. Iterate through the list of points $P = \{p_1, p_2, \ldots, p_n\}$. For each iteration, store the current point p_i and the previous point p_{i-1}. If i = 0, the value of the previous point is equal to null.
3. This step is used only for the first point of the list. Check the value of the previous point p_{i-1}. If it is not initialized (equal to null), instantiate the set of Segment Points S and populate it by p_i .Set the value of the Boolean variable junction-found as false. Skip to the next iteration.
4. Add the current point p_i to the set S.
5. Check if the current point p_i is the last point of the list $(i = n)$. If so, create a segment using the set S and exit.

6. If there is a junction that includes the current point p_i and at the same time the previous point p_{i-1} is not part of an intersection, a new segment is generated. For this purpose, create a new segment by using the set S. Clear S and initialize it with p_i. Set the value of the Boolean variable junction-found as true.

Skip to the next iteration.

6.1.2.3 Algorithm 3 - Segmentation by Junction combined with Previous Point inspection - Application of Formula

In the last section, a segmentation algorithm is discussed, that is based on the model of algorithm *6.1.2.2* combined with the utilization of the mathematical formula described in section *6.1.1*. The principal difference between the two methods is that in this proposed algorithm, the junctions are not only chosen when a segment point belongs to two or more highways and concurrently the previous segment point is not part of an intersection, but also when some supplemental conditions are verified. These conditions are introduced by the mathematical formula.

The addition of this formula to the basic rules of segment generation aims to confine the possibility of not finding any intersections among the highways. Apart from avoiding consecutive intersections points, the modified rules take into account the number of points P and the current length estimated for the under construction segment. Overall, It is very useful for roads with no junctions.

Algorithm

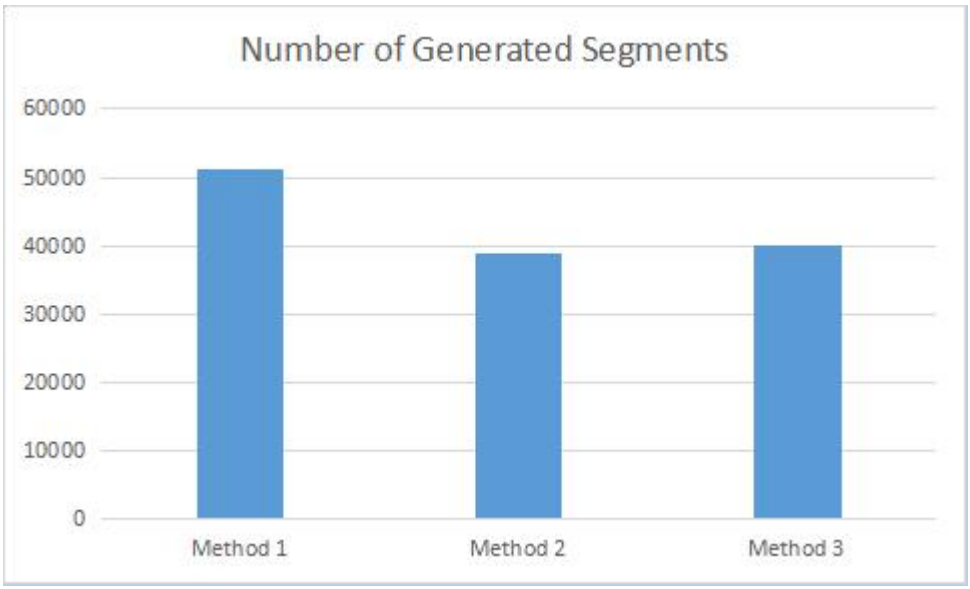
The steps 1 to 5 are the same as the proposed method *6.1.2.2*. The last step is formed as:

6. If there is a junction that includes the current point p_i and at the same time the previous point p_{i-1} is not part of a junction, a new segment is produced. Moreover, if the above condition is not verified, then the number of points that are included in the set S of the under construction segment, combined with their mean length, are taken into consideration. A verification of the previous conditions leads to the creation of a new segment. After a segment has been constructed, clear the set S and initialize it with p_i. Set the value of the Boolean variable junction-found as true.

Skip to the next iteration.

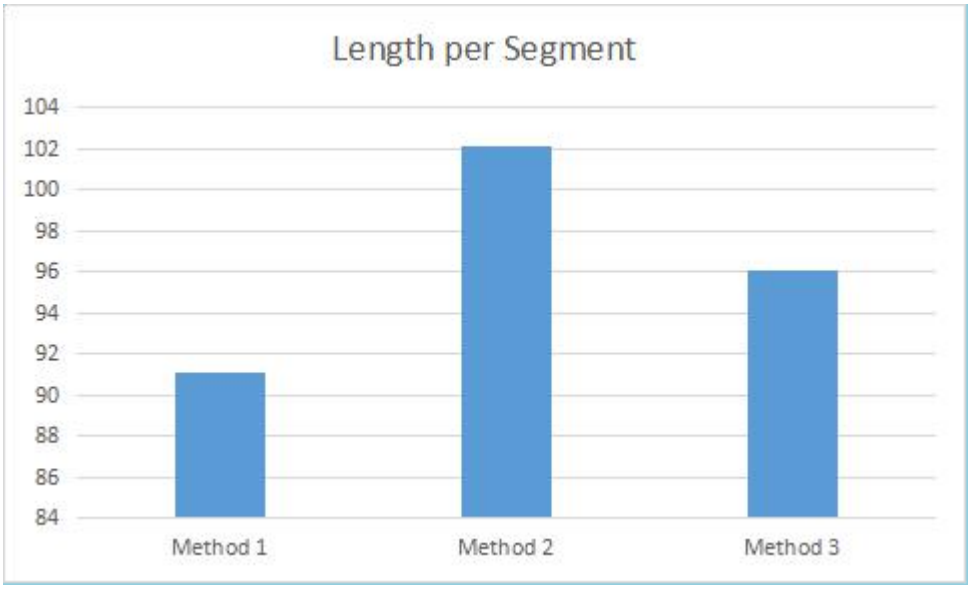
6.1.2.4 Charts

The following charts were produced by executing the road segmentation application. The sample that has been used, is a map of the city of Athens. This experiment focuses on the generation of segments for the high and low-performance roads, by applying each time a different algorithmic approach related to the segmentation by junction. The charts display the properties of the created segments and how these attributes are configured, depended on the corresponding method.



Number of Segments

The first bar chart illustrates the number of segments generated by the three methods. Overall, it is clear that the total amount of segments created by the method 1[1] has the greatest value. It is an expected result, because this method does not take into consideration any other limitation apart from checking for intersections. On the contrary, the other two methods lead to a lower number of segments, because they validate the construction of a new segment by using more conditions, such as inspecting the properties of the previous checked point or utilizing an upper bound in respect to the amount of points and the maximum road length for each segment.

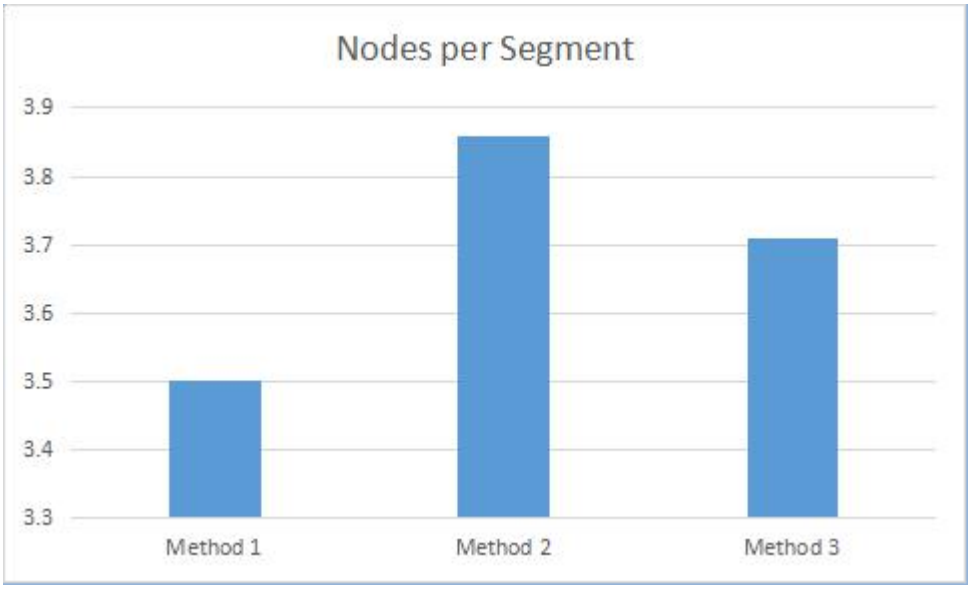


Mean length per Segment

[1] Algorithm 1 - Segmentation only by Junction

The second bar chart displays the average road length in meters for each segment. We understand that the approximate distance of the segments generated by method 1 is smaller than the respective length of the segments created by the other methods due to the fact the a new segment is produced when a junction is found. It is obvious that method 2[2], which additionally inspects the previous point and its participation in intersections, leads to a higher average road length per segment.

Finally, as it was mentioned, Method 3[3] utilizes the invented formula. We expected that the estimated mean road length for the generated segments would be lower than the mean length of the highways of a certain classification. That justifies the reason why method 3 displays a slightly lower average road length compared to the second algorithm.



Average number of points per Segment

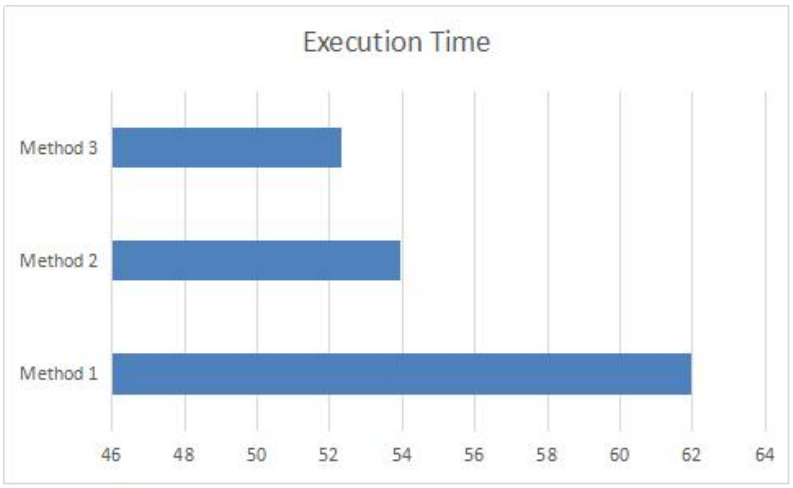
The above bar chart illustrates the average number of nodes per segment. The outcome is that method 1 results in a minimum number of points per segment. In the opposite direction, the fact that methods 2 and 3 use an extra rule involving the previous point, results in the usage of more nodes for the creation of a new segment. Method 3 displays a lower value than method 2, because the third method applies, additionally to the inspection for junctions for the previous point, an upper bound concerning the maximum number of the points and the maximum road length.

It follows the execution time diagram for each of the methods (in seconds). The algorithm we created for the first segmentation method performs the steps for the construction of segments more times compared to the other two methods.

---

[2] Algorithm 2 - Segmentation by Junction - Previous Point Inspection

[3] Algorithm 3 - Segmentation by Junction combined with Previous Point inspection - Application of Formula



Execution time

6.2 SEGMENTATION BY CURVATURE

Segmentation by curvature is another type of road segmentation. It is considered as an efficient way to produce the road segments by detecting the variation of curvature among a highway. Generally, *road curves* are regular bends in roads to bring a gradual change of direction.

Definition of Curvature[4] : The "curvature" of a way is determined by iterating over every set of three points in the line. Each set of three points forms a triangle and that triangle has a circumcircle whose radius corresponds to the radius of the curve for that set. Since every line segment (between two points) is part of two separate triangles, the radius of the curve at that segment is considered to be the smaller, larger, or average of the radii for its member sets. For the purpose of this research, the radius of the curve is equal to the mean radius.

Generally, segmentation by curvature is applied for roads that are classified as link roads or service roads. Given a highway as an argument, the system iterates through each of its nodes and creates the line segments. A set of nodes can produce a set of triangles. Each triangle consists of three consecutive nodes. After the validation of a *significant difference* in consecutive points' curvature, a segment is generated. During the conduction of the experiments, it is inevitable that some highways may not include any variation in curvature. To face this possibility and in order to create uniform polygonal curves, the formula that takes into account the number of nodes and the estimated mean length, is applied.

The construction of a segment is achieved when the estimated curvature of the current line segment has a greater value than the minimal curvature, computed by the line segments that are already used to create the current segment. In general terms, we utilize the rate of change concerning the highway curvature. To reduce the number of the generated segments, a similar algorithmic approach that involves the formula used in the segmentation by junction section, is applied.

[4] Definition of Curvature by Adam Franco [5]

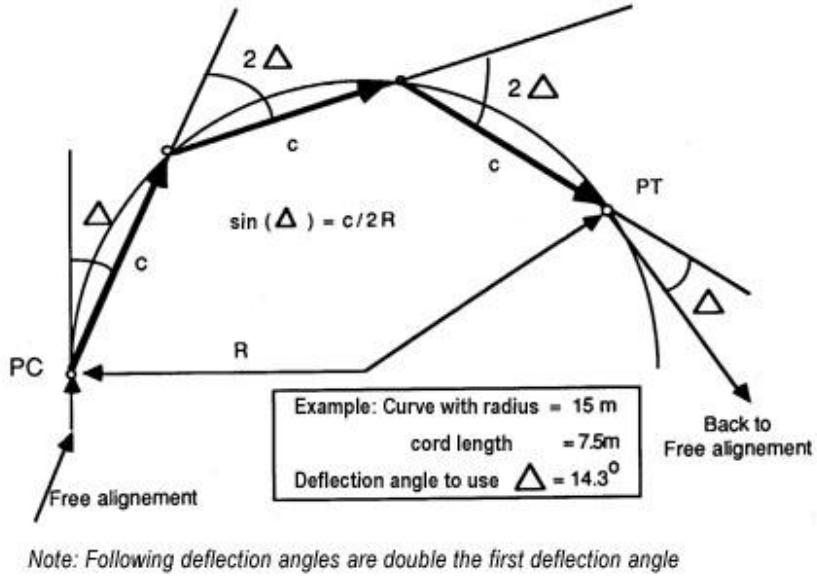


Image 6 : Definition of curvature

6.2.1 Segmentation by Curvature - Algorithms

In this thesis, we propose another set of *road segmentation methods*, based on the segmentation by curvature. The proposed methods are also independent from the highway content, so they can be used at the segmentation stage of verity applications. There are two implementations of the *road segmentation method by curvature.* The basic difference between the two proposed methods is that each one of them alternates in a unique way the rules that must be validated in order to generate a new segment.

6.2.1.1 Algorithm 1 - Segmentation by Curvature

In this section, a segmentation algorithm is presented that is based on the model of curvature variation between the segments. The proposed algorithm has six main steps. In the proposed segmentation method, the generation of the segments is only depends on the variation of curvature among the highway. No other condition is taken into consideration.

Description Given a highway, we want to create a number of segments. A highway contains a set of Points $P = \{p_1, p_2, \ldots, p_n\}$ in R^3 and it is defined by Cartesian coordinates generated by the respective geodetic. Every created segment includes a list of segment points S, populated by points of set P. Also, let L be the set of the line segments produced by the highway. The variable min-curvature stores the value of the minimum curvature of the under construction segment.

Algorithm

1. Define a list of line segments L. Call the Line Segments function that returns a list with the line segments of a given highway segment. If the set of Points contains less than three nodes, generate a segment and exit.

2. Iterate through the list of line segments L .For each iteration, store the previous line segment.

3. This step is used only for the first line segment. Check if the previous line segment has been initialized. If it is *null*, then instantiate the set S , populate it by the current line segment's points, set the value of the variable min-curvature equal to the current line segment curvature and compute the current distance using euclidean distance. Skip to the next iteration.

4. Check if the current line segment is the last. Then insert the second point of the current line segment into the set S , set the value of min-curvature equal to the minimal of min-curvature and current line segment curvature. Create a new segment using the set S, add it to the list and exit.

5. If the curvature of the current line segment displays a high augmentation in comparison with the minimum computed curvature, a new segment is generated. For this purpose, create a new segment by using the set S . Clear the set S and initialize it with p_i . Skip to the next iteration.

6. At the end of each iteration, if none of the above steps (3 to 5) is verified, then add the second point of the current line segment at the list and then update the min-curvature.

6.2.1.2 Algorithm 2 - Segmentation by Curvature - Formula

In the second section, another segmentation algorithm is presented and is depended on the model of curvature variation among the segments. The proposed algorithm has also six basic steps. The main difference between this method and the first proposed algorithm, is that at the last step, the conditions that generate the segments are enhanced by the invented formula.

Description Given a highway, we want to create a number of segments. A highway contains a set of Points $P = \{p_1, p_2, \ldots, p_n\}$ in R^3 and it is defined by the Cartesian coordinates generated by the respective geodetic. Every created segment includes a list of segment points S , populated by points of the set P. Also, let L be the set of the line segments produced by the highway. The variable min-curvature stores the value of the minimum curvature of the under construction segment. Also, the variable current_distance is responsible for saving the estimated distance.

Algorithm

1. Define a list of line segments *L.* Call the Line Segments function that returns a list with the line segments of a given highway segment. If the set of Points contains fewer than three nodes, generate a new segment and exit.

2. Iterate through the list of line segments. For each iteration, store the previous line segment.

3. This step is used only for the first line segment. Check if the previous line segment has been initialized. If it is null, then instantiate the set of Segment Points S , populate it by the current line segment's points, set the value of the variable min-

curvature equal to the current line segment curvature and compute the current distance using euclidean distance. Skip to the next iteration.

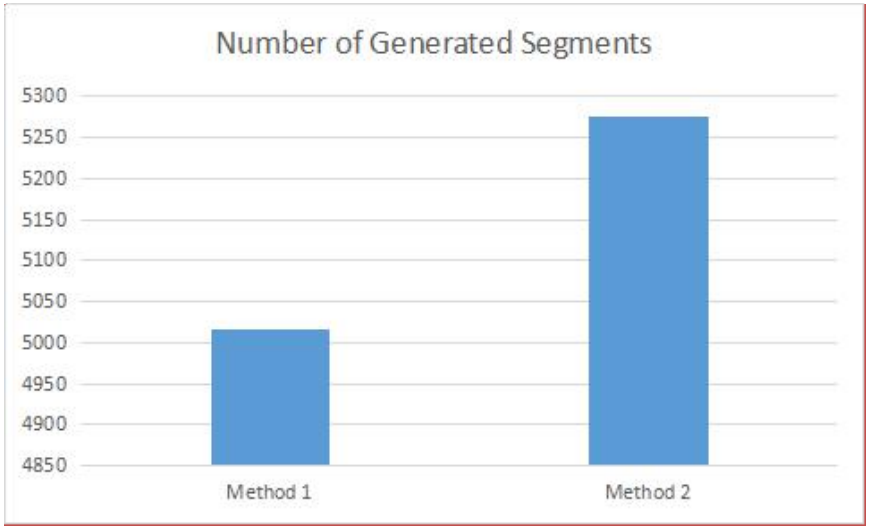
4. Check if the current line segment is the last. Then insert the second point of the current line segment into the set S, set the value of min-curvature equal to the minimal of min-curvature and current line segment curvature. Create a new segment using the set S, add it to the list and exit.

5. If the curvature of the current line segment has a greater value compared to the minimum computed curvature, a new segment is generated. If the above condition is not verified, then the number of points that are included in the set S, combined with the road mean length, are taken into account. The validation of either rule, leads to the creation of a new segment. Clear the set S and initialize it with p_i. Skip to the next iteration.

6. At the end of each iteration, if none of the above steps (3 to 5) is verified, then add the second point of the current line segment at the list and then update the values of min-curvature and current distance.

6.2.1.3 Charts

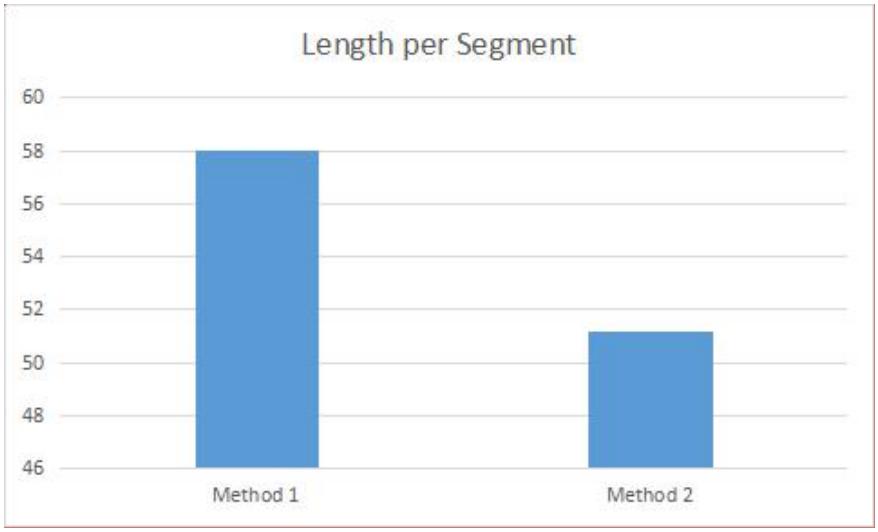
The following charts have been generated by executing the road segmentation application. The same sample has been used. This experiment focuses on the generation of segments for the high and low-performance roads, by applying each time a different algorithmic approach related to the segmentation by curvature. The charts display the properties of the created segments and how these attributes are configured depended on the corresponding method.



Number of generated segments

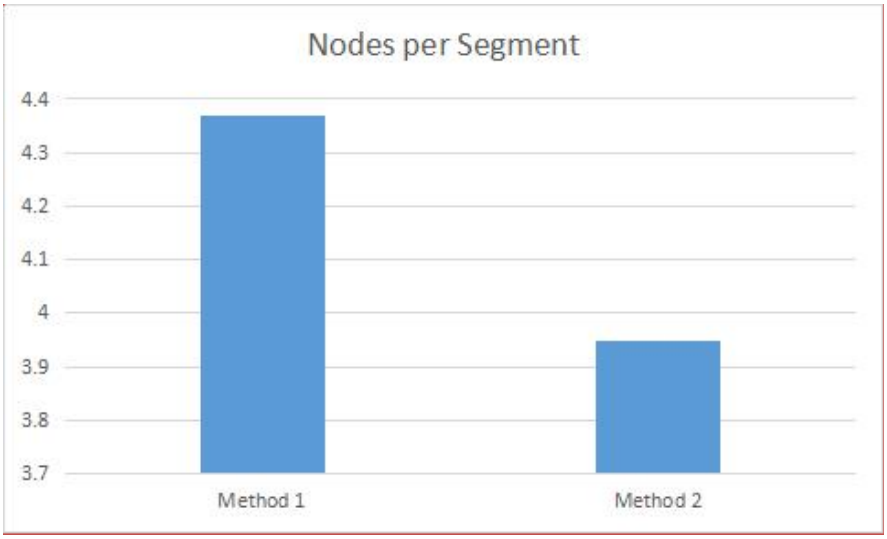
The first bar chart illustrates the number of segments that were generated by the two methods. Overall, the second method leads to a greater number of segments, because it validates the construction of a segment by utilizing an upper bound in respect to the number of points and the mean road length. On the contrary the

method 1[5] generates fewer segments, because it does not take into consideration any other limitation apart from checking for variation in curvature.



Mean length per segment

The second bar chart displays the average road length in meters for each segment. It can be seen that the approximate distance of the segments generated by the second method is smaller than the respective value of the segments created by method 1. Method 2 [6] displays the lowest value, because it applies the formula.



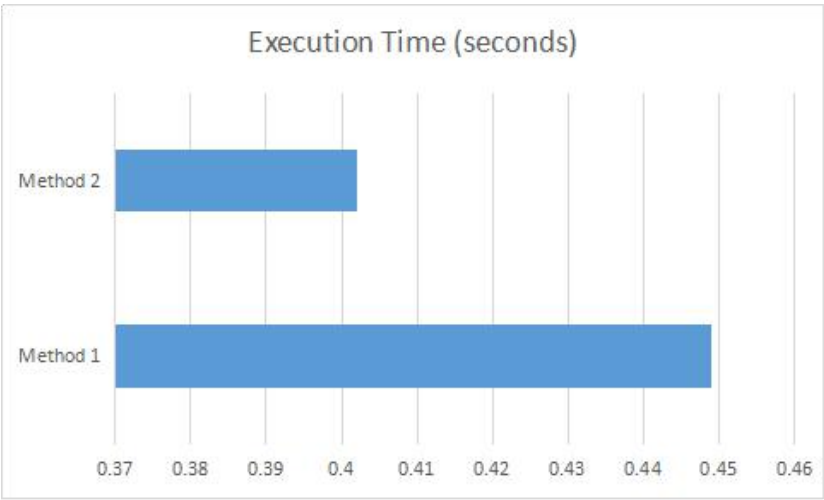
Average number of points per segment

The above bar chart illustrates the average number of nodes per segment. We expected that the first method leads to a higher number of points and on the contrary, method 2 has a lower value than method 1, because it inspects additionally to the

[5] Algorithm 1 - Segmentation by Curvature

[6] Algorithm 2 - Segmentation by Curvature - Formula

curvature variance, the maximum number of the points and the maximum road length.



Execution time

6.2.1.4 Line Segments Generation Algorithm

The line segment generation algorithm is used by the segmentation methods by curvature.

Definition: In real or complex vector spaces, if *V* is a vector space over $R\ or\ C$, and *L* is a subset of *V*, then *L* is a *line segment* if *L* can be parameterized as : $L = \{u + tv \mid t \in [0,1]\}$, for some vectors $u, v \in V$ in which case the vectors **u** and **u** + **v** are called the end points of *L*.

Description : Given a segment that contains a set of Points $P = \{p_1, p_2, \ldots, p_n\}$ in R^3 defined by the Cartesian coordinates generated by the geodetic, we want to produce a list with the line segments *L.*

Algorithm

1. Create each triangle as a set of three consecutive points. The number of the triangles is equal to $(n-2)$. Variable n is defined as the number of the points and is greater than 3. If a highway segment includes less than three points (and of course more than 2), then return a line segment with curvature equals to zero.

2. For each triangle, compute its curvature. Each set of three points forms a triangle and that triangle has a circumcircle whose radius corresponds to the radius of the curve for that set. Estimation of curvature requires the calculation of the three sides (a, b and c) and the radius of the circumcircle as the division of the sides' product $(a \cdot b \cdot c)$ by :

$$\sqrt{|(a+b+c)*(b+a-c)*(c+a-b)*(a+b-c)|}$$

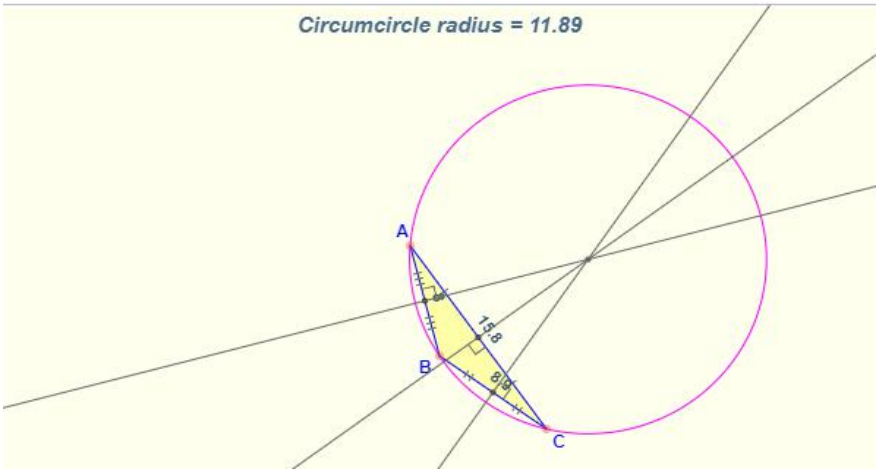


Image 7 : Definition of Circumcircle

The circumcenter is the point where the perpendicular bisectors of a triangle intersect. The circumcenter is also the center of the triangle's circumcircle - the circle that passes through all three of the triangle's vertices.

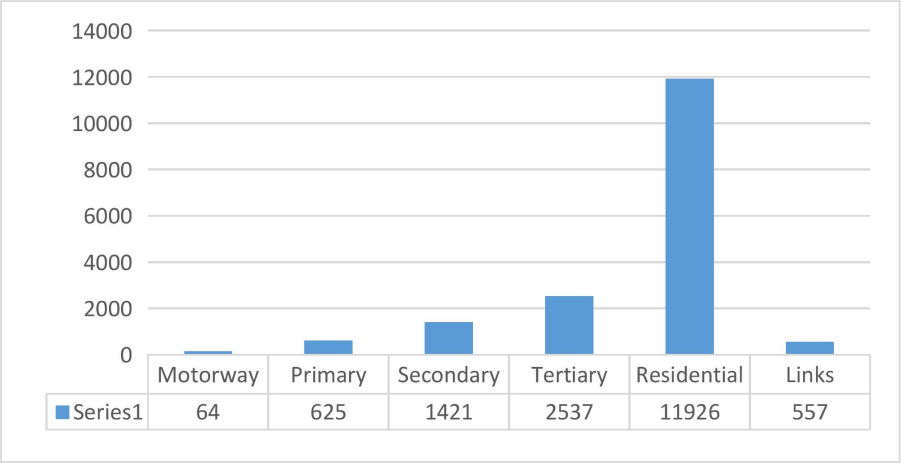
3. Iterate over each triangle and generate a line segment created by the two first points of the triangle. Since every line segment (between two points) is part of two separate triangles, the radius of the curve at that segment is considered to be the average of the radius for its member sets. Add the generated line segments to the list.

6.3 Choosing the suitable segmentation method

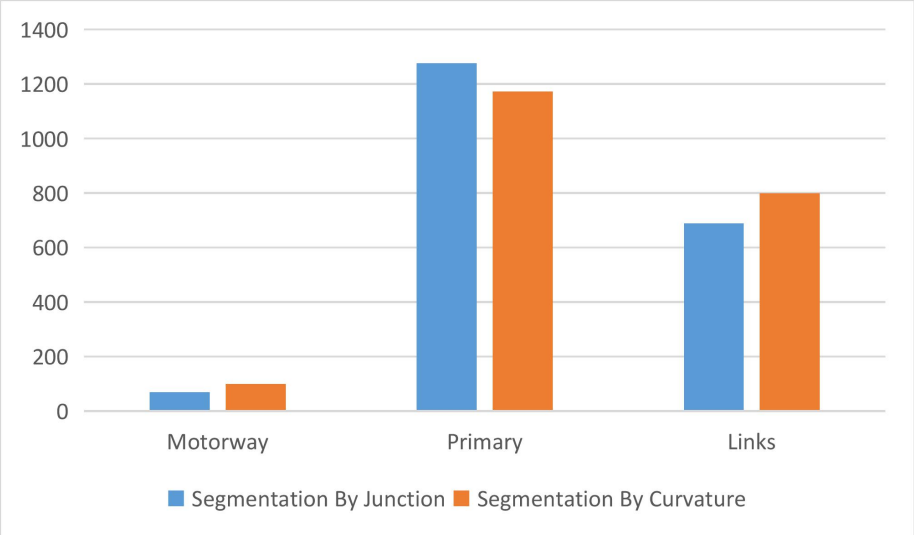
In order to determine which segmentation method is suitable and effective for each road classification, we conducted an experiment. For this experiment, we assume that the amount of the generated segments for each road category is depended on the properties of the highway, combined with the respective utilization of the proposed segmentation methods.

To confirm the previous assumption, we run the application by executing the two segmentation methods for some road classifications. We choose the road sample from the following categories: primary, secondary, tertiary, residential, links roads and motorways. The first bar charts demonstrates the number of the highways used for each category. The last two display how the number of segments fluctuates regarding the method that is applied.

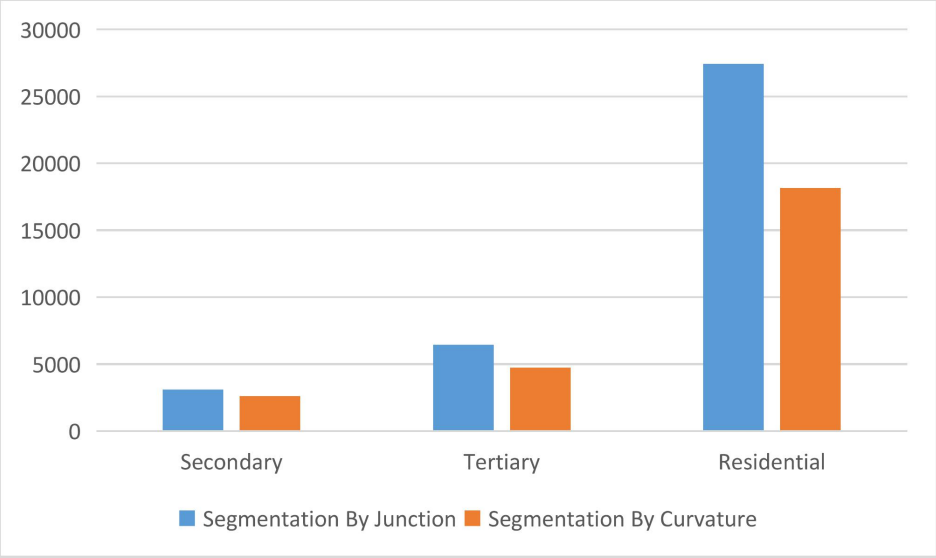
For the roads that are considered low-performance, i.e. residential, segmentation by junction shows significant difference in the total amount of the constructed segments. This is an expected outcome, caused by the great number of intersections. At the other end of the spectrum, the link roads are divided more effectively when the segmentation by curvature is applied, due to the fact that the existed intersections are less. For the high-performance roads, both methods display similar behavior.



Highways per Road Classification



Segmentation By Junction and By Curvature, Segments



Segmentation By Junction and By Curvature, Segments

7. NEAREST NEIGHBOR

7.1 Nearest neighbor search

Nearest neighbor search (NNS), as a form of proximity search, is the optimization problem of finding the point in a given set that is closest (or most similar) to a given point. Closeness is typically expressed in terms of a dissimilarity function: the less similar the objects, the larger the function values.

Formally, the nearest-neighbor (NN) search problem is defined as follows: given a set *S* of points in a space *M* and a query point $q \in M$, find the closest point in *S* to *q*. A direct generalization of this problem is a *k*-NN search, where we need to find the *k* closest points. Most commonly *M* is a metric space and dissimilarity is expressed as a distance metric, which is symmetric and satisfies the triangle inequality. Even more common, *M* is taken to be the *d*-dimensional vector space where dissimilarity is measured using the Euclidean distance.

Various solutions to the NNS problem have been proposed. The quality and usefulness of the algorithms are determined by the time complexity of queries as well as the space complexity of any search data structures that must be maintained. The informal observation usually referred to as the curse of dimensionality states that there is no general-purpose exact solution for NNS in high-dimensional Euclidean space using polynomial preprocessing and polylogarithmic search time. In this thesis, we study two parallel approximation methods, locality-sensitive hashing and clustering.

7.2 Euclidean Metric Space

A metric or distance function is a function that defines a distance between each pair of elements of a set. A set with a metric is called a metric space. A metric induces a topology on a set, but not all topologies can be generated by a metric. A topological space whose topology can be described by a metric is called metrizable.

In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in Euclidean space. With this distance, Euclidean space becomes a metric space.

Definition 7.2.1 The Euclidean distance between points p and q is the length of the line segment connecting them. In Cartesian coordinates, if $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the *Pythagorean formula*:

$$d(p,q) = d(q,p) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots (p_n - q_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

8. LOCALITY-SENSITIVE HASHING

Hashing is one of the popular solutions for approximate nearest neighbor search. In general, hashing is an approach of transforming the data item to a low-dimensional representation, or equivalently a short code consisting of a sequence of bits. The application of hashing to approximate nearest neighbor search includes two ways: indexing data items using hash tables that is formed by storing the items with the same code in a hash bucket, and approximating the distance using the one computed with short codes.

An approximation algorithm is allowed to return a point, whose distance from the query is at most ***c*** times the distance from the query to its nearest points. The appeal of this approach is that, in many cases, an approximate nearest neighbor is almost as good as the exact one. In particular, if the distance measure accurately captures the notion of user quality, then small differences in the distance should not matter. Hashing-based approximate nearest neighbor search algorithms generally use one main hashing method: a data-independent method, named as locality-sensitive hashing (LSH).

Locality sensitive hashing (LSH) is a technique for grouping points in space into 'buckets' based on some distance metric operating on the points. Points that are close to each other under the chosen metric are mapped to the same bucket with high probability. Locality-sensitive hashing hashes input items so that similar items map to the same “buckets” with high probability. In general, LSH reduces the *dimensionality* of high-dimensional data and differs from conventional and cryptographic hash functions because it aims to maximize the probability of a “collision” for similar items. Locality-sensitive hashing has much in common with data clustering and nearest neighbor search.

8.1 LSH Family

Definition 8.1.1 An LSH family ***F*** is defined for metric space $M=(M,d)$, a threshold R > 0 and an approximation factor c > 1. This family ***F*** is a group of functions $h: M \to S$ which map elements from the euclidean metric space to a bucket $s \in S$.

The LSH family satisfies the following conditions for any two points $p,q \in M$, using a function h ∈ ***F*** which is chosen uniformly at random:

- If $d(p,q) \le R$ **,**then $h(p)=h(q)$ (i.e.,***p*** and ***q*** collide) with probability at least P_1,
- If $d(p,q) \ge cR$,then $h(p)=h(q)$ with probability at least P_2,

Alternatively it is defined with respect to a universe of items *U* that have a similarity function $\phi: U \times U \to [0,1]$.

An LSH scheme is a family of hash functions H coupled with a probability distribution D over the functions such that a function $h \in H$ chosen according to *D* satisfies the property that $\Pr_{h \in H}[h(a)=h(b)]=\phi(a,b)$ for any $a,b \in U$.

8.2 LSH algorithm for nearest neighbor search

Definition 8.2.1 One of the main applications of LSH is to provide a method for efficient approximate nearest neighbor search algorithms. Consider an LSH family ***F***. The algorithm has two main parameters: the width parameter k and the number of hash tables ***L***.

1. We define a new family G of hash functions g ,where each function g is obtained by concatenating k functions $h_1,\ldots,h_k$ from ***F***, $g(p)=[h_1(p),\ldots,h_k(p)]$. A random hash function g is obtained by concatenating k randomly chosen hash functions from ***F***. The algorithm then constructs ***L*** hash tables, each corresponding to a different randomly chosen hash function g .

2. In the preprocessing step we hash all n points from the data S into each of the ***L*** hash tables. Given that the resulting hash tables have only n non-zero entries, one can reduce the amount of memory used per each hash table to $O(n)$ using standard hash functions.

3. Given a query point q, the algorithm iterates over ***L*** hash functions g . For each g considered, it retrieves the data points that are hashed into the same bucket as q. The process is stopped as soon as a point within distance ***cR*** from q is found.

Performance : Given the parameters k and L, the algorithm has the following performance guarantees:

- preprocessing time: $O(nLkt)$ **,** where *t* is the time to evaluate a function **f** ∈ ***F*** on an input point *p*;
- space: $O(Ln)$, plus the space for storing data points;
- query time: $O\left(L\left(kt+dnP_2^k\right)\right)$**;**
- the algorithm succeeds in finding a point within distance ***cR*** from *q* (if there exists a point within distance *R*) with probability at least $1-\left(1-P_1^k\right)^L$.

For a fixed approximation ratio $c=1+\varepsilon$ and probabilities P_1 and P_2, one can

$$set \quad k=\frac{\log n}{\log 1/P_2} \quad L=n^{\rho}, where \quad \rho=\frac{\log P_1}{\log P_2}$$

Then one obtains the following performance guarantees:

- preprocessing time: $O\left(n^{1+p}kt\right)$;
- space: $O\left(n^{1+p}\right)$ plus the space for storing data points;
- query time: $O\left(n^p\left(kt+d\right)\right)$;

8.3 Grid Curve

8.3.1 Linear approximation factor

The basic LSH has an approximation factor that is linear in the number of vertices that a curve can have.

Definition 8.3.1.1 We use a randomly shifted grid in our hashing scheme. Let the canonical d-dimensional grid of resolution δ be defined as an evenly spaced point set in R^d , as follows:

$$G_\delta = \left\{ (x_1, \ldots, x_d) \in R^d \mid \forall 1 \le i \le d \;\; \exists j \in N : x_i = j \cdot \delta \right\}$$

Consider a family of such grids parametrized by a shift **t**:

$$\hat{G}_\delta^t = \{ p + t \mid p \in G_\delta \}$$

Definition 8.3.1.2 Choosing t uniformly at random from the half-open hyper-cube $[0,\delta)^d$ we obtain a family of randomly shifted grids. Let $P \in \Delta^d$ be a polygonal curve with vertices $\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_m$ and let $h_\delta^t : \Delta^d \to \Delta^d$ be a hash function.

The curve $h_\delta^t(P)$ is defined as the result of the following two-stage construction.

1. We snap the curve to the grid $\hat{G}_\delta^t$. More precisely, we replace each vertex p_i with its closest grid point $p_i' = \arg\min_{q \in \hat{G}_\delta^t} \| p_i - q_i \|$ to obtain the curve P' .
2. We remove consecutive duplicates in P' . That is, we remove the vertex p_i' if it is identical to p_{i-1}'.

Definition 8.3.1.3 Let H_δ^L be the family of hash functions h_δ^t constructed this way. The value of h_δ^t *(P)* is the resulting polygonal curve. The resulting LSH family is $H_\delta^L = \left\{ h_\delta^t(\cdot) \mid t \in_R [0,\delta]^d \right\}$.

Lemma 8.3.1.4 Let $P, Q \in \Delta^d$ be two curves with m_1 and m_2 points, respectively, and let $\min\{m_1, m_2\}$. For any δ > 0, it holds that

$$\Pr_{H_\delta^L}\left[h_\delta^t(P) = h_\delta^t(Q) \right] \ge 1 - \left(2dm \cdot \frac{d_F(P,Q)}{\delta} \right). \qquad [8]$$

Lemma 8.3.1.5 For any value of δ and for any P, Q ∈ Δ^d , if there exists a value of t∈$[0,\delta)^d$ such $h_\delta^t(P) = h_\delta^t(Q)$, then it holds that $d_F(P,Q) \le \sqrt{d} \cdot \delta$.

$$d_F(P,Q) \le d_F\left(h_\delta^t(P), P \right) + d_F\left(h_\delta^t(P), h_\delta^t(Q) \right) + d_F\left(h_\delta^t(Q), Q \right) \le \sqrt{d} \cdot \delta \qquad [8]$$

Theorem 8.3.1.6 [8] Let P,Q be two polygonal curves with m_1 and m_2 points, respectively, $\mu = \min\{m_1, m_2\}$, and let $\delta = 4d\mu r$ and $c = 4d^{3/2}\mu$.

For r > 0, it holds that:

- If $d_F(P,Q) < r$, then $\Pr\left[h_\delta^t(P) = h_\delta^t(Q)\right] > 1/2$,
- If $d_F(P,Q) > cr$, then $\Pr\left[h_\delta^t(P) = h_\delta^t(Q)\right] = 0$

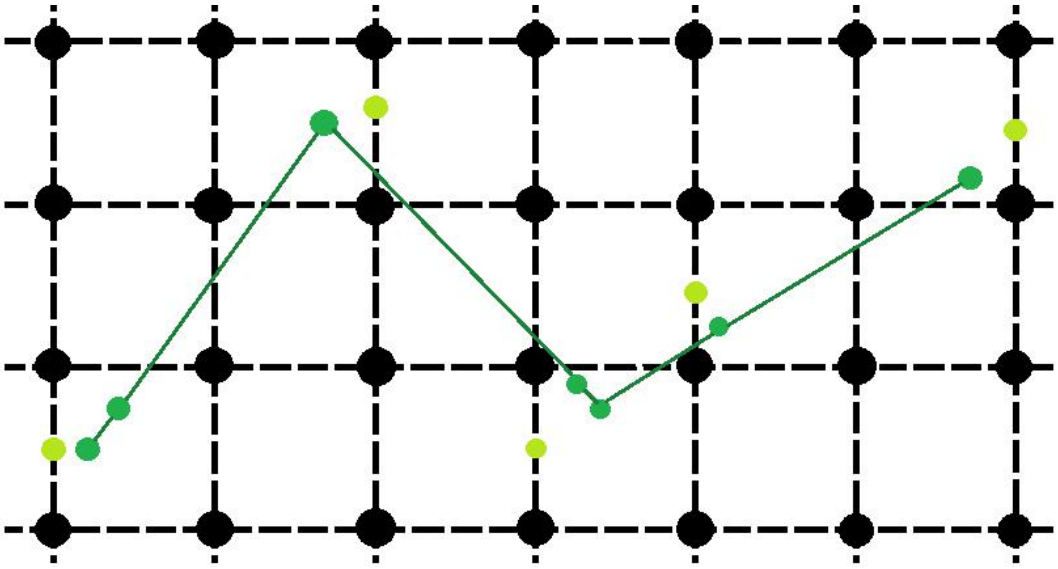
Proof. The first claim follows by plugging in the bounds of ***Lemma 1***:

$$\Pr\left(h_\delta^t(P) = h_\delta^t(Q)\right) > 1 - \left(2dm \cdot \frac{d_F(P,Q)}{\delta}\right) > 1 - \frac{d_F(P,Q)}{2r} > \frac{1}{2}$$

On the other hand, the second claim follows from ***Lemma 2***:

$$d_F(P,Q) > c \cdot r = 4d^{3/2}mr = \sqrt{d} \cdot \delta \Rightarrow h_\delta^t(P) \neq h_\delta^t(Q)$$

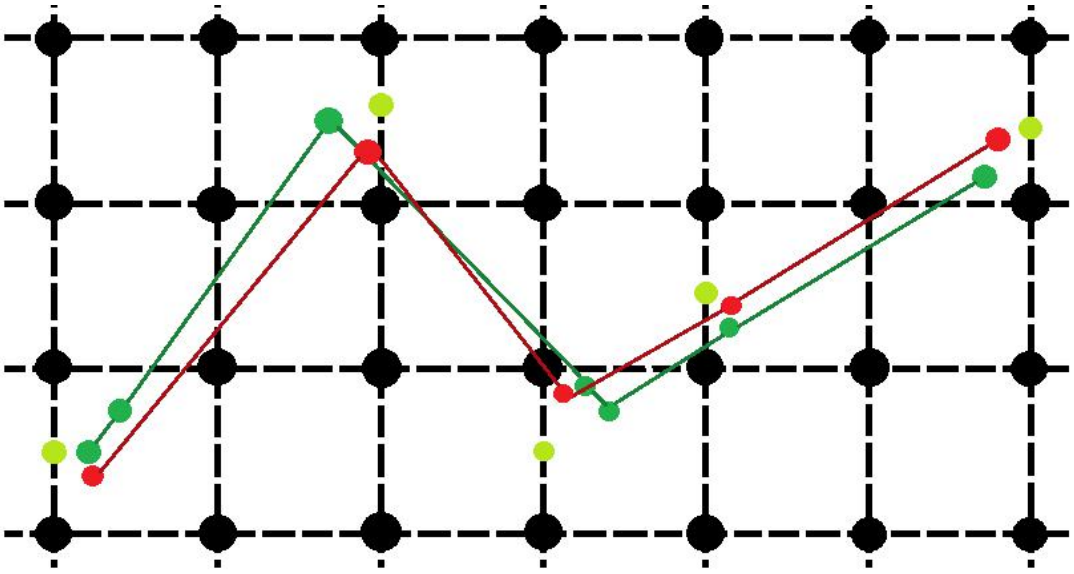
[8] Hence, no conflicts are observed for very different curves (non-symmetric).



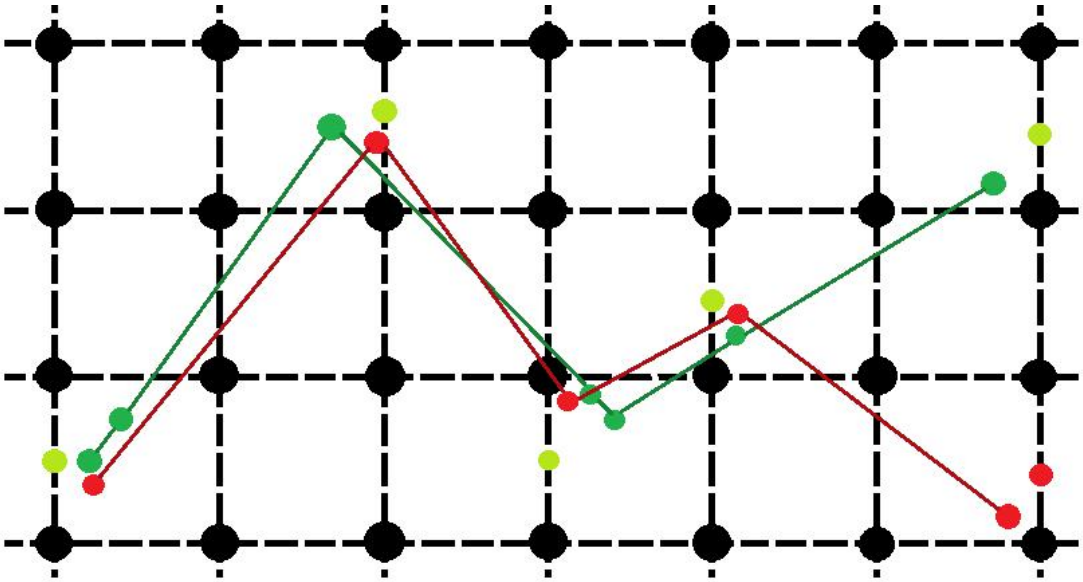
LSH for O(m) approximation

Algorithm

1. Create a grid of size $\delta = \mathrm{O}(mr)$.
2. Random shift of $[0,\delta)^2$.
3. Snap the curve on the grid and remove consecutive duplicates.
4. The signature is given by the remaining points.



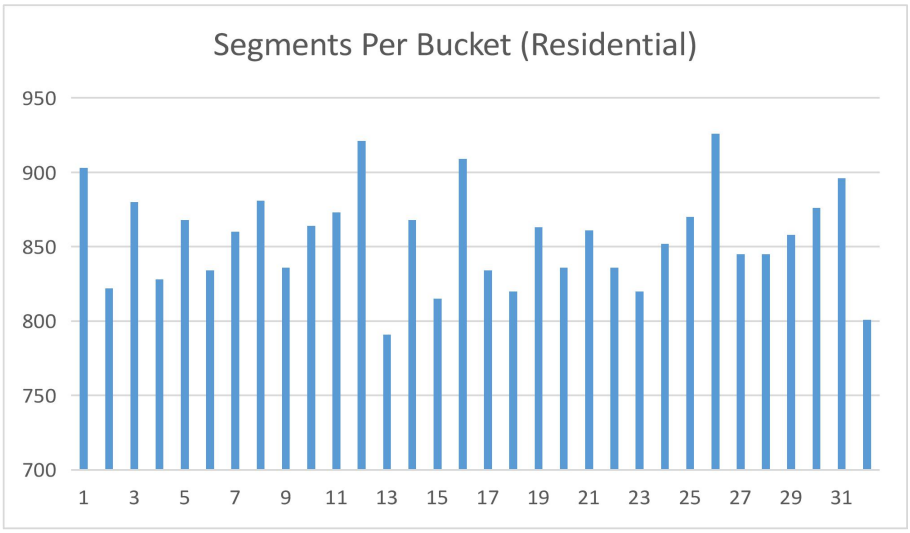
Collision of near curves : Two near curves collide with some probability



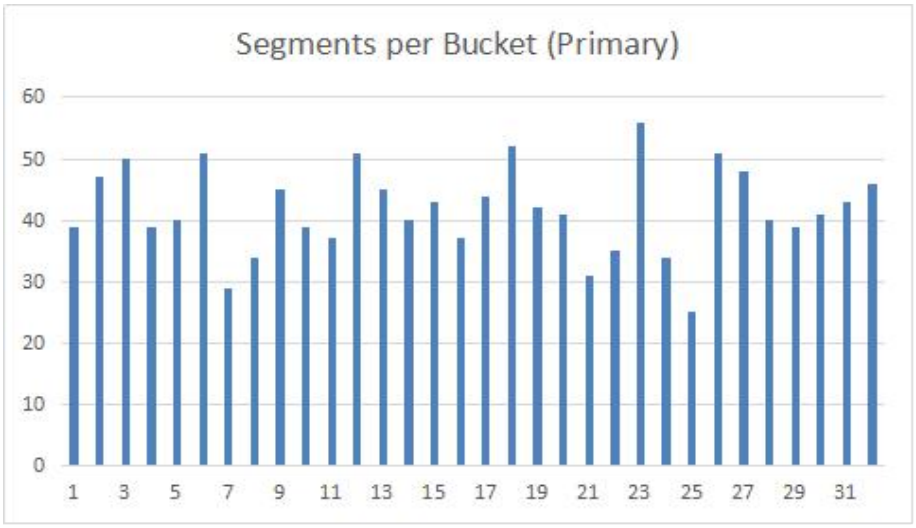
Collision of far curves : Two far curves never collide

8.4 Experiment - Locality Sensitive Hashing

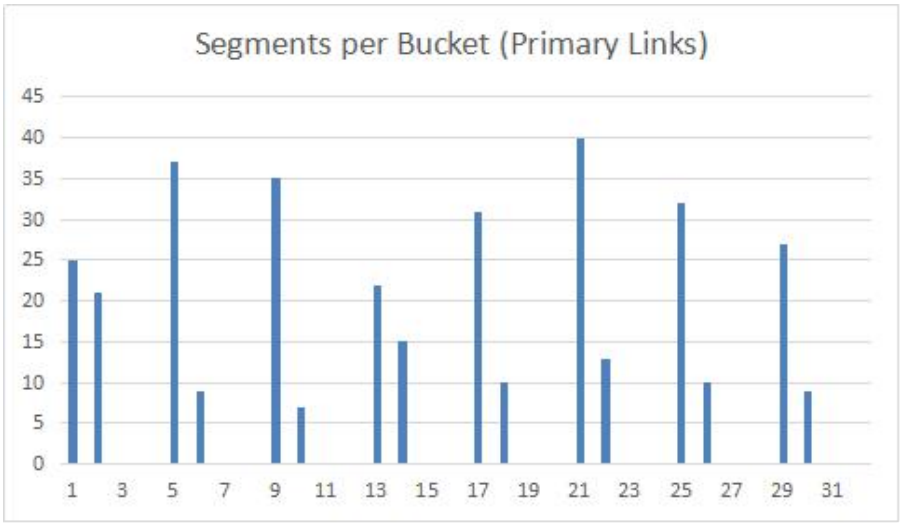
The following charts are produced by the conduction of an experiment aiming at hashing the generated segments. The charts display the number of segments for each bucket. Specifically, the road samples are distinguished into the following road classifications, residential, primary and primary-link roads. A general overview of the results is that there is an uniform distribution of the segments among the buckets.



Segments per Bucket (Residential)



Segments per Bucket (Primary)



Segments per Bucket (Primary Links)

9. CLUSTERING

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (cluster) are more similar to each other than to those in other groups.It is a main task of exploratory data mining, and a common technique for statistical data analysis. Clustering is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. [9]

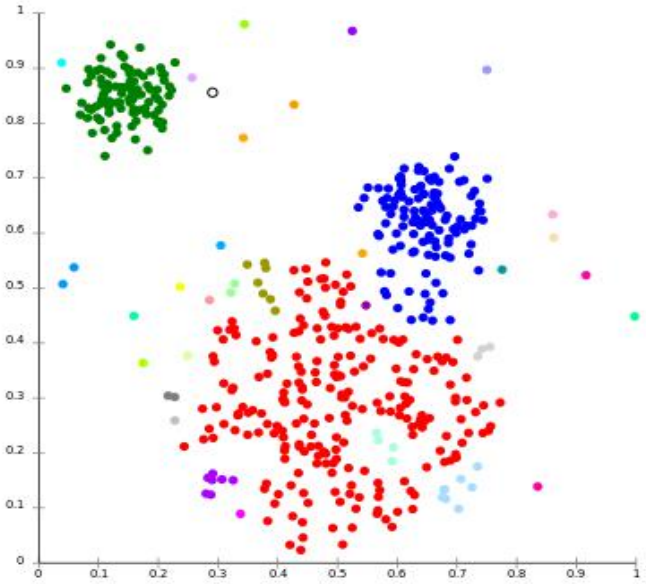


Image 8 : Cluster analysis with k-means on a Gaussian-distribution-based data set.

Clustering can therefore be formulated as a multi-objective optimization problem. The appropriate clustering algorithm and parameter settings (including parameters such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data. A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. A *cluster* is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

9.1 Centroid-based clustering

In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k

cluster centers and assign the objects to the nearest cluster center, such that the euclidean distances from the cluster are minimized.

The optimization problem is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximate method is *Lloyd's* algorithm, often just referred to as "k-means algorithm". It does however only find a local optimum, and is commonly run multiple times with random initialization. Variations of k-means often include such optimizations as choosing the best of multiple runs, but also restricting the centroids to members of the data set (k-medoids), choosing medians (k-medians clustering) or choosing the initial centers less randomly (k-means++).

Most *k*-means-type algorithms require the number of clusters - *k* - to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. This often leads to incorrectly cut borders of clusters (which is not surprising since the algorithm optimizes cluster centers, not cluster borders).

In order to define clustering for n objects, and k > 1, it is imperative to divide the objects into k subsets (clusters) so as to optimize some objective function. Objects in the same cluster are more "similar" (or closer) to each other than to those in other clusters. Optimizing an objective function means minimizing the total distance of cluster points to some center. It is described extensively in a following section.

9.2 Lloyd's Algorithm

In computer science, Lloyd's algorithm is an algorithm for finding evenly spaced sets of points in subsets of Euclidean spaces and partitions of these subsets into well-shaped and uniformly sized convex cells. Like the closely related k-means clustering algorithm, it repeatedly finds the centroid of each set in the partition and then re-partitions the input according to which of these centroids is closest. However, Lloyd's algorithm differs from k-means clustering in that its input is a continuous geometric region rather than a discrete set of points.

Definition 9.2.1 Algorithm:

1. Initialize k centers randomly (k-means initialization).
2. Assignment: Assign each object to its nearest center.
3. Update: Calculate mean $\frac{1}{T}\sum_{t=1}^{T}\vec{v}_i$ of each cluster, make it new center.
4. Repeat the two steps 2 and 3, until there is no change in the assignments.

Performance [7] **:** The complexity of this algorithm can be described by the following aspects. At first, for the computation of the each distance and for the updating of the clusters, the approximate complexity is equal to $O(nd)$. For assignment, it is $O(nkd)$.

9.3 K-means++ Initialization

The k-means method is a widely used clustering technique that seeks to minimize the average squared distance between points in the same cluster. Although it offers no accuracy guarantees, its simplicity and speed are very appealing in practice. By augmenting k-means with a very simple, randomized seeding technique, we obtain an algorithm that is $\Theta(\log k)$ -competitive with the optimal clustering. Preliminary experiments show that our augmentation improves both the speed and the accuracy of k-means, often quite dramatically.

It is the speed and simplicity of the k-means method that make it appealing, not its accuracy. Indeed, there are many natural examples for which the algorithm generates arbitrarily bad clustering. This does not rely on an adversarial placement of the starting centers, and in particular, it can hold with high probability even if the centers are chosen uniformly at random from the data points.

Definition 9.3.1 Algorithm : It follows a proposed method [7] that chooses centers at random from the data points, but weighs the data points according to their squared distance squared from the closest center already chosen.

(1) Choose a centroid uniformly at random; t ← 1.

(2) ∀ non-centroid point i = 1, . . . , n − t, let D(i) ← min distance to some centroid, among t chosen centroids.

(3) Choose new centroid: r chosen with probability proportional to $D(r)^2$:

$$prob\left[choose\ \ r\right] = D(r)^2 \Big/ \sum_{i=1}^{n-t} D(i)^2 .$$

Let t ← t + 1.

(4) Go to (2) until t = k = given #centroids.

Given D(i) > 0, i = 1, . . . , n − t, compute n − t (float) partial sums

$$P(r) = \sum_{i=1}^{r} D(i)^2, \quad r = 1,\ldots,n-t,$$

and store them in an array (or binary tree) P. To avoid the P(r)'s being very large, we may normalize all D(i)'s by dividing them by $\max_i D(i)$.

Pick a uniformly distributed *float* x ∈ [0, P(n − t)] and return

$$r \in \{1,2,\ldots,n-t\} : P(r-1) < x \le P(r),$$

where P(0) = 0: r chosen with probability proportional to P(r) − P(r − 1) ¬ $D(i)^2$. Can find r by binary search in array P.

9.4 Assignment

In the clustering method, each segment that was generated, may be a centroid or a part of a cluster. Considering the centroids have been chosen, each of the remaining segments is assigned to the cluster of the centroid that is the nearest.

9.5 Objective function

Generally, the objective of a linear programming problem is to maximize or to minimize some numerical value. In the case of clustering, the purpose is to minimize the total amount of the distances computed for each segment and the centroid of the cluster it belongs to. [1]

Definition 9.5.1 The clustering algorithm that was described previously, aims at minimizing an *objective function*, in this case a squared error function. The objective function :

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2,$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster center c_j , is an indicator of the distance of the *n* data points from their respective cluster centers.

9.5.1 Update of Objective cost function

Based on the previous section, we can define the objective function as:

$$J = \sum_{i=1}^{n-k} dist\left(i, c(i)\right), \;\; c(i) = centroid \;\; of \; i's \; cluster.$$

For each i store 2nd best centroid c' . Centroid m replaced by non-centroid t :

- For i : c(i) = m,

$$\Delta J = \left\{ \begin{array}{l} dist(i,t) - dist(i,m) \;\; if \;\; dist(i,t) \le dist(i,c') : do \;\; nothing \\ dist(i,c') - dist(i,m) \;\; if \;\; dist(i,t) > dist(i,c') : assign \;\; i \;\; to \;\; c', \\ \qquad\qquad\qquad\qquad update \;\; i's \;\; 2nd \;\; best \;\; centroid \end{array} \right\}$$

- For i : c(i) ≠ m,

$$\Delta J = \begin{Bmatrix} 0 & if\ \ dist(i,t) \geq dist(i,c(i)):\ \ do\ \ nothing \\ dist(i,t) - dist(i,c(i)) & if\ \ dist(i,t) < dist(i,c(i)):\ \ assign\ \ i\ \ to\ \ t \end{Bmatrix}$$

The complexity of updating the *Objective Function* is $O((n-k)d)$.

Suppose that n objects having p variables each should be grouped into k (k < n) clusters, where k is assumed to be given. Let us define j_{th} variable of object ***i*** as $X_{ij}(i = 1,\ldots,n;\ j = 1,\ldots,p)$.
The Fréchet distance will be used as a dissimilarity measure in this study although other measures can be adopted.

Algorithm

The proposed algorithm is composed of the following three steps.

1. *Select initial centroids*.

   - The k-means initialization has been chosen.

2. *Update centroids.*

   - Find the medoid of each cluster, which is the object minimizing the total distance to other objects in its cluster. Generally, The medoid of a set is the object of the set that minimizes total dissimilarity (distance) to all other objects in the set.

   - Update the current centroid in each cluster by replacing with the calculated medoid.

3. *Assign objects to medoids.*

   - Assign each object to the nearest medoid and obtain the cluster result.

   - Calculate the sum of distance from all objects to their medoids. If the sum is equal to the previous one, then stop the algorithm. Otherwise, go back to the Step 2.

9.6 Evaluation (Silhouette)

Silhouette refers to a method of interpretation and validation of consistency within clusters of data. The technique provides a succinct graphical representation of how well each object lies within its cluster.

The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative

value, then the clustering configuration may have too many or too few clusters. The purpose is to achieve evaluation of clustering without reference to objective function.

Internal evaluation considers the given point set and the clusters, produces quality coefficient for each partition; k may be a parameter. In the sequel we present an internal evaluation method, known as Silhouette.

Definition 9.6.1 Let k be the number of computed clusters.

For 1 ≤ i ≤ n, a(i) = average distance of i to other objects in same cluster.

Let b(i) = average distance of i to objects in next best (neighbor) cluster, i.e. cluster of 2nd closest centroid.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i) - b(i)\}} = \begin{cases} 1 - a(i)/b(i), & if\ a(i) < b(i) \\ 0 & if\ a(i) = b(i) \\ b(i)/a(i) - 1, & if\ a(i) > b(i) \end{cases} \in [-1,1]$$

Based on the value of silhouette, we can evaluate how successful the clustering is.

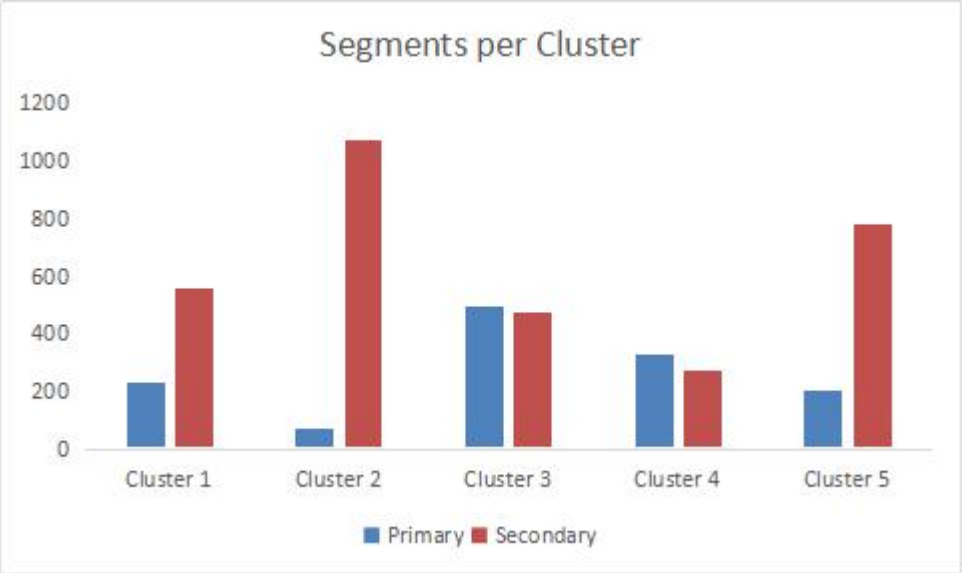
s(i) → 1 : i seems correctly assigned to its cluster;

s(i) ≈ 0 : borderline assignment (but not worth to change);

s(i) → −1: i would be better if assigned to next best cluster.

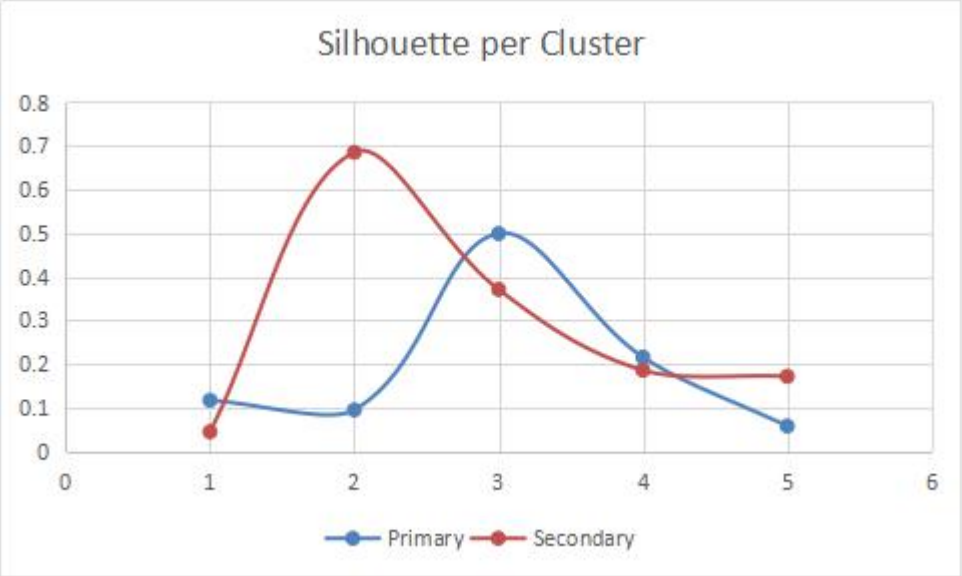
For $s(i)$ to be close to 1 we require $a(i) << b(i)$. As $a(i)$ is a measure of how dissimilar $b(i)$ is to its own cluster, a small value means it is well matched. Furthermore, a large implies that i is badly matched to its neighboring cluster. Thus an $s(i)$ close to one means that the data is appropriately clustered. If $s(i)$ is close to negative one, then by the same logic we see that I would be more appropriate if it was clustered in its neighboring cluster. An $s(i)$ near zero means that the datum is on the border of two natural clusters.

The average $s(i)$ over all data of a cluster is a measure of how tightly grouped all the data in the cluster are. Thus the average $s(i)$ over all data of the entire dataset is a measure of how appropriately the data have been clustered. If there are too many or too few clusters, as may occur when a poor choice of k is used in the clustering algorithm (e.g.: k-means), some of the clusters will typically display much narrower silhouettes than the rest. Thus silhouette plots and averages may be used to determine the natural number of clusters within a dataset. One can also increase the likelihood of the silhouette being maximized at the correct number of clusters by re-scaling the data using feature weights that are cluster specific.



Segments per Cluster

This chart displays the distribution of segments among the clusters. It was measured for primary and secondary roads.



Silhouette per Cluster

This chart illustrates the values of silhouette for each cluster. It was measured for primary and secondary roads.

9.7 Fréchet Distance

Given two curves in a metric space, the Fréchet distance δ_F between them can be defined intuitively as follows. We can imagine a man traversing a finite curved path while walking his dog on a leash, with the dog traversing a separate one. Assuming that the dog varies her speed to keep as much slack in her leash as possible: the Fréchet distance between the two curves is the length of the shortest leash sufficient for both to traverse their separate paths. It is imperative to note that the definition is symmetric with respect to the two curves and the Fréchet distance would be the same if the dog was walking her owner.

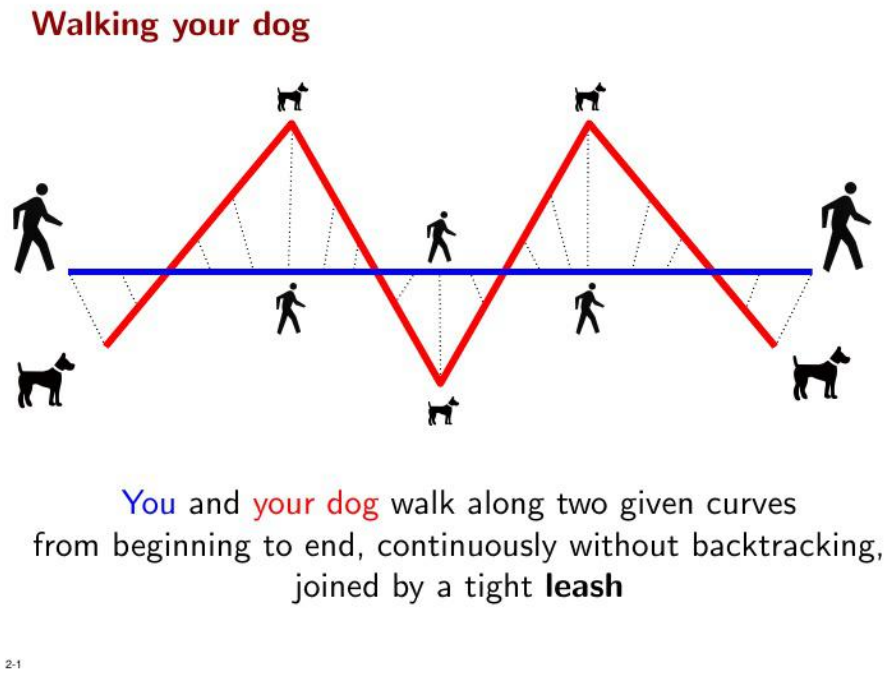


Image 9 : Discrete Fréchet Distance

The Fréchet distance is a superior measure for the similarity of polygonal curves, but it is very difficult to handle. The discrete Fréchet distance considers only positions of the leash where its endpoints are located at vertices of the two polygonal curves and never in the interior of an edge. This special structure allows the discrete Fréchet distance to be computed in polynomial time by an easy dynamic programming algorithm. When the two curves are embedded in a metric space, such as an Euclidean space, the distance between two points on the curves is most naturally defined as the length of the shortest path between them.

Definition 9.7.1 Given polygonal curves $P = p_1, p_2, \ldots, p_{m1}, Q = q_1, q_2, \ldots, q_{m2}$, a traversal $T = (i_1, j_1), \ldots, (i_t, j_t)$ is a sequence of pairs of indices referring to a pairing of vertices from the two curves such that:

1. $i_1, j_1 = 1,\ i_t = m_1, j_t = m_2$
2. $\forall (i_k, j_k) \in T : i_{k+1} - i_k \in \{0,1\}\ \ and\ \ j_{k+1} - j_k \in \{0,1\}$
3. $\forall (i_k, j_k) \in T : (i_{k+1} - i_k) + (j_{k+1} - j_k) \geq 1$

9.7.1 Discrete Fréchet Distance (DFD)

The discrete Fréchet distance, also called the coupling distance, is an approximation of the Fréchet metric for polygonal curves.The discrete Fréchet distance considers only positions of the leash where its endpoints are located at vertices of the two polygonal curves and never in the interior of an edge. This special structure allows the discrete Fréchet distance to be computed in polynomial time by an easy dynamic programming algorithm. [7]

Definition Let T be the set of possible traversals for curves P and Q. if $\|\cdot\|$ is the Euclidean distance in R^d, their DFD (metric) $d_F(P,Q)$ is defined as:

$$d_F(P,Q) = \min_{T \in T} \max_{(i_k, j_k) \in T} \left\| p_{i_k} - q_{j_k} \right\|$$

All traversals (hence the optimal one) start with pair (1, 1) and finishes with pair (m_1, m_2) .The optimal traversal with regards to the Discrete Fréchet Distance for curves P, Q of length m_1 and m_2 respectively, has *length* t:

$$\max\{m_1, m_2\} \le t \le m_1 + m_2.$$

One optimal traversal can be computed by back-propagation from the filled table of the dynamic programming algorithm for computing the Discrete Fréchet distance.

Algorithm : An effective algorithm that is used to estimate the Discrete Fréchet Distance $\delta_{d_F}(P,Q)$ for two polygonal curves P, Q , with $P = p_1, p_2, \ldots, p_n$ and $Q = q_1, q_2, \ldots, q_m$, follows :

Algorithm $d_F(P,Q)$: real;

Input: ca : array [1..n, 1..m] of real;

1. begin;
2. if $ca(i,j) > -1$ then return $ca(i,j)$
3. elsif $i = 1$ *and* $j = 1$ then $ca(i,j) := d(p_1, q_1)$
4. elsif $i > 1$ *and* $j = 1$ then $ca(i,j) = \max\{c(i-1,i), d(p_i, q_1)\}$
5. elsif $i = 1$ *and* $j > 1$ then $ca(i,j) = \max\{c(1,j-1), d(p_1, q_j)\}$
6. elsif $i > 1$ *and* $j > 1$ then
   $ca(i,j) = \max\{\min(c(i-1,1), c(i-1,j-1), c(1,j-1)), d(p_i, q_j)\}$
7. else $ca(i,j) = \infty$
8. return $ca(i,j)$;
9. end;

9.7.2 Mean Discrete Fréchet Curve

Definition Given curves P,Q , let $T=(i_1,j_1),\ldots,(i_t,j_t)$ denote any optimal traversal for the DFD, i.e.:

$$d_F(P,Q)=\max_{(i_k,j_k)\in T}\left\|p_{i_k}-q_{j_k}\right\|$$

The Mean Discrete Fréchet Curve is then defined (not uniquely) as:

$$MDFC\ (P,Q)=(p_{i_1}+q_{j_1})/2,\ldots,(p_{i_t}+q_{j_t})/2.$$

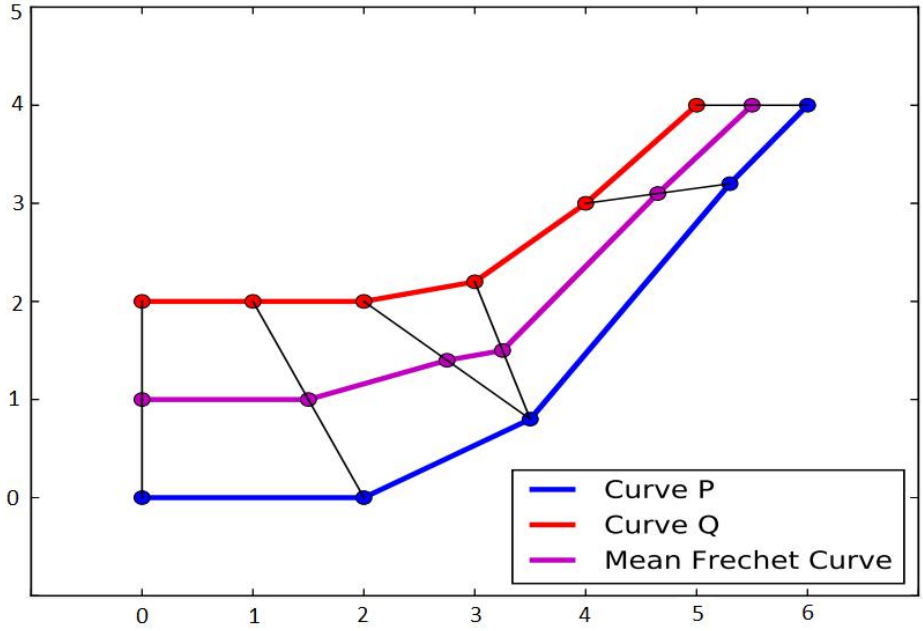


Image 10 : Estimation of the Mean Fréchet Curve for two curves P and Q

Performance : The Mean Discrete Fréchet Curve of a set of n curves is defined as the curve that minimizes the sum of *DFD*'s to all of them. Computing the exact Mean Discrete Fréchet Curve of n curves has a time complexity of $O(n^m)$,where m is the length of the longest curve. However, we can approximate the Mean Discrete Fréchet Curve in time $O(nm^2)$.

Let $h=\lfloor \lg n \rfloor$ *hence* $2^h \le n \le 2^{h+1}$

We construct a complete Binary Tree of height h, where each of the n curves corresponds to a single leaf. Recall that, a complete binary tree is a binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right.

Then, at each internal node, we compute the Mean Discrete Fréchet Curve of its two children. The final mean curve corresponds to the root of the tree.

Algorithm 9.7.2.2 : Computing the Approximate mean

1. Define a Complete Binary Tree of height h and root r.
2. randomly scatter the curves to the leaves of the tree .
3. $PostOrderTraversal(r)$

Algorithm 9.7.2.3 : $PostOrderTraversal(\cdot)$:

1. Check if the variable *node* is a leaf. If so, return the node's curve.
2. If node is not a leaf, then call the method $PostOrderTraversal$ recursively, with argument the leftChild of the node and store the result as the left curve.
3. If the right child is defined, call the method $PostOrderTraversal$ recursively, with argument the rightChild of the node and store the result as the right curve.
4. Estimate the *Mean Discrete Fréchet Curve* passing as arguments left and right curves.

9.8 Heuristics for matching 3D polygonal chains under translation and rotation

The next section takes a close look at two complicated heuristic methods used for the alignment of the polygonal chains that form the segments. [16]

9.8.1 Root-Mean-Square Deviation

The root-mean-square deviation (RMSD) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. The RMSD represents the sample standard deviation of the differences between predicted values and observed values. These individual differences are called residuals when the calculations are performed over the data sample that was used for estimation, and are called *prediction errors* when computed out-of-sample.

The RMSD serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. RMSD is a measure of accuracy to compare forecasting errors of different models for a particular data and not between datasets, as it is scale-dependent.

9.8.2 Heuristic 1

Definition 9.8.2.1 Given a 3D chain C of n vertices, the coordinates of each vertex c_i of C can be represented by a 3D vector $\overline{c}_i$.

(1) Translate to common origin by subtracting the centroid from all $x_i \in X$:

$$x_c = \frac{1}{n}\sum_{i=1}^{n} x_i \ ,$$

and by subtracting centroid y_c from all points y_i in "set" Y.

(2) Rotate to optimal alignment by 3 × 3 rotation matrix Q.

By definition, $Q^T Q = I,\ \det Q = |Q| = 1$ (orthonormal).

Recall rotated vector is vQ for row vector $v \in R^3$.

Counter-clockwise rotation in the plane about the origin by *θ*:

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \qquad \begin{bmatrix} x & y \end{bmatrix} Q = rotated\ \ vector,$$

where $Q^T Q = I,\ \det Q = |Q| = 1$.

Rotation on 3D sphere by θ, α:

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta\cdot\cos\alpha & \sin\theta\cdot\sin\alpha \\ \sin\theta & \cos\theta\cdot\cos\alpha & -\cos\theta\cdot\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} = \begin{bmatrix} x'y'z' \end{bmatrix}$$

Assume common centroid = 0, for pointsets $X, Y \in R^{n\times 3}$:

$c - RMSD(X,Y) = \min_Q |Y - XQ|_F$, for rotation matrix Q.

Lemma 9.8.2.2 : Optimizing rotation $Q \in R^{3\times 3}$ reduces to finding maximum trace:

$\max_Q tr\left(Q^T X^T Y\right)$, $Q^T Q = I_3, \det Q = 1$, where we compute rotation matrix Q.

Proof. Linear algebra calculations

Definition 9.8.2.3 : SVD (Singular value decomposition)

$$X^T Y = U \sum V^T$$

$$U^T U = V^T V = I,\ \Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} : \sigma_1 \geq \sigma_2 \geq \sigma_3,$$

where U, V, Σ are 3 × 3, singular values $\sigma_i = |e_i| \geq 0,\ e_i\ eigenvalue\ s\ of\ X^T Y.$

We search for rotation Q maximizing $tr\left(Q^T U\Sigma \cdot V^T\right) = tr\left(V^T \cdot Q^T U\Sigma\right) \leq tr(\Sigma).$

Theorem 9.8.2.4 Maximum occurs at $V^T Q^T U = I \Leftrightarrow Q = UV^T$, for rotation matrix Q.

Algorithm 9.8.2.5 ***: c-RMSD calculation***

Input: point sets $X, Y \in R^{n\times 3}$ of n corresponding points.

Output: minimum c-RMSD of translated and rotated sets

$x_c \leftarrow \sum_{i=1}^{n} x_i / n$, $y_c \leftarrow \sum_{i=1}^{n} y_i / n$.

$X \leftarrow \{x - x_c : x \in X\},\ Y \leftarrow \{y - y_c : y \in Y\}.$

$SVD: X^T * Y = U\Sigma V^T.$

Check: Confirm $\sigma_3 > 0$, where $\Sigma = diag[\sigma_1, \sigma_2, \sigma_3]$.

$Q \leftarrow U * V^T.$

if $\det Q < 0$ *then* $Q \leftarrow [U_1, U_2, -U_3] * V^T$

Return $|X * Q - Y|_F / \sqrt{n}$

9.8.3 Heuristic 2

Definition 9.8.3.1 Given a 3D chain C of n vertices, the coordinates of each vertex c_i of C can be represented by a 3D vector $\bar{c}_i$. The center c of the chain C corresponds to the vector:

$$\bar{c} = \frac{\sum_i \bar{c}_i}{n}. \qquad [16]$$

We observe that given 2 polygonal chains $A = \langle a_1, a_2, \ldots, a_m \rangle$ and $B = \langle b_1, b_2, \ldots, b_n \rangle$, if $d_F(A, B) = \varepsilon$, then we must have both $dist(a_1, b_1) \le \varepsilon$ and $dist(a_m, b_n) \le \varepsilon$.

If ε is smaller than half the minimum distance between two consecutive vertices in either A or B, then the Fréchet alignment of A and B must contain only one-to-one matches between vertices of A and B. That is, we must have m = n and, for 1 ≤ i ≤ n, $dist(a_m, b_n) \le \varepsilon$ .It follows that $dist(a_m, b_n) \le \varepsilon$, where a and b are the centers of A and B, respectively.

The observation above suggests that we can use the three points, the two endvertices and the center, as the reference points for each chain. For two polygonal chains with a small discrete Fréchet distance, their corresponding reference points

must be close. In general, the position and orientation of each polygonal chain is determined by the positions of its three reference points. [16]

We have the following heuristic for matching A and B under translation and rotation :

(1) Translate B such that the center a of A and the center b of B coincide.

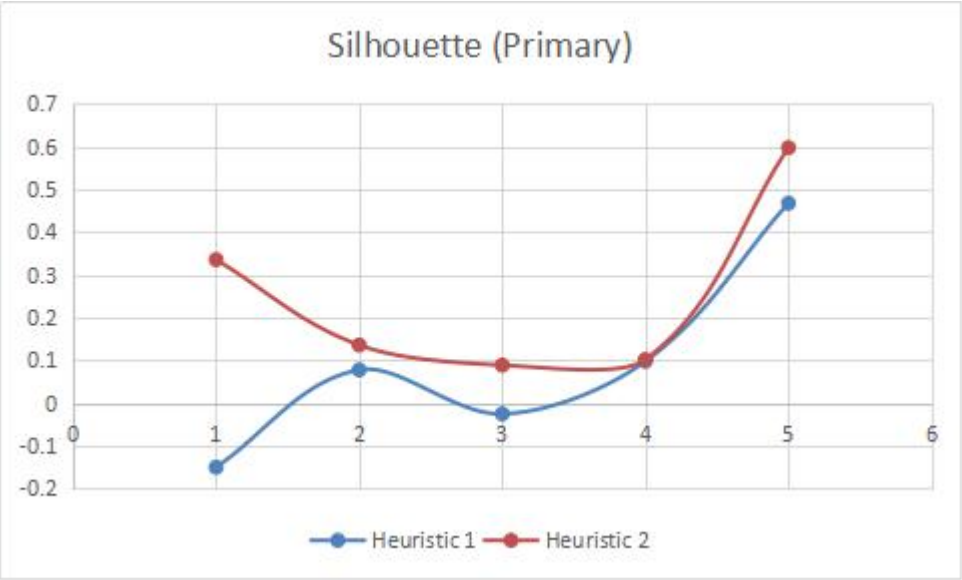
(2) Rotate B around b such that the two triangles $\Delta a a_1 a_m$ and $\Delta b b_1 b_n$ are coplanar, and such that the two vectors $\frac{\overline{a}_1 + \overline{a}_m}{2} - a$ and $\frac{\overline{b}_1 + \overline{b}_n}{2} - b$ have the same direction.

(3) Rotate B for around the axis through its two randomly chosen vertices. The rotation method that is used is the one described in the previous section (heuristic 1- *step 2: rotation*).

9.8.4 Results

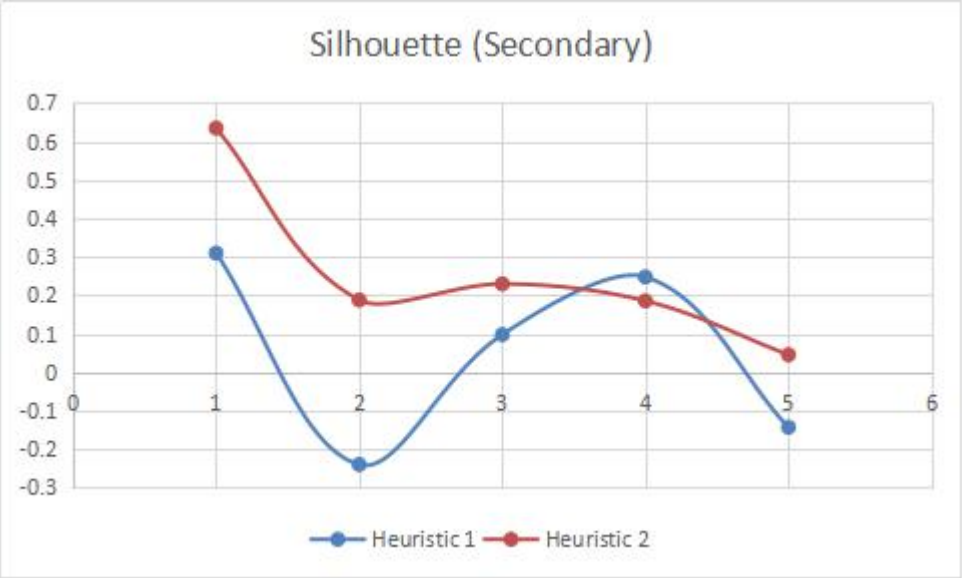
Generally, by applying the 2nd heuristic, it is possible to match two polygonal chains with m and n vertices in R^3 .The following charts illustrate the produced outcomes that showing its effectiveness in matching two similar polygonal chains.

Contrary to the second heuristic, the first one indicates that the alignment of the segments, after translation and rotation, is inferior and less effective. The silhouette values show that the computation of the discrete Fréchet distance for the first implementation is accomplished in a more efficient way. For similar polygonal curves, we achieve lower Fréchet distances and as a result, the distribution of the segments is more uniform and distinguishable. These circumstances are mandatory for a productive and sufficient clustering analysis.



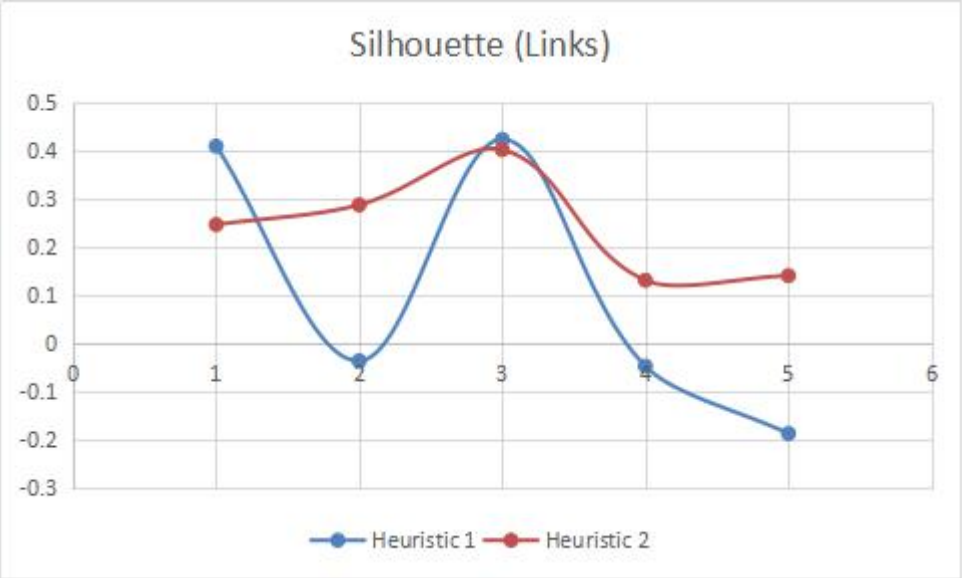
Silhouette per Cluster for Heuristics 1 and 2 (Primary)

The values of silhouette for each cluster. Applied for primary roads.



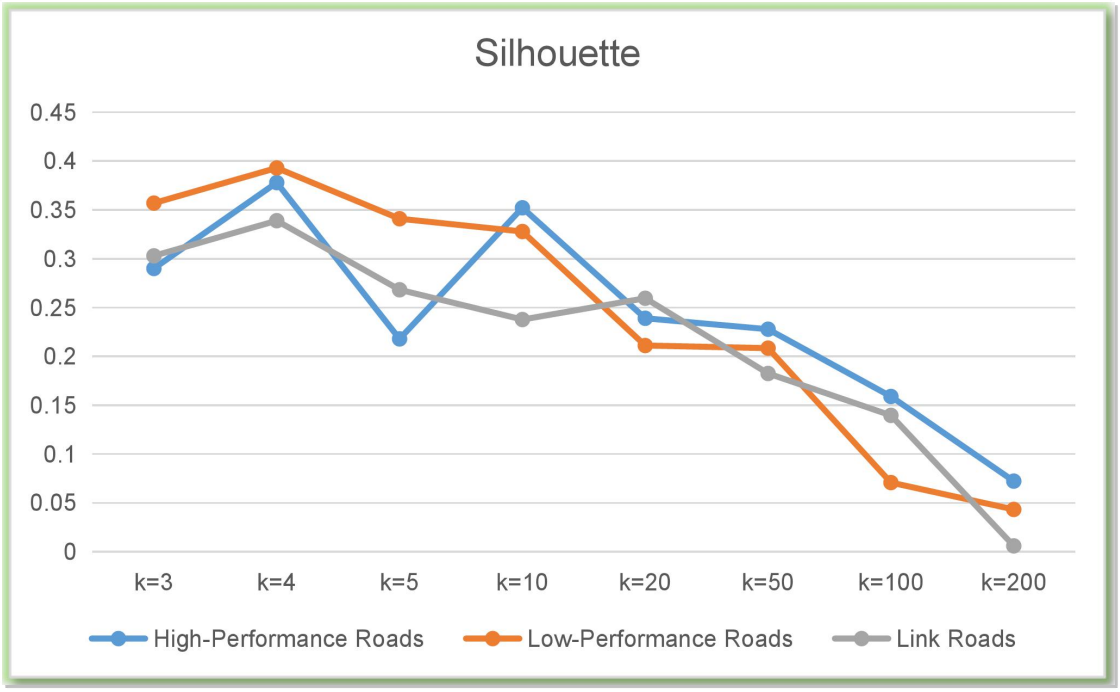
Silhouette per Cluster for Heuristics 1 and 2 (Secondary)

The values of silhouette for each cluster. Applied for secondary roads.



Silhouette per Cluster for Heuristics 1 and 2 (Links)

The values of silhouette for each cluster. Applied for link roads.



Silhouette per number of clusters

The values of silhouette for different number of clusters. For higher number of clusters (k) we get lower value of silhouette. This measurement suggests that the amount of different segment types (- polygonal curves shapes) can be distinguished to few categories, fact that explains that if we increase the number of cluster, clustering becomes less effective accurate.

10. CONCLUSIONS

In this thesis, we have implemented and performed experiments on a set of various algorithms, in order to divide the roads into segments and group the generated data in complicated structures. We define the classifications of the highways based on their morphology and their attributes and we manage to represent them as polygonal curves.

We designed two algorithms that transform roads into segments. These generated objects can be more flexible and can be easily utilized by our experiments. We observed that each segmentation method is affected by the road classification. In more details, roads that are considered as high-performance don't display any significant variation in their curvature. However, those roads can be divided into segments by looking for junctions. The usage of the proposed formula leads to better results, as the number of points and the length in the polygonal curves are taken into consideration. For the low-performance or links roads, where the number of junctions is minimal, segmentation by curvature combined with our formula is considered as the best option.

Furthermore, we develop approximation methods that group the data by comparing their similarity. Both locality-sensitive hashing and clustering achieved to distinguish the road segments by using similar approaches. The projection of polygonal curves to grid displays sufficient results for the road identification and comparison. Also, regarding clustering, we presented two different heuristic methods for curve alignment. The best results were achieved when the second heuristic was performed.

The results presented above are encouraging and can be improved. The parallelization of the segmentation methods and the approximation algorithms we analyzed could offer better time measurements. Moreover, a new heuristic for the translation and the rotation of the polygonal curves can be designed and implemented, so that we can determine which of them can provide better experimental results for the curve alignment. Future work on the road segmentation can include experiments focused on highways that are related to traffic jam or their location at a city.

ABBREVIATION – ACRONYMS

OSM	Open Street Map
WGS	World Geodetic System
XML	eXtensible Markup Language
HTTP	HyperText Transfer Protocol
URL	Uniform Resource Locator
GPS	Global Positioning System
API	Application Programming Interface
ECEF	Earth-Centered, Earth-Fixed
NNS	Nearest Neighbor Search
LSH	Locality-Sensitive Hashing
DFD	Discrete Fréchet Distance
RMSD	Root-Mean-Square Deviation
SVD	Singular value decomposition

IMAGES DIRECTORY

REFERENCES

[1] Leonard Kaufman, Peter J. Rousseeuw, “Clustering by means of Medoids”, 1987.

[2] James Newling, Francois Fleuret, “K-Medoids for K-Means Seeding”, 2016.

[3] Hartigan, J., & Wang, M. “A K-means clustering algorithm. Applied Statistics”, 28, 100–108, 1979.

[4] Hastie, T., Tibshirani, R., & Friedman, J., “Elements of statistical learning”, 2001.

[5] Adam Franco, https://github.com/adamfranco/curvature .

[6] Springer Verlag. Jain, A., & Dubes, R. “Algorithms for clustering data”, 1988.

[7] Hae-Sang Park, Chi-Hyuck Jun, “A simple and fast algorithm for K-medoids clustering”, Expert Systems with Applications, Vol 36, Else- vier, 2008.

[8] Anne Driemel, Francesco Silvestri, "Locality-sensitive hashing of curves", 2017.

[9] A.K. Jain, M.N. Murty, and P.J. Flynn, “Data Clustering: A Review”, ACM journal of Computing Surveys, Vol 31, pp: 264-323, NewYork, 1999.

[10] Sergios Theodoridis,”Pattern Recognition” ,chapter 14, 2003.

[11] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In Proceedings of the 33rd international conference on Very large data bases, VLDB ’07, pages 794–805, 2007. ISBN 978-1-59593-649-3.

[12] Thomas Eiter and Heikki Mannila, “Computing Discrete Frechet Distance”, 1994.

[13] Nicholas Jing Yuan, Yu Zheng, Xing Xie, “Segmentation of Urban Areas Using Road Networks”, 2012.

[14] Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: Proc. KDD. pp. 63–72 (1999).

[15] Open Street Map wiki, https://wiki.openstreetmap.org/wiki.

[16] Minghui Jiang, Ying Xu, Binhai Zhu, “Protein Structure, Structure Alignment with Discrete Frechet Distance”, 2008.

[17] Olson, Donald Karl, “Calculation of Geodetic Coordinates from Earth-Centered, Earth-Fixed Coordinates”, 2003.