National Technical University of Athens

Crowd-sourced vs proprietary routing web APIs

A Web GIS app for comparing routing web APIs over proprietary and crowd-sourced road network data

By Dimitris Leader

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science (M.Sc.) in Geoinformatics

National Technical University of Athens

School of Rural and Surveying Engineering

Athens, February 2020

Supervised by Vasileios Veskoukis

Table of Contents

Acronym List

ACRONYM	MEANING
API	Application Programming Interface
AOI	Area of interest[1]
CRS	Coordinate Reference System
dH	Hausdorff Distance
ELSTAT	Hellenic Statistical Authority
GIS	Geographical Information System
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
OS	Operating System
OSM	OpenStreetMap platform
OSRM	Open-source Routing Machine
QGIS-LTR	QGIS Long Term Release (most stable release)
VGI	Volunteered Geographic Information

[1] The term "area of interest" is used interchangeably and in cases refers to all four municipalities studied while in others refers to one specific municipality. The meaning depends on the context it is used in.

List of Figures

List of Tables

List of Equations

List of Graphs

Page intentionally blank

Abstract

Travelling has been at the epicentre of human activity for millennia. In the modern era, global navigation satellite systems, mobile phones, and the Internet have all revolutionised the way we seek optimum ways to travel to our destinations. We now have reached the point where we have a selection of routing services to choose from. Their providers depend on geographical data collections. Two major data collections have dominated this routing service industry. Google, who provides its map services through various business models and pricing schemes and the open-source community and more specifically the OpenStreetMap platform relying on open data and the crowdsourcing model. The desire to perform comparative tests between the two is unavoidable and serves several purposes.

This paper describes the steps of a project that aims at creating a tool that provides an attempt to such a comparison. Specifically, it aims at creating an application in the form of a web GIS App that will display pairs of points – an origin and a destination - and the routes provided by the equivalent services of Google and the Open-source Routing Machine. These are displayed on a common map background with data on the side for the comparison. For this to be achieved, three major steps have been identified as key processes leading to implementation. The first phase contains the process for the creation of the points that will be candidates for a route in a designated area of interest. The points need to be random but must also be representative of the area and spatially well distributed. The probability of an incident must be factored in. Additionally, the population of points must be adequate to cover several cases. The second phase involves the creation of an automated process that will create the collection of requests for the services. This tool after creation and execution of the request collects the replies and handles them so that mappable objects are created in the form of GeoJSON format along with files that hold additional data useful to the App. The third and final stage involves the creation of the viewing tool, the web App, that will display the two suggested routes. It also provides the relevant data such as length and duration as well as the possible differences between the points requested and the ones used by the service. In addition, it uses the Hausdorff distance metric between the two route vectors to illustrate the degree of divergence between them. To facilitate the identification of cases that appear interesting, tables with colour maps of the Hausdorff distances as well as distribution graphs are provided.

The paper ends with a demonstration of examples of conclusions that can be derived after examining a sample of pairs and their relevant routes.

Page intentionally blank

Ελληνική Περίληψη – Greek Summary

Η μετακίνηση αποτελεί αναπόσπαστο κομμάτι της ανθρώπινης ύπαρξης. Ο σύγχρονος τρόπος ζωής όχι μόνο επιτρέπει την μετακίνηση του ανθρώπου αρκετά έξω από τη κλίμακα της οντότητάς του, αλλά ένα άτομο καλείται σχεδόν καθημερινά να καλύπτει αρκετά μεγάλες αποστάσεις και συχνά να περιηγηθεί σε άγνωστες για αυτόν περιοχές. Το ερώτημα της βέλτιστης διαδρομής δεν αποτελεί πλέον ζητούμενο αλλά ανάγκη που όμως τα τεχνολογικά επιτεύγματα του τρέχοντος αιώνα έχουν καλύψει με πληθώρα λύσεων. Σήμερα δεν διατίθενται απλώς υπηρεσίες πλοήγησης με λεπτομερείς περιγραφές μετάβασης από το ένα σημείο στο άλλο αλλά ολοκληρωμένες λύσεις παροχής υπηρεσιών δρομολόγησης από τις οποίες μπορούμε να διαλέξουμε και να παρέχουμε τις δρομολογήσεις ως υπηρεσίες οι ίδιοι. Οι πάροχοι αυτών των υπηρεσιών φυσικά στηρίζουν τα αποτελέσματα σε γεωγραφικά δεδομένα τα οποία είτε συλλέγουν, είτε συντηρούν. Δύο βασικοί πάροχοι που στηρίζονται σε εκ διαμέτρου αντίθετα επιχειρηματικά μοντέλα είναι η Google που κρατά τα δεδομένα ιδιωτικά και έχει μία τεράστια γκάμα υπηρεσιών που καλύπτουν όλο το φάσμα τιμολογιακής πολιτικής και η Κοινότητα Ανοιχτού Λογισμικού/Δεδομένων που έχει βρει την έκφρασή της στη πλατφόρμα του OpenStreetMap όπου όλα είναι ανοιχτά και συντηρούνται σε εθελοντική βάση. Η επιθυμία για την σύγκριση των δύο είναι αναπόφευκτη και μπορεί να εξυπηρετήσει πλήθος σκοπών μεταξύ των οποίων και το εντοπισμό διαφορών στα υποκείμενα δεδομένα.

Η παρούσα Διπλωματική Εργασία επιχειρεί μια τέτοια σύγκριση μέσω της δημιουργίας εργαλείων που θα μπορέσουν να αναδείξουν τις διαφορές των υπηρεσιών web δρομολόγησης ανάμεσα στην υπηρεσία της Google που διατηρεί κλειστά δεδομένα και της πλατφόρμας του OSM που διατηρεί ανοιχτά δεδομένα. Πιο συγκεκριμένα, στοχεύει στη δημιουργία μιαw Web GIS εφαρμογής που θα προβάλει ζεύγη σημείων, ένα σημείο αφετηρίας και ένα προορισμό, και τις διαδρομές που προτείνουν οι δύο πάροχοι των δρομολογήσεων. Οι διαδρομές προβάλλονται σε ένα κοινό χάρτη ως υπόβαθρο και συνοδεύονται από τα στοιχεία που αφορούν την κάθε διαδρομή που προτείνει ο κάθε ένας καθώς και συγκριτικά μεγέθη. Στόχος της εργασίας δεν είναι η ανάδειξη του καλύτερου παρόχου αλλά η δημιουργία ενός εργαλείου που βοηθάει το χρήστη να εξάγει τα δικά του συμπεράσματα. Για να επιτευχθεί το ζητούμενο απαιτείται η υλοποίηση τριών διαφορετικών σταδίων ανάπτυξης που το καθένα στοχεύει σε διαφορετικό διακριτό τομέα ή δομικό στοιχείο.

Το πρώτο στοιχείο είναι η δημιουργία των σημείων που θα αποτελέσουν τα σημεία εκκίνησης και τερματισμού της κάθε διαδρομής. Τα σημεία πρέπει να προκύψουν με τυχαίο τρόπο ενώ

ταυτόχρονα δεν πρέπει να παραβλέπεται η δημοφιλία μιας περιοχής έναντι άλλων. Επίσης πρέπει να εμφανίζουν μια ικανοποιητική χωρική διασπορά.

Το δεύτερο στοιχείο είναι η αυτόματη διαδικασία παραγωγής, διαχείρισης και επεξεργασίας των διαδρομών. Το πλήθος των διαδρομών αυξάνεται με εκθετικό ρυθμό σε σχέση με το πλήθος των σημείων, γεγονός που οδηγεί στην απαίτηση αυτοματοποιημένων διαδικασιών. Αυτές οφείλουν να αναλάβουν τη δημιουργία και εκτέλεση των ερωτημάτων δρομολόγησης προς τις υπηρεσίες, την αποθήκευση των πρωτογενών απαντήσεων καθώς και την περαιτέρω επεξεργασία τους με στόχο την μεταποίηση πληροφοριών για την προβολή τους στην εφαρμογή.

Το τρίτο και τελευταίο στοιχείο είναι η δημιουργία της εφαρμογής που θα προβάλει τα αποτελέσματα σε ένα εύκολο και εύχρηστο διαδικτυακό περιβάλλον. Στόχος είναι η ταυτόχρονη προβολή των δύο διαδρομών σε κοινό υπόβαθρο μαζί με πίνακες που προβάλουν τα ποσοτικά χαρακτηριστικά του μήκους, της διάρκειας, των διαφορών τους σε αυτά, την απόσταση του δοσμένου σημείου από το σημείο πραγματικής δρομολόγησης στο χάρτη καθώς και μιας μετρικής που καταδεικνύει τον βαθμό απόκλισης των δύο διαδρομών.

Ως περιοχές ενδιαφέροντος για εξέταση επιλέχθηκαν οι 4 Καποδιστριακοί Δήμοι που περιβάλλουν το όρος Πεντέλη, Κηφισιά, Νέα Ερυθραία, Εκάλη, και Διόνυσος επειδή είναι προάστεια, με συνοχή και με εντελώς διαφορετικά ποιοτικά χαρακτηριστικά μεταξύ τους. Οι διαδικασίες που εφαρμόστηκαν για τα παραπάνω στοιχεία είναι οι εξής:

Για το πρώτο, τη δημιουργία σημείων, ακολουθείται μια επαναληπτική διαδικασία που στηρίζεται σε μια αρχική αποτύπωση του οδικού δικτύου σε μια μορφή νέφους σημείων. Το νέφος αυτό δημιουργείται από τους κόμβους που προκύπτουν από τον τοπολογικό τεμαχισμό του οδικού δικτύου και βοηθάει στον να αποτυπώσει την πυκνότητα μιας περιοχής έναντι άλλων. Στη συνέχεια εφαρμόζεται μια επαναληπτική διαδικασία ομαδοποίησης με τον αλγόριθμο k-means για ένα αρχικό τυχαίο k, την ανάδειξη ενός κεντρικού σημείου για κάθε ομάδα, την προβολή του στο οδικό δίκτυο και τη δημιουργία ενός πίνακα αποστάσεων εγγύτερων σημείων για κάθε ένα από αυτά. Από τα στατιστικά του πίνακα εξετάζεται η μέση απόσταση καθώς και η στατιστική κατανομή των αποστάσεων και κρίνεται εάν το δείγμα ικανοποιεί τα κριτήρια. Εάν δεν τα ικανοποιεί η διαδικασία επαναλαμβάνεται με διαφορετικό k ανάλογα με τις επιδιώξεις μας, μεγαλύτερο εάν θέλουμε μικρότερες μέσες αποστάσεις ή μικρότερο για το αντίθετο. Όταν επιτευχθεί τελικό σύνολο, παράγεται αρχείο τύπου csv με τις συντεταγμένες του κάθε σημείου.

Για το δεύτερο στοιχείο, γράφτηκαν αρχεία κώδικα σε Python τα οποία παίρνουν ως είσοδο το αρχείο σημείων, δημιουργούν ερωτήματα HTTP request για κάθε μία από τις υπηρεσίες και τα εκτελούν. Στη συνέχεια αρχειοθετούν τις απαντήσεις που είναι της μορφής JSON και τις επεξεργάζονται. Η επεξεργασία περιλαμβάνει τη δημιουργία των χωρικών αναπαραστάσεων σε μορφή GeoJSON, τον υπολογισμό διαφορών στα ποσοτικά χαρακτηριστικά, την συνολική στατιστική ανάλυση, τον υπολογισμό της μετρικής και την αποθήκευση όλων των παραπάνω σε αρχεία JSON για τη χρήση τους από το εργαλείο της προβολής. Ως μετρική για το βαθμό απόκλισης μεταξύ των δυο διαδρομών χρησιμοποιήθηκε η απόσταση Hausdorff.

Για το τρίτο στοιχείο, την εφαρμογή, δημιουργήθηκε σελίδα που στηρίζεται στη χαρτογραφική πλατφόρμα Leaflet. Δημιουργήθηκε μια ιστοσελίδα η οποία δίνει την δυνατότητα επιλογής της περιοχής ενδιαφέροντος, η οποία προβάλει τις θέσεις των σημείων στο χάρτη και κατόπιν επιλογής των 2 σημείων απεικονίζονται με διαφορετικό χαρτογραφικό συμβολισμό οι δύο προτεινόμενες διαδρομές. Ταυτόχρονα, σε παρακείμενο πίνακα προβάλλονται τα ποσοτικά στοιχεία της κάθε διαδρομής καθώς και οι διαφορές τους. Επιπλέον, δίνεται η δυνατότητα της κατάδειξης των σημείων στα οποία προκύπτει η μέγιστη απόκλιση της απόστασης Hausdorff αλλά και πινάκων με όλες τις αποστάσεις και την στατιστική κατανομή τους για την περιοχή ώστε να μπορεί να γίνει καλύτερη ερμηνεία των ευρημάτων. Για τη δημιουργία της σελίδας χρησιμοποιήθηκαν οι απαραίτητες τεχνολογίες HTML και CSS ενώ όλη η λειτουργικότητα γράφτηκε σε κώδικα JavaScript.

Τέλος, στη παρούσα εργασία γίνεται επίδειξη του τρόπου με τον οποίο μπορούν να ερμηνευτούν τα στοιχεία μέσω ενδεικτικών περιπτώσεων σύγκρισης και παρουσιάζεται ο τρόπος εξαγωγής συμπερασμάτων. Παράλληλα, παρουσιάζονται ιδέες για περαιτέρω μελέτη και ανάπτυξη καθώς επίσης και ιδέες για εφαρμογή σε ένα μελλοντικό εργαλείο σύγκρισης.

Page intentionally blank

1. Introduction

Man has been on the move from the very first moment of existence on the planet. In the beginning in search of food, later on in exploration of the world in search of resources and later even as a hobby. In fact, it was a Sunday's pastime hobby that instigated the birth of graph theory [1]. Towards the end of the 20th century, an abundance of technological advancements laid the foundation for the routing services we enjoy today. Geoinformatics gave us a way to model our world, global satellite systems enabled us to find our whereabouts accurately on the planet and the Internet gave us a fast highway of accessing the required knowledge and tools. By the end of the first decade in the 21st century, vehicle navigation services had become a service that all people have access to and are free. Web routing APIs not only give us solutions to routing queries but enable us to provide the means to offer such services to others on a custom application of our own.

With the pace of technological advancements ever increasing, new concepts and ideas follow and add to the availability of services. Business models where proprietary data is offered without having to pay, such as Google's street maps, and expertise work provided voluntarily on a global scale through the idea of crowdsourcing [2] and gamification [3], are examples of such concepts – not technological advancements as such – that have given us more than a service, they have given us choice. But choices come with dilemmas since we all want to make the best choice among those available. Choosing a route to move from A to B relies on several factors and the decision whether it was a "good" choice or not, is very often a matter of personal taste. It could rely on factors such as the scenery, the choice of roads made, the number of manoeuvres or the amount of traffic we encounter. On other occasions, it could be completely objective like when we need to reach a certain destination by a certain time or even in an important situation where human lives are at risk. Situations like these are for example when the route is part of an evacuation plan or when a first responder like an ambulance or a fire engine needs to get to an incident. In this context, investigating ways and tools that can give us means to compare our choices, are crucial.

This thesis attempts such an investigation and suggests a simple application that can provide objective insight before that choice is made. By performing a series of simultaneous route requests to two different service providers, comparing the results and projecting key quantitative facts and values with a map of the routes, we hope it will provide the minimum basis of such a comparison. The candidates for the provision of the routes for the comparison could be no others than the two global giants of road maps and routing services, Google Maps [4] and OpenStreetMap [5]. Google has a very refined business model in which a large number of services are provided for free, others are charged but much information, such as the road data that it maintains for its Maps, is kept private. OpenStreetMap is on the other hand by definition and name open and transparent by nature. The data is available not only for viewing but for changing as well. Identifying the advantages and disadvantages that each one brings is at the core of the investigation and to reach a conclusive and reliable result, a particular methodology needs to be established. This methodology provides a case study area (or a number of areas), with descriptors that provide analysis of the road data, a process for the creation of the routes to be compared, and specific indicators upon which an evaluation can be made. The proposed methodology can be broken down as follows:

- the sample area is selected with the aim of creating random pairs of points representing points of origin and points of destination in that area
- simultaneous route queries are requested for the same pair of points from the two service providers
- the results are analysed, compared in terms of their attributes, and displayed with the aid of a map application for direct evaluation and comparison

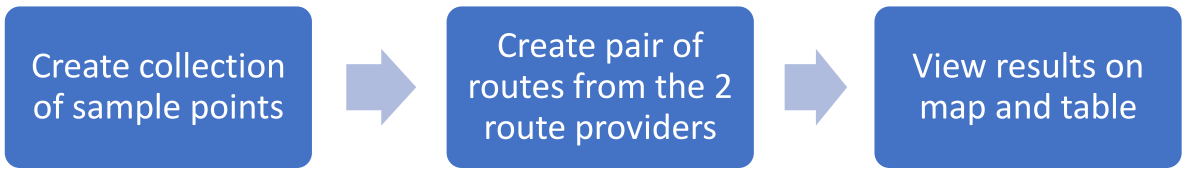

Create collection of sample points
Create pair of routes from the 2 route providers
View results on map and table

Figure 1: The 3 main phases of development.

This paper has been organised in a way that presents the 3 phases, explains the challenges and available choices, and presents the viewing tool that was developed.

The area for the case study was chosen from the list of municipality units in the Attica basin so as to investigate suburban environments and have as wide a range of road diversity and environments as possible. The collection of the sample points needs to be carefully created so that the pairs are enough to produce meaningful and insightful routes. The closer the points are, the higher the chance of the routes being similar there is. On the other hand, longer routes will have a bigger distance between the two points which can limit the number of points over the area thus reducing the degree of examination of that area. Furthermore, the aim is to try and simulate realistic events of origin and destination which requires a certain degree of consideration of the possibility of the events happening. For example, routes originating or ending in the middle of a mountainous road leading to a fire lookout have less of a chance happening as one from the city centre to one of the neighbourhoods or the local hospital. The routes created cannot be between completely random points. This research proposes a systematic way for the creation of points on a road network that involves a semi-stochastic process. A semi-stochastic process will provide a balance between the deterministic nature resulting from location popularity and the stochastic nature deriving from the 50/50 chance of incidents occurring.

The number of points needs to be sufficient to provide coverage of the area and a variety of results. Each point will have to connect to every other point so that all possible routes are calculated for the total of points. This results in an exponential increase in the number of routes. The exact number is expressed by the formula

$$r = p^2 - p$$

Equation 1: Number of routes r involving locations p

where r is the number of routes and p is the number of points and $(p \geq 2)$ must apply. This leads to the conclusion that the process of requesting the routes will have to be automated since to manually perform these requests and record the resulting data would be time-consuming, tiresome, and error-prone. Especially since the task of requesting a route has to be performed twice, once for each route provider. These services are also provided in the form of a web Application Programming Interface (API). According to [6]:

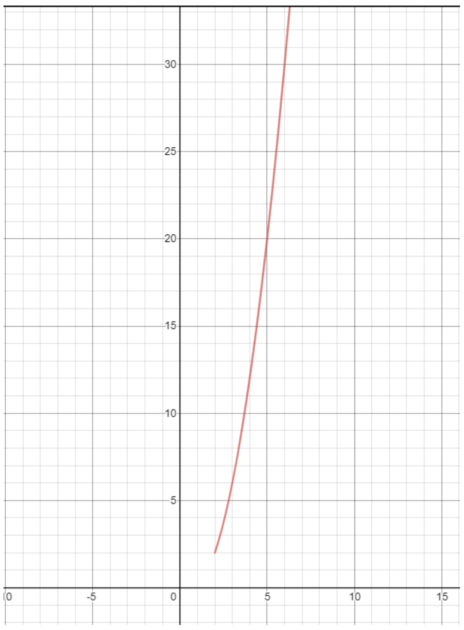


Figure 2: Graph displaying the exponential number of routes per extra point

> *A web API is the pattern of HTTP requests and responses that is used to access a website that is specialized for access by arbitrary computer programs, rather than (or as well as) web browsers used by humans.*

Both route providers offer the services in the form of an API which facilitates the task of creating the routes for any number of points. Furthermore, since the process can be coded, it can be directly integrated into the next step of the analysis. The analysis step is necessary for two reasons:

a) it converts the reply to a format that can be humanly understood. Humanly understood refers to all the elements of the route, including its map projection[2]

b) it performs calculations on data such as distance and time to provide information for the comparison

Since routes move in the two-dimensional space of the terrain, one metric for the comparison is each one's length. An additional metric that would illustrate how apart these routes are in space could be implemented to facilitate the comparison. The thesis proposes the use of the Hausdorff Distance used in pattern recognition and computer vision. Pattern recognition and computer vision rely on determining the extent to which one shape differs from another, according to [7]. This is like the object of the comparison which aims at identifying how similar the shape of one route is to another.

The final step includes the development of a viewing tool. As mentioned earlier, routes are very often a matter of personal preference. Despite the ability to automatically estimate which provider gives us the shortest or the fastest routes, it is impossible to evaluate each route without human oversight. The need for a medium on which the results can be displayed was apparent. Because understanding the value of a route does not require expertise, the idea of using expert software such as GIS was immediately discarded. The obvious direction to move towards was the development of a web GIS App that allows as many people as possible to view the routes and data and make an evaluation. The choice of a web GIS App provides the two basic advantages of simplicity of creation due to a selection of libraries to choose from and user-friendliness. A web GIS App cuts across platforms and operating system divides thus enabling easy access over a variety of systems and devices.

This paper is organised into 5 additional chapters:

Chapter 2 presents the general options and choices made on aspects involved. These aspects include the study area, the tools used in the development and methodologies involved.

Chapter 3 presents a detailed account of the first phase which involves the creation of the points that represent the route incidents.

Chapter 4 details the processes involved in the creation of the routes and the files with the analysis that support the viewing App.

Chapter 5 is a brief presentation of the tools and methods used in the creation of the App.

Chapter 6 contains conclusions and future suggestions.

[2] Projection is used in the context of display i.e. shown on a map and not in the cartographic context meaning a way to represent the globe's curved surface on a flat two-dimensional surface.

2. Study area, tools, and methodologies

This chapter covers a detailed presentation of the following topics. In the first subchapter, an overview of the area of study is presented. The area of interest involves four municipal units that surround Mount Pentelicus (or as it is commonly called in Greece, Penteli) in the north of Attica and the outskirts of Athens. The reasons for the choice of the areas are explained and a detailed presentation together with a statistical analysis of each municipality is provided. The individual municipality presentation follows the presentation of tables with comparative figures for all four municipalities. The second subchapter focuses on the presentation of the various sources of data that were used to extract the information and to perform the necessary spatial and non-spatial statistical tasks. In the third subchapter, a brief presentation of the tools that were used is presented together with the choice of methodologies and techniques. Finally, a brief explanation of the Hausdorff distance metric is presented.

2.1 Municipalities of choice

The basic concept was based around the question of how navigable a suburban area is and whether the data and mechanisms from open sources can be used for reliable route analysis compared to the solutions the leading provider Google gives. The areas of interest would have to be suburban areas where the routes would not only be based on road classification and direction but would also include a variety of surface types such as unpaved roads or service roads and where a variety of vehicle access such as steps, paths, footways etc is an issue. In this context and having the conviction that the area of interest should not be some arbitrary area but an administrative unit, we were limited to candidates on the outskirts of the city and on the foothills of one the mountains that form the Attica basin. The four adjacent municipalities that surround Mount Pentelicus seemed to fit the profile of municipalities for our research. They are all suburban areas and contain parts that are densely populated, they provide a satisfying variety of road classification from motorways to cycleways and footpaths, they display a wide range in population density and finally, two out of four contain large uninhabited areas. A small description accompanied by a small table of statistics for each area is given in the following chapters. They are presented in alphabetical order. Before the analysis of each area, tables that display all the facts and figures for all the municipal units for comparative reasons are provided.

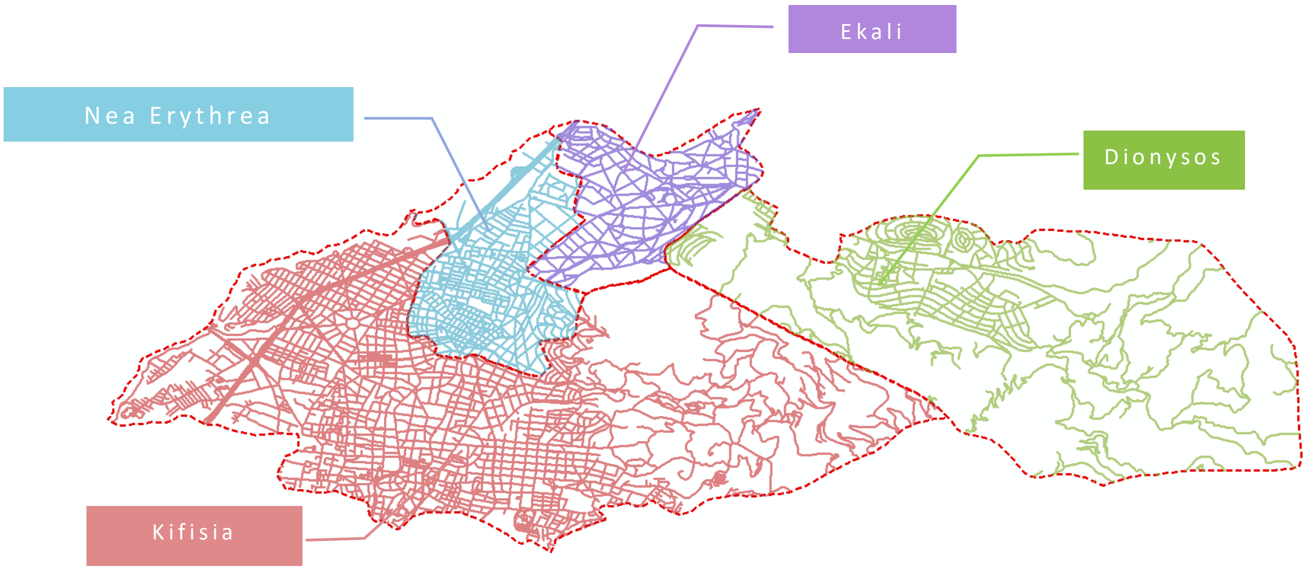


Figure 3: Map of the four municipality units and the road network each contains

Table 1: Road classification distribution (count and length) per municipality and totals

			Dionysos		Ekali		Kifisia		Nea Erythrea		Sums	
Highway Class	Level	%	#[3]	ℓ	#	ℓ	#	ℓ	#	ℓ	#	ℓ
Motorway	1	2.06%			8	678	52	8510	15	4107	75	13295
Motorway Link	1	0.48%					18	2371	6	720	24	3091
Primary	3	1.31%			39	3383	58	3538	29	1548	126	8470
Primary Link	3	0.02%			2	112	3	38			5	150
Secondary	4	6.84%	52	8643	19	1229	367	25893	116	8457	554	44222
Secondary Link	4	0.38%	3	160	4	168	40	1781	9	324	56	2434
Tertiary	5	10.93%	60	7236	97	8634	565	47693	138	7101	860	70665
Tertiary Link	5	0.06%			2	53	9	293	1	18	12	364
Unclassified	6	3.18%	21	9679			83	8571	24	2275	128	20525
Residential	7	54.16%	414	48938	483	48885	2178	189047	830	63172	3905	350042
Track	7	12.66%	107	42188	2	201	164	38333	12	1105	285	81827
Service	7	2.81%	5	315	15	1789	251	13692	63	2364	334	18159
Footway	7	1.68%	1	8			206	10698	3	165	210	10872
Path	7	1.58%	16	5953			39	4253			55	10206
Living Street	7	1.18%					53	3958	58	3648	111	7606
Pedestrian	7	0.38%					16	1247	25	1196	41	2443
Cycleway	7	0.26%							25	1685	25	1685
Steps	7	0.03%	2	59			13	160			15	220
SUMS			**681**	**123180**	**671**	**65132**	**4115**	**360077**	**1354**	**97885**	**6821**	**646273**

[3] # is count of edges and ℓ is sum length of edges in metres.

Table 2: Population density and road allocation statistics per municipality

Municipality	Population	Pop/km²	Pop/km	m/Resident
Dionysos	5338	264.72	43.34	23.08
Ekali	4520	1045.55	69.40	14.41
Kifisia	38282	1466.12	106.32	9.41
Nea Erythrea	12489	2481.51	127.59	7.84

Table 3: Road density statistics per municipality

Municipality	Area km²	Roads km	Edges	Avg. Length	m/km²
Dionysos	20.16	123179.57	681	180.88	6108.72
Ekali	4.32	65131.60	671	97.07	15066.00
Kifisia	26.11	360076.83	4115	87.50	13790.15
Nea Erythrea	5.03	97885.42	1354	72.29	19449.45

2.1.1 Dionysos

Dionysos used to be an independent local community according to the previous administrative plan in effect. In 2011 it was upgraded to a municipal unit and together with six other municipal units it forms the greater Dionysos municipality that bears the same name. Because the greater municipality of Dionysos extends beyond Athens we have limited our research to the municipal unit that covers the northeast side of the mountain. It includes a settlement that according to the last census of 2011 has a population of 5338 inhabitants, covers a land area of about twenty square kilometres and is at an average altitude of 500 metres. It is adjacent to 5 other municipalities. It is one of the newest suburbs as it was formed towards the end of the 19th century when workers of the nearby quarries settled down in the area. Until 1975 it was part of the Stamata community and became an independent community in 1979. The following table provides statistical insight into the area. The population data is from the official census results and a lot of the figures are calculated in our GIS as outlined in chapter 2.2.

Table 4: Dionysos population fact sheet

Dionysos Fact Sheet

Population	**5338**
Area km²	**20.16**
Pop/km²	**264.72**
Roads km	**123179.57**
No of Edges	**681**
Average Edge Length	**180.88**

Road Length to Area ratio m/km²	**6108.72**
Road length per resident m/resident	**23.08**
Residents / km	**43.34**

Dionysos is the second largest area but the most sparsely populated of the four with an average of 264.72 residents/km², one-quarter of the value the second sparsest area of the four displays. It also has the second-longest road network covering 123 kilometres in total. This is mostly due to its size and not its density. This becomes apparent by the fact that its average edge length is almost double compared to the other three and the length over area ratio has an index of approx. 6km per square kilometre (6108.72). Most of the area in Dionysos is uninhabited and mountainous and this is reflected in the road classification breakdown presented in the following table.

Table 5: Dionysos road class statistics

Dionysos road data

Highway class	Count	Length (m)	% length	Level
Residential	414	48938	39.73%	7
Track	107	42188	34.25%	7
Unclassified	21	9679	7.86%	6
Secondary	52	8643	7.02%	4
Tertiary	60	7236	5.87%	5
Path	16	5953	4.83%	7
Service	5	315	0.26%	7
Secondary link	3	160	0.13%	4
Steps	2	59	0.05%	7
Footway	1	8	0.01%	7
SUMS	**681**	**123180**		

The figures in the above table illustrate the type of area that is under examination. The high percentage of unpaved roads – a third of the total – shows it is an area that has a large proportion of mountainous landscape. The relatively high degree of paths indicates large parks or forested land coverage that adds to the sparse suburban image. The high percentage of low-level road classification such as residential and unclassified shows that most of the roads are not to accommodate drive-by traffic nor that the area displays commercial characteristics or operates as a traffic hub. It is a highly suburban area that lacks the characteristics of a heavily urban environment. This will probably yield routes that will frequently use the same few central roads for long-distance crossing. It is on the other hand ideal to examine the level of detail of the data each route provider has on outlying and infrequent areas.

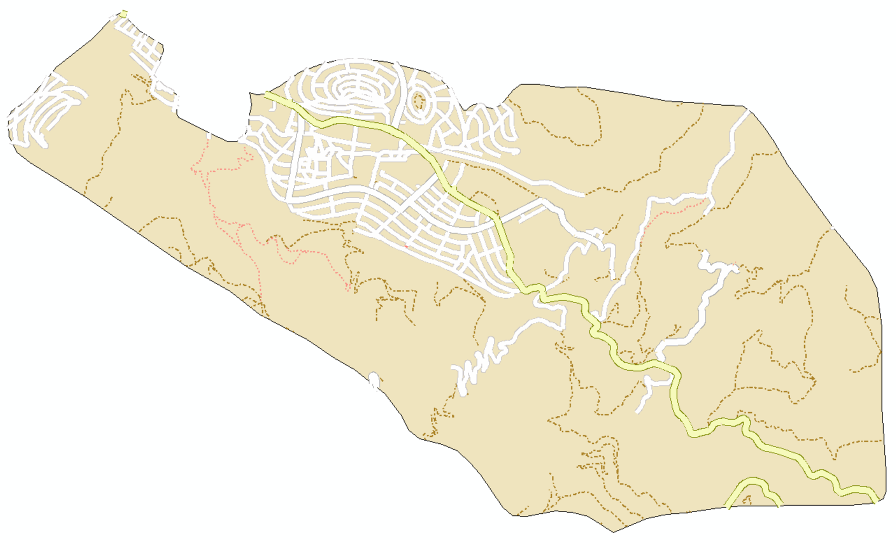


Figure 4: Map of Dionysos road network

2.1.2 Ekali

Ekali is another area that was "promoted" from a local community to a municipal unit since it passed the 5000 inhabitants limit. Together with the municipal units of Kifisia and Nea Erythrea, they form the greater municipality of Kifisia. It is a strictly residential area and any commercial activity is prohibited in the municipality except for the zone along the side road of the motorway that runs through it and is part of the Athens - Lamia National Road (E75). It lies on the northern part of the greater municipality of Kifisia and is on the west of the Dionysos municipal sector. Its name was first used by a building society that established a model urban settlement based on the patterns of an English Garden City[4] in 1922. The idea that belonged to the architect Spilios Agapitos was to create an outdoor-loving urban settlement near the Pentelicus landscape. Ekali was originally recognised as a community of Attica until the beginning of the 21st century when it was admitted as part to the greater Athens City area. Like all the municipal units in the municipality of Kifisia, it is a green suburb with tree-lined streets and has, despite its popularity, managed to maintain very slow expansion or development. This is because of actions taken by its residents to maintain its original urban planning character. It is home to some of the richest families of Greece, coming from a range of activities such as politics and shipping and most houses are of high architectural standards.

[4] An idea initiated in 1898 find by Ebenezer Howard in the UK aiming to capture benefits of a countryside environment and a city environment while avoiding the disadvantages of both [32].

Table 6: Ekali population fact sheet

Ekali Fact Sheet	
Population	4520
Area km²	4.32
Pop/km²	1045.55
Roads km	65131.60
No of Edges	671
Average Edge Length	97.07
Road Length to Area ratio m/km²	15066.00
Road length per resident m/resident	14.41
Residents / km	69.40

Ekali has the lowest population among the four areas but is not as sparsely populated as Dionysos due to its smaller size. The average population density of 1045 residents per square kilometre is still 4 times higher compared to the sparsest municipal Dionysos. It has a relatively extensive road network, considering its size, with an average of 15 kilometres of road surface per square kilometre placing it 2nd among the four. Its highly residential character can be identified in the table of road data provided below. It is also apparent that despite its character, it is not very accommodating to pedestrians as the lack of paths, footways and pedestrian streets shows.

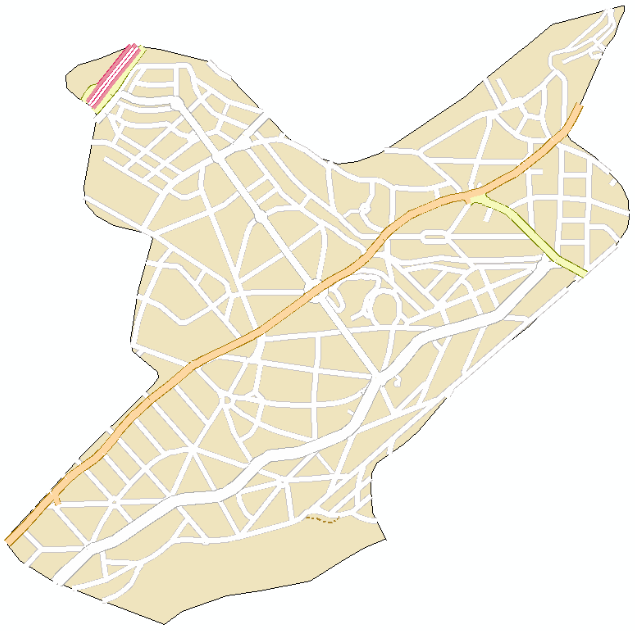


Figure 5: Map of Ekali road network

Table 7: Ekali road class statistics

Ekali road data				
Highway class	**Count**	**Length (m)**	**% length**	**Level**
Residential	483	48885	75.06%	7
Tertiary	97	8634	13.26%	5
Primary	39	3383	5.19%	3
Service	15	1789	2.75%	7
Secondary	19	1229	1.89%	4
Motorway	8	678	1.04%	1
Track	2	201	0.31%	7
Secondary Link	4	168	0.26%	4
Primary Link	2	112	0.17%	3
Tertiary Link	2	53	0.08%	5
SUMS	**681**	**123180**		

The facts in the table above display the type of suburban area under examination, a district where 75% of the roads are residential roads and the majority of the remaining 25% serve as feeders to those residential streets.

2.1.3 Kifisia

Kifisia is the oldest of the four municipal units. The area is famous for having a highly active commercial centre and a well-designed urban plan covering a lot of its areas with a grid system. The city centre has a wide network of pedestrian streets and cycleways and parts of the municipal unit have been characterised as a traditional settlement and maintain its original architecture. It is built on the western side of Penteli, at an altitude higher than Athens (the centre is at 290 metres above sea level), in a pine tree valley. The first statute of the inhabitants over the area was written out in 1870 and aimed at financial independence. Its economic growth was boosted in 1882 by the development of the railway that reached the area from the coastal town of Pireas making it in the following years a popular resort with locals and foreigners. In 1925, it is officially recognised as an autonomous community of the region of Attica and in 1942 it is promoted to a municipality having passed the 10,000 residents' margin. Its proximity to the motorway leading to the North of Greece or the city centre is considered advantageous.

Table 8: Kifisia population fact sheet

Kifisia Fact Sheet

Population	38282
Area km²	26.11
Pop/km²	1466.12
Roads km	360076.83
No of Edges	4115
Average Edge Length	87.50
Road Length to Area ratio m/km²	13790.15
Road length per resident m/resident	9.41
Residents / km	106.32

It is the largest of the four municipal units under examination with an area of approximately 26 square kilometres and a population that nears 40,000 people. Parts of the area are uninhabited as they cover the southern slopes of Penteli mountain. It has the most extensive road network of the four having a total of 360 kilometres of roads, which is more than twice as much as the 2nd with 123 kilometres. The lower average edge length compared to the previous two municipalities presented above is an indication of a tighter urban plan that is organised around smaller building blocks.

Table 9: Kifisia road class statistics

Kifisia road data

Highway class	Count	Length (m)	% length	Level
Residential	2178	189047	52.50%	7
Tertiary	565	47693	13.25%	5
Track	164	38333	10.65%	7
Secondary	367	25893	7.19%	4
Service	251	13692	3.80%	7
Footway	206	10698	2.97%	7
Unclassified	83	8571	2.38%	6
Motorway	52	8510	2.36%	1
Path	39	4253	1.18%	7
Living Street	53	3958	1.10%	7
Primary	58	3538	0.98%	3
Motorway Link	18	2371	0.66%	1
Secondary Link	40	1781	0.49%	4
Pedestrian	16	1247	0.35%	7

Tertiary Link	9	293	0.08%	5
Steps	13	160	0.04%	7
Primary Link	3	38	0.01%	3
SUMS	**4115**	**360077**		

Kifisia has the largest variety of roads classification, having roads from almost every level and demonstrating its careful urban planning. As is apparent in the table above, half of the roads are residential roads with the highest road classification after that being occupied by the class of tertiary roads. The uninhabited character of the municipality is visible in the data above by demonstrating a 10% of track roads. One striking feature that is indicative of the character of the municipal unit, is that if one adds all road classes that do not allow vehicle access, namely footways, paths, pedestrian streets and steps, the sum would show a sizeable 5% which represents nearly 17 kilometres of road surface dedicated to pedestrians.

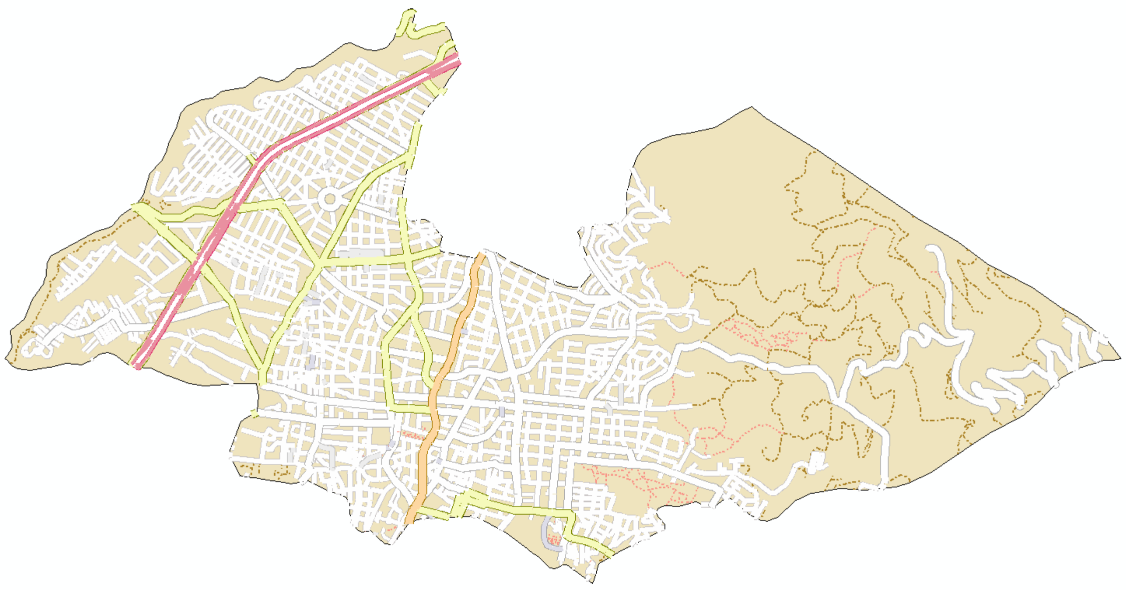


Figure 6: Map of Kifisia road network

2.1.4 Nea Erythrea

Nea Erythrea, which is the final of the four areas, originated as a neighbourhood of the nearby municipal unit of Kifisia. It was first inhabited in 1923 by refugees who came from Erythrea in Minor Asia and justifies the addition of the word “Nea” which means new. It started with the creation of the union of immigrants of Kifisia in 1923 and saw its first buildings when the Ministry of Providence expropriated land and gave it out to 364 families that were members of the union. The area was originally established as a settlement that belonged to the municipality of Athens in 1925. By 1928 it numbered 1844 residents and under the pressure of local needs, it managed to achieve autonomy and recognition as an individual community in 1934. Between the 1991 census and 2001 census, it displayed a steep increase (around 18%) in demographics, triple that of the national increase which shows its popularity.

Table 10: Nea Erythrea population fact sheet

Nea Erythrea Fact Sheet

Population	12489
Area km²	5.03
Pop/km²	2481.51
Roads km	97885.42
No of Edges	1354
Average Edge Length	72.29
Road Length to Area ratio m/km²	19449.45
Road length per resident m/resident	7.84
Residents / km	127.59

Nea Erythrea displays the highest density in population among all four municipal units, with almost 12 and a half thousand inhabitants in 5 square kilometres. The building blocks in the area are smaller since the average edge length seems to drop to 72.29 metres per road segment. It also has the longest relative road network with almost 20 kilometres of road for every square kilometre it occupies. This is at least 3 times higher compared to the sparsest of the four areas. Its density in population is at 2481 compared to the second highest in density that has a value of 1466 signifying an extra 1000 inhabitants per square kilometre.

Table 11: Nea Erythrea road class statistics

Nea Erythrea road data

Highway class	Count	Length (m)	% length	Level
Residential	830	63172	64.54%	7
Secondary	116	8457	8.64%	4
Tertiary	138	7101	7.25%	5
Motorway	15	4107	4.20%	1
Living Street	58	3648	3.73%	7
Service	63	2364	2.41%	7
Unclassified	24	2275	2.32%	7
Cycleway	25	1685	1.72%	7
Primary	29	1548	1.58%	3

Pedestrian	25	1196	1.22%	7
Track	12	1105	1.13%	7
Motorway Link	6	720	0.74%	1
Secondary Link	9	324	0.33%	3
Footway	3	165	0.17%	7
Tertiary Link	1	18	0.02%	5
SUMS	**1354**	**97885**		

The suburban character of the area is apparent from the data in the table since close to 70% of the road network is either residential or of a similar class. It displays variety and has road segments of most classes like motorways, primary roads, secondary roads, tertiary roads and even track roads and cycleways. As is visible in the image and the data above, parts of the motorway that lead to the North of Greece go through this municipal unit. Its collection of roads displays a balanced distribution in terms of both quantity and space.

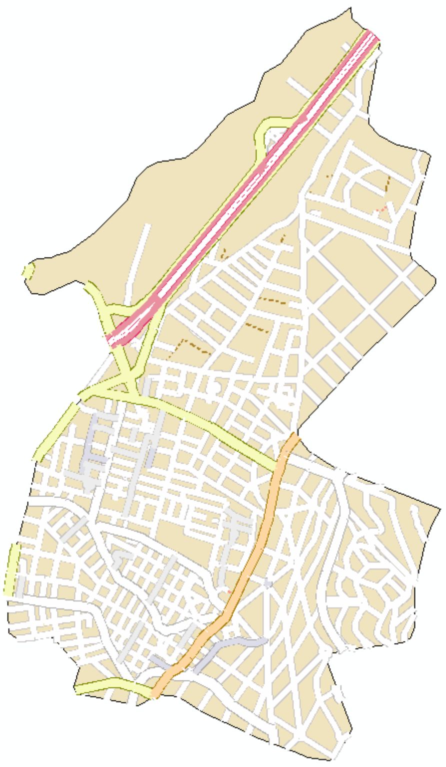


Figure 7: Map of Nea Erythrea road network

2.2 Data sources

As mentioned before, the general plan followed the diagram shown below. To implement it, a set of administrative units and road data was required. The following chapters provide a detailed explanation of the sources of data and how each was used.

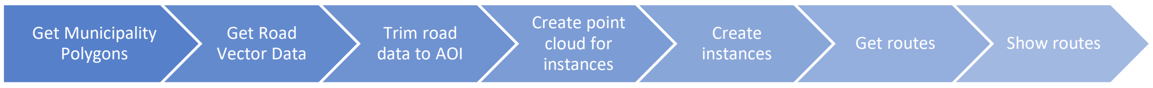


Figure 8: General steps to implementation

2.2.1 Polygons of areas of interest

The first thing required was to identify the area of interest and its coverage. Consequently, authoritative data in the form of digital cartographic data that can be used in a GIS was needed. The Hellenic Statistical Authority (ELSTAT) webpage was used and digital cartographical data that represented the municipalities of Greece was downloaded. The data provided by ELSTAT is in shapefile format, so it was easily entered into the GIS. The areas of interest were extracted for later processing. Specifically, the four polygons of the municipalities of interest were isolated in order to be used for the next steps which involved identifying the road network and creating the sample points collection. The hyperlink of the source for polygons is provided in Appendix A.
Besides the cartographic data, the Hellenic statistical authority was also the source of the latest census data that was needed for the population figures. The hyperlink for the source of

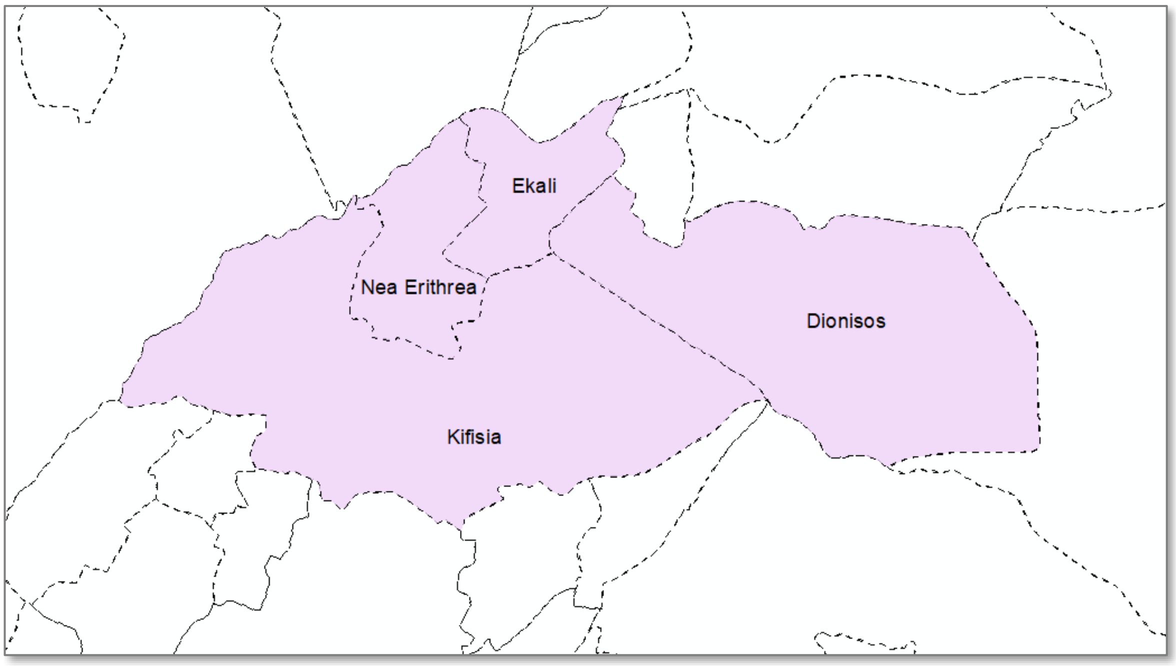


Figure 9: Polygonal representation of the areas of interest used in the study

population data is also provided in Appendix A. The four municipalities that were chosen are shown in the image.

2.2.2 Road network data

Once the surrounding polygons for each of the areas of interest were established, a set of data representing the road network for each municipality was required. The source of the data coincides with one of the sources of the routing service, the OpenStreetMap (OSM) platform. OpenStreetMap is a crowdsourcing webpage that provides "*an openly editable wiki nature of a mapping system*" [8]. The OpenStreetMap Foundation, which is in charge of maintaining the OpenStreetMap platform, specifies that the platform does not provide a complete or an accurate map, that it should not be used in such a manner and provides a detailed disclaimer. It is, however, if not the richest, one of the richest sources of geographical data provided openly and for free. It has also added the routing feature that calculates routes based on the data on the platform, the Open Source Routing Machine (OSRM) which as mentioned provides the API that was used for getting the routes for the comparison of the thesis. The OSRM API is extensively presented in chapter 4.1.

OSM allows users to voluntarily enter geographical information data that complies with a database schema that provides both flexibility and formality. This is achieved by providing a system of tagging. Users can attach tags to the basic data structures and use these tags to describe the attributes of the geographic features that have been digitised. The community has agreed on certain key-value combinations but also allows users to include new tags to improve the style of the map. Contrary to most geographical information systems, the basic data structures are points, lines, and relationships or as they are specified in the OpenStreetMap specifications, Nodes, Ways and Relations. A line that represents a road is entered as a way which is a collection of nodes along with a set of tags that describe the attributes of the road. An example of such a road is shown in the following snippet.

In this snippet, the way tag contains the attributes that are related to the way such as its unique ID, the time and date it was created, the user who created it and more. It then includes the collection of node IDs that make up the geographic entity of the road as well as tags that describe the characteristics of the road. The road seen in the snippet is made up of 7 nodes and has attributes such as 3 lanes, a maximum speed of 60 kilometres an hour, an excellent smoothness on the surface, a sidewalk on the right, a direction of one-way and other attributes.

```
  <way id="435969922" version="11" timestamp="2019-12-
07T19:37:43Z" changeset="78094662" uid="936280" user="beatnickg
r">
    <nd ref="279591132"/>
    <nd ref="2517680711"/>
    <nd ref="279591133"/>
    <nd ref="2656090050"/>
    <nd ref="5586937690"/>
    <nd ref="6122403549"/>
    <nd ref="279591134"/>
    <tag k="highway" v="primary"/>
    <tag k="int_name" v="Kifisias"/>
    <tag k="lanes" v="3"/>
    <tag k="lit" v="yes"/>
    <tag k="maxspeed" v="60"/>
    <tag k="name" v="Κηφισίας"/>
    <tag k="oneway" v="yes"/>
    <tag k="ref" v="E083"/>
    <tag k="sidewalk" v="right"/>
    <tag k="smoothness" v="excellent"/>
    <tag k="surface" v="asphalt"/>
    <tag k="wikidata" v="Q785707"/>
    <tag k="wikipedia" v="el:Λεωφόρος Κηφισίας"/>
  </way>
```

Figure 10: A "way" element declared in an osm file

The following snippet shows the node tag of the first of the nodes from the list of nodes in the way above. This is where the actual coordinates of the point are specified. Once declared, a node can be used in as many “ways” or “relations”, as necessary.

```
  <node id="279591132" lat="38.0655496" lon="23.8096050" version="8"
timestamp="2013-04-29T18:31:53Z"    changeset="15912026"    uid="1016290"
user="ika-chan! UK-USA"/>
```

Figure 11: A "node" element declared in an osm file

OpenStreetMap raw data is downloaded in a file that has a .osm file format. This is an XML type file, as noticed in the snippet above, and can be manipulated either programmatically with the use of libraries such as ”Beautiful Soup”, which is a Python XML parser, or directly loaded into a Geographical Information System such as QGIS.

The data required for our project is strictly road network data which means that once the data of the study area was downloaded it needed to be processed for the next steps. To be precise, the tasks performed were the following:

i. all way features that were not road features were deleted. This included railway lines, power lines, bus routes etc
ii. the road data was trimmed to the boundary of each municipality and attributed to that municipality, and finally
iii. the roads were split at their intersection as OpenStreetMap maintains roads without topological correctness. The lack of topology is not convenient since we rely on the creation of points at the ends of each road segment - thus discovering the parts of our area that have a denser street network - and not at the beginning and end of each road

One more reason for performing the above tasks on the road data is because of the need to use the lines' set to extract statistical information that will give us insight over our areas. The statistical information includes the following elements:

- the sum length of roads per AOI
- length over area ratio (a normalising index of how extensive the network is in the area)
- percentage of road classification per AOI
- number of edges per AOI
- average length per edge

One of the most useful tags in the OSM data schema is that with a k value of highway.

```
<tag k="highway" v="primary"/>
```
Figure 12: A "highway" tag declared in an osm file

This gives us a value (v) that is equal to the classification of the road. In the example above, v equals primary which means our highway is a primary road. The OSM documentation provides an extensive guideline over its classification system and an extraction guide combined with pictures of each class for the highway tag. Further tagging includes tags that can provide attributes for the following general categories:

- names and references
- speed limits and restrictions
- surface, width, and lighting
- parking
- number of lanes
- cycle lanes and tracks

- crossing points and barriers
- turn restrictions etc

All the above characteristics are particularly useful for the reconstruction of a local network and running a local routing algorithm. They are, however, not utilised in the current case. The main focus is on road categories.

The following table provides a list of the specific road classes that were identified in the data that covers the areas of interest. For each road class, the image provided by the OpenStreetMap extraction guide has been added, together with the general description for ease of use. These are displayed as provided by the OSM Wiki [9]. The order in which they appear is by frequency of appearance in our specific area of study. For example, the first row refers to the residential category which covers approximately 350Km and represents 54% of our total road coverage as opposed to the final row that refers to the primary link class which has only 5 segments and a total length of 150m in all four municipalities.

Table 12: List of road classes identified in the data that as provided by the OpenStreetMap extraction guide

Highway value	Description	Photo	Level
Residential	Roads which serve as an access to housing, without the function of connecting settlements. Often lined with housing.		7
Track	Roads for mostly agricultural or forestry uses.		7
Tertiary	The next most important roads in a country's system. (Often link smaller towns and villages)		5

Highway value	Description	Photo	Level
Secondary	The next most important roads in a country's system. (Often link towns.)		4
Unclassified	The least important through roads in a country's system – i.e. minor roads of a lower classification than tertiary, but which serve a purpose other than access to properties		6
Service	For access roads to, or within an industrial estate, campsite, business park, car park etc.		7
Motorway	A restricted access major divided highway, normally with 2 or more running lanes plus emergency hard shoulder. Equivalent to the Freeway, Autobahn, etc.		1
Footway	For designated footpaths; i.e., mainly/exclusively for pedestrians. This includes walking tracks and gravel paths.		7

Highway value	Description	Photo	Level
Path	A non-specific path.		7
Primary	The next most important roads in a country's system. (Often link larger towns.)		3
Living Street	For living streets, which are residential streets where pedestrians have legal priority over cars, speeds are kept very low and where children are allowed to play on the street.		7
Motorway Link	The link roads (slip roads/ramps) leading to/from a motorway from/to a motorway or lower-class highway.		1
Pedestrian	For roads used mainly/exclusively for pedestrians in shopping and some residential areas which may allow access by motorised vehicles only for very limited periods of the day.		7
Secondary Link	The link roads (slip roads/ramps) leading to/from a secondary road from/to a	-	4

Highway value	Description	Photo	Level
	secondary road or lower-class highway.		
Cycleway	For designated cycleways.		7
Tertiary Link	The link roads (slip roads/ramps) leading to/from a tertiary road from/to a tertiary road or lower-class highway.	-	5
Steps	For flights of steps (stairs) on footways.		7
Primary Link	The link roads (slip-roads/ramps) leading to/from a primary road from/to a primary road or lower-class highway.		3

2.3 Technology options and choices

The object of the thesis is twofold. The first is creating an automated process that will quickly create many routing results for a specific area from 2 different routing providers. The data acquired needs to be processed for the creation of tables that will illustrate the differences between the two sets provided. However, once this large number of results has been acquired, a flexible viewing tool is necessary for sharing the data and reviewing it. These 2 discrete tasks require 2 different IT skillsets and 2 different sets of tools.

The first task requires tools of programming nature especially capable of computational processes, data manipulation, file management and mathematical ability. It would be advantageous if the chosen programming language is also supported by several libraries and functions for performing geographical information analysis. The second task requires web page creation skills and tools. The choices for each one of the abovementioned tasks are presented in each subchapter below.

2.3.1 Route acquisition and data handling

For the task of acquiring the routes and handling the data, Python was the language of choice for several reasons. Python has displayed exceptional growth and recently became the most popular programming language in the world according to the 2019 Stack Overflow survey[5] [10].

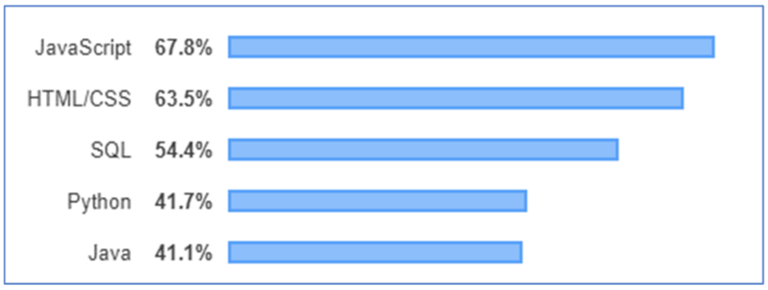


Figure 14: List of popularity of programming languages according to 2019 Stack Overflow

The widespread use of Python has been based on the existence of very extensive libraries with scientific and more specifically mathematical functions. Popular libraries such as NumPy and SciPy which can be deployed very easily and run very efficiently. This has led to commercial GIS products such as those by ESRI and open source GIS products such as QGIS to utilise Python. As a result, Python has, for the last couple of years, become the most widely used language by GIS programmers and users [11]. Its open-source character and widespread use of the language has boosted its usability. As a result, it comes with lots of functionality like the JSON encoder and decoder as part of the standard library or the HTTP protocol client which provides classes that implement the client side of HTTP/HTTPS protocols and is also part of the standard library. In summary, the advantages of choosing Python are:

Figure 13: Python logo

- an extensive standard library that provides more than basic functionality
- a large list of modules and libraries for additional functionality especially for scientific or geographic tasks

[5] Javascript is considered a scripting language, HTML/CSS a markup language and SQL a querying language leaving Python as the highest-ranking programming language.

- it is relatively easy to learn
- it is efficient
- it integrates directly with many GIS tools and resources such as QGIS, Google Maps, ArcGIS

The external libraries that were used in the current project are presented in the following table. The reason for the usage of each package is given in the "usage" column.

Table 13: Python packages used

Package	Version	Usage
NumPy	1.18.1	Not explicitly but as core package for SciPy
PyProj	2.4.2.post1	Used for transforming geographic data to and from projected data
SciPy	1.4.1	Used for useful distance functions
Requests	2.22.0	Used for making simple HTTP requests

These were all added through the Python Package Installer (pip) which facilitates the installation of packages and libraries in Python.

The version of Python was Version 3.7.3 since the Python 2.0 series was due for retirement at the end of 2019.

The extended use of Python has brought an abundance of Python programming environments and IDEs. The preferable tool for the tasks was the PyCharm IDE provided by JetBrains. It provides a clear writing environment and a large set of tools such as Git support, virtual environments, integrated debuggers, and others.

A detailed presentation of the tasks that were developed in Python and how each library was implemented is presented in Chapter 4.3.

Figure 15: PyCharm IDE logo

2.3.2 Viewing tool

One of the most useful aspects of GIS is its ability to help us visualise data. Just managing the results and creating tables is not as useful or informative as being able to view the differences and similarities between the 2 routes over a map.

Figure 16: Leaflet logo

Once the routes had been created, the need to be able to view the results was also created. Both APIs provided results in JSON format that could not be directly loaded into a viewer. Although GeoJSON can be very easily created from the JSON results and GeoJSON files can be loaded onto GIS software, such software is not capable of easily handling the large numbers of layers that are created. Besides, manually loading pairs of files two at a time would be a highly time-consuming and a cumbersome task. The solution that was implemented was based on the concept of creating a web page with a map control capable of displaying GeoJSON files. This would also provide the advantage of distributing and displaying the results to a much broader audience that does not have access to GIS software. Furthermore, the use of a browser solves a lot of OS issues and is cross-platform. The webpage solution relies on using the classic technologies of HTML, CSS, and JavaScript so the only question that required an answer was "which map control should be used for the map display". The advancements in GIS and Web GIS have resulted in the availability of many map web libraries, but one has stood out for its simplicity and performance. The Leaflet framework has become exceedingly popular not only because it provides many features that come "out of the box" but also because the application of those features is quite simple and effective. Features like showing and hiding a layer or zooming in/out on a selected feature. A detailed presentation of the Leaflet features that were specifically used and how they were implemented is given in chapter 5.

Besides Leaflet, the only other library that was used was Bootstrap. Bootstrap is a front-end component library that provides ready classes for a responsive grid system, prebuilt components and plugins built on jQuery. Its usage provides readymade solutions for table-less and responsive design that can adapt to various sized screens.

Just as PyCharm was used with the Python programming language, WebStorm by JetBrains was the IDE of choice for writing code for the webpage app.

Figure 17: WebStorm IDE logo

2.4 Hausdorff Distance

The Hausdorff distance is a max-min distance between the points of two sets of a metric space. Given two finite sets $A = \{a_1, \dots, a_m\}$ and $B = \{b_1, \dots, b_n\}$ the directed Hausdorff Distance of a vector A to a vector B is given by

$$h(A, B) = max_{a \in A}\, min_{b \in B}\, \|a - b\|$$

Equation 2: Directed Hausdorff Distance where $\|a - b\|$ is a norm like the Euclidean norm that defines the Euclidean distance between point a and b. Due to the bidirectionality of the two sets, the Hausdorff distance $H(A, B)$ between A and B is:

$$H(A,B) = max\big(h(A,B), h(B,A)\big)$$

Equation 3: Hausdorff Distance between sets A and B

A practical presentation on [12] provides a simple explanation and a demonstration that can be seen in the following image. The process of calculating the Hausdorff distance between set A = {a_1, a_2} and set B = {b_1, b_2, b_3} is the following:

- the closest point from B to a_1 is calculated. This is $d_{11} = min_{b \in B}\|a_1 - b\|$ and is with point b_1
- the closest point from B to a_2 is calculated. This is $d_{23} = min_{b \in B}\|a_2 - b\|$ and is with point b_3
- the directed Hausdorff distance is the maximum of the two, $h(A, B) = \max(d_{11}, d_{23}) = d_{11}$

We can now say that any point of vector A is no further than h from any point of B. If the process is performed by calculating the distance from B to A, we will obviously have different results. Points b_1 and b_3 will again pair with points a_1 and a_2, but we would also have an extra distance calculated between points b_2 and whichever of the two is nearest, a_1 or a_2. As it is apparent, this distance d_{2a} will be greater than any of the two we already know thus yielding a larger Hausdorff distance between the two sets. We then say that the distance of the two sets is the maximum of the two directed distances which in this case would be from B to A since $d_{2a} \gg d_{11}$.

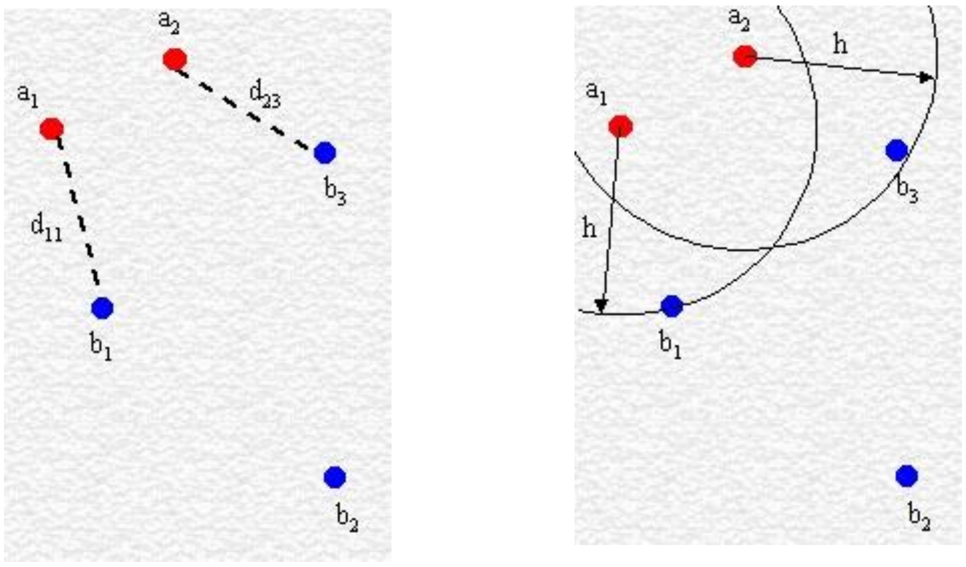


Figure 18 After the calculation we declare that any point of A is at most distance h from any point of B. [12]

Use of the Hausdorff distance in comparing the two route results will be easy to implement since each route is defined as a set of points with a pair of coordinates.

Important note! Special provisions will have to be made to convert the coordinates of each point returned by the service with the use of a Geographic CRS transformation to a Projected CRS and to get correct calculations of the Euclidean distance between the points.

3. Point determination

A route is a list of directions/instructions that define the course of a line connecting two points in a manner that satisfies certain criteria. An example of such criteria is that routes should use roads that are suitable for the means of travel and can be travelled in a short period of time or by covering the least distance. In the comparison of the two APIs that provided the routes, one of the key issues was the determination of the number of pairs of points that populated the area upon which the routes would be calculated. Second and most important is the question of where these points that will form the pairs should be. This aspect was especially important because it had to take into consideration two opposing conditions. On the one hand, the randomness of the points' occurrence i.e. the fact that the point of origin or destination could occur at any point on the road network in the area. On the other hand, the fact that there is a higher chance of the instance that requires a route happening where the population is denser and/or the road network is also denser displays a probabilistic element to the instance. The need for a "semi stochastic" approach was apparent. Travelling points occur on the road axis and not in the space that is defined between the roads. The methodology that was used was based on the idea of point clouds. Point clouds are sets of points in space and are generally popular in 3D representation. They illustrate points on the surfaces of objects. Respectively, the points that indicate the positions of junctions on the road can give us a similar cloud of points and indicate where we have a denser road network. This is a valid approach supported by the fact that the road network is denser in more populated areas and more populated areas are likely to have more people travelling on those roads, therefore, increasing the probability of an instance occurring on that part of the network. The final events' points that will form pairs should be enough in number so that the whole area is covered and at the same time be as few as possible so that redundant routes are avoided. This was moderated with the use of a metric that indicated the average Euclidean distance among the set of points. An iterative process that would lead to a specific set of points in our area of interest was designed. The exact process that was used is presented in the following chapter and subchapters.

3.1 Point generation process

The whole process of point generation can be broken down into 5 steps. The first involves the creation of an initial collection of points that represent the area of interest. This is performed only once for each municipality. The following four are repeated until our set satisfies our criteria. These are:

- organize points into groups (clusters)
- calculate a central point that represents the group

- assign a position on the road network from that central point
- calculate the proximity table and statistics

All these tasks are easily performed with a GIS tool. A detailed description is presented in the following subchapters.

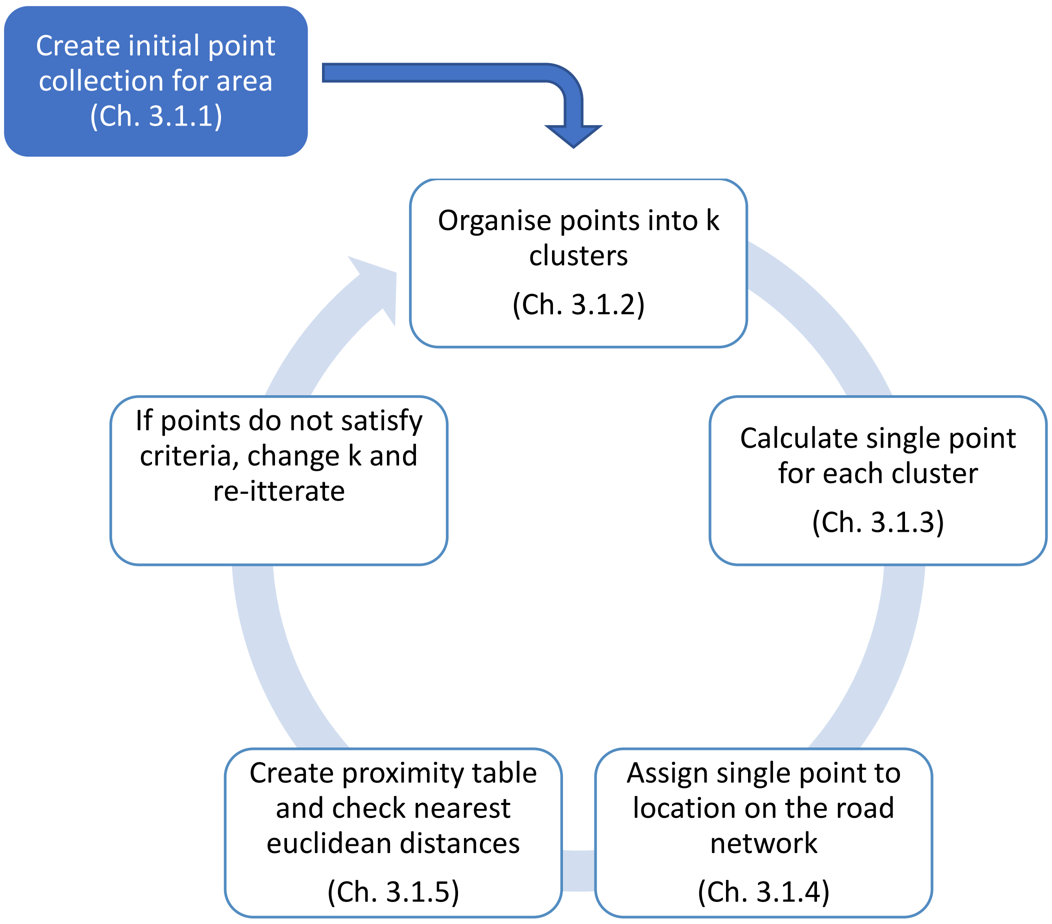

Create initial point collection for area (Ch. 3.1.1)
Organise points into k clusters (Ch. 3.1.2)
Calculate single point for each cluster (Ch. 3.1.3)
Assign single point to location on the road network (Ch. 3.1.4)
Create proximity table and check nearest euclidean distances (Ch. 3.1.5)
If points do not satisfy criteria, change k and re-itterate

Figure 19: Flow chart for the creation of the semi-stochastic location set

3.1.1 The initial collection of points

As mentioned in chapter 2.2, to create the initial collection of points, the set of actual data that represented the road network in each municipality was required. For the main source of data, the OSM platform (https://www.openstreetmap.org) was used to extract the file of an area that included all four municipalities. Road network data can be downloaded from this platform or third-party providers in a variety of GIS formats including shapefile, osm which is the XML type file format presented previously, and others. Osm file of the area was chosen because the data is more complete compared to other formats which are provided through the third parties. Osm files can be practically imported to most GIS software such as QGIS and ArcGIS and managed. Once imported the data can be used to extract the information we need and create our initial set of points. The OpenStreetMap data schema maintains roads and not road segments. A road is considered an edge for the entire length that shares the same attributes such as name, road surface etc. This means that the points occur only where roads begin and end and not at their junctions where they intersect. The following task required road segmentation to create a topologically correct network that would provide all the necessary points. A topologically correct network is one where a node exists wherever there is a physical junction between two or more roads. In the case of an overpass or an underpass, no node is required but if one is added it does not affect the result since the number of such points is relatively small to the total of nodes on the network (less than 1%). The difference between points that lie at the ends of roads as opposed to points at intersections can be seen in the image in which blue points represent points at junctions and yellow represent the ones at the ends of the road. Choosing only points at ends would have omitted the blue points from the original point collection.

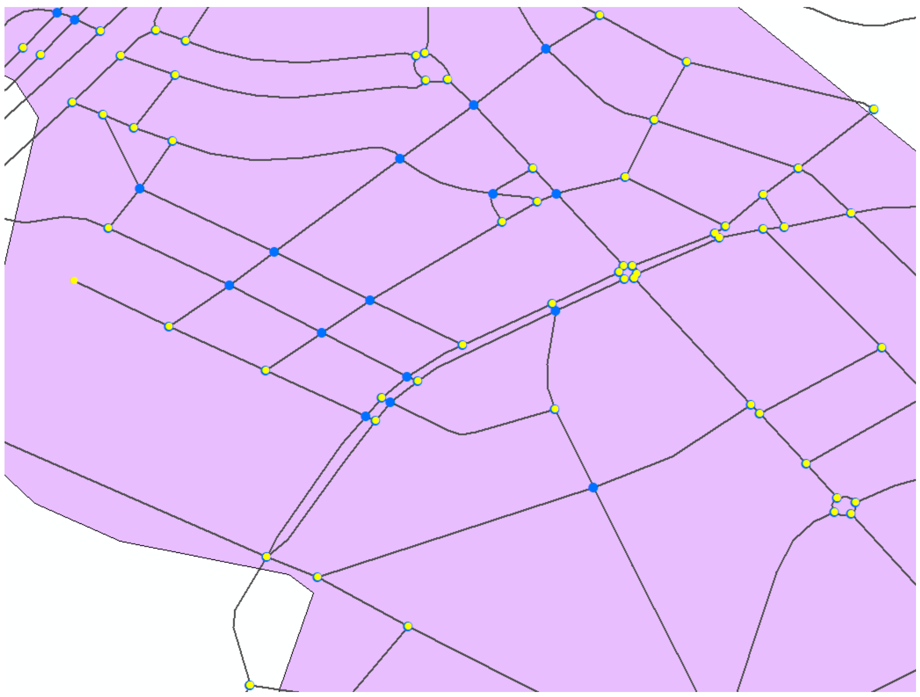


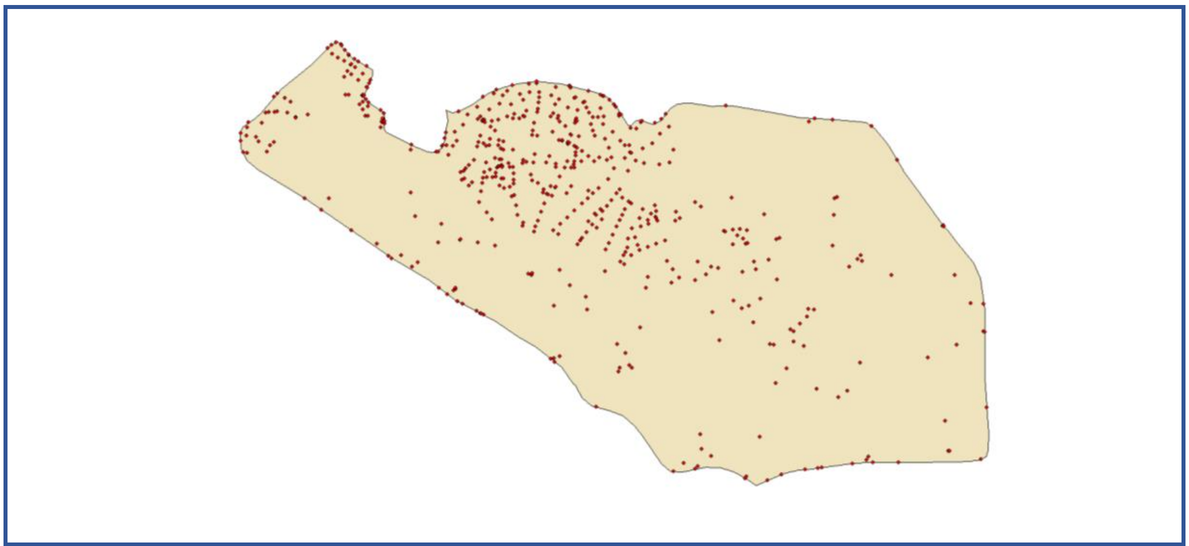
Figure 20: The blue points would be left out if the topological process did not take place prior to the initial point creation

The total set of points after the topological correction establishes our initial collection. The following table shows the number of points that were created for each one of the municipalities after this step.

Table 14: Number of point per area and statistical information

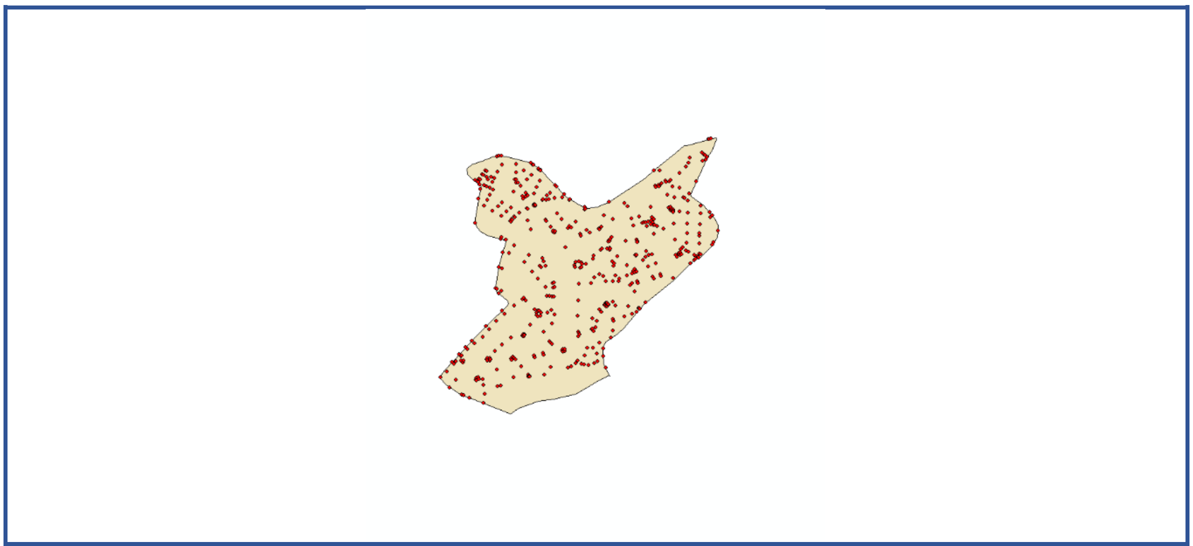
Municipality	Points	Area (km²)	Points/km²
Dionysos	547	20.16	27.13
Ekali	474	4.32	109.64
Kifisia	2869	26.11	109.88
Nea Erythrea	941	5.03	186.97

The next images show the initial point collection for each municipality together with statistical information on the municipality.



547 points	20.16 km²	27.13 points/km²

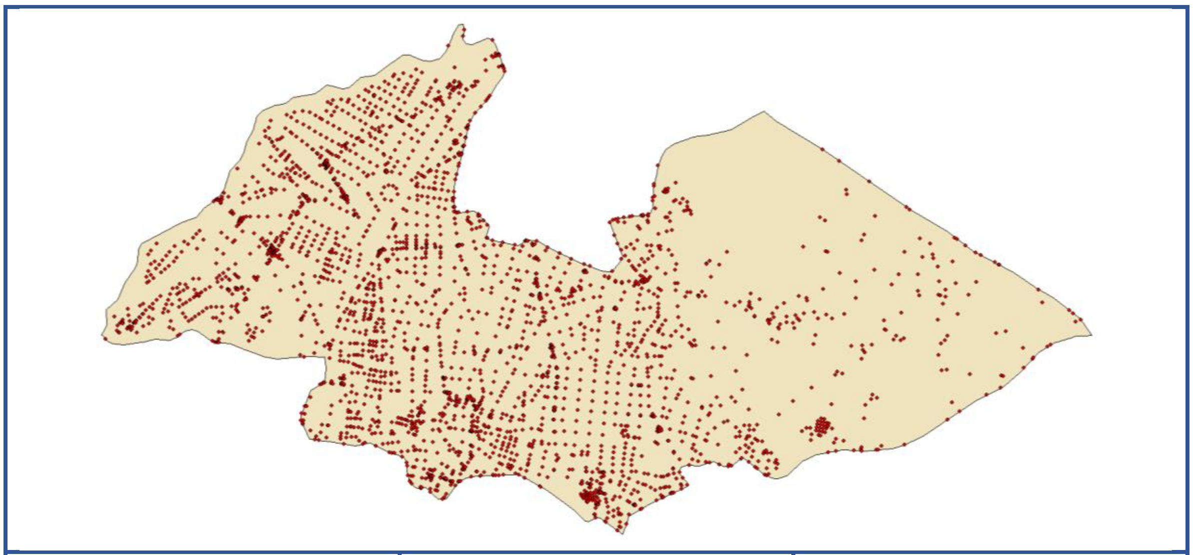
Figure 21: Dionysos map of points and data



474 points	4.32 km²	109.64 points/km²

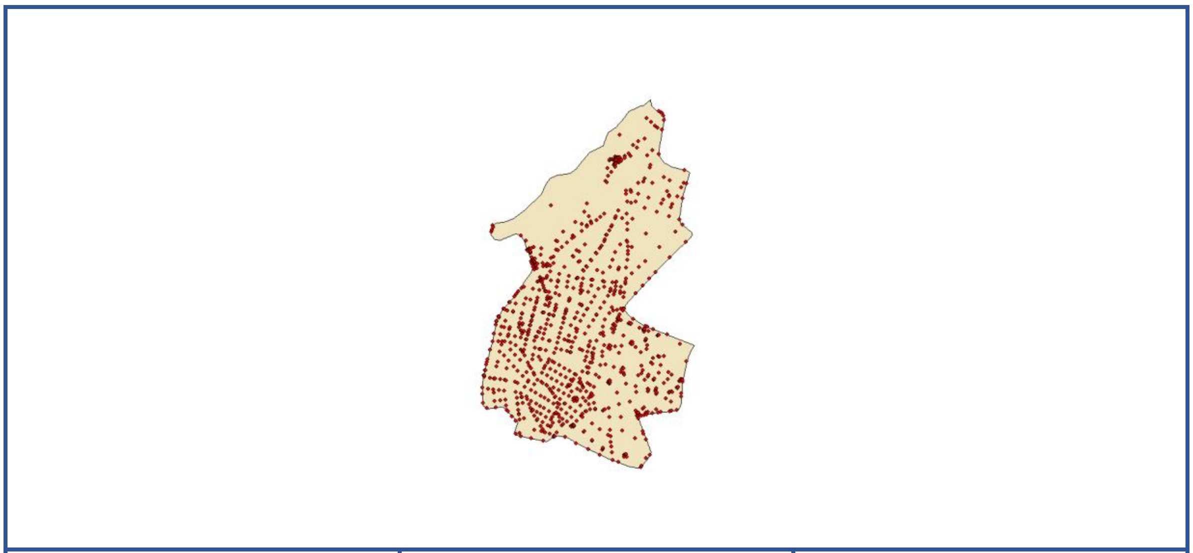
Figure 22: Ekali map of points and data

The density of the points depends heavily on the density of the road network and the urbanisation of the municipality!



2869 points	26.11 km²	109.88 points/km²

Figure 23: Kifisia map of points and data



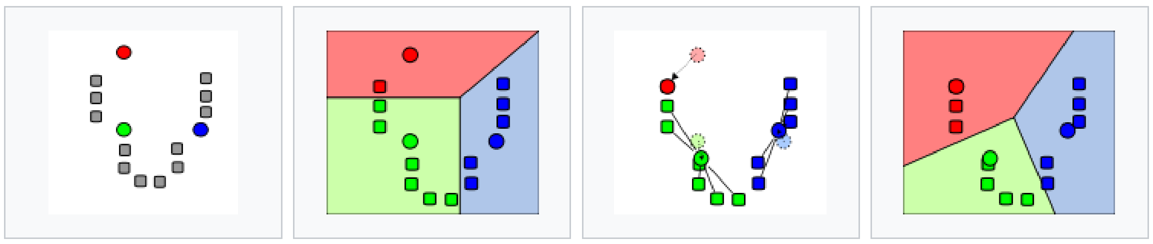
941 points	5.03 km²	186.97 points/km²

Figure 24: Nea Erythrea map of points and data

3.1.2 k-Means clustering

With the original set of points for each area defined, determination of the central points of concentration followed. This is performed with cluster analysis which is the task of organising objects in such a way that they form a homogeneous group, also called a cluster. We do not have some attribute that the points share but we are looking at spatial proximity and grouping. There are several clustering algorithms. Some evaluate the grouping based on some hierarchy, others depend on density while others are centroid based. k-means relies on a centroid model which means that a central vector will represent each cluster. It is one of the most popular algorithms because despite addressing an NP-hard problem, it employs a greedy heuristic and always converges to a local minimum. But this also has the disadvantage of not always finding the most optimal answer. The k in the name refers to the number of groups or clusters it will try to partition the initial set into. The number of observations in the original set does not affect the outcome of the algorithm. This method is appropriate for our task since we do not seek a partition over the area that is optimal by some standards but provides us with an indication of the places that should draw our attention and have a higher chance of producing a routing event. The standard algorithm, or as it is known as naive k-means, is based on an iterative process in which:

a) it creates an initial random selection of k observations called seeds
b) assigns each observation to the nearest seed
c) calculates the centroid of the cluster that has been formed from the observations that have been assigned to that seed
d) re-positions the seed to the location of the centroid
e) if convergence has not been achieved, it returns to step B



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Figure 25: Demonstration of the standard algorithm

Image provided by [13]

The following image shows the result of a k-means calculation for k = 5 on the 941 initial points in the area of Nea Erythrea.

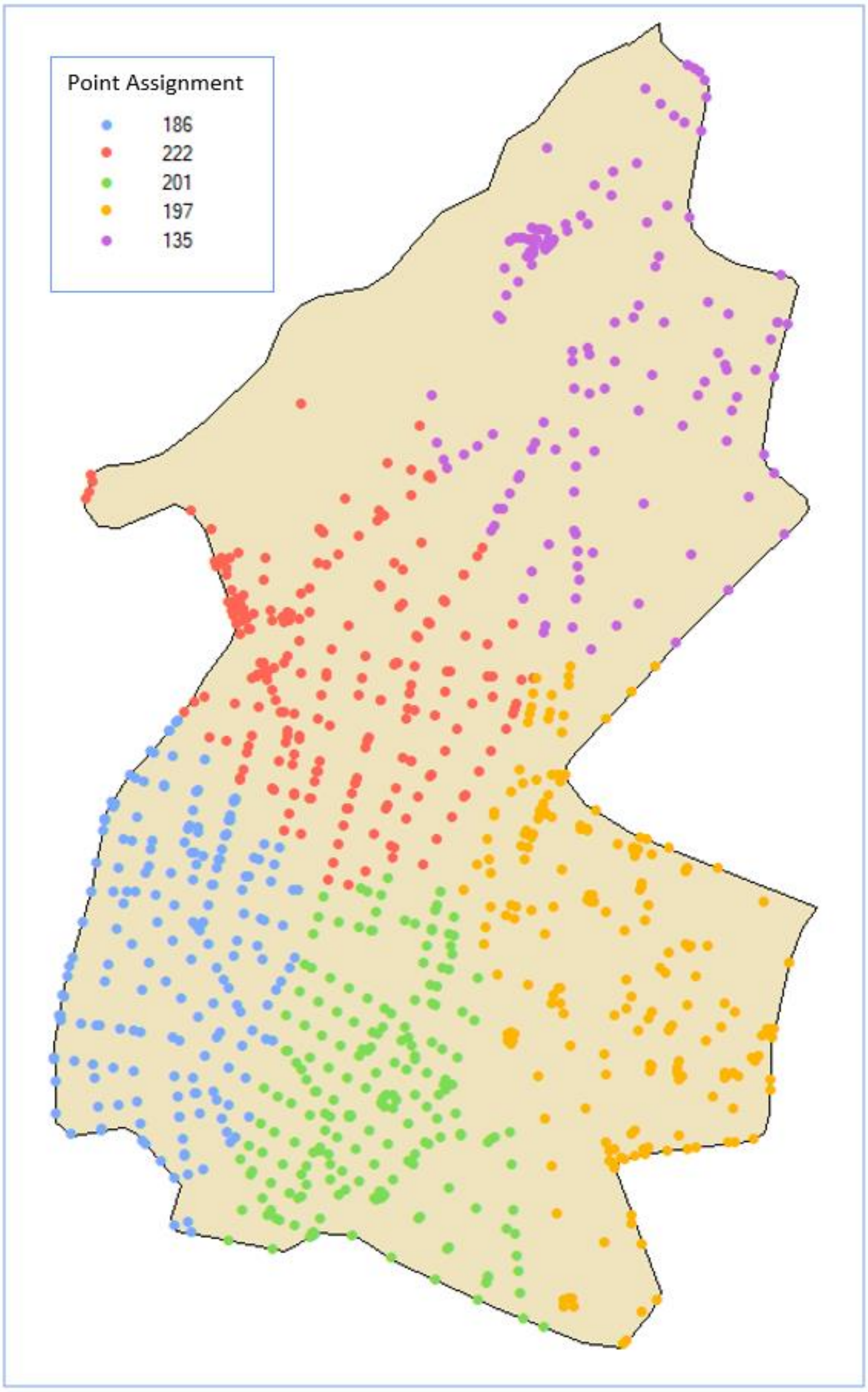


Figure 26: Nea Erythrea sample clustering and number of points per cluster for 5 clusters (k = 5)

Important Note! For the above process as well as the following averaging tasks to take place, all data must be projected for accurate distance measurements. The projected coordinate system that was used was the Greek Grid / GGRS87 (EPSG:2100).

3.1.3 Averaging

The previous process facilitates the partitioning of our area into segments of similar gravity. These groups of points must, in the following step, be represented by a central vector that will be reflected as a single position on the road network. This is achieved by exercising a spatial averaging process. Because there is more than one way to perform this measure of central tendency, the 3 most common that were examined are presented next.

3.1.3.1 Mean Centre

The mean centre calculation is the most used and most intuitive approach in which a point is calculated having as coordinate X the average x value of the input set of n points and for coordinate Y the average y value of the n points.
The mathematical expression of this concept is presented in the next 2 formulas.

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Equation 4: Mean X

$$\bar{Y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Equation 5: Mean Y

There are variations to the tool that could consider an attribute as a weighing factor but no use of such a factor was considered for our approach.

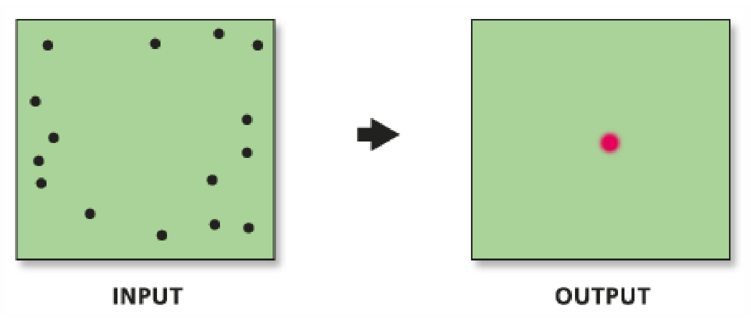


Figure 27: Example of Mean central point

Image provided by [14]

This measure will identify our geographic centre, not the point of concentration necessarily as the above picture displays. It essentially finds the location with the minimum sum of squared distances and not the location with the minimum sum of distances [15].

3.1.3.2 Median Centre

The measure of the median centre is a calculation of the point that minimises the sum of distances from the central point to each one point in our set. It is estimated with an iterative method by using an initial location (which could be the mean centre) and through the repetitive process reaches a result when the new point is similar to the previous.

Assuming d_i^t is the distance between a given point i and the current centre in our iteration t with coordinates (x^t, y^t), the coordinates of the new location (x', y') are calculated based on the following:

$$x' = \frac{\sum_{i=1}^{n} \frac{x_i}{d_i^t}}{\sum_{i=1}^{n} \frac{1}{d_i^t}}$$

Equation 6: Median X

$$y' = \frac{\sum_{i=1}^{n} \frac{y_i}{d_i}}{\sum_{i=1}^{n} \frac{1}{d_i^t}}$$

Equation 7: Median Y

with

$$d_i^t = \sqrt{(x^t - x_i)^2 + (y^t - y_i)^2}$$

Equation 8: Distance of point i from median point at iteration t

There are variations to the tool that could consider an attribute as a weighing factor but no use of such a factor was considered for our approach.

From a practical point of view, this measure differs from the previous in that outliers (distant points) have a much smaller impact on the results location. This method does not necessarily have one single correct answer like the mean centre. Consider a set of precisely two points. Only one point can be in the middle of the 2 separating the distance in half, but any point on the connecting line will have a sum of distances to the two points that will equal the length of that line.

It is for this reason that some GIS applications return both points in such a case. Results of the sort did appear in some of the tests with the unexpected production of more central points than the number of clusters.

3.1.3.3 Central Point

The final measure that was considered is that of Central Point. It is like the previous measure of the Median Centre in that It seeks a minimum sum of distances. It is however different in that the location that will satisfy the criteria must be one of the points of our initial set. This measure does not have some mathematical formula but relies on an iterative process of going through the entire set of points, one at a time, calculating the distance between it and every other point, finding the sum of these distances and keeping the result as the new smallest sum If it is indeed smaller to the previous minimum found.

3.1.3.4 Conclusion

Tests were performed with all three methods to examine which one would have the most appropriate results. Because all our areas displayed similar density in points, all 3 methods produced similar results showing that, in this case, the measure of choice would not have a large impact on the results. However, the median centre was preferred over the other 2 for the following reasons:

A. The central point was rejected because it produced results that were limited to a pre-existing point and did not reflect the span of the area the cluster covered. Furthermore, although it helped in eliminating one more step in preparing our points for routing by identifying a point that already lies on our network, this could produce false negatives since the point always lay on a junction. To be specific, since the two data sets used in the background for the calculation of the routes will differ, even if the differences are minute, one of the two routing providers will be at a disadvantage if the point is not exactly at the junction but at a small offset on a one-way street. The result for such a route might result in being vastly different, not because of the underlying data or routing algorithm, but because of the seeming differences of the origin or destination.
B. The mean centre was rejected because of the sensitivity it displays with outliers. The reason the data was processed in the beginning was to identify parts of the road network which are denser and have a higher chance of being used. An outlier in our set will affect the position of the mean centre without it being a particularly important position on our network.

The preferred measure of spatial averaging was by calculation of the median centre which must be used with an extra examination for the case of having precisely two points in the cluster. The results were examined for every area and in the case of having such a situation one of the two actions were chosen. If the two points where at proximity, one of the two could be deleted without greatly affecting the results of the routing request. If on the other hand, the two points were at great distance apart then the decision of keeping both seemed more appropriate.

3.1.4 Road snapping

The location results from the previous process would not necessarily lie along the line of the road. To have realistic instances, a GIS tool was used to calculate the nearest point on a road segment which would represent our final cluster representative.

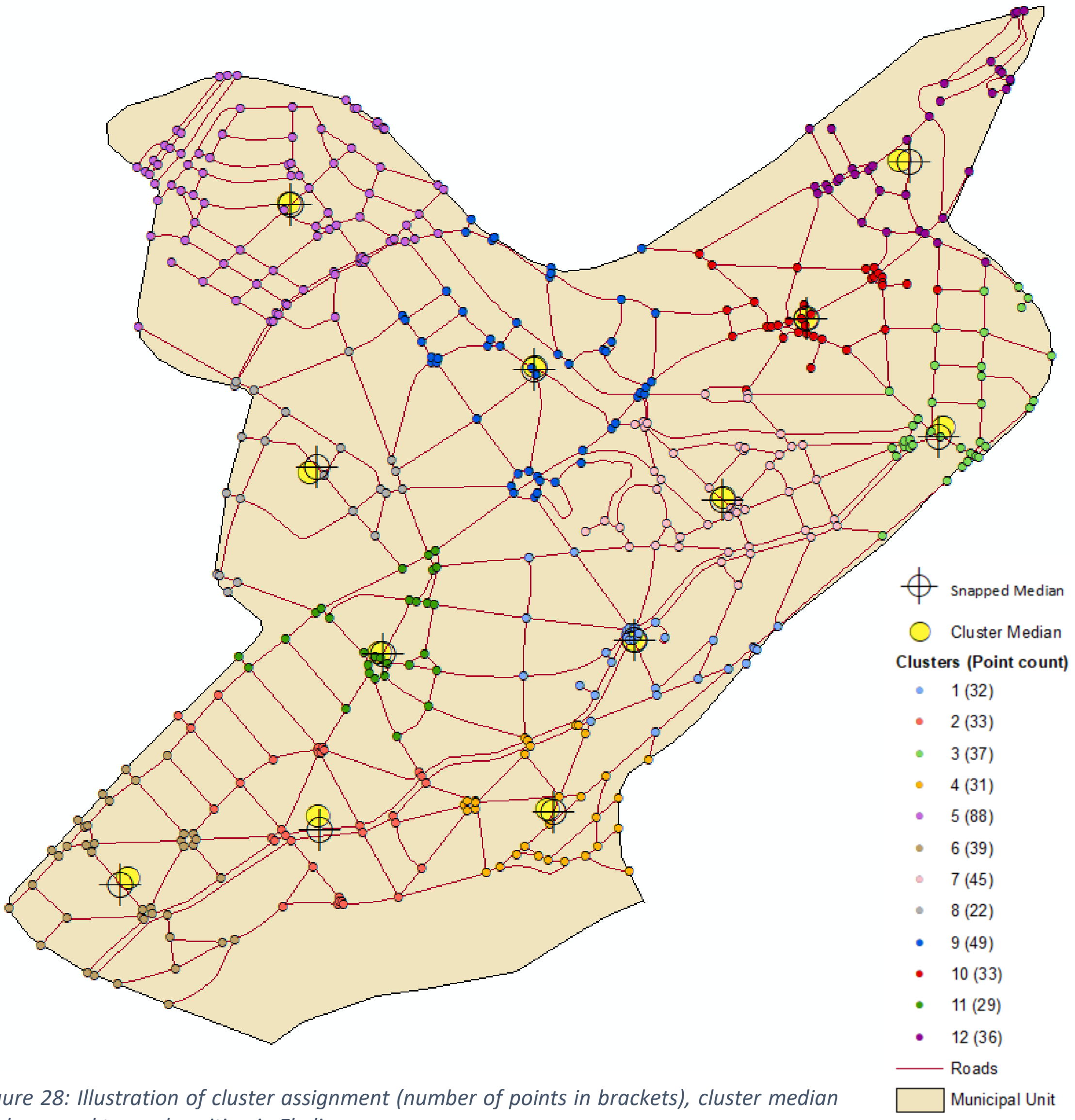
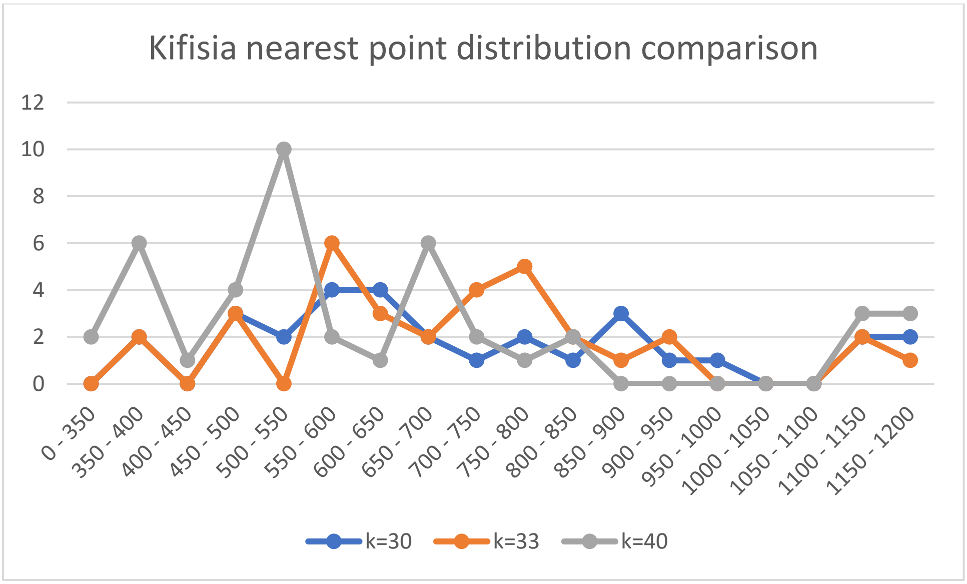


Figure 28: Illustration of cluster assignment (number of points in brackets), cluster median and snapped to road position in Ekali

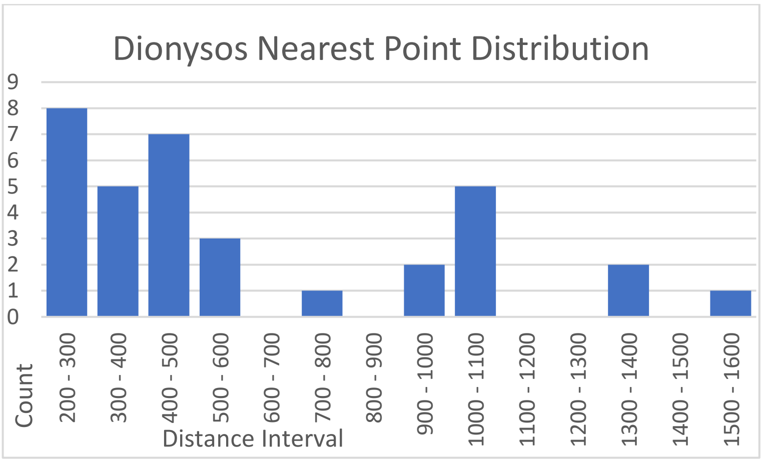
3.1.5 Nearest points distribution tables

As mentioned in the beginning, the aim was to create a distribution of sample points that was neither too large nor too small for reasons previously stated. The iterative process presented in chapter 3 relies on making an initial estimation at the number of necessary clusters, calculating the central points of each cluster, and examining the proximity of the central points by employing a nearest point distribution table. Such a table would have the point index and the distance of the nearest point to it. Certain GIS tools will also provide the ID of the nearest point to the one examined. The produced table of nearest points would then be used to estimate the mean nearest distance among the points. The threshold that appeared to satisfy our criteria as far as the resulting quantity of points for our areas was estimated at 500 metres distance. After the calculation of the nearest distribution table for an area, the mean nearest distance was estimated and examined along with a distribution graph to evaluate if the process had to be repeated with a different number of clusters k. If the mean nearest distance was very high, a higher number of clusters was required. If on the other hand, the mean nearest distance was very low then the number of clusters should be reduced. Presenting the various attempts made to finalise our results displays no particular interest and is omitted. Graphs and data on the conclusions for each area are given in the following tables and illustrations with notes where necessary. The following graph is presented to illustrate the differences in the distribution of minimum distances for 3 different values of k (30, 33 and 40) for the area of Kifisia. When k = 40, there are 10 points with nearest other points within a radius 500m to 550m.



Graph 1: Nearest point distribution in Kifisia for 30, 33 and 40 clusters

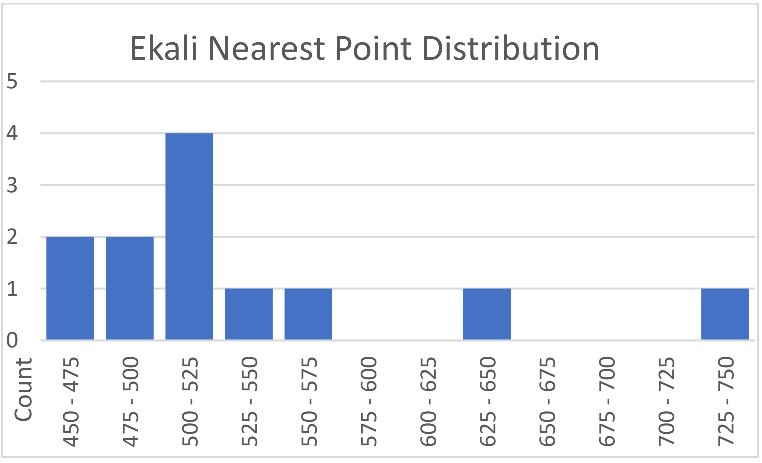
In the case of Dionysos, the concluding value for k was 34 and yielded a mean average nearest distance of 615 metres, higher than the desired 500 metres. k could not be increased to a higher value than 34 to bring the mean closer to our target because as the graph below displays there was already high concentration of points at proximity and is not a normal distribution. The largest count of distances, which is 8, is in the 200 - 300 interval.



k	34
Minimum	230.54
Maximum	1538.43
Mean	615.36
Standard Deviation	376.56

Figure 29: Nearest distances per point graph and data in Dionysos

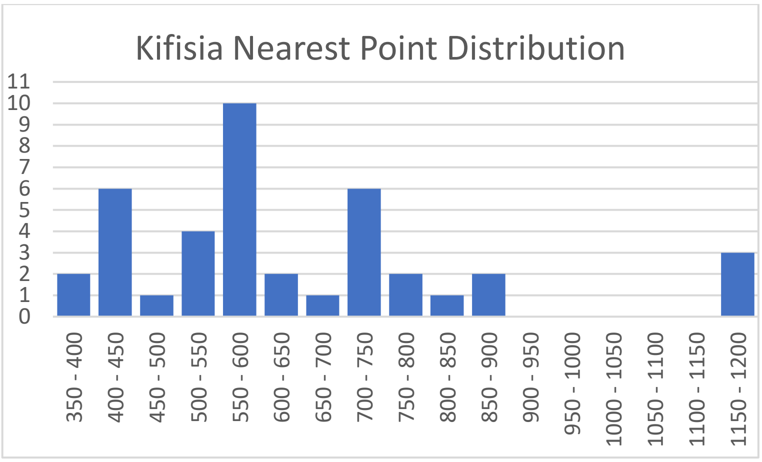
Ekali was finalised at k = 12 with a distribution closer to normal. More points are below the mean value but an attempt with k less than 12 did not seem to cover the area adequately.



K	12
Minimum	452.81
Maximum	726.84
Mean	534.36
Standard Deviation	74.49

Figure 30:Nearest distances per point graph and data in Ekali

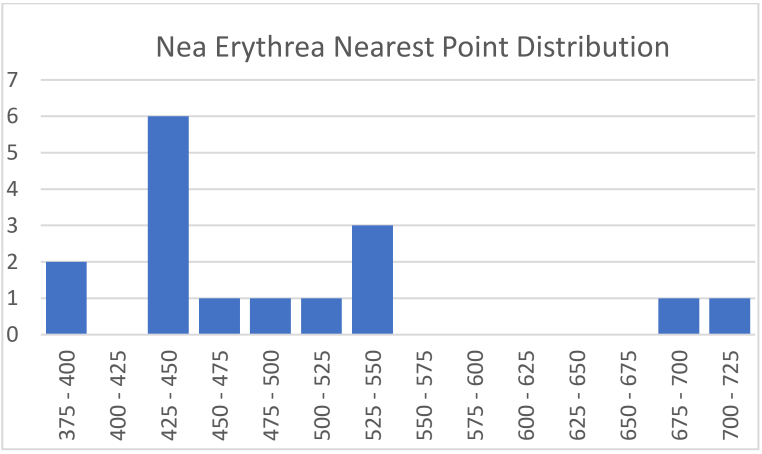
Kifisia was unique in that it contained both very sparse and very dense areas. Due to its size and overall density, it gives the largest k value.



Count	40
Minimum	367.43
Maximum	1172.54
Mean	636.90
Standard Deviation	202.69

Figure 31:Nearest distances per point graph and data in Kifisia

Nea Erythrea has a mean value close to the desired target with the majority of points below the average.



Count	16
Minimum	385.97
Maximum	718.35
Mean	492.58
Standard Deviation	90.55

Figure 32:Nearest distances per point graph and data in Nea Erythrea

Just as in the case of Ekali, the range of distances is smaller with a range of ~403m in Nea Erythrea and ~274m in Ekali. Kifisia has a range of ~805m and Dionysos ~1308m

3.2 Forming a final set of incidents

The purpose of all the previous tasks was to create a set of hypothetical incidents that will serve as points of origin and destination for a routing request. These locations will, in the next steps, be prepared to serve as input for the routing Web API. These steps aim at converting the locations we have in our GIS tool to a format that can be easily fed to the automated process. The tasks performed are the following:

- our collection of features was projected to a geographical coordinates system that is appropriate for the Web API, usually WGS 84
- the latitude and longitude for each point were calculated in decimal degrees
- the collection was saved in a text file format such as CSV

Specifically, the file maintained the data in the following schema so it can be properly read by the following automated process. The file was in .txt format with values separated by a comma and contained the following data: Each point's objectID, WGS84 latitude and longitude, and the Greek Grid coordinates so that the distance between the point and the point of departure/destination can be calculated later on. The clusterID can be included but is not useful and in cases has duplicates. The order of the above should be OBJECTID, CLUSTER_ID, Lat, Long, X, Y and have these exact title headers. Finally, the English numbering format was used, and decimals were distinguished using the point and not the comma. An example is provided next (spaces were added, and decimals were trimmed for readability purposes).

```
OBJECTID, CLUSTER_ID, Lat, Long, X, Y
 1,  5, 38.092648, 23.870710, 488513.37, 4215814.67
 2,  8, 38.072170, 23.918687, 492718.41, 4213537.81
 3,  0, 38.072980, 23.928521, 493580.99, 4213626.97
 4, 12, 38.098852, 23.876984, 489064.48, 4216502.25
 5, 20, 38.090710, 23.918163, 492674.27, 4215594.94
 6, 10, 38.092181, 23.891665, 490350.84, 4215760.53
 7, 28, 38.081008, 23.890707, 490265.36, 4214520.95
 8, 18, 38.103587, 23.884212, 489698.85, 4217026.88
 9, 22, 38.104620, 23.849107, 486621.21, 4217145.83
10, 25, 38.105462, 23.859911, 487568.65, 4217237.82
11, 13, 38.081645, 23.881771, 489481.76, 4214592.54
12, 13, 38.081288, 23.881871, 489490.46, 4214552.97
13, 11, 38.092775, 23.855142, 487148.31, 4215830.73
14,  3, 38.072513, 23.900318, 491107.29, 4213577.41
15, 23, 38.068681, 23.910653, 492013.34, 4213151.28
```

Page intentionally blank

4. Route creation and response manipulation

The purpose of this chapter is to present the choices of routing web APIs and explain how these were implemented. Since the object of the thesis was to compare the route web APIs of proprietary and crowdsourced road network data, the options that were investigated were those of the open-source routing machine OSRM and the Google API services. The Google API routing service is the highly popular service provided by Google which is also used for its Google Maps page. The OSRM, as the name suggests, is a project for developing a routing machine that provides routes based on the data from the OpenStreetMap platform. These choices were immediately adopted based on their popularity. Their popularity also ensured extensive documentation, simplicity in implementation and a plethora of resources on their use on the Internet. Each one is presented individually in the following 2 chapters.

4.1 OSRM Routes API

The open-source routing machine (OSRM) [16] is an open-source project [17] for a routing engine written in C++ that calculates shortest paths in road networks based on OSM data and supports a variety of services and options such as vehicle type.

The services provided by the open-source routing machine include the following:

- nearest - returns the nearest match from a given set of coordinates to the street network
- route - finds the fastest route between points
- table - calculates all distances or times between all pairs although given coordinates
- match - snaps GPS traces to the road network
- trip - provides a solution to the travelling salesman problem using a greedy heuristic
- tile - generates map box vector tiles with internal routing metadata

We are interested in the route service alone. The machine calculates routes on one of either two pre-processed graphs. One prepared with contraction hierarchies (CH) and one with Multi-Level Dijkstra (MLD), so it can efficiently handle both small-sized and continental-sized requests. The routing machine can be used in ways other than the HTTP interface. It can be used as a library via C++ (Docker images with lightweight backend and frontend implementations for Debian provided) or as a Node.js wrapper. Advantages in doing so include the ability to use customised OSM data (pre-processors provided for both graph reduction methods), fine-tuning of the results and much less overhead. None of the above was tested or used in any way as it extends beyond the scope of this paper.

According to [18], the HTTP request must conform to the following format:

```
GET
/{service}/{version}/{profile}/{coordinates}[.{format}]?option=value&option=value
```

The description of the parameters is given in the following table:

Table 15: OSRM request mandatory parameters

Parameter	Description
`service`	Is the service type and takes one of the following values: `route`, `nearest`, `table`, `match`, `trip`, `tile`
`version`	The version of the protocol implemented by the service. `v1` for all OSRM 5.x installations
`profile`	Mode of transportation is determined statically by the Lua profile that is used to prepare the data using `osrm-extract`. Typically, `car`, `bike` or `foot` if using one of the supplied profiles.
`coordinates`	String of format: `{longitude},{latitude};{longitude},{latitude}[;{longitude},{latitude} ...]` or `polyline({polyline}) following Google's polyline format with precision 5 or polyline6({polyline6})`.
`format`	Only `JSON` is supported at the moment. This parameter is optional and defaults to `JSON`.

The API documentation provides an example of such a request using cURL.

```
# Query on Berlin with three points:
curl 'http://router.project-osrm.org/route/v1/driving/13.388860,52.517037;13.397634,52
.529407;13.428555,52.523219?overview=false'
```

Entering the requested URL in a browser (without the cURL command and quotes) will return the following response.

```
{
  "routes": [
    {
      "legs": [
        {
          "summary": "",
          "weight": 766.9,
          "duration": 606,
          "steps": [],
          "distance": 1884.9
        },
        {
          "summary": "",
          "weight": 720.3,
          "duration": 524,
          "steps": [],
          "distance": 2948.2
        }
      ],
      "weight_name": "routability",
      "weight": 1487.1999999999998,
      "duration": 1130,
      "distance": 4833.1
    }
  ],
  "waypoints": [
    {
      "hint":
"yffjiFn444gcAAAAdQEAAAAAAAAAAAAASjFaQU1xpUEAAAAAAAAAAABwAAAB1
AQAAAAAAAAAAAAAABpgAA_kvMAKlYIQM8TMwArVghAwAA7wqMrqJ-",
      "distance": 4.231665624816857,
      "name": "Friedrichstraße",
      "location": [13.388798, 52.517033]
    },
    {
      "hint": "LHgegHlg5YgLAAAADAAAAAAAAAAACAQAAW7-
PQOKcyEAAAAAApq6DQgsAAAAMAAAAAAAAAIoAAAABpgAAf27MABiJIQOCbswA
_4ghAwAAXwWMrqJ-",
      "distance": 2.7893928415656375,
      "name": "Torstraße",
      "location": [13.397631, 52.529432]
    },
    {
      "hint":
"aXYegP___38iAAAAyAAAACkAAABKAAAAsowKQkpQX0Lx6yZCvsQGQiIAAABk
AAAAKQAAACUAAAAABpgAASufMAOdwIQNL58wA03AhAwMAvxCMrqJ-",
      "distance": 2.2265954222656257,
      "name": "Platz der Vereinten Nationen",
      "location": [13.428554, 52.523239]
    }
  ],
  "code": "Ok"
}
```

The above data is in JSON format. Anything in curly braces is considered an object and anything in brackets is an array. Items left and right of colons are field names and their equivalent values. The response object has 3 fields: Routes, Waypoints, and Code. The value of OK for the code field indicates that the response had no issues. The routes' array contains objects of routes; the array in the example above contains only one route. For each route, there is a legs array and the attributes of the route. This route is made up of two legs, one with a duration of 606 seconds and a distance of 1884.9 metres and the 2nd one with a duration of 524 seconds and a distance of 2948.2 metres. The file provides the sum after the legs indicating that it has a total duration of 1130 seconds and a distance of 4833.1 metres.

The weight points array contains 3 waypoint objects providing their location in Lat/Long coordinates, the distance from the input coordinate to the snapped coordinate, the name of the road and a hint. The hint is a unique internal identifier of the segment that can be used on subsequent requests to speed up the query and connect multiple services.

Specific to the route service are the following general options that can be supported. They are presented together with the possible values and the description as provided by the API documentation.

Table 16:OSRM request optional parameters

Option	Description	Values[6]
Alternatives	Search for alternative routes. Passing a number (`alternatives=n`) searches for up to n alternative routes. A result is not guaranteed if requested.	`True` `False` (default) Number n
Steps	Returned route steps for each route leg	`True` `False` (default)

Annotations	Returns additional metadata for each coordinate along the route geometry.	`True`
		`False` (default)
		`Nodes`
		`Distance`
		`Duration`
		`Datasources`
		`Weight`
		`Speed`

Geometries	Returned route geometry format (influences overview and per step)	`Polyline` (default)
		`Polyline6`
		`GeoJSON`

Overview	Add overview geometry either full, simplified according to highest zoom level it could be displayed at, or not at all.	`Simplified` (default)
		`Full`
		`False`

Continue _ straight	Forces the route to keep going straight at waypoints constraining U-turns there even if it would be faster. Default value depends on the profile.	`Default` (default)
		`True`
		`False`

Waypoints	Treats input coordinates indicated by given indices as waypoints in returned Match object. Default is to treat all input coordinates as waypoints.	`{index};{index};{index}...`

The requests formed for the thesis were of the following format:

```
http://router.project-
osrm.org/route/v1/car/a;b?geometries=geojson&overview=full&st
eps=true
```

This is a request for a route between 2 points (a and b) for which the mode of transport is car, the geometry reply is in GeoJSON format and a full overview of the route with steps information will be provided. A note must be made that the points must be in Long/Lat order as opposed to the popular Lat/Long that is often encountered.

4.2 Google Directions API

The Google Directions API which is part of the Google Maps platform is a service that calculates directions between points using an HTTP request in a similar way to the one presented above. It can return results for several modes of transportation like cycling, walking, driving and even public transport. It returns multipoint directions using a series of waypoints and can even accept

[6] Options and values have been capitalized. See online documentation for proper case usage in http request.

addresses or place IDs as well as latitude/longitude coordinates as designated points for the route.
At the moment of writing, the Google Maps platform provided several APIs with services connected to their products. These fell into 3 major categories **Maps**, **Routes** and **Places** [19].
The 3 Routes APIs provided by Google [20] were the following:

- the **directions API**, explained above
- the **distance matrix API**, which provided travel distance and time for a matrix of origins and destinations
- the **roads API**, which identifies the roads a vehicle was travelling along and provided additional metadata about those roads such as speed limits

Prior to using any of Google's API services an API key is required. To obtain one, an account must be created with the Google Cloud Platform. The Google cloud platform is a suite of services running on infrastructure provided by Google and used by the company itself. The services provided could be networking services, cloud AI services, computational services, API platforms, IoT services and many more.

According to [21], the API key is part of the request which takes the following general form:

```
https://maps.googleapis.com/maps/api/directions/outputFormat parameters
```

In the above syntax, the output format may be a JSON or an XML value. The request parameters can be either required or optional. The required parameters are the following:

- origin, this is a parameter that indicates the position from which you want the calculated route to start. It can be entered in either the address form, a Lat/Long value, or the id of a place from the list of places Google has on its Maps
- destination, the position to which you want the calculated route to arrive. It can be entered in one of the formats described above
- the API key

The optional parameters are the following:

- mode
- waypoints
- alternatives
- avoid
- language

- units
- region
- arrival time
- departure time
- traffic model
- transit mode
- transit routing preference

The typical request formed in the automation code takes the following form

```
https://maps.googleapis.com/maps/api/directions/json?origin=3
8.11656516,23.82942926&destination=38.11343398,23.82478286&ke
y=ABcdEfGhIJkL_1MN2OPQRstUv345wXy6zaB7CDE
```
[7]

This is broken down into the following parts

`https://maps.googleapis.com/maps/api/directions/`	The request URL for the Directions API
`json?`	The output format
`origin=38.11656516,23.82942926&`	The point of departure in Lat/Long decimal degrees format
`destination=38.11343398,23.82478286`	The point of destination in Lat/Long decimal degrees format
`key=ABcdEfGhIJkL_1MN2OPQRstUv345wXy6zaB7CDE`	The API key

The response from the previous request will be in JSON format (as declared in the request) and is presented in the contracted form next[8].

```
{
   "geocoded_waypoints" : […],
   "routes" : [
      {
         "bounds" : {…},
         "copyrights" : "Map data ©2020",
         "legs" : […],
         "overview_polyline" : {…},
         "summary" : "A/D PAThE/NEO Athinon Thessalonikis/A1/E75",
         "warnings" : [],
         "waypoint_order" : []
      }
   ],
   "status": "OK"
}
```

[7] The key displayed is fictional based on an actual key obtained for the purposes of the thesis
[8] Repeated elements within the response have been omitted for clarity

Status

The status element indicates the response is a valid result.

Geocoded Waypoints

The geocoded waypoints element contains the list of points (in this case the origin and destination) in the order approached along with metadata for the result of identifying the location as well the type and an ID for use with other Google APIs. An example follows.

```
{
   "geocoder_status": "OK",
   "place_id": "ChIJi1lWkD6eoRQRSTWnz45FbLs",
   "types": [ "street_address"]
}
```

Routes

The routes element is an array of other elements that contain information pertaining to the result of the request if it is successful. These elements will be presented individually.

Routes | Bounds

The bounds element contains 2 diagonal coordinates that can define the viewport of the route.

```
"northeast": {
      "lat": 38.1405523,
      "lng": 23.845441
},
"southwest" : {
   "lat" : 38.1134199,
   "lng" : 23.8247848
}
```

Routes | Copyright

The copyright element is an attribution to the map owner to be used by the user.

```
"copyrights": "Map data ©2020",
```

Routes | Legs

The legs element is an array of other elements. Each leg is a route between 2 waypoints. Since there are only 2 points in the request, the legs in the reply will contain only one leg.

Each leg contains the following

```
{
   "distance" : {…},
   "duration" : {…},
   "end_address" : "A/D PAThE, Kifisia 146 71, Greece",
   "end_location" : {…},
   "start_address" : "Agiou Gerasimou 20, Ekali 145 78, Greece",
   "start_location" : {…},
   "steps" : […]
}
```

The contents of each element in a leg are presented more thoroughly after the elements of routes.

Routes | Overview polyline

The overview_polyline element contains an encoded polyline representing a single collection of points That make up the approximated line of the directions.

```
"overview_polyline" : {
   "points" :
"sssgFsdmpCg@U]~@mBsBu@eAsAuAkBkBsBkByAiAcB}AcEgEmBsBq@y@sAwAoAmA_CgBmD
_Ba@GaAc@cCgAiBs@yAi@_Cw@gCaAyB}@cAa@a@MgBe@oBw@kFsBgFqBsEiB_EgBmFmBeBo
@cBy@gE_Cy@o@U_@Se@OcACg@AmBEi@Ki@Og@Y_@WUUI{AKs@A@nA?zAKz@W`Aa@hA}@zB_
@dAm@`BOn@`@Jb@Db@Bb@?hAGdAUh@Kh@q@NQZOR?\\Hz@TZD^AtBfArFxBdOvFv_@vNlGx
BdBl@vBx@bBv@lAp@jBfA|AdAlAv@zAnA`EpDnKdKdEbExC~C`EhEdEdEpDbDrBlBx@t@HZ
r@`BdBfBPRC\\"
}
```

As explained in [22], the encoded polyline algorithm is a lossy compression algorithm allowing the storage of a series of coordinates in a single string. The process converts the value into a series of character codes for ASCII characters using the base64 encoding scheme. For example, the following 3 points would be encoded into the following polyline:

- Points: (38.5, -120.2), (40.7, -120.95), (43.252, -126.453)
- Encoded polyline: `_p~iF~ps|U_ulLnnqC_mqNvxq`@`

Detailed steps for the encoding can be found at [22]

Routes | Summary

The summary element provides a short description for the route which in the example is a referral to the location of the destination.

"summary": "A/D PAThE/NEO Athinon Thessalonikis/A1/E75"

Routes | Warning

The warnings element Is an array of warnings to be displayed when the directions are shown. It is empty in the example provided.

Routes | Waypoint order

The waypoint order element is another array indicating the order of the waypoints for the calculated route. This is used in the cases where more than two points have been provided for a route and the request contains the “optimise route” parameter set to true.

Each leg element contains the following elements.

Routes | Legs | Distance

The distance element contains an indication of the distance of the route in metres, in both number and approximated text format.

```
"distance" : {
   "text" : "6.9 km",
   "value" : 6856
}
```

Routes | Legs | Duration

The duration element contains an indication of the travel time over the route. it is provided in both seconds and approximated text.

```
"duration" : {
   "text" : "8 mins",
   "value" : 489
},
```

Route | Legs | Start/End Address

The start_address and end_address elements contain addresses in a human-readable format. These derive from the reverse geocoding of the origin and destination points

```
"start_address" : "Agiou Gerasimou 20, Ekali 145 78, Greece",
"end_address" : "A/D PAThE, Kifisia 146 71, Greece",
```

Route | Legs | Start/End Location

The start and end location elements contain the actual latitude/longitude coordinates of the first and final points in a route. This is provided for every request to indicate the exact location on the map that was matched to the points of the request. This is useful in the cases where points provided in the request do not fall exactly on a location that is directly navigable.

```
"start_location": { "lat" : 38.11658449999999, "lng" : 23.829377 },
"end_location": { "lat" : 38.113442, "lng" : 23.8247848}
```

Route | Legs | Steps

The steps element Is an array containing a collection of single steps. One step is one specific single travelling instruction which is part of the entire journey. The following element contains two steps.

```
"steps": [
   {…},
   {…},
],
```

Each step contains the following elements:

```
{
   "distance": {"text" : "24 m", "value" : 24},
   "duration": {"text" : "1 min", "value" : 3},
   "start_location" : {"lat" : 38.11658449999999, "lng" : 23.829377},
   "end_location": {"lat" : 38.1167783, "lng" : 23.8294924},
   "html_instructions": "Head \u003cb\u003enortheast\u003c/b\u003e
toward \u003cb\u003eAgiou Gerasimou\u003c/b\u003e",
   "polyline" : {"points" : "sssgFsdmpCGC_@Q"},
   "travel_mode" : "DRIVING"
}
```

Some of the elements in a leg have been explained already e.g. distance and duration. The elements that appear for the first time are explained below.

Route | Legs | Steps | Html Instructions

This contains instructions that would be provided to a human to explain the directions each step requires. It is given in HTML format and can be used “as is” in an application.

For example, the following part from the response

```
"html_instructions": "Head \u003cb\u003enortheast\u003c/b\u003e toward
\u003cb\u003eAgiou Gerasimou\u003c/b\u003e",
```

is rendered in html as

```
Head <b>northeast</b> toward <b>Agiou Gerasimou</b>
```

and ends being displayed as

Head **northeast** toward **Agiou Gerasimou**

Route | Legs | Steps | Travel mode

This specifies the type of travel selected

4.3 Task flow

This chapter describes the code written that performs the automated process of the route requests. The whole process of requesting the routes, converting, extracting, and analysing the information from the replies is spread over 4 different files written in Python.

The first file is responsible for the creation of the routes, the archiving process of the replies and the creation of files from the replies that can be used on the map. It contains 3 functions the main function, the osrm() function and the grm() function.

The general purpose of this file is to open and read the text file with the list of points and their coordinates and go into an iterative process in which for every pair of points it creates a route. To cover all possibilities without repetition the code performs two loops, one embedded in the other, with the first point always being a precedent of the one in the second loop and by creating the routes for both directions for every pair before proceeding with a new pair. For example, for 5 points it would start by appointing point 1 for the outside loop and every remaining point in the inside loop thus creating the routes for the following pairs (1,2) & (2,1), (1,3) & (3,1), (1,4) & (1,4), (1,5) & (5,1). This will create all the routes to and from point 1. The point in the outside loop would then go to 2 and through the inside loop create all the remaining routes from and to point 2. The process is repeated until the outside loop point is the one before last. At every repetition of the loop, the program calls the function that creates the request for the first of the two providers, OSRM, by providing it with the two points and the name of the file in which the reply needs to be saved. It then proceeds with flipping the latitude and longitude in the coordinates of the points and by calling the function that creates the request for the second of the two providers, Google, again giving it the two points and the name of the file in which the reply needs to be saved.
Each of the functions forms the URL for the HTTP request and through an exponential backoff algorithm sends the request to the server. The purpose of the exponential backoff algorithm is to wait and perform the request at a later time if the request is unsuccessful due to the server being busy. The amount of time the program waits for doubles after every failure leading to an exponential increase of the waiting time. The request is performed with the use of the "Requests" library [23] as mentioned in chapter 2.3.1.

Each reply is saved in a file with the following naming convention.

```
<provider>_<fromID>_to_<toID>.json
```

For example, the names for the routes between points 5 and 22 would be:

```
osrm_5_to_22.json
google_5_to_22.json
osrm_22_to_5.json
google_22_to_5.json
```

The flowchart on the next page displays the general steps of the process.

When all the files are saved, the program calls the JSON to GeoJSON conversion python file. This programme reads the folder where the files were saved, uses the filename to identify the route provider and depending on that, reads the file contents for the coordinates of the route. It then uses those coordinates to create a file of GeoJSON format that can be used for a route to be displayed on a map and saves it. The most challenging part of this process was to decode the google polyline string but a ready solution was found at [24].

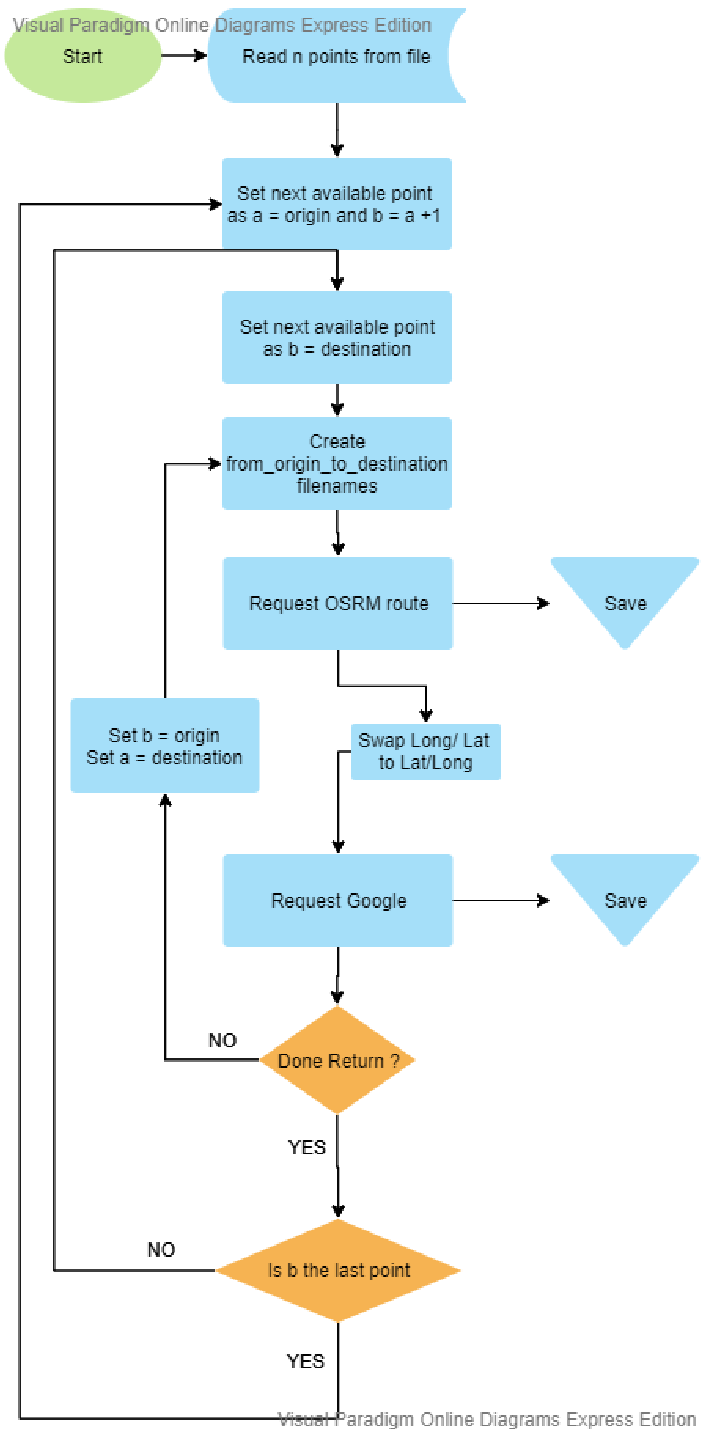

Visual Paradigm Online Diagrams Express Edition
Start
Read n points from file
Set next available point as a = origin and b = a +1
Set next available point as b = destination
Create from_origin_to_destination filenames
Request OSRM route
Save
Swap Long/ Lat to Lat/Long
Set b = origin Set a = destination
Request Google
Save
NO
Done Return ?
YES
NO
Is b the last point
YES
Visual Paradigm Online Diagrams Express Edition

Figure 33: Request creation flow chart

After all the replies have been collected in JSON and GeoJSON format, a python file responsible for the creation of all point data is run. Because the route might not start or end at the given point, if for example, the point is not accessible by car, we need to calculate the distance between the original point and the point the route actually approaches. This can happen to routes from either provider so all points from the collection of the ones in the list, must be found and examined. Whether the point is the origin or the destination in the route does not matter since either way the routing machine will just try and match to the nearest point on the network. To calculate the distance between the original point and the point that the route finally used, all points need to be transformed to a projected coordinate system. This program uses the PyProj library mentioned in chapter 2.3.1 to quickly convert the coordinates of a point from one system to another. The library can be found on [25]. One more library used in this programme was the SciPy library [26] and specifically the distance computations on [27]. More specifically the Euclidean Distance function explained on [28] was used to calculate the distance between the original point and the point on the route returned.

This file reads the collection of points from the list of points of the area and creates a Point Class object for each location. The class diagram of the point object is provided in the picture. It opens two JSON route files, one from each provider, pertaining to this point. It then sets on to extract, calculate and assign the following values to the equivalent properties of the point instance:

- the id of the point,
- the decimal degree coordinates
- the projected coordinates of the point
- the decimal degree coordinates of the nearest point on the OSRM route
- the projected coordinates of the nearest point on the OSRM route
- the decimal degree coordinates of the nearest point on the Google route
- the projected coordinates of the nearest point on the Google route

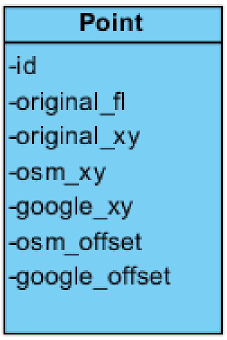


Figure 34: Class diagram for Point

After all the data for all the points have been assembled, it creates a file named pointData.<*name_of_the_area*>.json and dumps it in JSON format.

The file is ready to be used by the viewing tool. Part of it appears in the following snippet:

```
{
  "points": [
    {
      "id": 1,
      "original_fl": ["38.10652476", "23.84526065"],
      "original_xy": [ 486284, 4217357],
      "osm_xy": [ 486284, 4217357 ],
      "google_xy": [ 486282, 4217357 ],
      "osm_offset": 0.0,
      "google_offset": 2.0
    },
    {
      "id": 2,
      "original_fl": ["38.10180262", "23.83540469"],
      "original_xy": [485419, 4216835],
      "osm_xy": [485419, 4216835],
      "google_xy": [485419, 4216836],
      "osm_offset": 0.0,
      "google_offset": 1.0
    }
}
```

The final python file creates a JSON file with information on routes between the pairs of points. During runtime, a Route Class is defined with the specifications seen in the next class diagram.

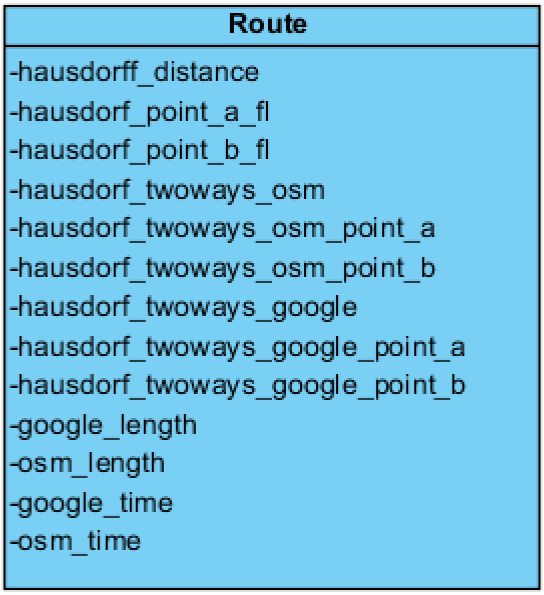


Figure 35: Class diagram for Route

A route instance is created for every pair of points in the area. The programme then goes through each pair, opening the routes from both providers, extracting the necessary data, performing the calculations, and setting the values for the equivalent attributes in the object. The Hausdorff distance is measured using the directed_hausdorff function found at [29]. This function will receive two arrays u, v and return the directed distance between u and v, the index of the point in u and the index of the point in v.

After running the function, the points are identified in the array and their coordinates are found, converted to geographic coordinates, and entered as Hausdorff_point_a_fl and Hausdorff_point_b_fl in the object's properties. This is used later to show the exact points where the distance occurs. The function runs for both directions of the directed Hausdorff and the result with the max distance is kept as the distance between the two sets.

The same process is applied to measure the distance between the route and its reverse route according to the same provider. An example for clarification is given. For points 1 and 5, we calculate the Hausdorff distance between the routes from 1 to 5 between Google and OSRM. Then, the Hausdorff distances between the routes from 1 to 5 and the returning route from 5 to 1 for both Google and OSRM are also calculated. This is done to discover cases where the return is substantially different to the original. When all routes have their data calculated, the file is saved with the naming format routeData.<*name_of_the_area*>.json.

The file is ready to be used by the viewing tool. Part of it appears in the following snippet:

```
{
 "routes": {
  "1_to_2": {
   "hausdorff_distance": 45.23,
   "hausdorff_point_a_fl": [38.10403098688873, 23.841903991446365],
   "hausdorff_point_b_fl": [38.1038899868894, 23.841419991446905],
   "hausdorff_twoways_osm": 96.77,
   "hausdorff_twoways_osm_point_a":
        [38.10166098689931,23.834315991455117],
   "hausdorff_twoways_osm_point_b":
        [38.10180298689819, 23.835404991454148],
   "hausdorff_twoways_google": 96.49,
   "hausdorff_twoways_google_point_a":
        [38.101689986899274, 23.83430999145508],
    hausdorff_twoways_google_point_b":
        [38.101809986898196, 23.83539999145414],
   "google_length": 1140,
   "osm_length": 1141.8,
   "google_time": 131,
   "osm_time": 127.2
  },
  "1_to_3": {
  "hausdorff_distance": 61.61,
  "hausdorff_point_a_fl": [38.10876998687331, 23.84953999143413],
  "hausdorff_point_b_fl": [38.10865298687406, 23.84885299143478],
  "hausdorff_twoways_osm": 107.07,
  "hausdorff_twoways_osm_point_a":
        [38.110286986867834, 23.852710991429703],
  "hausdorff_twoways_osm_point_b":
        [38.10970798686974, 23.851733991431235],
  "hausdorff_twoways_google": 60.69,
  "hausdorff_twoways_google_point_a":
        [38.11194998686339, 23.854119991426295],
  "hausdorff_twoways_google_point_b":
        [38.11140998686439, 23.854229991427008],
  "google_length": 1061,
  "osm_length": 1052.3,
  "google_time": 126,
  "osm_time": 121.6
  }
 }
}
```

Page intentionally blank

5. Online Viewing Tool

As explained previously, the choice of a Web GIS App for the presentation of the results was made based on the need to display spatial and tabular data in a fast and easy way that facilitates sharing. The tool should be simple so that both experts and non-specialists alike can view the results and draw conclusions. The basic concept involves the existence of a web page that contains a dropdown selector for choosing an area, has a map displaying the points of the selected area, provides two drop-down lists for the selection of the origin and destination points and upon selection will portray the routes between the two on the map. Information and comparative figures of the selected routes from the two providers should be displayed on the side of the map.

The appearance of JavaScript mapping libraries like Leaflet [30] has made the process very easy and someone with basic HTML skills and some JavaScript can quickly and with little effort set up a web page containing a map. Unlike other applications in which there is a GIS server serving the data in an OGC compatible format like WMS, all data has been pre-processed and stored in JSON and GeoJSON files using Python. The functionality of the page starts with the selection of the area of interest from the ones available in the dropdown menu. The list of points is then dynamically populated with the number of points of that area. At every change of the pair of points, the equivalent files are requested asynchronously and loaded as layers onto the map. The point and route data related to the selection is extracted from the equivalent pointData.<area> and routeData.<area> files and displayed in the appropriate fields on the webpage. The whole page is contained in a standard index.html file for the elements and a main.js file for the functionality. This functionality is provided through 14 functions which are called at specific user caused events. The activity diagram on the following page displays the sequence of functions and chain of other functions called subsequently. The only function that is triggered automatically is the creation of the legend object for the map which is called once the page has finished loading.

This approach based on HTML and JavaScript that loads GeoJSON onto a map control and reads JSON files simplifies the deployment and migrating process since it requires no server software like Apache or GeoServer to be setup. To make the app more responsive, user-friendlier and more applicable across various viewport sizes, the bootstrap library [31] was used in the development stages.

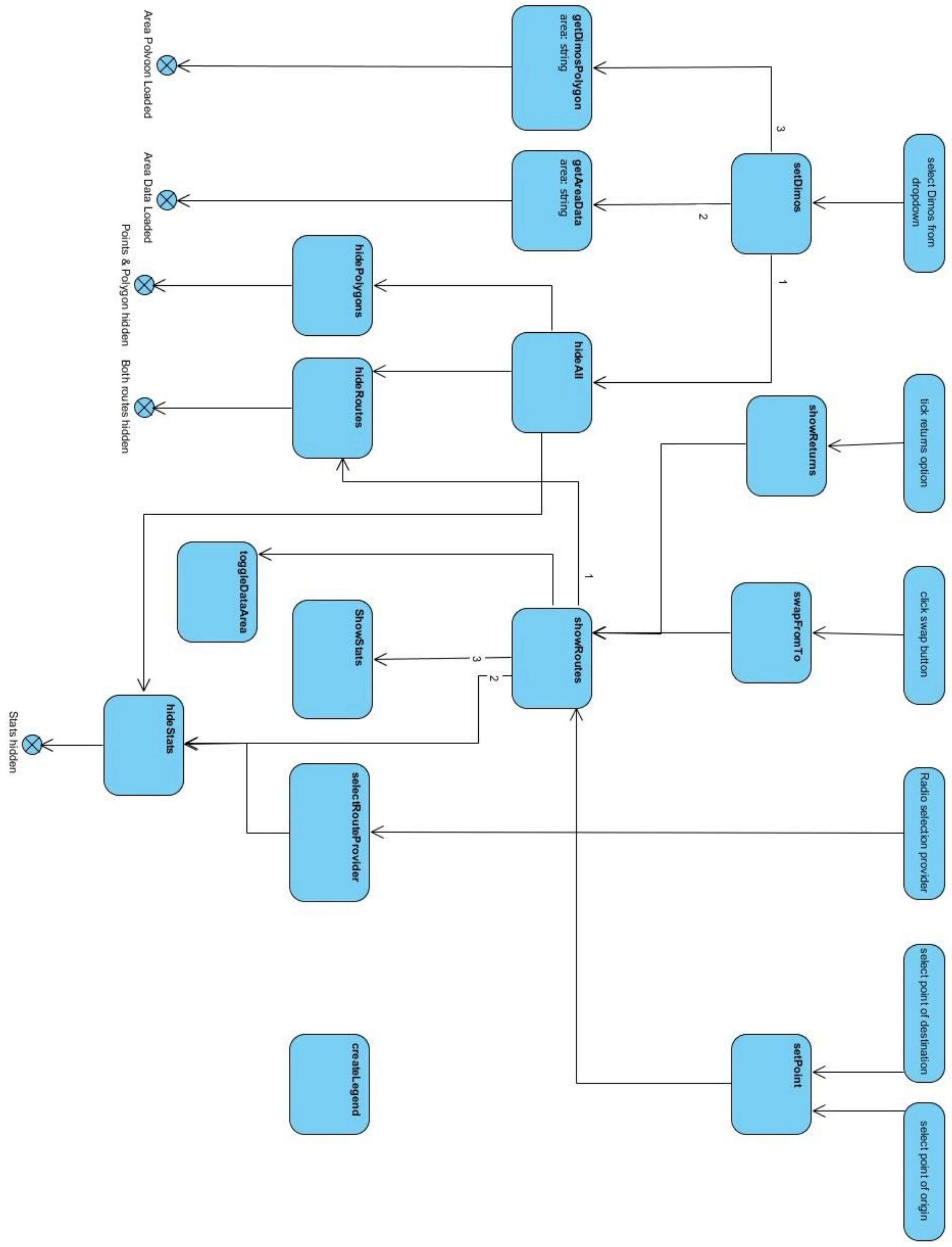
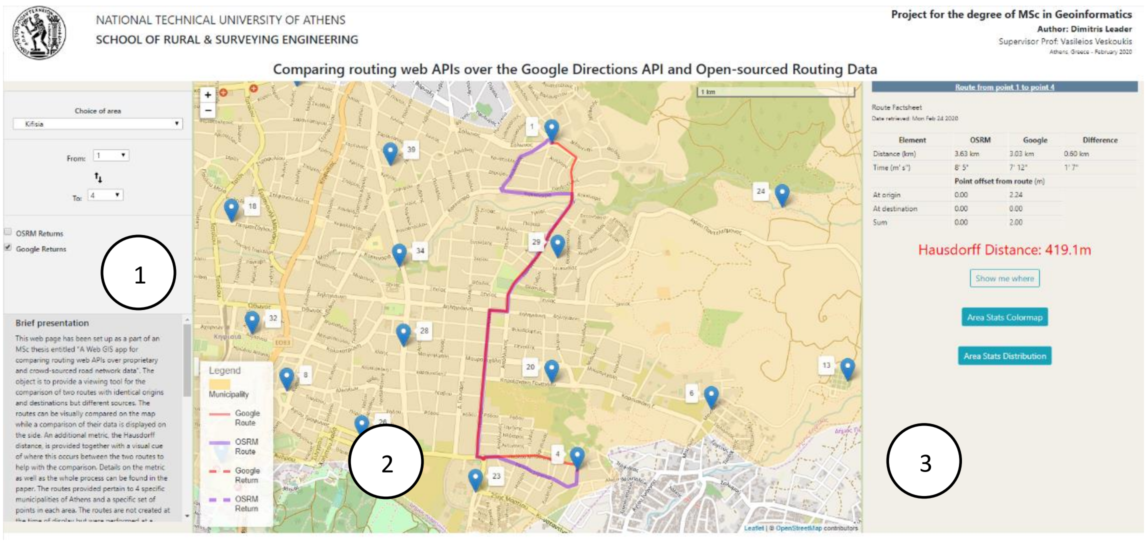


Figure 36: Activity diagram for Javascript functions in viewing tool

The following image depicts the final web App with the area of Dionysos selected.



NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF RURAL & SURVEYING ENGINEERING

Project for the degree of MSc in Geoinformatics
Author: Dimitris Leader
Supervisor Prof: Vasileios Veskoukis
Athens, Greece - February 2020

Comparing routing web APIs over the Google Directions API and Open-sourced Routing Data

Choice of area: Kifisia

From: 1

To: 4

☐ OSRM Returns
☑ Google Returns

Brief presentation

This web page has been set up as a part of an MSc thesis entitled "A Web GIS app for comparing routing web APIs over proprietary and crowd-sourced road network data". The object is to provide a viewing tool for the comparison of two routes with identical origins and destinations but different sources. The routes can be visually compared on the map while a comparison of their data is displayed on the side. An additional metric, the Hausdorff distance, is provided together with a visual cue of where this occurs between the two routes to help with the comparison. Details on the metric as well as the whole process can be found in the paper. The routes provided pertain to 4 specific municipalities of Athens and a specific set of points in each area. The routes are not created at

Route from point 1 to point 4

Route Factsheet
Date retrieved: Mon Feb 24 2020

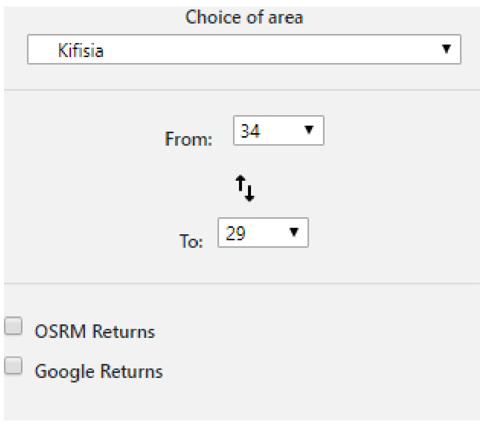
Element	OSRM	Google	Difference
Distance (km)	3.63 km	3.03 km	0.60 km
Time (m' s")	8' 5"	7' 12"	1' 7"
	Point offset from route (m)		
At origin	0.00	2.24	
At destination	0.00	0.00	
Sum	0.00	2.00	

Hausdorff Distance: 419.1m

Figure 37: Web App in full implementation on desktop

The page is separated into three discrete areas. On the left of the map we have the data input area (1) where the choice of the parameters of the route is set, the map is in the middle (2) and the area where the route data is displayed is on the right (3).

The parameters settings in area 1 include the choice of area of interest, two drop-down lists programmatically populated with the number of points according to the selected area, a swap button to easily compare the data of the returning routes, and a tick box for each provider if we wish to compare the travel route to its return.



Choice of area: Kifisia

From: 34

To: 29

☐ OSRM Returns
☐ Google Returns

Figure 38: Area, point and route parameters

The map area displays the two routes to be compared. The Leaflet library provides ready to use solutions for implementing scale bars, zoom buttons, a legend, feature labels, and many more features.

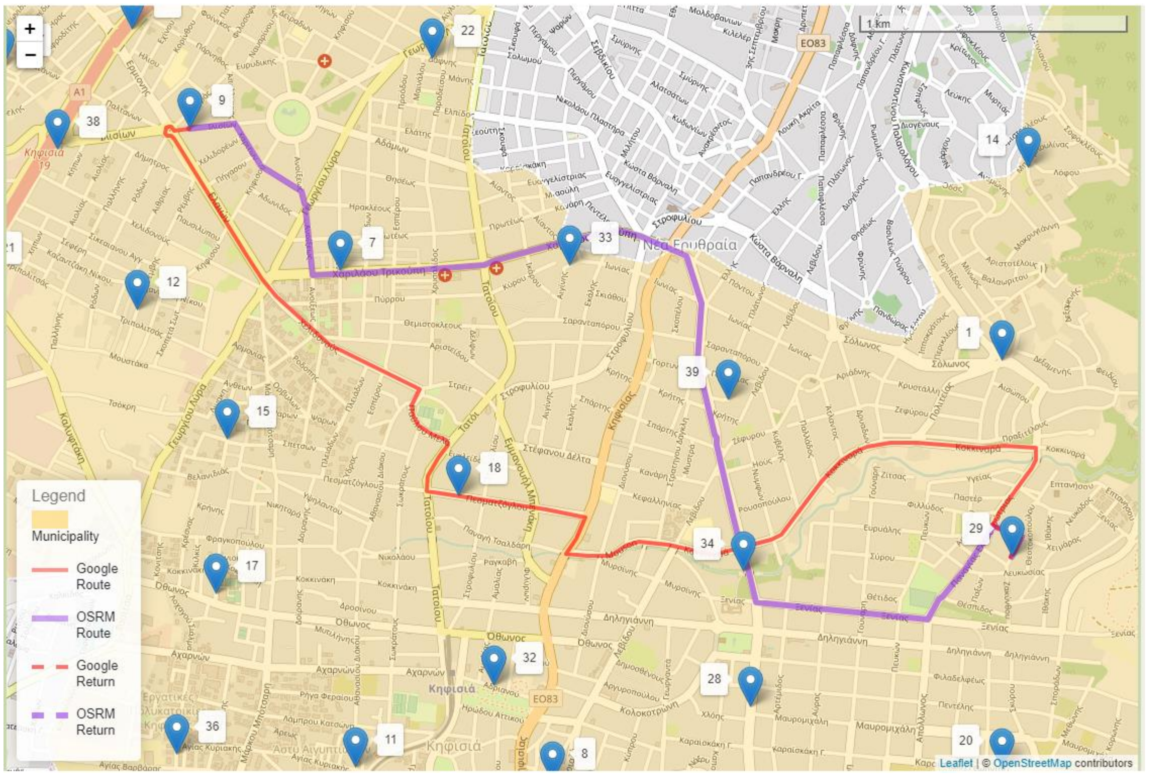
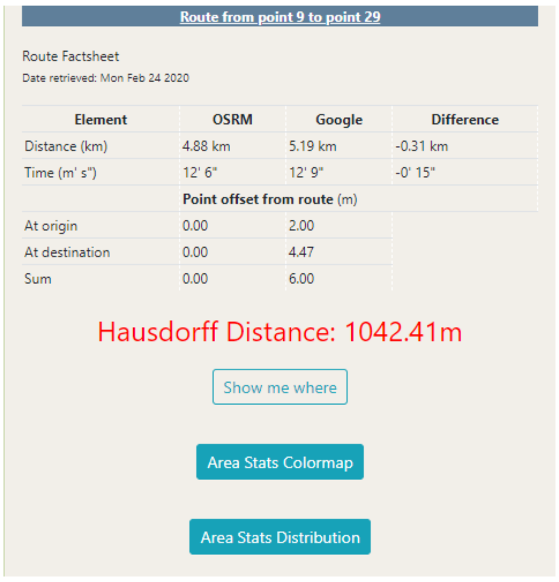


Figure 39: Map in viewing tool displaying the routes from the 2 service providers

The data area on the right provides the data of the routes, sums and differences as well as buttons to auxiliary metrics, graphs, and tables



Route from point 9 to point 29

Route Factsheet
Date retrieved: Mon Feb 24 2020

Element	OSRM	Google	Difference
Distance (km)	4.88 km	5.19 km	-0.31 km
Time (m' s")	12' 6"	12' 9"	-0' 15"
	Point offset from route (m)		
At origin	0.00	2.00	
At destination	0.00	4.47	
Sum	0.00	6.00	

Hausdorff Distance: 1042.41m

Figure 40: Route information display

In the case of comparing a route to its return, the two lines appear on the map in the same colour with the transit being a solid line and the return a dashed. The points where maximum divergence was found is displayed if desired.

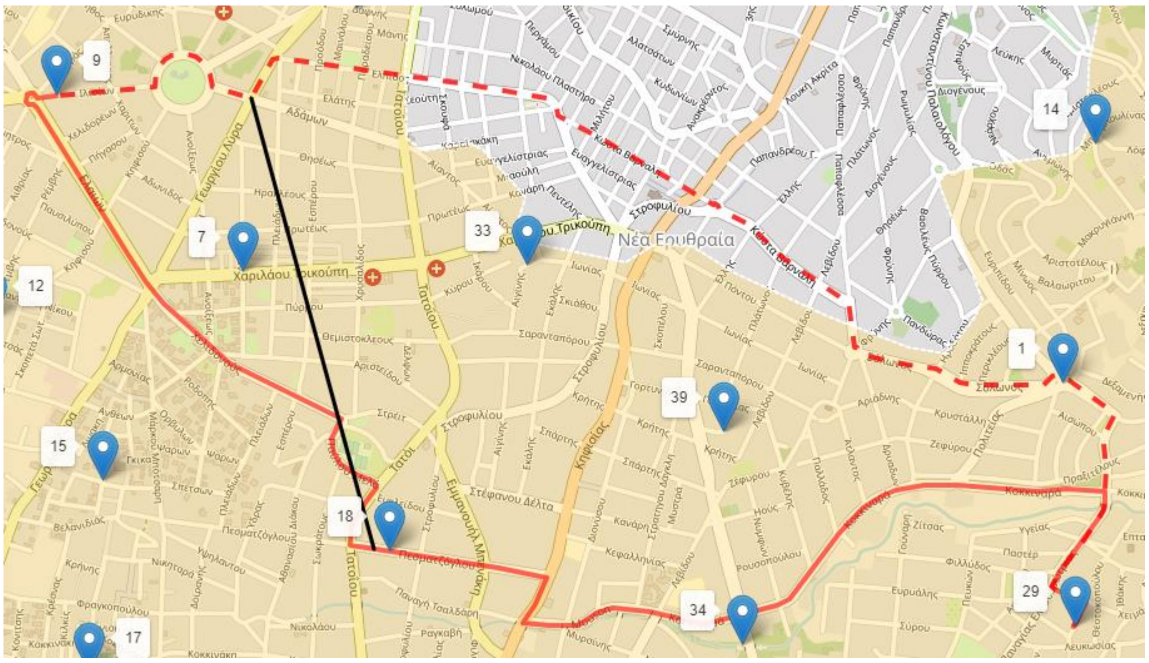
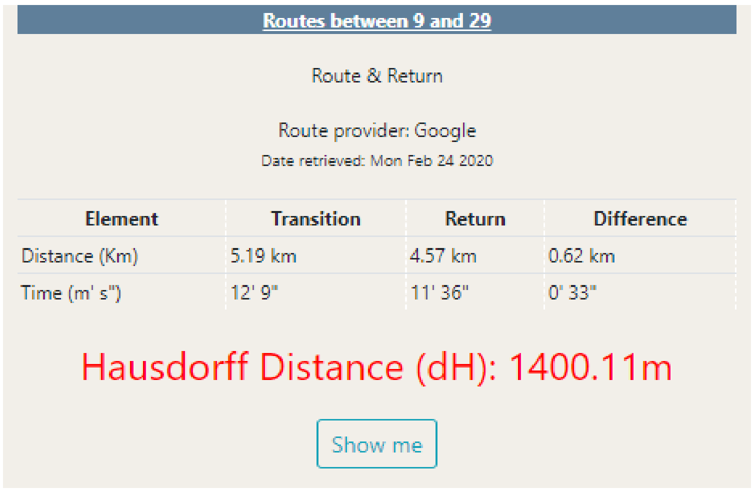


Figure 41: Route compared to its returning course as provided by Google (black line displays points of max divergence)

Subsequently, the data area displays a comparison between the two opposite directions.



Routes between 9 and 29

Route & Return

Route provider: Google

Date retrieved: Mon Feb 24 2020

Element	Transition	Return	Difference
Distance (Km)	5.19 km	4.57 km	0.62 km
Time (m' s")	12' 9"	11' 36"	0' 33"

Hausdorff Distance (dH): 1400.11m

Figure 42:Route information display for comparison of route to its returning course

In the example above we see that the return route is 33 seconds faster and 620 metres shorter.

Page intentionally blank

6. Conclusions and future suggestions

As stated in the introduction, the “approval rating” of a route is in most cases subjective. In this light, this paper does not aim to categorise the results from a provider as good or bad or decide which of the two has a better solution to the routing problem. The objective is to provide the tools for each examiner to reach their own conclusion. However, we can demonstrate cases of interest that were identified after sifting through the results.

As demonstrated by various routes in the area of Dionysos, one of the key issues is that the routes provided by the OSRM do take advantage of any non-asphalt roads. For example, the route between points 5 and 26 in Dionysos display an offset of the points to the origin and destination of the route at 590.27 and 410.5 equivalently. Google, on the other hand, accurately reaches the points since the offsets are at 1.00 and 5.10 metres.

The use or not of non-asphalt roads is a severe issue. If there is no alternative way to reach the destination, the omittance is a serious problem. On the other hand, the use of such roads as shortcuts to minimise travelling distance or time is met with high disapproval by many drivers.

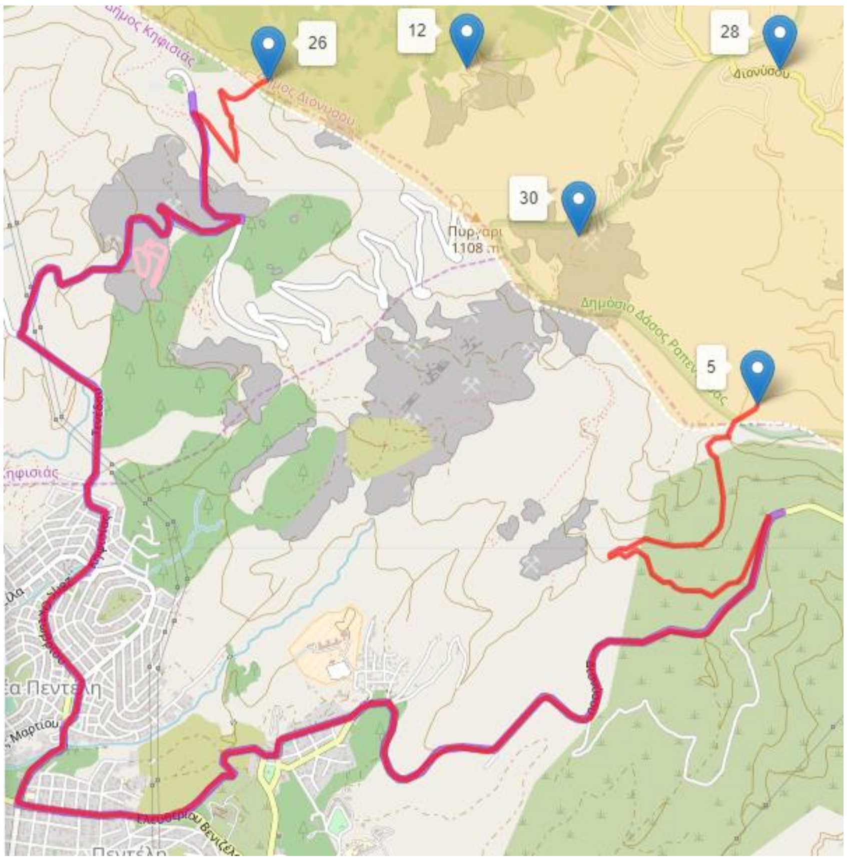
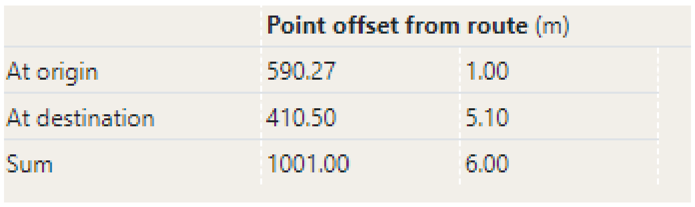


Figure 43: Route between points 5 and 26 in Dionysos show where one providers reaches points and other doesn't due to track road usage



	Point offset from route (m)	
At origin	590.27	1.00
At destination	410.50	5.10
Sum	1001.00	6.00

Figure 44: The data display panel shows distance between point of navigation sent as request and point actually directed to

The application is useful in identifying underlying differences in the data used in each case. Some examples of this are the following:

a) routes from point 8 to various others in Dionysos shows Google going through a forested area that OSRM does not have recorded. This is shown in the highlighted circle in the image below. By examining the area on Google street view we find that OpenStreetMap indeed lacks the part of the road at the point

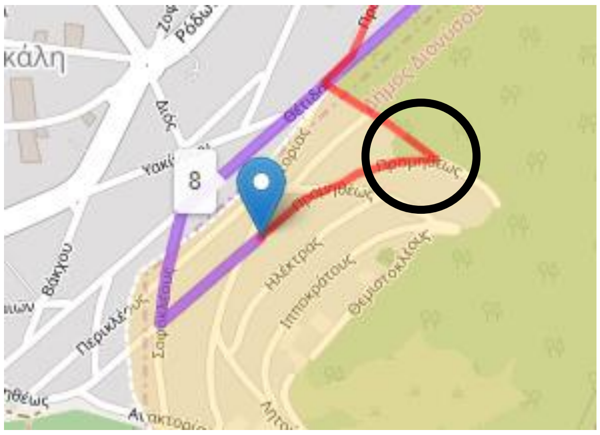


Figure 45: Google route appears to be travelling on non-existent road

One of the useful tools that could be developed in the future is a split-screen where the two providers are placed side by side and the route is shown on both. This will help by providing a satellite background which is something OSM does not have.

b) route from 1 to 9 in Nea Erythrea shows a divergence near point 7.

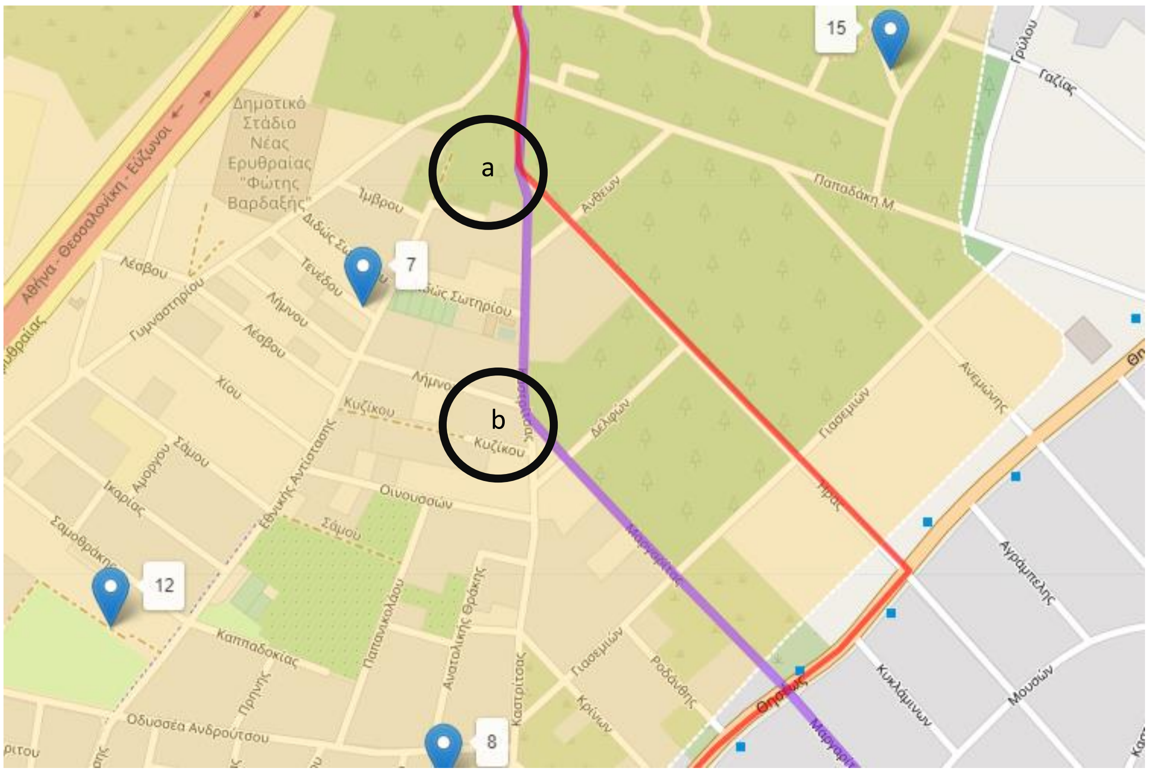


Figure 46: Travelling from north to south, the two providers choose to turn easterly at different locations a and b

We can see Google taking a turn at "a" while OSRM turns left at "b". Examination of the two on Google Street View verifies the differences and indicates that OSM has a road in its data where Google does not.

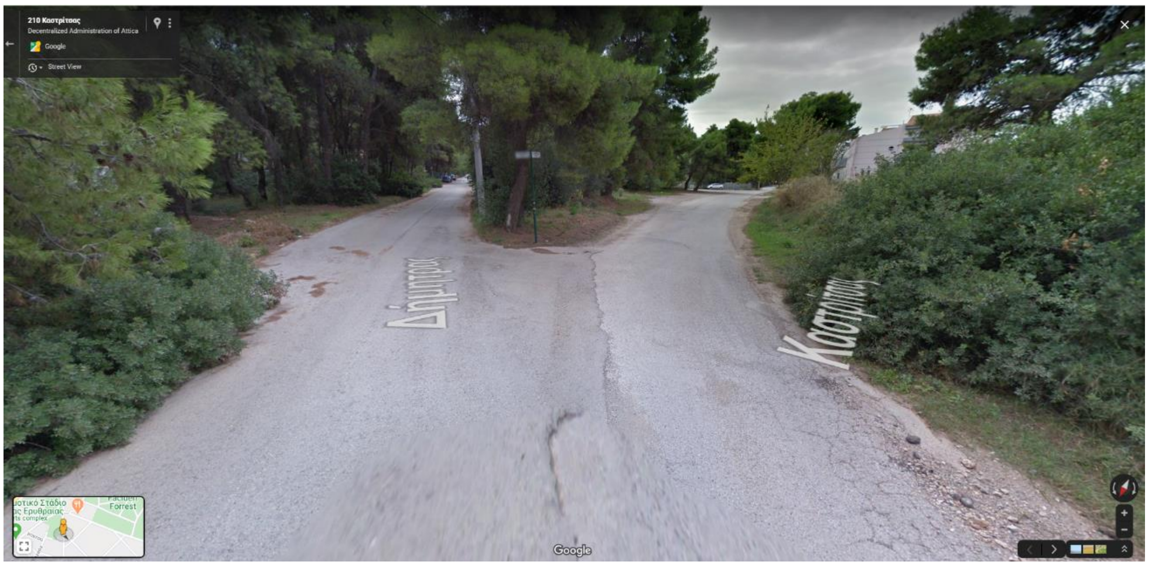


Figure 47: This is what the driver will see at point a

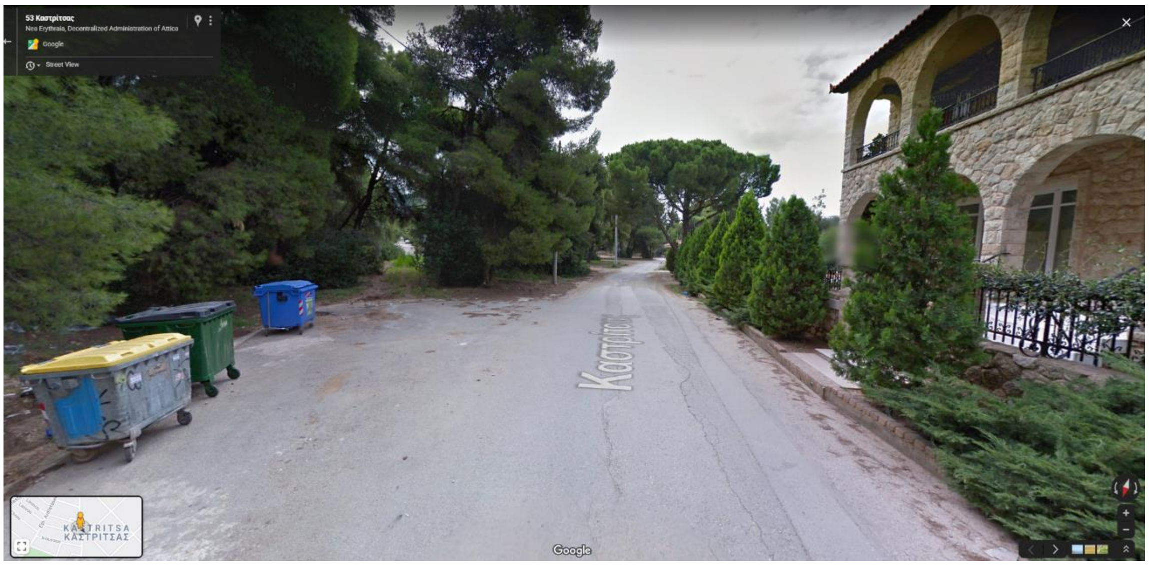


Figure 48: This is what the driver will see at point b

One might not necessarily consider this a mistake as the images in Street View are not recent and can be misleading, but recent satellite imagery does not support this.

This leads to a fundamental issue, that the evaluation of the routes should be performed by people who are familiar with the areas and use the streets. One of the suggestions proposed is to attach a voting system where locals can examine and vote for the best alternative.

This could help in situations where there seems to be a small difference in duration and distance, but the choice of roads is not. Such a situation can be found in the route from 1 to 11 in Kifisia.

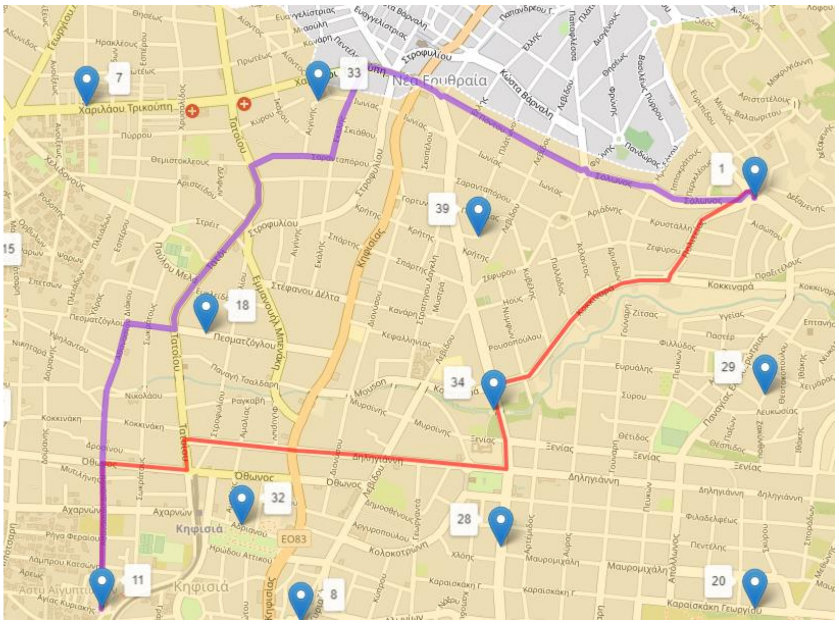
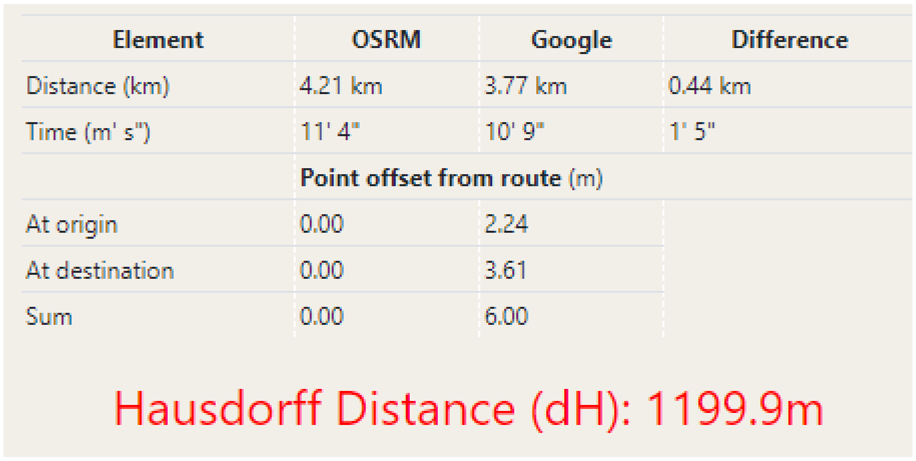


*Figure 49: Case where for no apparent reason the two providers choose almost completely different roads (*from point 1 to 11 in Kifisia)*

Google seems to suggest a faster by 1’ 5” and shorter by 440 m route compared to OSRM but whether it is, in general, a better suggestion cannot be evaluated by someone unfamiliar with the streets and area.

Table 17: Data for the two routes from point 1 to 11



Element	OSRM	Google	Difference
Distance (km)	4.21 km	3.77 km	0.44 km
Time (m' s")	11' 4"	10' 9"	1' 5"
	Point offset from route (m)		
At origin	0.00	2.24	
At destination	0.00	3.61	
Sum	0.00	6.00	

Hausdorff Distance (dH): 1199.9m

The reasons that one might prefer the route by OSRM over Google’s is that it has a fewer number of instructions. This was estimated manually but this is another point for future improvement since it could be extracted from the data programmatically.

The WebGIS App helps identify big differences by providing a colour map of all the Hausdorff distances (dH) for all routes in an area. This is useful in two ways:

a) it pinpoints the extremities and

b) provides statistical information (min, max, average, standard deviation) for the evaluation of the area as a whole

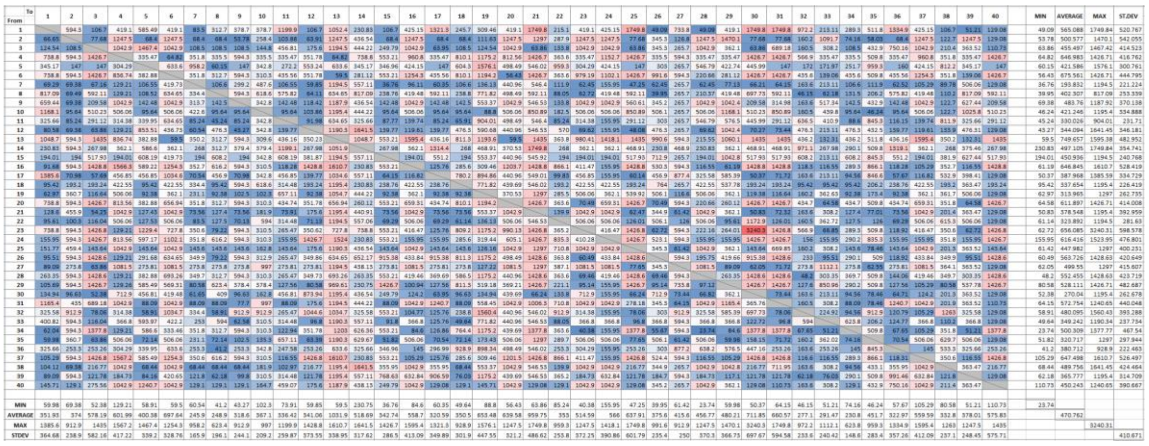


Figure 50: Picture of auxiliary colormap table of all Hausdorff distances in an area

The colourmap of Kifisia easily shows the biggest difference in the area, found in the route from 23 to 30. The dH found in this case was 3240.31

To get a sense of the area and overall results, a distribution chart is given with the above colourmap.

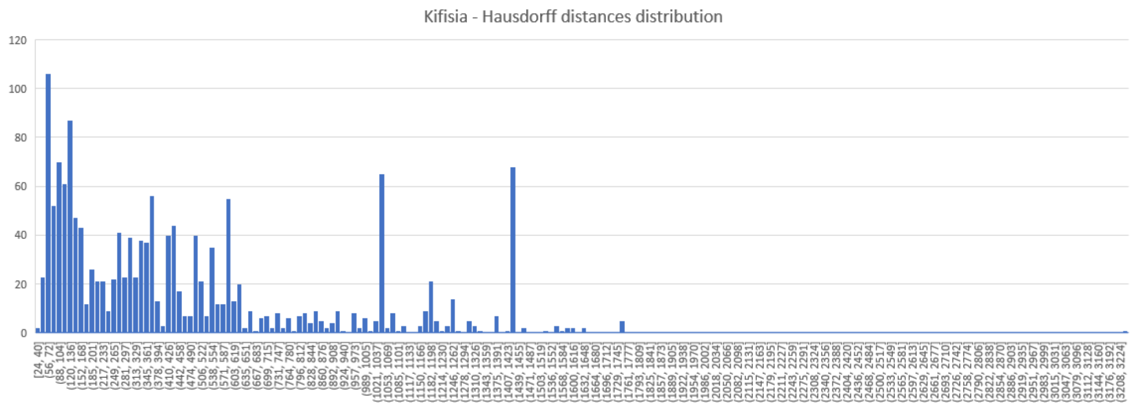


Figure 51: Picture of auxiliary chart of all Hausdorff distance distribution in an area

This illustrates all the dH in Kifisia categorised in200 bins and helps with the evaluation of the results in the colour map table.

The approach presented in this thesis and its implementation provides a wealth of situations to be explored. It merges and shares the information and allows users to place the two route providers under scrutineering and to identify the advantages and disadvantages each has to offer.

Bibliography

[1] T. Paoleti, “Leonard Euler's Solution to the Konigsberg Bridge Problem,” [Online]. Available: https://www.maa.org/press/periodicals/convergence/leonard-eulers-solution-to-the-konigsberg-bridge-problem. [Accessed 10 February 2020].

[2] "WikiProject Haiti," [Online]. Available: https://wiki.openstreetmap.org/wiki/WikiProject_Haiti. [Accessed 4 Μαΐου 2019].

[3] OpenStreetMap Wiki contributors, “Kort Game,” 26 April 2019. [Online]. Available: https://wiki.openstreetmap.org/w/index.php?title=Kort_Game&oldid=1846453. [Accessed 10 February 2020].

[4] Google, “Google Maps,” Google, [Online]. Available: https://www.google.com/maps/@48.9102256,4.2487779,4.8z. [Accessed 10 February 2020].

[5] OpenStreetMap Foundation, “OpenStreetMap,” [Online]. Available: OpenStreetMap. [Accessed 10 February 2020].

[6] apigee, “Web API Design: The Missing Link, Best Practices for Crafting Interfaces that Developers Love,” Google Cloud, 2018.

[7] D. P. Huttenlocher, G. A. Klanderman and W. J. Rucklidge, “Comparing images using the Hausdorff distance,” *IEEE Transactions on pattern analysis and machine intelligence,* vol. 15, no. 9, pp. 850--863, September 1993.

[8] OpenStreetMap Wiki contributors, “Using OpenStreetMap,” 16 January 2019. [Online]. Available: https://wiki.openstreetmap.org/w/index.php?title=Using_OpenStreetMap&oldid=1772189. [Accessed 9 February 2020].

[9] OpenStreetMap Wiki contributors, “Map Features,” 18 July 2019. [Online]. Available: https://wiki.openstreetmap.org/w/index.php?title=Map_Features&oldid=1879758. [Accessed 10 February 2020].

[10] StackOverflow, “Developer Survey Results 2019,” [Online]. Available: https://insights.stackoverflow.com/survey/2019#technology-_-programming-scripting-and-markup-languages. [Accessed 1 Feb 2020].

[11] M. Altaweel, “The Use of Python in GIS,” 13 June 2017. [Online]. Available: https://www.gislounge.com/use-python-gis/. [Accessed 10 February 2020].

[12] N. Grégoire and M. Bouillot, “Hausdorff distance between convex polygons,” Fall 1998. [Online]. Available: http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html. [Accessed 10 February 2020].

[13] Wikipedia, “k-means clustering,” [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering. [Accessed 10 February 2020].

[14] ESRI, “Mean Center,” [Online]. Available: https://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-statistics-toolbox/mean-center.htm. [Accessed 10 February 2020].

[15] P. Rogerson and I. Yamada, Statistical Detection and Surveillance of Geographic Clusters, Boca Raton: CRC Press, 2009.

[16] “Project OSRM,” [Online]. Available: http://project-osrm.org/. [Accessed 10 February 2020].

[17] “GitHub:Project-OSRM:osrm-backend,” [Online]. Available: https://github.com/Project-OSRM/osrm-backend.

[18] OSRM, “OSRM API Documentation,” [Online]. Available: http://project-osrm.org/docs/v5.22.0/api/#general-options. [Accessed 10 February 2020].

[19] Google, “Google Maps Platform | Products,” [Online]. Available: https://cloud.google.com/maps-platform/products. [Accessed 10 February 2020].

[20] Google, “Google Maps Platform | Routes & Directions,” [Online]. Available: https://cloud.google.com/maps-platform/routes. [Accessed 10 February 2020].

[21] Google, “Google Web Services | Directions API Documentation,” [Online]. Available: https://developers.google.com/maps/documentation/directions/start. [Accessed 10 February 2020].

[22] Google, “Google Maps Platform | Encoded Polyline Algorithm Format,” [Online]. Available: https://developers.google.com/maps/documentation/utilities/polylinealgorithm. [Accessed 10 February 2020].

[23] “Requests: HTTP for Humans,” [Online]. Available: https://requests.readthedocs.io/en/latest/. [Accessed 10 February 2020].

[24] M. Davlantes, “decode-google-maps-polyline,” [Online]. Available: https://github.com/geodav-tech/decode-google-maps-polyline/blob/master/polyline_decoder.py. [Accessed 10 February 2020].

[25] J. Whitaker, “pyproj Documentation,” pyproj Documentation, 2019. [Online]. Available: http://pyproj4.github.io/pyproj/stable/index.html. [Accessed 10 February 2020].

[26] The SciPy community, “SciPy Reference Page,” 19 December 2019. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/index.html. [Accessed 10 February 2020].

[27] The SciPy community, “Distance computations (scipy.spatial.distance),” 19 December 2019. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/spatial.distance.html. [Accessed 10 February 2020].

[28] The SciPy community, “scipy.spatial.distance.euclidean,” 19 December 2019. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.euclidean.html#scipy.spatial.distance.euclidean. [Accessed 10 February 2020].

[29] The SciPy community, “scipy.spatial.distance.directed_hausdorff,” 19 December 2019. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.directed_hausdorff.html#scipy.spatial.distance.directed_hausdorff. [Accessed 10 February 2020].

[30] V. Agafonkin, “Leaflet,” [Online]. Available: https://leafletjs.com/. [Accessed 10 February 2020].

[31] Bootstrap Team, “Bootstrap Library,” [Online]. Available: https://getbootstrap.com/. [Accessed 10 February 2020].

[32] Wikipedia, “Garden city movement,” [Online]. Available: https://en.wikipedia.org/wiki/Garden_city_movement. [Accessed 10 February 2020].

Appendix A

Table of data resources

Source	Hyperlink	Description
Hellenic Statistical Authority	https://www.statistics.gr/el/digital-cartographical-data	Municipality polygon data
Hellenic Statistical Authority	https://www.statistics.gr/documents/20181/1210503/FEK_Nomimos_rev.pdf/9c385e3d-cd1c-4a9f-8caa-036bb1febe69	Official 2011 Census release