POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

A Path Learning Approach for Building Accessibility Topologies

Polo Territoriale di Como
Master of Science in Computer Engineering

Advisor: Prof. Matteo Matteucci
Co-Advisor: Prof. Sara Comai
Co-Advisor: Ing. Emanuele De Bernardi

Master Thesis by:
Andrea Peccini, ID 823759

Academic Year 2015-2016

A Roberta.

Abstract

According to World Healt Organization statistics, about 15% of the world's population, which represents over a billion people, presents some form of physical disability or mobility impairment, which may be congenital or as a result of injury, where aging disease is the most common reason. This number is supposed to grow rapidly, as the EU population gets progressively older.

Our cities collect a multiplicity of obstacles and barriers because they are usually built for able-bodied users. This represents one of the main issue for those users who are obliged to move around in their everyday life with manual or motorized wheelchairs.

This thesis, as a part of the Maps for Easy Paths (MEP) polisocial project, aims at defining and developing a set of tools for the enrichment of cartographic maps with accessibility information regarding the pedestrian routes usually adopted by users affect by mobility impairments. This work takes care of building a system able to elaborate and supply information through an automatic procedure, in order to show a suitable path for each user based on its mobility conditions.

The research work described throughout this thesis aspires to the problem resolution adopting an unsupervised and neural network based algorithm, designed to cluster the collected routes, and to extract the accessibility of areas within a city. The described method takes into account the geotagged information supplied by users using two applications developed for mobile devices, i.e., smartphones and tablets, considering the collected routes and physical barriers. This correspond to a step toward the visualization of cartographic heat maps where the degree of accessibility can be identified by simply looking at different colored areas.

Sommario

In accordo con quanto riportano le statistiche dell'Organizzazione Mondiale della Sanità, circa il 15% della popolazione mondiale, approssimativamente circa un miliardo di persone, presenta una qualche forma di disabilità fisica o menomazione motoria, la quale può essere congenita o dovuta ad una ferita, dove l'invecchiamento è una delle cause più comuni. Queste cifre sono destinate ad un rapido aumento, poiché il numero di anziani nell'Unione Europea è in forte crescita.

Le nostre città a misura di utenti normodotati, collezionano una molteplicità di barriere fisiche e ostacoli. Questi rappresentano un serio impedimento per chi è costretto a muoversi quotidianamente con uno mezzo quale puó essere una carrozzina manuale o motorizzata.

Questo lavoro di tesi, in quanto parte del progetto polisocial Maps for Easy Paths (MEP), ha come obiettivo quello di definire e sviluppare un insieme di strumenti per l'arricchimento di mappe cartografiche contenenti informazioni sull'accessibilità dei percorsi pedonali urbani abitualmente utilizzati da utenti affetti da disabilità motorie. Questo lavoro si fa carico di costruire un sistema capace, tramite una procedura automatica, di elaborare e fornire dati, al fine di mostrare il percorso ottimale per ogni utente basato sulle sue condizioni di mobilità.

Il lavoro di ricerca, descritto in questo lavoro di tesi, mira alla risoluzione del problema adottando un algoritmo non supervisionato basato su reti neurali, capace di raggruppare i percorsi tracciati e di estrarre l'accessibilità delle aree all'interno di una città. Il metodo descritto prende in considerazione informazioni geo-localizzate, riportate dagli utenti stessi attraverso due applicazioni sviluppate per dispositivi mobili (smartphone e tablet), considerando i percorsi tracciati dagli utenti e le barriere architettoniche. Questo corrisponde ad un ulteriore passo verso la visualizzazione di mappe cartografiche colorate, in cui il grado di accessibilità possa essere individuato osservando semplicemente la colorazione delle diverse aree.

Acknowledgements

Per prima cosa vorrei esprimere la mia gratitudine nei confronti del Professor Matteo Matteucci e della Professoressa Sara Comai per la loro guida, i loro consigli e interesse durante lo svolgimento di questo lavoro.

Un sincero grazie lo dedico al Dot. Emanuele De Bernardi, per il suo costante aiuto e per il tempo che mi ha dedicato per risolvere ogni singolo problema incontrato durante tutto il periodo di svolgimento di questa tesi.

A tutti i Membri dell'Assistive Technology Group e in particolare al Professor Salice, per l'interesse dimostrato per il progetto. Un interesse non solo formale che ha portato tutti a partecipare attivamente all'acquisizione iniziale dei dati, senza i quali, difficilmente questa tesi avrebbe potuto raggiungere risultati così notevoli.

Vorrei ringraziare i miei genitori, Marilena e Sauro, per avermi dato la possibilità di intraprendere questo fantastico percorso che mi ha permesso di crescere come persona e come professionista in un modo che non credevo nemmeno possibile. Ai miei nonni Maria, Pietro e Francesca per i forti valori che mi hanno trasmesso.

Con immenso affetto vorrei ringraziare Roberta con cui ho condiviso questo lungo viaggio, per aver sempre creduto ciecamente in me senza mai vacillare e senza la quale difficilmente sarei riuscito a raggiungere tale traguaro.

Devo anche un meritato ringraziamento alla mia famiglia comasca Iacopo, Luca, Emilio, Nino, Sara, Clara, Stella, Francesco, Carmelo, Rosario e tutti i compagni di "Sound", i quali hanno allietato e reso indimenticabile la mia permanenza comasca in tutte quelle giornate che non sembravano finire mai tra progetti ed esami.

Come non ringraziare inoltre la mia personale equipe medica e in particolare Carola e Zain, per per avermi supportato e forse troppo spesso "sopportato", soprattutto durante l'ultimo periodo di questo lavoro.

Al mio gruppo di fisici, soprattutto Andrea ed Erica, con cui ho condiviso pause pranzo tra torte e partite a carte, voglio dire grazie.

Agli amici di sempre Erika e Riccardo per esserci sempre stati e per i momenti condivisi insieme.

A Walter, Monica, Valentina, Simone ma soprattutto Rosy, per la sua fantastica pizza, una costante ormai irrinunciabile della domenica sera che mi dà la carica per affrontare tutta la settimana.

Contents

Abstract **i**

Sommario **iii**

Acknowledgements **v**

1 Introduction **1**

1.1 Thesis Contribution within the MEP-Project 2

1.2 Structure of the Thesis 5

2 State of the Art **7**

2.1 About statistical learning 7

2.1.1 Reducible and Irreducible Errors 10

2.2 Estimation of f 10

2.2.1 Supervised vs. Unsupervised Learning 12

2.2.2 Regression vs. Classification Problems 13

2.3 Model Accuracy 13

2.3.1 The Bias-Variance Trade-Off 16

2.3.2 The Classification Setting 18

2.4 Clustering 20

2.4.1 Spectral Clustering 22

2.4.2 Pairwise & Path-Based Clustering 23

2.4.3 In between Spectral Clustering and Path-Based Clustering 30

2.5 The Neural Gas Network Approach 33

2.5.1 Growing Neural Gas Network 35

2.5.2 Adaptive Incremental Growing Neural Gas Network . 39

3 Methods and Procedures **43**

3.1 The MEP Data Collections 43

3.1.1 Implicit Data Collection 46

3.1.2 Explicit Data Collection 47
3.1.3 Issue and Requirements 48
3.1.4 Android Sensor Framework 49
3.2 The MEP-Fusion Engine 50
3.2.1 The ROAMFREE Library 52
3.3 Geodetic System 54
3.3.1 Geodetic Coordinates 55
3.3.2 Conversion 57
3.3.3 Cartographic Correction of Global Navigation Satellite System Trajectories From Low Cost Devices . . . 58
3.4 PostgreSQL and PostGIS 60
3.5 Python Programming Language & Libraries 62
3.6 The MEP Project Workflow 64
3.6.1 Decompression Phase 67
3.6.2 Building and Highway Tables Population Phase 68
3.6.3 MEP-Fusion Correction Phase 70
3.6.4 GNSS Correction Phase 73

4 The MEP-Clusterpath Algorithm 75
4.1 Algorithm Preparation and Definitions 75
4.1.1 Nodes, Arc and Graph 77
4.2 The MEP-Clusterpath Algorithm 78
4.2.1 Initialization 78
4.2.2 The MEP-Clusterpath Input Set 79
4.2.3 The MEP-Clusterpath Training Phase 81
4.2.4 The Merging Procedure 85
4.3 Neurons' Heat Maps Representation 88

5 Experimental Results and Validation 93
5.1 Algorithms' Performances 98
5.2 The MEP Heat Maps Representations 101
5.3 Results Validation 106
5.4 The MEP APIs 108

6 Conclusions 111
6.1 Future Works 112

Bibliography 115

List of Figures

1.1 The MEP project scenario. 3

1.2 Heat maps results examples. 4

2.1 On the left a generic initial data set. On the right the same generic data set with the curve which generates the data. . . 8

2.2 Generic multi-variable fitting example. 9

2.3 On the left panel the data simulated from f are shown in black. Three estimates of f are presented: the linear regression line(orange), and two smoothing spline fits(blue and green). On the right the training MSE(gray curve), test MSE(red curve), and minimum test MSE over all methods(dashed line) can be observed. Squares represent the training and test MSEs for the three fits shown in the left hand panel. 14

2.4 Graphical illustration of bias and variance 17

2.5 Squared bias(the blue curve), variance(the orange curve), $Var(\epsilon)$ (the dashed line) and test MSE(the red curve) for different underlying function f, respectively an general one, an highest linear function and finally an highly non linear function. . . . 18

2.6 Simulated dataset consisting of 100 observations, distributed into two groups. The dashed line represents the Bayes decision boundary. The orange background grid represents the region in which a test observation will be assigned to the orange class, and the same for the blue background grid area respectively. 19

2.7 Generic examples of spectral clustering and path-based clustering. 21

2.8 Generic examples of pairwise clustering on a generic data set. 24

2.9 Two subsamples from the same input data set. Agglomerative algorithms sometimes fail to extract the correct underlying structure of the clusters, due to noise link between objects. . 26

2.10 Distance matrices of three-circles dataset. 29

2.11 Final clustering results using K-means after the application of multi-dimensional scaling 30

2.12 Clustering results for a noisy 2-circle data set. 32

2.13 Representation of two ways of defining closeness among a set of points. 35

2.14 A "Growing Neural Gas" example. 38

2.15 "Growing Neural Gas" vs. "Neural Gas" 39

2.16 The three possible different cases in which the AING operates. 40

3.1 MEP-Traces Activities. From left to right: the MEP-Traces Splash screen, displayed at the application launch; the Welcome screen of the application after the login; the Main menu with the application's features buttons. 44

3.2 MEP-APP activities. From left to right: the initial screen with the MEP logo; the welcome screen; the map which can be consulted by the user. 45

3.3 Implicit Data Collection general schema. 46

3.4 General processing schema applied to data coming from MEP-Traces in order to perform the trajectories reconstruction. . . 51

3.5 Instance of hyper-graph for pose tracking and self-calibration with four pose nodes. 53

3.6 Ellipsoid approximation of the Earth 55

3.7 ENU reference system 56

3.8 "Bring-outside" correction produced by the GNSS algorithm from the DICA. 59

3.9 On the left a real case result after the application of the GNSS correction. On the right there is an example in which the GNSS correction results in an error. 59

3.10 The MVC design pattern general block schema. 65

3.11 The Main Controller subdivision into the different elements cooperating into the scientific computation. 66

3.12 Worker general process tasks. 67

3.13 The *.zip* file check procedure. 68

3.14 Population of the Building and Highway tables required by the GNSS correction to operate. The MEP_NMEA_INFO file contains the points' sequence corresponding to the recorded path. This can be used to create a series of overlapping windows. Each window corresponds to a query to the OSM server to retrieve building and highway falling within its area. The server's response contains all the demanded information, which can then be loaded locally into the related MEP-Server schema. Subsequently to the schema, also the related tables have to be updated. 69

3.15 The MEP-Fusion pre-processing schema. This schema tackle many different problems, focusing on the purpose to get heterogeneous data as input to the MEP-Clusterpath algorithm. The results of this step get loaded into the MEP-Database in the *pose_pt* table and activate the GNSS correction trigger. . 72

3.16 GNSS correction workflow adopted in order to keep updated the tables *public.building_area*, *public.building_perimeter*, *public.building_segment*, *public.highway_segment*. These tables contains all the data required for the application of the GNSS "*outside-building*" projection. We decide here to represent also the population of the four tables underlying the GNSS correction although the correction take place when the new points from the Fusion correction come into the MEP-Database. 74

4.1 Three different real data collection referred to three different cities, i.e., Cernobbio, Siena and Milano respectively. 80

4.2 Real examples of raw and cleaned data representation of Cernobbio coming from the MEP-Database. 81

4.3 MEP-Clusterpath threshold definition reflects the one of the base AING algorithm. 82

4.4 Green square points correspond to the three possible cases based on the distance of the point x with respect to first and second nearest nodes. The black squares are the samples already assigned to a neuron node (blue circle points). 84

4.5 Two-dimensional representation of a generic Gaussian function. 89

5.1 Real examples of raw and cleaned data representation of Cernobbio and Orta San Giulio coming from the MEP-Database. 94

5.2 The urban canyon problem around cities. In panel 5.2(a) Cernobbio recorded data; in panel 5.2(b) an information' slice from the city of Siena. 94
5.3 Both slices represent two cases in which the AING algorithm produced sub-optimal results. Into the left panel, blue points show the displacement of the neurons Cernobbio. In particular, the red circle points out the area in which the AING algorithm produced sub-optimal results. Into the right panel orange squares indicate the neurons' position over Novara. Once again the algorithm fails to fuse information while creating the accessibility topology of the city. 96
5.4 Neurons' displacement in Cernobbio. 97
5.5 Detailed zoom of the AING's weakness. Resulting nodes that characterize a graph, may sometimes distribute over tight areas, leading to an undesired situation of overlapping information. 99
5.6 Each panel presents the final topology arising with a different set of parameter. The "id" links the panel to the corresponding parameter set in Table 5.2. In such a way its easy to compare the results obtained from different tunings. 102
5.7 Two and three dimensional heat maps representations of the accessibility obtained by the application of the AING algorithm on the data contained into the MEP-Database related to Cernobbio. Both panels represent the same input topology from a different point of view. In particular, the accessibility scale valid for both figures appears in Figure 5.7(b). Starting from 0 (unrecorded areas) the scale describes values up to 4. . 104
5.8 Heat maps referred to Cernobbio. In both figures the green blobs refer to the accessible areas while the red peaks correspond to the places where an obstacle or an inaccessible barrier has been encountered by a user. The lowest purple level instead marks the areas with no reference path recorded by users. 105
5.9 Resulting heat map obtained over the city of Cernobbio. Green shaded areas locate the accessible areas within the city's surface while the red blobs precisely locate the inaccessibilities descending by the presence of an obstacle. 107

List of Tables

3.1 Complete sensor overview of the Android platform. 50
3.2 Geodetic defining parameters. 56
3.3 Geodetic derived geometric constants. 56
3.4 Database Features 61

5.1 Results obtained by testing all the possible settings configuration. Each setting can be uniquely identified through a corresponding identification (id) number. The column time refers to the time laps spent by the algorithm to produce the results. The d parameter indicates the mean distance from all the existing neurons with respect to the center of-mass of the observed data-points. The d_{min} parameter collects the different values for the minimum granularity tested with our algorithm. The max_{age} parameter fixes an upper bound seniority for the nodes links; by setting it to $+\infty$, the algorithm will never eliminate outdated links. Nodes' number column collects the final number of neurons produced by the algorithm. 98
5.2 Collection of results obtained by varying the parameters set of the MEP-Clusterpath algorithm. We refer to each test through an id number which uniquely indicates the parameters set that we employ for the test itself. All the described features reflect the ones it Table 5.1 and they can be used to operate comparisons between different settings for each algorithm, or evaluations between tunings of the two different algorithms. 100

Chapter 1

Introduction

"Tutti abbiamo dei limiti, ma non tutti siamo limitati."

Angela Gambirasio

Urban mobility is recently one of the most discussed thematic in the international debate, which mainly focuses its attention on environmental sustainability of movements and social sustainability as key aspects in the improvement an reorganization of urban life and of the cities themselves. Public areas' accessibility represents a benefit for all the inhabitants considering that the movement around the urban space must be easy, safe and pleasant for all the citizens. In particular, some extra care should be taken considering the difficulties of people with mobility impairments, considering dedicated solutions to solve the problem of this specific class of users. According to World Healt Organization statistics, about 15% of the world's population, which represents over a billion people, presents some form of physical disability or mobility impairment, which may be congenital or as a result of injury, where aging disease is the most common reason.

Traveling through cities for such users, is one of the concerns which people with mobility impairments deal with. In particular physical barriers cause uneven access to public services or public areas, reducing the impaired people's everyday-life quality of mobility.
In daily life wheelchair users face off with a huge amount of obstacles along their routes represented by poles, steps, poor surfaces or, lack of ramp on the footpaths, inappropriate urban design and inaccessible transportation services. These are the basic issues that they are dealing with, and may sometimes represent a serious problem, forcing them to dangerous situations or leading them to consider longer alternative routes to reach their favorite/preferred destination.

As a consequence of the presence of this physical barrier dislocated around cities there is the frustrating situation in which an entire class of users could not freely participate in what is considered “normal” in everyday society on their own. People who are not affected by any kind of mobility impairment are usually unaware of the impact of this kind of obstacle, which characterizes the daily moving of the impaired users instead.

Regardless of the need for further enhancements in city routes accessibility conditions and facilities, the huge worldwide smartphone and tablet spread affecting nowadays world offers the possibility to turn mobile devices into an enormous sensing platform, enabling large-scale analysis of the human behavior in all the aspects of life. Mobile device technology really introduces the possibility to improve the people’s experience of the modern cities daily life by offering services which can be specialized on the different user’s needs.

Many projects developed in different mobile applications, e.g., ComuniPerTutti, Mapability and Rota Accessible offer the possibility to display the Point of Interests (POIs) like restaurants, museums, etc. They present locations with a good accessibility level around a specific city, but they lack the accessibility as information of the path that the user has to take to reach the desired destination, resulting in an incomplete information.

Popular modern navigation systems on the market, such as TomTom, Garmin or Google Maps Navigation do not consider information about the accessibility of the routes for people affected by mobility impairments. This lack of information about the accessibility level of the routes represents the starting point for a project called MEP(Map for Easy Paths) which will be described into the following paragraph.

1.1 Thesis Contribution within the MEP-Project

Despite the recent attitude toward making modern urban cities accessible, many barriers and obstacles still represent a serious issue for impaired people’s mobility in every day life. The effort of keeping updated maps of both accessible paths and obstacles, in order to support people with limited mobility in their daily movement, would represents a too high expense for the majority of local administrations. For this reason the notification of such elements is usually left to volunteers and no-profit associations. The community objective should be to reach noticeable results by involving directly the interested users in the mapping process keeping its costs low while offering a constantly updated map of the location of the barriers, obstacles,

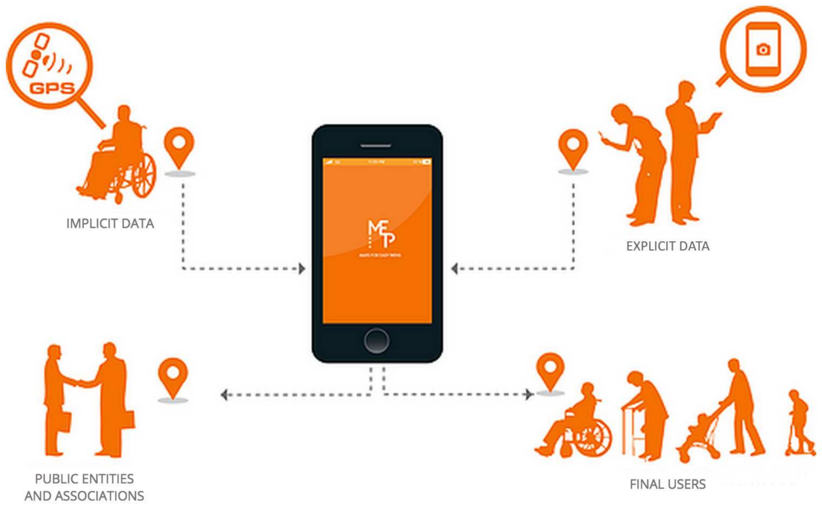


Figure 1.1: The MEP project scenario.

and accessible points within the city area, renewing the attention and the effort toward this extremely sensitive and social argument.

The challenge of how to improve the life's quality of people affected by mobility impairments using common mobile-device, has been caught by the Maps for Easy Paths polisocial project, a program of social effort and responsibility of Politecnico di Milano whose main target is the development of a set of automated tools and innovative solutions to enrich the geographic map with information about the accessibility of the urban pedestrian areas.

The MEP project represents an attempt to build accessibility maps (i.e., maps with accessible routes and city areas) fusing, in an automatic fashion, data implicitly collected through mobile devices (e.g., smartphone and dedicated electronic systems) which are able to automatically detect the location (through the Global Positioning System, i.e., the GPS) and when possible the characteristics of the ground condition (e.g., the presence of steps, holes, etc.). All the interested citizens can participate and give their own contribution to the project in order produce a more fair society.

The MEP scenario is well described by the representation in Figure 1.1. Two type of data were discovered to be fundamental for the project's purpose i.e., "implicit" and "explicit" data. "Implicit data" refers to automatically collected data, specifically to those caught without the need for the user to be aware off. "Explicit data" refers to those information which the users explicitly decide to make available to the MEP community, recording problems

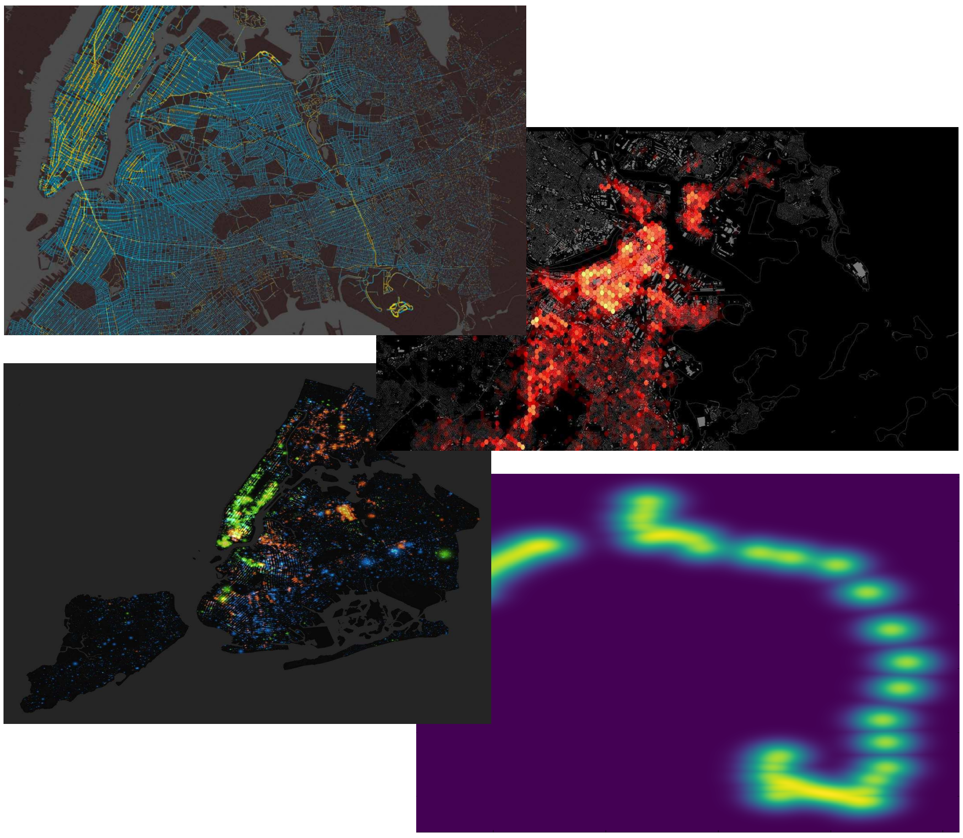


Figure 1.2: Heat maps results examples.

encountered on his/her path.

The MEP project aims to promote a solution of the problems reported by the users through the direct involvement of public entities and associations, which may be interested in the process by sponsoring the removal of physical barriers.

The knowledge about this problems produces a better understanding of the obstacles distribution which impaired users encounter in every day life focusing the attention on people with impairments in order to produce a more fair society.

Information's relevance goes together with human readability. A cartographic representation of areas through accessibility heat maps offers the possibility to easy locate places characterized by high level of accessibility. Nonetheless visual inspection of a cartographic heat map leaves also the possibility for further definition of accessibility indexes, which may arise as considerations about the colored areas' distribution around cities. Some examples of this powerful data visualization tools appear in Figures 1.2, and well describe the thesis final contribution within the MEP project.

1.2 Structure of the Thesis

The thesis is structured in the following way:

- Chapter two offers a more specific definition of the MEP project data collection and a particular method for the processing of the data is theoretically defined.
- Chapter three and four presents the development of a new processing method for the solution of the problem.
- Chapter five presents the application of the developed method upon the MEP data collection, the results which have been obtained and the problem encountered during the development of the solution.
- Finally Chapter six presents the results and the considerations about the processing of the solution, and possible future development with the enhancement of this method.

Chapter 2

State of the Art

“Un piccolo passo per l’uomo, un miracolo per la disabilità.”

Angela Gambirasio

The ease which characterizes humans’ behavior in recognizing everything surrounds them all, belies the extremely complex processes that underlie the acts of pattern recognition. Pattern recognition, i.e., the act of taking in raw data and taking an action based on the “category” of the pattern, has been crucial for humans’ survival from the moment we appear on the earth [8]. Sophisticated neural and cognitive systems has been extensively studied within the fields of neurosciences, neurophysiology and neuropsychology among others.

The successful application of cognitive reasoning to computer science field, lead to the build of machines able to recognize patterns in an autonomous and flexible fashion.

In particular, statistical learning refers to a set of tools for modeling and understanding complex datasets, usually encountered in the computer science field. It arises as a recently developed area in statistics blends with parallel developments in computer science and, in particular, Machine Learning.

Recent explosion of “Big Data” problems lead the Machine Learning field to be one of the hottest in many scientific areas as well as marketing, finance and other business disciplines [18].

2.1 About statistical learning

The main goal of statistical learning is the development of an accurate model which can be used to predict the outputs, when the inputs are given. The

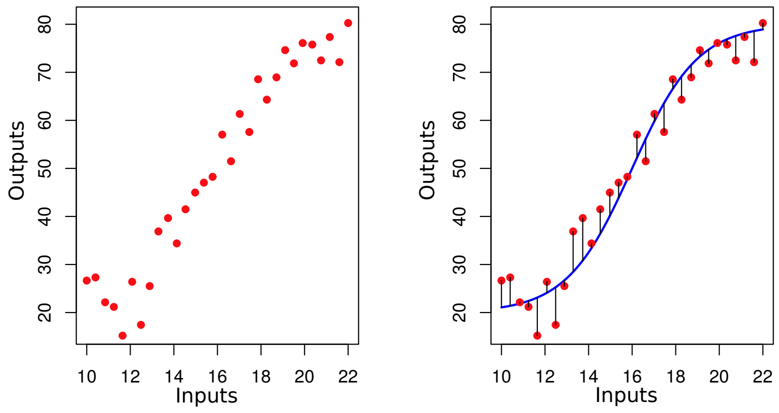


Figure 2.1: On the left a generic initial data set. On the right the same generic data set with the curve which generates the data.

symbols X and Y are respectively used to denote the inputs and the outputs. Suppose to observe a quantitative response Y and p different predictors, respectively $X_1, X_2, \cdots, X_p$, we can assume the existence of a relationship between Y and the p predictors $X = (X_1, X_2, \cdots, X_p)$ and in particular we could write its general form as:

$$Y = f(X) + \epsilon \tag{2.1}$$

where f represents some fixed, even still unknown, function of $X_1, X_2, \cdots, X_p$ and the parameter ϵ expresses a random error term, assumed independent from X, with 0 mean. The function f expresses the systematic information that X provides about Y, providing a connection between the input variable to the output one. Figure 2.1 gives a representation of the previous analytic expression. Left hand panel displays a generic relation connecting inputs and outputs variables. The true f which generates the data shows in the right hand panel of Figure 2.1, in particular deviation between the real data and the true f(the black vertical lines) allows to figure out what the parameter ϵ stands for.

Machine Learning refers in particular to the set of approaches adopted for the estimation of the unknown function f. The two main reasons demanding the function f are: *prediction* and *inference.*

- *Prediction* refers to those situations in which a set of inputs X is available, but the output Y cannot be easily obtained. In those cases, thanks to the error term averaging to zero, the output estimate $\hat{Y}$ can be obtained as:

$$\hat{Y} = \hat{f}(X) \tag{2.2}$$

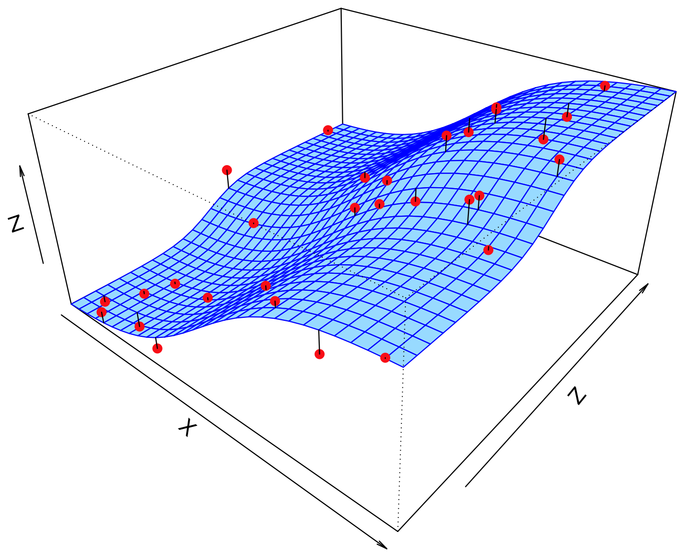


Figure 2.2: Generic multi-variable fitting example.

where $\hat{f}$ describes the unknown function f's estimate. In particular the accuracy of $\hat{Y}$, in predicting Y, depends on two relevant quantities denoted as *reducible* an *irreducible* error(discussed in details in Section 2.1.1).

- *Inference* refers to situations in which the estimate of the function f represents the relevant information itself. In such cases the target knowledge is embedded into the relation linking the predictors $X_1, X_2, \cdots, X_p$ change to the output Y.

Several reasons may guide through the choice between *inference* and *prediction*. Often only a small fraction of the predictors are effectively related with Y. For this reason the subset's identification of relevant predictors may produce an improvement. Moreover, some predictors may have a positive relationship with the Y, i.e., an increase of the associated predictor results into an increase of the output value Y, but other predictors may depict the opposite relationship.

Even though most of the relationships are more complicated with respect to linear one, this basic and simple relation has been widely considered in order to describe the input-output relationship. Clarity and interpretability regarding *inference* comes out adopting linear models to explain the input-output relationship at the cost of poor *prediction* results. Elementary models in fact, neglect the incredible real-world complexity embedded into the data.

At the same time, even though highly complex models provide accurate results regarding *prediction*, they come at the expense of lower interpretability, which negatively affects the *inference* process.

The estimation of the function f might also involve multiple variables assuming geometrical form closer to the thin plate represented in Figure 2.2 with respect to the line previously represented in Figure 2.1.

2.1.1 Reducible and Irreducible Errors

Considering the expression of f outlined in Function 2.2, the amount of error which can be encountered when fitting a data set, may be of two types, i.e., *reducible* or *irreducible.*

The inaccuracy introduced by estimating f through $\hat{f}$ results in an error afflicting the estimate. This error can be addressed as "reducible" in the sense that it can be potentially removed by adopting the proper learning technique to estimate the unknown function f.

Even if it were possible to obtain the perfect estimate of f, so that $\hat{Y} = f(X)$, the prediction would still produce an approximation of the true f. This uncertainty, denoted with ϵ, represents the variability which affects the accuracy of the prediction. Unlike the previous error, this one cannot be reduced and so denoted as "irreducible". No matter how accurate the estimate $\hat{f}$ of the model is, it will always contain an error which cannot be avoided. The quantity ϵ expresses a mixture of unmeasured variables or unmeasurable variations.

The analytical explanation takes into account an estimate $\hat{f}$, a set of predictors X, where the input-output relation takes the form $\hat{Y} = \hat{f}\,(X)$. Fixing both $\hat{f}$ and X, then

$$E(Y - \hat{Y})^2 = [f(X) - \hat{f}(X)]^2 + Var\,(\epsilon)\,; \tag{2.3}$$

$E(Y - \hat{Y})^2$ represents the expected value of the squared difference between the predicted and actual value of Y, and $Var\,(\epsilon)$ represents the variance associated with the error term ϵ.

In particular, terms encountered in Equation 2.3 outline two fundamental quantities: *bias* and *variance* respectively.

2.2 Estimation of f

Estimation of the function f requires a collection of n different observed points, denoted through the term *observations.* Observations compose the training data set and they will be used to train or "teach" the method, with the objective of estimate f.

Let x_{ij} represents the value of the j-th predictor for observation i, where $i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, p$. Let also y_i represents the response variable for the i-th observation. Then the training data set assumes the form $\{(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)\}$ where $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$.

A statistical learning method responsibility is the estimation of the unknown function f using the set of observation, defined then as "training data set". The estimate, denoted through $\hat{f}$ is such that $Y \approx \hat{f}(X)$ for any observation (X, Y).

Parametric vs. Non-Parametric

Statistical learning methods can be mainly divided into two classes, i.e., *parametric* and *non-parametric*.

Parametric methods involve a two-step model based approach. In particular they leverage on an assumption about the model underlining f, reducing the problem of estimating the unknown function f down to the one of estimating a set of parameters.

The first step outlines an assumption about the functional shape of the unknown function f. Assumptions on the linearity of f with respect to X have the advantage to be simple and easy to understand. Linear model's assumption can be represented as:

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p. \tag{2.4}$$

Instead of estimating an entirely and arbitrary p-dimensional function $f(X)$, one only needs to estimate the $p + 1$ coefficients $\beta_0, \beta_1, \cdots, \beta_p$.

The second step consists of training the model using the observations. As result comes out an estimate of the parameters $\beta_0, \beta_1, \cdots, \beta_p$, such that:

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \tag{2.5}$$

Among others, the most common approach adopted to fit the model is the *(ordinary) least squares* [4] one.

Model-based approaches can be referred as parametric, in the sense that they reduce the problem of estimating f down to the one of estimating a set of parameters. Of course the estimation of a set of parameters only is simpler with respect to the estimation of an arbitrary function f.

Disadvantage of parametric approaches relies on the fact that usually the elected model will not match the true unknown form of f. Expressions too far from the true one will result in a poor estimation and the adoption of more flexible models solves only partially the problem, because it requires the estimation of an higher number of parameters. Moreover, too complex models can lead to the phenomenon of *"overfitting"*, i.e., too close explanation of the intrinsic error lead to a perfect estimation on the training data, which is unlikely to perform well on novel patterns [18], discussed in Section 2.3.

Non-parametric methods, also referred as "sample-based" or "instance based" methods, do not make explicit assumptions about the functional form of f, and they exploit the training data "directly". They seek an estimate of f that get as close as possible to the data points, without being too rough wiggly. Non-parametric approaches have the potential advantage to accurately estimate a wider range of possible shapes for f, since they do not rely on any assumption on the form it can assume.

The problem with non-parametric approaches regards to the number of observations considered before obtaining an accurate estimate of the function f: it can potentially grow till prohibitive values, thus producing computational issues.

2.2.1 Supervised vs. Unsupervised Learning

Most statistical learning problems fall into one of two of the following categories: *"supervised"* or *"unsupervised"* [18].

The main purpose of supervised learning consists in fitting a model relating the response to the predictors. A teacher drives this method by providing a category label or cost for each pattern within the training, trying to minimize the sum of the costs for the patterns. Many statistical learning methods such as linear regression, logistic regression and support vector machines, operate under the supervised learning domain. Supervised learning problems can be further divided into *Regression* and *Classification* problems, which will be further described in Section 2.2.2.

As opposite to supervised, unsupervised learning describes the situation in which for every observation vector of measurements there is no associated response. For this reason would be impossible fitting a linear regression model, due to the lack of response variable to predict. In unsupervised learning there is no more an explicit teacher and the system forms clusters or "natural grouping" of the input patterns. "Natural" is defined explicitly or implicitly in the clustering system itself, and each set of patterns or cost function definitions lead to different clustering results. One fundamental statistical learning tool which goes under the unsupervised learning domain is the cluster analysis, or clustering, whose main goal is the ascertain of whether the observations fall into relatively distinct groups. Cluster analysis will be described in detail in Section 2.4.

2.2.2 Regression vs. Classification Problems

Variables denoting inputs and outputs can be characterized as either *"quantitative"* or *"qualitative"*(also denoted as *"categorical"*) [18].

Quantitative variables take on numerical values. Example of those variables are height, weight, response time, temperature, etc. Qualitative variables instead, take on values in one of K different classes, or categories, and they can be distinguished from quantitative ones because no ordering or measuring holds between them, i.e., color, religion, city of birth, etc.

Problems with a quantitative response are typically denoted as "Regression" problems, while those involving a qualitative response as "Classification" problems. The choice of the statistical learning method must take care of whether the response is quantitative or qualitative.

2.3 Model Accuracy

The selection of the best approach on a given data set is one of the most challenging task when performing statistical learning in the real world. This is due to the fact that no statistical learning approach dominates all other methods over all possible data sets.

In the regression setting, evaluation of the performance of a statistical learning method on a given data set, expresses the extent to which the predicted response value for a given observation is close to the true one. The performances of a statistical learning approach can be evaluated through the *"mean squared error(MSE)"*

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.6}$$

where $\hat{y}_i$ is the prediction obtained through $\hat{f}(x_i)$ for the i-th observation. Training is designed to make the MSE small on the training data. Looking at Equation 2.6, MSE will displays small values if the predicted response is close to the true one, while it will be large if the true responses differ substantially from the predicted one, for some of the observations.

Training MSE does not represent the absolute quality information of the model. What really does instead, is the "test MSE", obtained applying the learning approach on novel data. Moreover, results obtained on the training set in fact, are already available and do not represent whichever interesting information.

Given a set of training observations $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, suppose to fit a statistical learning method on them: the computation of the

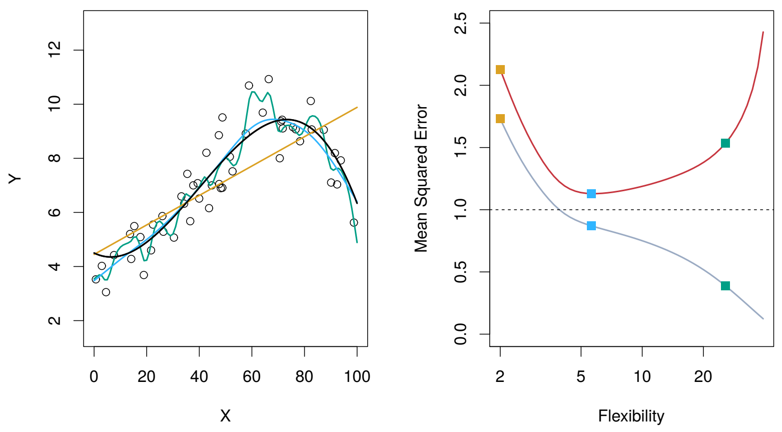


Figure 2.3: On the left panel the data simulated from f are shown in black. Three estimates of f are presented: the linear regression line(orange), and two smoothing spline fits(blue and green). On the right the training MSE(gray curve), test MSE(red curve), and minimum test MSE over all methods(dashed line) can be observed. Squares represent the training and test MSEs for the three fits shown in the left hand panel.

estimate over all the observations produces $\hat{f}(x_1), \hat{f}(x_2), \cdots, \hat{f}(x_n)$ and if these are approximately equal to $y_1, y_2, \cdots, y_n$, then the training MSE will be small. Knowing whether $\hat{f}(x_i) \approx y_i$ does not give a sufficient information about the quality of the elected model. Whether $\hat{f}(x_0)$ is approximately equal to y_0, where (x_0, y_0) is a previously unseen test observation not used to train the statistical learning method, offers the possibility to get more interesting results. The method offering the lowest test MSE, as opposed to the lowest training MSE, represents the criterion for the choice of the best model.

Considering a large number of test observations, we would be able to compute

$$Ave(y_0 - \hat{f}(x_0))^2. \tag{2.7}$$

Equation 2.7 represents the average squared prediction error for test observations (x_0, y_0). By adopting the method with the lowest training set MSE there would be no certainty for the test set MSE to be the lowest possible too. Nevertheless many statistical methods specifically estimates coefficients so as to minimize the training set MSE, producing large errors in the test MSE.

A non-linear function f(the black curve) generats the observations displayed in the left hand panel of Figure 2.3. Methods with increasing flexibility produces the orange, blue and green curves when trying to estimate the undelying model. In particular the orange line is the inflexible linear regression fit, while the blue and green curves were produced using smoothing splines with different level of smoothness. Growing levels of flexibility, make the curves fit the obsersevations closer. The green curve matching data very with high flexibility, but it ends up fitting poorly the true f because of it's sinuosity. Adjusting the level of flexibility of the smoothing spline, many different fits can be obtained.

Moving on the right hand panel of Figure 2.3, the gray curve displays the average training MSE as a function of flexibility in terms of degrees of freedom, for a number os smoothing splines. The degrees of freedom summarize the flexibility of a curve: more restricted and hence smoother curve has fewer degrees of freedom than a wiggly curve. The training MSE declines monotonically as flexibility increases. The true f is non-linear, so the orange linear fit is not flexible enough to estimate f well. The green curve instead, has the lowest training MSE and corresponds to the most flexible of the tree curves fit.

The red curve in the right-hand panel of Figure 2.3 represents the test MSE. As with the training MSE, the test MSE initially declines as the level of flexibility increases, but at some point it levels off and then starts to increase again. The orange and green curves, both have high test MSE, while the blue curve minimizes the test MSE. The horizontal dashed line indicates $Var(\epsilon)$, i.e., the irreducible error expressed in Equation 2.3, which corresponds to the lowest achievable test MSE among all possible methods. Hence, the smoothing spline(blue curve)is close to optimal.

As the flexibility of the statistical learning method increases, a monotone decrease in the training MSE appears as well as a U-shape like curve for the test MSE. This represents a fundamental property of statistical learning that holds regardless of the particular data set or the statistical method being used. The increase of model's flexibility produces a lower training MSE which may not correspond to a lower test MSE. Overfitting describes the situation in which a given method yields a small training MSE and, at the same time, a large test MSE. This means that the statistical learning procedure is working too hard while trying to discovery patterns into the training data, and may be picking up some patterns caused only by random change rather than by the true properties of the unknown function f.

Overfitting the training data, results in large test MSE. In fact, the patterns supposed to be present also into the test data, simply does not exist there producing then large errors. Irrespective of overfitting occurrence, the training MSE will always be lower with respect to the test MSE, because most statistical learning methods either directly or indirectly seek to minimize the training MSE.

2.3.1 The Bias-Variance Trade-Off

The U-shaped curve described in Section 2.3 underlies the competition between two forces that govern the choice of any statistical learning method:

- **Bias** refers to the error introduced by modeling a real life problem by a much simpler problem. The more flexible/complex a method is, the less bias it will have.

  The error related to the bias is taken as the difference between the expected (or average) prediction of the model and the correct value to be predicted. Imagine to repeat the whole model building process more than once: each time new data becomes available, a new analysis can take place producing a new model. Due to the randomness in the underlying data sets, the resulting models will have a range of predictions. At the end the Bias will measures how far off in general these models' predictions are from the correct value.

- **Variance** refers to how much the estimate for f would change by adopting a different training data set. In general, it can be observed that the more flexible a method is, the more variance it will have. The error corresponding to the variance is taken as the variability of a model prediction for a given data point. Repeating the entire model building process multiple times, the variance measures how much the predictors for a given point vary between different realizations of the model.

The bulls-eye diagram in Figure 2.4 offers the bias-variance graphical representation, presenting a plot of all the possible different combinations of both high and low bias and variance . Imagine the center of the target to be a model that perfectly predicts the correct values. Moving away from the bulls-eye, predictions get worse and worse. Repeating the entire model-building process getting as result a number of separate hits on the target representing the individual realization of the model. Results can display a large variability going from the center, meaning a very good prediction, to the exterior, indicating the presence of outliers or non standard values producing then a poorer prediction. These different realizations result in a scatter of hits on the target.

According to [18] estimating a model f trough $\hat{f}$ using whichever modeling technique in a given point x_0 produce an error denoted as expected squared prediction error:

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon) \tag{2.8}$$

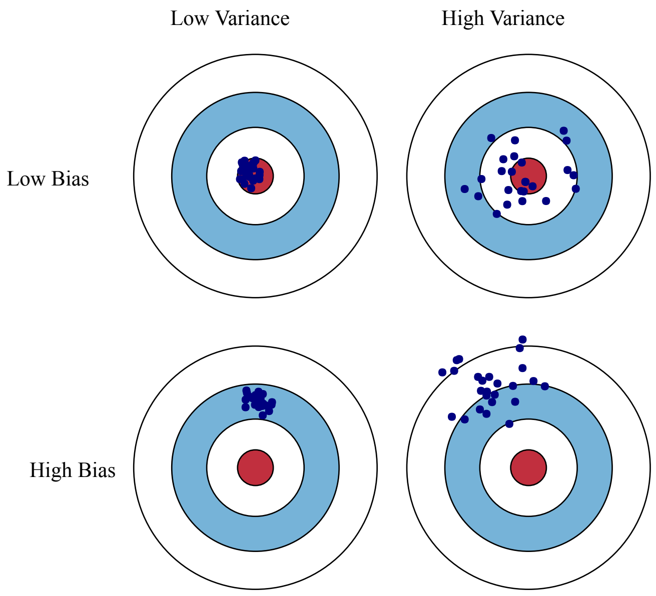


Figure 2.4: Graphical illustration of bias and variance

The expected test MSE refers to the average test MSE that would be obtained by repeatedly estimating f using a large number of training sets. The overall expected test MSE can be computed as the average $E(y_0 - \hat{f}(x_0))^2$ over all possible values of x_0 in the test set. The minimization of the expected test MSE goes through the selection of the model which simultaneously achieves low bias and low variance. Adopting more flexible methods will make the variance increase and the bias decrease. The relative changes of the two quantities determines whether the test MSE increases or decreases.

Relationship between bias, variance, and test set MSE defines in Equation 2.8. Good test set performances of statistical learning methods require low variance as well as low squared bias. Obtaining optimal performances from the bias and variance point of view is referred as "trade-off", since to extremely low bias may correspond an high variance and vice versa. The challenge lies in finding a method for which both the variance and the squared bias are simultaneously low.

Figure 2.5 presents the bias-variance trade-off considering different orders of flexibility of the function f generating the data. On the left hand panel of Figure 2.5 shows up a generic function f underlying the data. In general, as discussed throughout this Section, the test MSE displays an initial decrease, before starting increasing. The increase means that the method starts to fit the noise which is naturally present into the data. Cen-

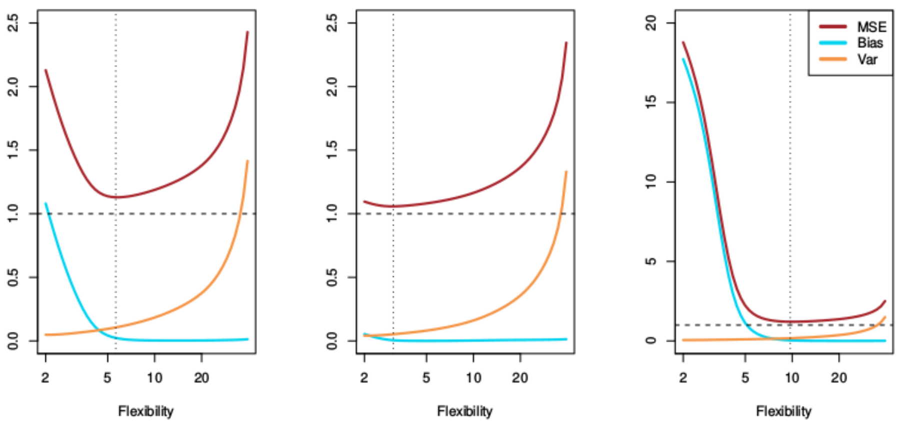


Figure 2.5: Squared bias(the blue curve), variance(the orange curve), $Var(\epsilon)$ (the dashed line) and test MSE(the red curve) for different underlying function f, respectively an general one, an highest linear function and finally an highly non linear function.

ter and right hand panel represent instead an almost-linear and an highly non-linear function f respectively. In the first case the test MSE starts growing immediately due to the fact that the function is linear. In the second case instead, the error starts immediately to grow since the generating model is linear. Finally in the third, case the high non linearity makes the error rapidly decrease as the complexity of the model increases, before leveling off to a stable value and starting increasing again at the end.

2.3.2 The Classification Setting

The estimate of the function f in a classification setting, based on training observations $\{(x_1, y_1), \cdots, (x_n, y_n)\}$, assume $y_1, \cdots, y_n$ to be qualitative measurements. Most common approach adopted for the quantification of the accuracy of the estimate $\hat{f}$ is the training error rate, i.e., the proportion of mistakes that were made applying the estimate to the training observations:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{2.9}$$

where $\hat{y}_i$ is the predicted class label for the i-th observation using $\hat{f}$. $I(y_i \neq \hat{y}_i)$ is an indicator variable, it is equal to 1 if $y_i \neq \hat{y}_i$ and equal to 0 if $y_i = \hat{y}_i$. If $I(y_i \neq \hat{y}_i) = 0$ then the i-th observation was classified correctly by the classification method adopted, otherwise it was misclassified. Equation 2.9 is referred as "*training error rate*" since it is computed on data used to train

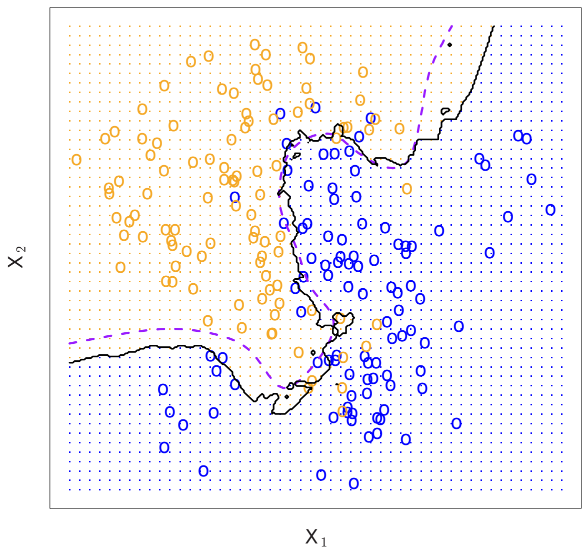


Figure 2.6: Simulated dataset consisting of 100 observations, distributed into two groups. The dashed line represents the Bayes decision boundary. The orange background grid represents the region in which a test observation will be assigned to the orange class, and the same for the blue background grid area respectively.

the system. By applying the classifier to previously unseen test observations of the form (x_0, y_0) we obtain the "*test error rate*" as:

$$Ave(I(y_0 \neq \hat{y}_0)), \tag{2.10}$$

where $\hat{y}_0$ is the predicted class label that results from the application of the classifier to the test observation with predictor x_0. Good classifiers are those achieving low test error rate.

The test error rate expressed in Equation 2.10 is minimized, on average, by a simple classifier that assigns each observation to the most likely class, given its predictor values. A test observation with predictor vector x_0 is assigned to the class j for which

$$Pr(Y = j|X = x_0) \tag{2.11}$$

is largest. The conditional probability is the probability that $Y = j$, given the observed predictor vector x_0. This classifier is addressed as "*Bayes Classifier*".

In a two-class problem where there are two only possible response values, lets say *class 1* or *class 2*. The Bayes classifier corresponds to predict *class 1* if $Pr(Y = 1|X = x_0) > 0.5$, and *class 2* otherwise. The orange and blue circles of Figure 2.6 correspond to the training observations that belong to two different classes. For each value of X_1 and X_2, there is a different probability of the response for being orange or blue. Computing the conditional probabilities for each value of X_1 and X_2, the orange shaded region reflects the set of points for which $Pr(Y = orange|X)$ is greater than 50%, while the blue shaded region indicates the set of points for which the probability is below 50%. The dashed line represents the points where the probability is exactly 50%, also denoted as "*Bayes decision boundary*". Depending on which side of the Bayes decision boundary an observation fall in, it will be assigned to the corresponding class. The Bayes classifier is the one that produces the lowest possible test error rate, called the "*Bayes error rate*". The Bayes classifier will always choose the class for which the Equation 2.11 is largest and, precisely, the error rate at $X = x_0$ will be $1 - max_j \; Pr(Y = j|X = x_0)$. The overall Bayes error rate is given by:

$$1 - E(1 - max_j \; Pr(Y = j|X)), \tag{2.12}$$

where the expectation averages the probability over all the possible values.

2.4 Clustering

Clustering can be considered the most important unsupervised learning problem[5]. In particular it deals with finding an inherent structure in a collection of unlabeled data. A *cluster* represents a collection of objects which are "similar" between each other and are "dissimilar" to the objects belonging to other clusters. Clustering devise a set of tools intended for the setting in which only a set of features measured on n observations becomes available.

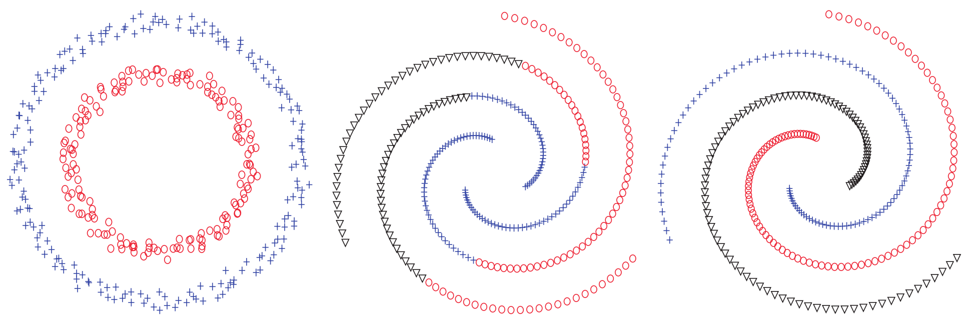
Two type of clustering can be distinguished:

- *distance-based clustering* adopts as similarity criterion the distance between pair of objects (i.e., two or more objects belong to the same cluster if they are "close" according to a given distance);

- *conceptual clustering*, in which two or more objects belong to the same cluster if this one defines a concept common to all the objects (i.e. the objects are grouped according to their fit to descriptive concepts and not according to simple similarity measures).

The goal of clustering is to extract the intrinsic grouping in a set of unlabeled data. The main requirements that a clustering algorithm should satisfy are

- *scalability*: the ability to handle a growing amount of work;



(a) Spectral clustering result for two-circle data set.

(b) Spectral clustering result for three-spiral data set.

(c) Path-based spectral clustering for three-spiral data set.

Figure 2.7: Generic examples of spectral clustering and path-based clustering.

- *capability*: discover clusters with arbitrary shape even when noise and outliers are present into the data set;
- *interpretability and usability of the results.*

Whichever clustering algorithm needs to be robust enough to cope with imperfect data and to extract new knowledge from the available dataset.

Some important issues can be identified when dealing with a large number of dimensions and large number of data items: the effectiveness of the method strictly depends on the definition of "distance" adopted.

According to [17] cluster analysis has received, during the past few years, an increasing attention in machine learning and statistics fields. Although many traditional clustering algorithms have been developed in literature, there are two methods that produce particularly interesting results on some challenging data sets, represented by elongated in addition to compact clusters. This two relevant methods are "*spectral clustering*" [25, 33] and "*path based clustering*" algorithm [26, 6, 9].

Despite the theoretical performances of this two clustering algorithm's family on some challenging data set, there exist situations in which their behavior is heavily affected by the presence of noise and outliers. On the two-circle dataset in Figure 2.7(a), spectral clustering offers very interesting results but reveal to be inadequate for the three-spiral data set of figure 2.7(b). The poor results obtained on the three-circle data set of Figure 2.7(b) can be mitigated adopting an in between approach, i.e., path-based spectral clustering, whose relevant results can be found in Figure 2.7(c) and whose description is left for Section 2.4.3.

2.4.1 Spectral Clustering

Success of spectral clustering [36] resides on the lack of the a priori assumptions about the shape of the considered clusters. As opposed to traditional cluster algorithms as K-means, where resulting cluster are always convex sets, spectral clustering can operate over very general problems such as interwined spirals or nested clusters. Moreover, spectral clustering can be implemented efficiently for large data sets, as long as they are based on sparse similarity graph, reducing the memory and computational effort. Once the similarity graph is chosen, the solution can be obtained solving a linear problem, without issue of getting stuck in local minima or restarting the algorithm for several times with different initializations. The choice of the proper similarity graph represents the non trivial point and its wrong choice may also lead to instability. Spectral clustering should not be confused for a " black box" algorithm automatically detecting the correct clusters, but it can be considered a powerful tool producing good results even when complex cases are encountered.

Many traditional clustering methods adopt a spherical or elliptical metric to group data points [14], hence they wont be able to work optimally with non-convex clusters. Spectral clustering is a generalization of standard clustering methods designed to work on data sets containing non-convex shapes. Concentric circles data sets, such as the one in Figure 2.7(a), highlight the weakness of traditional cluster algorithms such as K-means. The globular assumption on which they strictly rely does not allow the search for nested or overlapping clusters.

An undirected "*similarity graph*", denoted through $G = \langle V, E\rangle$, represents the observations and pair of connected vertexes, starting from an $N \times N$ matrix of pairwise similarities $s_{ii'} \geq 0$ weighted by $s_{ii'}$. Once the undirected similarity graph has been defined, the clustering problem can be rephrased as a graph-partition problem, where each connected component can be identified with a cluster.

In an undirected similarity graph weights between edges of different group are low, while ones between edges within the same group high is finally obtained. Spectral clustering algorithms requires the construction of a similarity graph for the representation of the local neighborhood relationship between observations. The similarity matrix and the similarity graph reflecting the local behavior can be obtained in several ways.

With "*adjacency matrix*" we refer to the edge weights matrix $W = \{w_{ii'}\}$ obtained from a similarity graph. The degree of vertex i, i.e., $g_i = \sum_{i'} w_{ii'}$, collects the sum of the weights of the edges connected to it. Let G be a diagonal matrix with the diagonal elements g_i.

The unnormalized *graph Laplacian* can be defined as:

$$L = G - W. \tag{2.13}$$

Spectral clustering finds the m eigenvectors $Z_{N \times m}$ corresponding to the m smallest eigenvalues of L. Using a standard method the rows of Z can be grouped yielding a clustering of the original data points.

For any vector f we have:

$$f^T L f = \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} w_{ii'} (f_i - f_{i'})^2. \tag{2.14}$$

Equation 2.14 suggests that a small value of $f^T L f$ can be achieved if pairs of points with large adjacencies have coordinates f_i and f_i' close together.

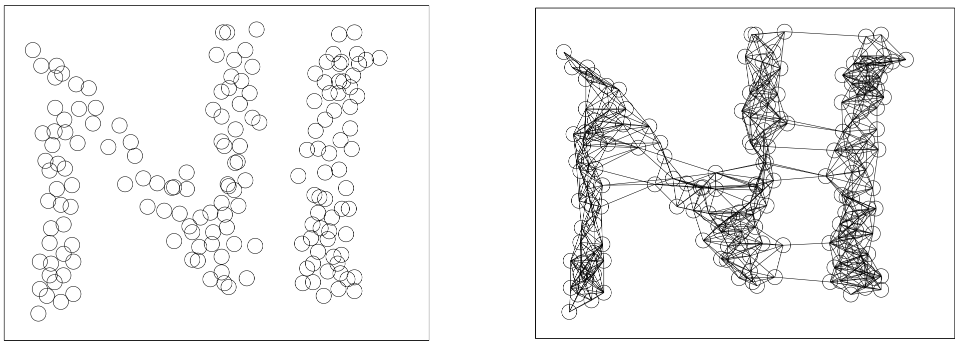
Since $1^T L 1 = 0$ for any graph, the constant vector is a trivial eigenvector with eigenvalue zero. Less obvious is the fact that if the graph is connected, it is only the zero eigenvector. For a graph with m connected components, the nodes can be reordered so that L is block diagonal with a block for each connected component. Then L has m eigenvectors of eigenvalue zero, and the eigenspace of eigenvalue zero is spanned by the indicator vectors of the connected components. In other words, it has strong and weak connections, so zero eigenvalue are approximated by small eigenvalues.

Spectral clustering is an interesting approach for finding non-convex clusters, but there are a number of issue that one must deal with in applying spectral clustering in practice. The first one is the choose of the type of similarity graph to be used and the associated parameters. Another element to choose is the number of eigenvectors to extract from L and finally, as with all the clustering methods, the number of clusters.

2.4.2 Pairwise & Path-Based Clustering

Despite spectral methods' success over a wide range of data collections, some problems still persist demanding for a different dissertation. Perhaps the most important problem is the lack of a straightforward probabilistic interpretation, which makes difficult the automatic parameters' tuning using the available training data. On the other hand significant progress in clustering has been achieved by algorithms based on pairwise affinities, which somehow relate data points [32].

A class of data collection, such as the one in Figure 2.8(a), which cannot be easily described through parametric models, can be successfully clustered



(a) Generic data set presenting pairwise affinities between data points.

(b) Pairwise clustering results for the generic presented data set.

Figure 2.8: Generic examples of pairwise clustering on a generic data set.

using pairwise clustering methods. These algorithms start by building a graph where:

- *vertexes* corresponds to data points.
- *Edges* corresponds to connections existing between nearby points, characterized by a weight that decreases with distance.

Adopting this kind of structure a clustering operation is equivalent to a graph partitioning.

The partitioning of a set of objects into groups aspires to the extraction of the hidden structure of the data collection relying on pairwise comparisons between different objects. In particular the pairwise comparisons are based on *proximity* or *(dis)similarity* information, which can be estimated even if objects are not elements of a metric space.

Pairwise clustering starts by grouping adopting a configuration of exactly one object per cluster and then successively merges the two most similar clusters. While general agglomerative clustering methods adopt algorithmic considerations rather than an optimization principle, a systematic approach is preferred in pairwise clustering. The objective function is based on an axiomatization of invariance properties and robustness for data grouping.

By replacing the pairwise object comparisons with a path-based dissimilarity measure, the cluster connectedness is emphasized. The effective dissimilarity between objects is defined as the largest edge cost on the minimal intra cluster path connecting both objects into the feature space. Two objects assigned to the same cluster are either similar or there exists a set of mediating objects such that two consecutive objects in the chain, are similar[10].

The final goal of path-based clustering algorithms is the extraction of elongated structures from data collections. This information can be obtained adopting a cost function considered as a generalization of graph-based clustering cost function. The cost function can be seen as a measure evaluating improvements produced over each step of the process. Despite agglomerative optimization is very fast, it can be influenced by fluctuations present within data. In order to reduce noise sensitivity the approach presented in [9] proposes a data resampling technique, employed to measure the instability of the resulting grouping.

The reliability measure aims to define a statistical criterion in order to determine the number of cluster. In fact, the number of clusters is rarely known a priori and has to be chosen as the value providing the highest stability.

While in most data analysis application distances are measured by Euclidean distances(or more general $\ell_p - norms$), this method assumes a general situation describing objects with their pairwise dissimilarities, which may also violate the triangular inequality. The representation of the objects space is not explicit and the dissimilarity representation of all the existing object is a priori defined.

A clustering instance combines a set of objects $\mathcal{O} = \{o_1, \cdots, \o_n\}$ and the respective mutual dissimilarities D, building an undirected graph through a vertex set $\mathcal{O}$ and edge set $\{D_{i,j} : o_i.o_j \in \mathcal{O}\}$. Dissimilarities relies on a symmetric assumption and a clustering solution $c : \mathcal{O} \rightarrow \{1, \cdots, k\}$, maps each object to one of the k labels.

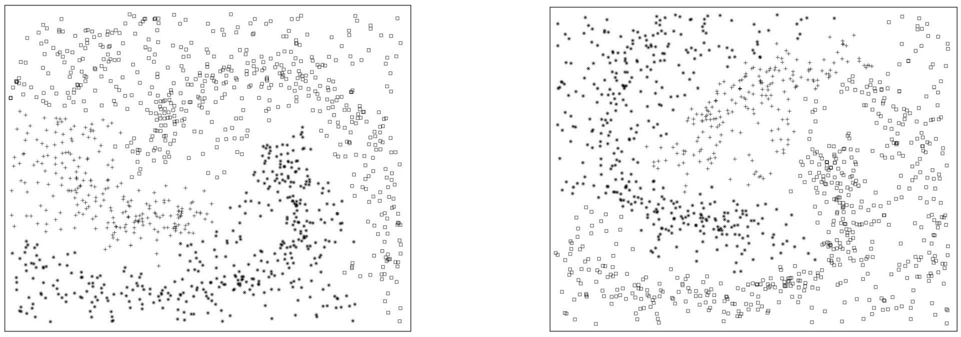
The quality of an "object-cluster" assignment relates to a cost function $\mathcal{H} : \mathcal{C} \rightarrow \mathbb{R}_+$, projecting the space of clustering solutions $\mathcal{C}$ to the non-negative reals.

According to pairwise clustering, two objects o_i, o_j should be assigned to the same cluster if their pairwise similarity is high, or, in other words, their dissimilarity is small.

Considering paths $\mathcal{P}_{i,j}(c)$ connecting two object o_i to o_j, the dissimilarity corresponding to a particular path $p \in \mathcal{C}$ can be defined as the maximal dissimilarity encountered on that path. The effective dissimilarity D_{ij}^{eff} can be computed as the minimum overall path distances:

$$D_{ij}^{eff} = min_{p \in \mathcal{P}_{ij}(c)} \{max_{1 \leq i \leq |p|-1} D_{p[i]p[i+1]}\} \tag{2.15}$$

$D_{i,j}^{eff}$ provides the discriminative information to decide if two objects are jointly assigned to the same cluster.



(a) First subsample. (b) Second subsample.

Figure 2.9: Two subsamples from the same input data set. Agglomerative algorithms sometimes fail to extract the correct underlying structure of the clusters, due to noise link between objects.

The cost for each cluster ν represents the mean effective dissimilarity of cluster ν scaled by the number of objects $|\mathcal{O}_\nu|$:

$$\mathcal{H}^{pbc}(c; D) = \sum_{\nu in\{1,\cdots,k\}} |\mathcal{O}_\nu(c)| \overline{D_\nu^{eff}}(c; D) \tag{2.16}$$

where the mean effective dissimilarity averages all the pairwise effective dissimilarities

$$\overline{D_\nu^{eff}}(c; D) = \frac{1}{|\mathcal{O}_\nu(c)|^2} \sum_{o_i \in \mathcal{O}_\nu} \sum_{o_j \in \mathcal{O}_\nu} D_{ij}^{eff}(c; D) \tag{2.17}$$

Cost function in Equation 2.17 is invariant with regard to additive shift and dissimilarities' scaling. The best clustering solution is then obtained as the one which groups regions of high probability within the same cluster.

Two problems subsist with agglomerative optimization:

1. It is not guaranteed that the global minimum of the path-based clustering energy function is found.

2. Clustering results for two data sets drawn from the same data distribution should be comparable.

A suitable solution can be obtained by adopting "*bootstrap aggregation*" or "*bagging*". Bagging is a general-purpose procedure for reducing the variance of a statistical learning methods. The desired advantage of procedures with low variance is that they will yield similar results if they are applied repeatedly to distinct data sets.

Given a set of n independent observations $Z_1, \cdots, Z_n$, each with variance σ^2, the variance of the mean $\overline{Z}$ of the observations is given by σ^2/n. This means that averaging a set of observations reduces variance. Hence it is possible to reduce the variance and, at the same time, increase the prediction accuracy by taking many training sets from the population, build a separate prediction model and finally average the resulting predictions. Compute $\hat{f}^1(x), \hat{f}^2(x), \cdots, \hat{f}^B(x)$ using B separate training sets, and average them in order to obtain a single low-variance statistical learning model:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x). \tag{2.18}$$

The B different sets are drawn by sampling n objects with replacement from the empirical probability distribution defined by the input collection. Because some objects might occur more than one times, the B sets of objects containing the sampled elements, can be considered as a bootstrap replications, denoted as $\mathcal{O}^b$ for $(1 \leq b \leq B)$. The computation of the mapping function c^b for each of the B replications relies on agglomerative optimization.

The probability of a cluster assignment, considering that the costs of a mapping is invariant with respect to the permutation of the cluster labels, can be obtained combining the B mapping solutions:

$$\hat{f}_b ag(x) = \frac{1}{B}\sum_{b=1} B\hat{f}^{*b}(x). \tag{2.19}$$

Averaging with bagging has the effect of reducing the variance giving impressive improvements in accuracy, combining together hundreds or even thousands of “data-replications” into a single procedure.

This bagging-based method[9] reside on the Minimum Spanning Tree clustering algorithm[27], an elementary method to find elongated(connected) structures into the data and to detect clusters with irregular boundaries. This ability comes from the fact that it does not rely on the spherical shaped clustering assumption, adopted in many traditional clustering algorithms.

According to [16] the minimum spanning tree is denoted as a subset of edges of a connected, edge-weighted undirected graph connecting all vertexes together, without cycles and with the minimum possible total edge weight.

Minimum Spanning Tree algorithm starts by considering n different clusters, each containing exactly one object, and proceeds fusing objects into bigger groups adopting an agglomerative optimization.

The relative low cost of building a minimum spanning tree represents a huge plus for this algorithm. It can be approximately around $\mathcal{O}(m \cdot log(n))$, where m is the number of edges and n is the number of vertexes in the graph.

On the other hands, an high sensitivity to outliers, i.e., a single object far away from all the other, represents a relevant drawback. If the input collection contains noisy data, then the probability to encounter outliers may represent a serious issue.

An Euclidean Minimum Spanning Tree is a spanning of a set of n points in a metric space ($\mathbb{E}_n$), where the length of an edge is the Euclidean distance between a pair of points in the point set.

A Minimum Spanning Tree based clustering algorithm adopting Euclidean Minimum Spanning Tree (EMST) of a graph [27] produces the structure of point clusters in the n-dimensional Euclidean space. A measure of optimality, such as minimum intra-cluster distance or maximum inter-cluster distance, develops the required ability to perform cluster detection. According to this optimality measure, two possible different ways to produce a group of clusters arises:

- Given the number of clusters in advance, the algorithm sorts the edges in descending order of weights and then remove the edges with the heaviest weights till reaching the desired number of clusters. This is denoted as standard EMST(SEMST) algorithm which first builds the EMST of a data set and then removes the inconsistent edges that satisfy the inconsistency measure. The process repeats until the number of clusters reach the desired one, creating a hierarchy of clusters.

- If the clusters' number is a priori unknown, the algorithm proceeds deleting edges which do not satisfy a predefined inconsistency measure from the tree. This is denoted as ZEMST because the method employs the Zahn inconsistency measure. Groups of clusters arise partitioning the input collection, maximizing the overall standard deviation reduction. Finally the clusters' number can be determined by finding the local maximum of the standard deviation reduction function.

The intuitive idea behind path based dissimilarity[26] expresses the situations in which even if two objects are very far from each other in space, but there exists a link connecting them through a path composed by points which can be considered close each other, then their distance should be adjusted to a smaller value, in order to reflect the existing connection.

When a points interconnection rises up, the membership to the same cluster can be evaluated. Denoting P_{ij} as the set of all possible paths connecting the graph pair points $x_i - x_j$, their effective dissimilarity characterizing each path $p \in P_{ij}$ can be obtained as the maximum of all edge weights in the path p. The path-based distance d'_{ij} between x_i and x_j, de-

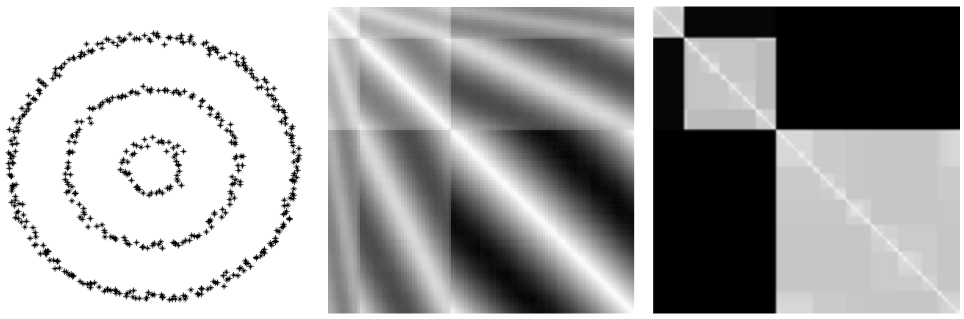

(a) Input data. (b) Original distance matrix. (c) Transformed distance matrix.

Figure 2.10: Distance matrices of three-circles dataset.

noted as $pbdis(x_i, x_j)$, can be computed as the minimum among effective dissimilarities of all paths in P_{ij}:

$$d'_{ij} = pbdis(x_i, x_j) = min_{p \in P_{ij}} \{ max_{1 \leq h \leq |p|} (dis(p[h], p[h+1])) \} \qquad (2.20)$$

where $p[h]$ denotes the object at the h^{th} position in the path p and $|p|$ denotes the length of path p.

The only definition of path-based distance doesn't offer the total solution for the purpose of path-based clustering. A multi-dimensional scaling (MDS) technique could be adopted for the visual exploration of data based on dissimilarity information, improving results of the clustering process.

Multi-dimensional scaling technique produces a new data representation trying to preserve original distances with a better representation of the data collection. Multi-dimensional scaling analytical solution for the problem employs the well known linear algebra operation of eigendecomposition[11]. Eigendecomposition operates a factorization of a given matrix into a canonical form, representing it through eigenvalues and eigenvectors: eigenvalues' magnitude will indicate the relative contribution of the resulting coordinate matrix's column with respect to the original distance matrix. A shrinkage of the dimensions' number results considering only those eigenvalues with a small loss of information.

This transformation carries the advantage that even classical and wide spread clustering algorithm such as K-means [1] will produce interesting results. Another advantage resides in the single transformation operated on the whole data set at the beginning without further operations for the procedure to work. The clusters' number definition instead, represents the main drawback of this method. Clusters' number parameter is rarely available in advance and limits the scenarios in which this algorithm can be applied.

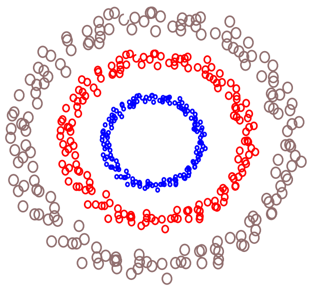


Figure 2.11: Final clustering results using K-means after the application of multi-dimensional scaling

Figure 2.10 visually presents results of the application of this methods on a commonly used three circle data set. Figure 2.10(a) shows the original collection in a two-dimensional space while its distance matrices before and after the multi-dimensional scaling appear respectively in Figures 2.10(b) and 2.10(c). The difference between Figures 2.10(b) and 2.10(c) clearly displays how the application of the path-based dissimilarity transformation enhanced the cluster structures on the distance matrix. After the application of the transformation even a classical algorithms, such as K-means, can easily detect and identify the three circles as shown in Figure 2.11.

2.4.3 In between Spectral Clustering and Path-Based Clustering

Robust path based spectral clustering approach proposed in [6], develops a solution in between spectral clustering and path-based clustering. This approach is strictly based on the concept of M-estimation [15] statistics, in order to enhance the poor results obtained by the separate application of the two approaches.

Recalling the definition of path-based distance expressed in Equation 2.20, the dissimilarity between two points x_i and x_j should take on small values when they belong to the same cluster, and viceversa high values when they do not.

Problems arise in presence of outliers because in that cases the dissimilarity between points residing in different cluster may become smaller than it should be. This means that the path-based dissimilarity measure is sensitive to noise and outliers.

To address the robustness problem [6] propose to adopt a weighted path-based similarity measure, used in combination with spectral clustering to obtain a robust path-based spectral clustering method.

A fully connected graph with n vertexes corresponding to the n points $\mathscr{X} = \{x_1, x_2, \cdots, x_n\}$ represents the initial data collection.

To each edge (i, j) that connects two points in the graph corresponds a weight $s'_{i,j}$ reflecting the original similarity between the two vertexes x_i and x_j:

$$s'_{ij} = \begin{cases} exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) & for\ i \neq j, \\ 0 & for\ i = j \end{cases} \tag{2.21}$$

where σ represents a scaling parameter controlling the fall off speed of $s'_{i,j}$ with respect to the Euclidean distance between x_i and x_j.

By denoting $\mathscr{P}_{ij}$ as the set of all paths $p \in \mathscr{P}_{ij}$ from vertex i to j through the graph, the effective similarity s^p_{ij} is the minimum edge's weight along the path. Finally the total similarity s_{ij} take on the maximum value of all path-based effective similarities s^p_{ij}'s for paths $p \in \mathscr{P}_{ij}$:

$$s_{ij} = max_{p \in \mathscr{P}_{ij}} \{min_{1 \leq h \leq |p|} s'_{p[h] \cdot p[h+1]}\}. \tag{2.22}$$

Into Equation 2.22 $p[h]$ denotes the h-th vertex along the path from vertex i to vertex j and $|p|$ denotes the number of vertexes that p goes through. According to the definition the similarity between points that belongs to different clusters should be small, but the presence of outliers leads similarity to assume larger values than they should be. This high sensitivity to noise and outliers can be reduced by adopting an M-estimator.

The squared residual error e^2_{ij} of i,j based on the x_i-x_j distance can be defined as:

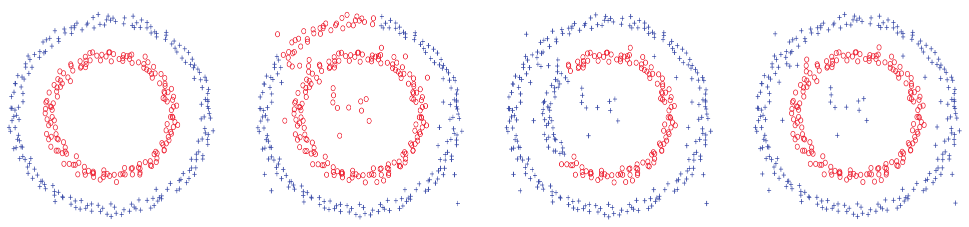
$$e^2_{ij} = \frac{\|x_i - x_j\|^2}{|\mathcal{N}_i|} \tag{2.23}$$

where $\mathcal{N}$ denotes the neighborhood of x_i and $|\mathcal{N}_i|$ the number of neighbors it contains. Using e^2_{ij} the total squared residual error of the estimators on x_i can be expressed as $\sum_{x_j \in \mathcal{N}_i} \|x_i - x_j\|^2 / |\mathcal{N}_i|$. Therefore the total squared error E can be obtained as:

$$E = \sum_{i=1}^{n} \sum_{x_j \in \mathcal{N}_i} e^2_{ij} = \sum_{i=1}^{n} \sum_{x_j \in \mathcal{N}_i} \frac{\|x_i - x_j\|^2}{|\mathcal{N}_i|}. \tag{2.24}$$

While the standard least squared would try to minimize the error in Equation 2.24, the adoption of a robust estimation technique replaces it with a robust statistical estimator, minimizing the follwoing expression:

$$E_\rho = \sum_{i=1}^{n} \sum_{x_j \in \mathcal{N}_i} \rho(e_{ij}) \tag{2.25}$$



(a) Original data before the application of noise points.

(b) Spectral clustering results for the original data set.

(c) Path-based spectral clustering results for the original data set.

(d) Robust path-based spectral clustering results for the original data set.

Figure 2.12: Clustering results for a noisy 2-circle data set.

where $\rho(\cdot)$ usually grows more slowly than the quadratic function, reducing the outliers' influence. Using robust estimation techniques, E_ρ can be minimized by solving an *iterative re-weighted least squares* problem[13].

At the end of the optimization process the weights w'_i for each point x_i can be obtained as:

$$w'_i = \sum_{x_j \in \mathcal{N}_i} a_{ij} = \sum_{x_j \in \mathcal{N}_i} exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) = \sum_{x_j \in \mathcal{N}_i} s'_{ij}, \tag{2.26}$$

expressed recurring to the original similarity values.

A data point's weight is large if many points are in its proximity while it is small when few point are close to it. In order to reduce the effect of the parameter σ over the w'_i's range, each weight has to be normalized through

$$w_i = w'_i / max_{x_i \in \mathscr{H}} \{w'_i\} \tag{2.27}$$

so that all weights finally reside in range $(0, 1]$. Large values for w_i points out that x_i is likely to be inside a compact cluster, while small ones that x_i may be an outlier.

Robust path-based similarity can be finally reformulated as:

$$s_{ij} = max_{p \in \mathscr{P}_{ij}} \{min_{1 \le h \le |p} w_{p[h]} w_{p[h+1]} s'_{p[h]p[h+1]}\}. \tag{2.28}$$

Equation 2.28 reflects the genuine similarity between x_i and x_j neglecting outliers' presence. This method reflects the simple idea that if there exists a path connecting x_i to x_j, going through only points with high weights values, then the similarity should be large as the total similarity, revealing the two points' membership to the same cluster.

Although both spectral clustering(2.4.1) and path-based clustering(2.4.2) can detect the two clusters in the two-circle data set of Figure 2.12(a), they reveal to be not robust enough against the presence of outliers as in Figure 2.12(b). Figures from 2.12(b) to 2.12(d) compare spectral clustering and path-based clustering with robust path based spectral clustering[6] when 30 noise points are added. Some of the introduced noise points, located between the two circles, ends up connecting the two clusters introducing further problems. Figure 2.12(b) shows that spectral clustering produces no interesting results for this data set. On data set in Figure 2.12(c) the presence of noise points reduces the measures for the dissimilarity values affects negatively the cluster separation when path-based clustering has been adopted. Finally, Figure 2.12(d) presents the results obtained by the application of the robust path-based spectral clustering: the introduction of the robust similarity measure is reveal to be very effective in reducing the influence of the outliers. This brings to better identification of the different groups.

2.5 The Neural Gas Network Approach

Lack of information about the desired output characterizes unsupervised learning problems. Only input data are available with the aim of inferring a function to describe the hidden structure from an unlabeled data collection. “Dimensionality reduction”, i.e., find a low-dimensional subspace of the input vector space containing most or all the input data represents one possible objective. Linear subspace with this particular property can be computed directly by principal component analysis[1] or iteratively with a number of network models.

Dimensionality reduction’s objective can be gathered through the use of self-organizing map(SOM) sometimes called Kohoen feature map, which are particular type of artificial neural network whose input space is general wider with respect to the output one. Kohoen feature maps in conjunction with the “growing cell structures” allow projection onto non-linear discretely sampled subspaces whose dimensionality has been chosen a priori. Self organizing maps differ from usually adopted artificial neural network because they apply “*competitive learning*” instead of classical techniques of error-reduction learning such as back-propagation with gradient descent.

[1] Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal of the number of original variables.

Given some high-dimensional data distribution $P(\xi)$, the objective is to find a topological structure closely reflecting the topology of the data distribution. This process is denoted as *Topological Learning.* The data structure's construction can be elegantly performed adopting "competitive Hebbian learning" which in turns refers to a vector quantization method[2].

Competitive Hebbian learning(CHL)[12] assumes a number of centers in $\mathbb{R}^n$ and successively inserts topological connections among them by evaluating input signals collected from a data distribution $P(\xi)$. For each input signal x, it connects the two closest centers by an edge, where closeness is evaluated according to the Euclidean distance. Figure 2.13 represents two ways of defining closeness among a set of points.

A subgraph of the "Delaunay triangulation" corresponding to the set of centers appears in Figure 2.13(a), connecting points having neighboring Voronoi polygons(thin lines). Basically this reduces to points having small Euclidean distance with respect to the given set of points.

The "induced Delaunay triangulation"(thick lines) in Figure 2.13(b) can be obtained by masking the original Delaunay triangulation with a data distribution(shaded). Two centers are connected if and only if the common border of their Voronoi polygons lies at least partially in those areas of the input space $\mathbb{R}^n$ where $P(\xi) > 0$. In other words only centers lying on the input data submanifold or in its vicinity develop any edges. The others are useless for the purpose of topology learning and are often called dead units.

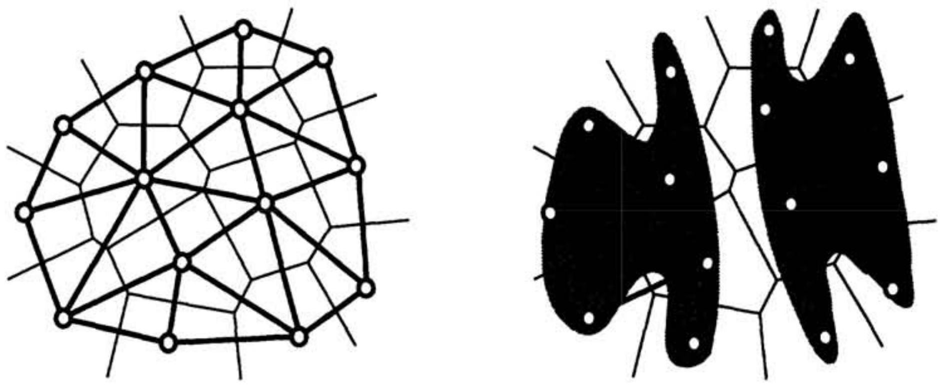
To make use of all centers they have to be placed in those regions of $\mathbb{R}^n$ where $P(\xi)$ differs from zero. This could be done using vector quantization[3] procedure.

Neural Gas Network algorithm[12] operates an adaptation of the k-nearest centers toward the input signal x where k decreases from a large to a smaller final value. A large initial value for k causes adaptation toward the input signal of a large fraction of centers, while it's decrease brings to a low number of center adapted time by time. Adaptation strength underlies a similar decay schedule. For the parameter decay operation, the total number of adaptation steps must be a priori defined.

Initially it requires to run the Neural Gas algorithm in order to distribute

[2]Vector quantization is a classical quantization technique usually adopted in the field of signal processing, which allows the modeling of probability density function by the distribution of prototype vectors.

[3]Vector Quantization (VQ). The density matching property of vector quantization is powerful, especially for identifying the density of large and high-dimensioned data. Since data points are represented by their closest centroid's index, commonly occurring data have low error, and rare data high error.


(a) The Delaunay triangulation(thick lines) connects point having neighboring Voronoi polygons(thin lines).

(b) Induced Delaunay triangulation(thick lines) arises by masking the original Delaunay triangulation with a data distribution $P(\sigma)$(shaded).

Figure 2.13: Representation of two ways of defining closeness among a set of points.

a certain number of centers into the space according to an input data distribution. In a second moment the competitive Hebbian learning process can be applied to generate the final topology. The execution of the two different techniques can be even applied concurrently. The last operation required is a procedure for removing obsolete edges. In fact, the centers' motion makes previously generated edges invalid when they overpass a fixed age threshold.

Competitive Hebbian learning method does not influence in any way the outcome of the Neural Gas method, because adaptations employ input distances and not network topology. On the other hands, Neural Gas algorithm does not influence in any way the topology generation of competitive Hebbian learning since it moves the centers around.

Combination of Neural Gas and competitive Hebbian learning represents a successful strategy to operate topology learning. The main drawback relies on the fact that the number of the centers must be a priori defined. This is usually not trivial and, depending on the situations, many different values can be adopted.

2.5.1 Growing Neural Gas Network

Despite Neural Gas network (Section 2.5) approach represents a very effective method to perform topology learning, it's lack of an automatic procedure to detect an a priori suitable number of centers and adaptation steps, makes it dependent on a human intervention. A second method, strictly relying on Neural Gas network approach, tackle this problem with an interesting adaptation avoiding to fix a priori this two fundamental parameters.

This method considers:

- A set A of units. To each $c \in A$ corresponds an associated reference vector $w_c \in \mathbb{R}^n$, which can be regarded as positions in input space of the corresponding unit. Each units can be represented through a node.

- A set N of non-weighted connections among pairs of units, whose purpose is the definition of a topological structure. The pairs units' connections can be represented through edges.

In principle there are possible infinite number of n-dimensional input signals obeying to a probability density function $P(\xi)$. Growing Neural Gas network approach[12] proposes to add new units over time to an initially small network. Local statistical measures obtained throughout the adaptation steps evaluate over this small network. In this way, the network topology can be generated incrementally adopting the competitive Hebbian learning method, and the final dimension of the network will depend on the input local statistical behavior.

Topology learning process starts by placing randomly two units into the space which have to be learned. Successively an input signal ξ, according to the probability distribution $P(\xi)$, is initially generated. According to the generated input signal ξ, the two closest units can be identified. Each unit contains a local counter, responsible for the tracking of the distance between inputs and the unit itself. At each step the local counter of the closest unit is updated by incrementing the aging of all the emanated edges. In particular the update of the local counter adopts the following expression:

$$\Delta_{error}(s_1) = \|w_{s_1} - \xi\|. \tag{2.29}$$

The closest unit s_1 and its direct topological neighbors moves toward ξ respectively by fractions ϵ_b and ϵ_n of the total distance, which can be computed as:

$$\Delta w_{s_1} = \epsilon_b(\xi - w_{s_1}) \tag{2.30}$$

$$\Delta w_n = \epsilon_n(\xi - w_n) \tag{2.31}$$

for all direct neighbors n of s_1. If units s_1 and s_2 are connected each other, the age is reset to zero. If they are not connected, a new link between them have to be introduced. Obsolete edges removal process occurs when the nodes display ages larger than a fixed threshold a_{max}. Input signals' cardinality multiple of a fixed parameter λ causes the insertion of a new unit. The new unit's placement is not performed randomly, but supporting the units with accumulated the largest error q. Finally the new unit r take place between units q and its neighbor f with the largest error variable:

$$w_r = 0.5(w_q + w_f). \tag{2.32}$$

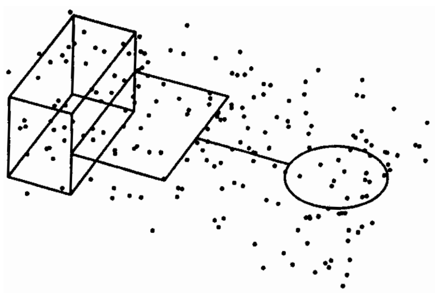
Existing edges between the previous units q and f must be removed and new edges between the new r and q, f created. The q and f's error variables decrease multiplying them by a constant α and the r's error variable initializes with the new value of the error variable of q instead. The process ends up decreasing all the error variables by multiplying it with a constant d.

Adaptation steps towards the input signals lead to a general movement of all units towards those areas of the input space where signals come from $P(\xi) > 0$. The insertion of edges between the first and the second-nearest unit with respect to an input signal generates a single connection of the "induced Delaunay triangulation" with respect to the current position of all units. Removal of edges is necessary to get rid of those edges which are no longer part of the "induced Delaunay triangulation", due to the fact that their ending points have been moved. Local edge aging in combination with the edges' age re-setting procedure achieves the objective of keeping the network updated.

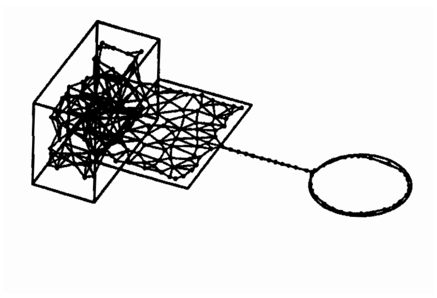
With edges insertion and removal the model tries to construct and then track the "induced Delaunay triangulation", which is a slowly moving target due to the adaptation of the reference vectors. The accumulation of squared distances during the adaptation helps to identify units lying in areas of the input space where the mapping from signals to units causes much error. To reduce this error, new units are inserted in such regions.

Results obtained with Growing Neural Gas network algorithm are particularly interesting when topology learning is the objective, even when the input distribution follows complicated structures evolving over different dimensionalities.

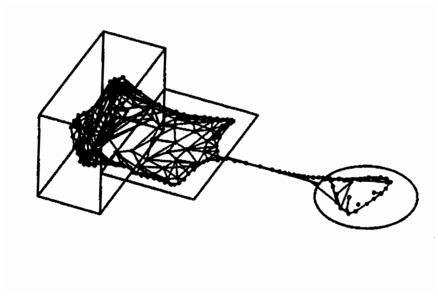
Figure 2.14 shows the Growing Neural Gas network adaptation steps to a signal distribution with different dimensionalities in different areas of the input space. Figure 2.14(a) displays the initial placement of two random units on the surface. Figures 2.14(c) to 2.14(f) present the adaptations produced by the application of 600, 1800, 5000, 15000 and 20000 generated signals respectively to the initial small network. Especially Figure 2.14(f) highlights how the Growing Neural Gas method efficiently operates the learning of



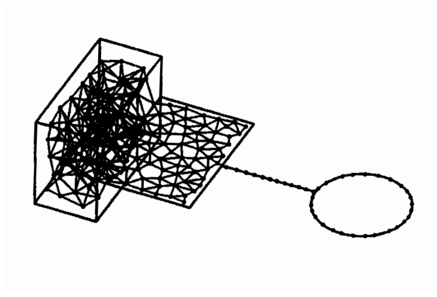
(a) Initial situation.



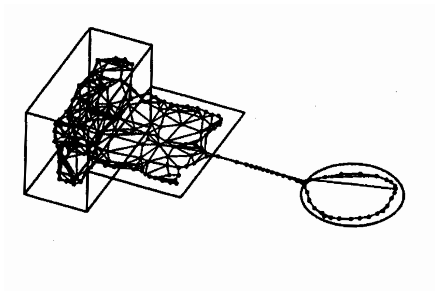
(b) Introduction of 600 generated input points.



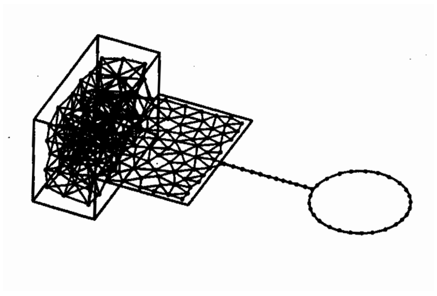
(c) Introduction of 1800 generated input points.



(d) Introduction of 5000 generated input points.

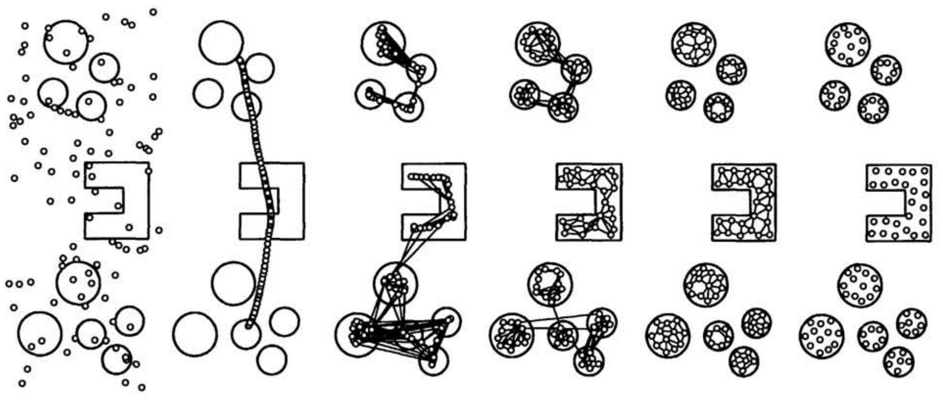


(e) Introduction of 15000 generated input points.



(f) Introduction of 20000 generated input points.

Figure 2.14: A "Growing Neural Gas" example.


(a) “Neural Gas” and “competitive Hebbian learning”.

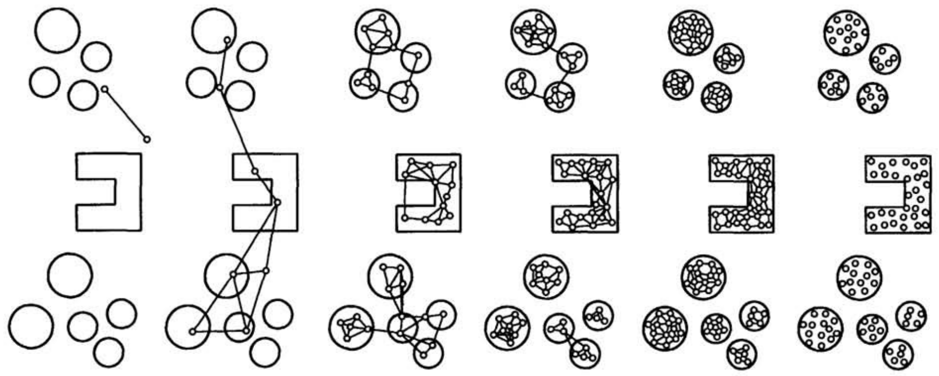

(b) “Growing Neural Gas” with the use of “competitive Hebbian learning”.

Figure 2.15: “Growing Neural Gas” vs. “Neural Gas”

complicated topologies. Hypothetically the learning process could be operated indefinitely, so Figure 2.14(f) representation is not necessarily the final one.

Another example represented in Figure 2.15 displays the main difference between the current method and the Neural Gas method of Section 2.5. Both methods are perfectly able to learn the clusters' distribution, but the Growing Neural Gas Network approach is able to indefinitely continue until the complete disclosure of the smaller clusters is completed.

2.5.2 Adaptive Incremental Growing Neural Gas Network

Each edge, in Growing Neural Gas, has an associated age used to remove old edges in order to keep the topology dynamically updated. While in Growing Neural Gas method the decision of introducing a new neuron is taken upon a fixed parameter λ, in Adaptive Incremental Growing Neural

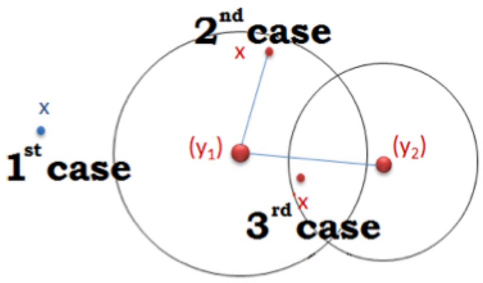


Figure 2.16: The three possible different cases in which the AING operates.

Gas Network procedure the decision is taken upon an adaptive parameter-free distance threshold instead. An excessive growth of neurons' number is avoided by condensing them using a probabilistic criterion, in such a way a new topology arises to preserve memory constraints.

Let y_1 and y_2 be respectively the nearest and the second nearest neurons from a new data-point x, such that $dist(y_1, x) < dist(y_2, x)$:

1. if x is far enough from y_1: a new neuron y_{new} is created at x (1^{st} case);
2. if x is close enough to y_1 but far enough from y_2: a new neuron y_{new} is created at x, and linked to y_1 by a new edge (2^{nd} case);
3. if x is close enough to y_1 and close enough to y_2 (3^{rd} case):
    - move y_1 and and its neighboring neurons towards x, i.e. modify their reference vectors to be less distant from x;
    - increase the age of y_1s edges;
    - link y_1 to y_2 by a new edge (reset its age to 0 if it already exists);
    - activate the neighboring neurons of y_1;
    - delete the old edges if any.

Figure 2.16 displays intuitively this procedure in a graphical manner, representing the three possible different cases which can be encountered.

An age parameter characterizes each existing edge emanating from a given neuron, keeping track of the seniority of a particular neuron-to-neuron link. The age of neuron edges increments each time a data-point assigns to the winning unit y_1 (such as in the 3^{rd} case). Each time an input-generated point is close enough to neurons y_1 and y_2, the age of the link connecting the two different units reset to 0. In such a way their interconnection in refreshed to smaller values, reflecting the youngness of the link. If the age of an edge increases without being never reset, it will reach a maximum age value. By reaching this threshold value, the node will be considered “old” and thus removed though the edge removal procedure.

Point-to-point distances’ considerations rely on a parameter-free adaptive threshold T_y, which can be defined for a generic neuron y as:

$$\frac{\sum_{e \in X_y} dist(y,e) + \sum_{e \in V_y} |X_e| \times dist(y,e)}{|X_y| + \sum_{e \in V_y} |X_e|}. \tag{2.33}$$

In case y has no assigned data-points (X_y is empty) and it has no neighbor (V_y is empty), then we consider the adaptive threshold to be the half distance from y to its nearest neuron, according to the expression:

$$\frac{dist(y,\tilde{y})}{2}, \tilde{y} = argmin_{\tilde{y} \neq y}\ dist(y,\tilde{y}). \tag{2.34}$$

By fusing together Equation 2.33 and 2.34 the expression becomes:

$$T_y = \begin{cases} \frac{\sum_{e \in X_y} dist(y,e) + \sum_{e \in V_y} |X_e| \times dist(y,e)}{|X_y| + \sum_{e \in V_y} |X_e|} & \text{if } X_y \neq \emptyset \vee V_y \neq \emptyset, \\ \frac{dist(y,\tilde{y})}{2}, \tilde{y} = argmin_{\tilde{y} \neq y}\ dist(y,\tilde{y}) & otherwise. \end{cases} \tag{2.35}$$

The adaptive threshold updates incrementally as the data become available by keeping information related to each neuron up to date over time.

Once again, the merging process(as in Growing Neural Gas network approach depicted in Section 2.5.1) tries to reduce the number of neurons generated by the algorithm. When the number of neurons within the network reaches an upper bound fixed a priori, some of them can optionally fuse together, reducing the whole dimension of the network itself.

Dimensionality reduction by neurons merging employs a probabilistic criterion evaluating the probability for y to be far enough from its nearest neuron $\tilde{y}$:

$$P_{y,\tilde{y}} = \frac{|X_y| \times dist(y,\tilde{y})}{\kappa}. \tag{2.36}$$

Equation 2.36 highlights that the probability is proportional to the distance between the two nearest neurons and the number of data point assigned to y: in particular, the more is y is far from $\tilde{y}$, the more likely they wont be merged together. As we can see, the probability is inversely proportional to a parameter κ, used for the tuning of the merging probability between two nearest neurons. In Expression 2.36, $P_{y,\tilde{y}}$ may also assume values higher then 1 when κ is not sufficiently big. Values greater than 1 has no mathematical sense for a probability, then a better formulation consists in:

$$P_{y,\tilde{y}} = min\left(\frac{|X_y| \times dist(y,\tilde{y})}{\kappa}, 1\right) \tag{2.37}$$

which guarantees a probability included in 0 and 1.

Chapter 3

Methods and Procedures

"In materia di accessibilità, una denuncia vale più di mille suppliche. "

Angela Gambirasio

3.1 The MEP Data Collections

MEP-Traces and MEP-APP applications were both developed specifically for Android devices, in collaboration with the "Design department of Politecnico di Milano" in order to make them intuitive and easy-to-use for any kind of user[30].

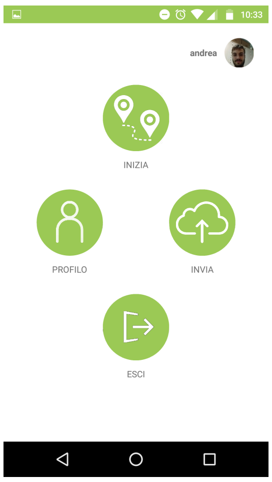
MEP-Traces offers the possibility to track users' habitual path within the city area. Figure 3.1 shows the elements the users face off when using the MEP-Traces application. The main menu, in Figure 3.1(c), gives the access to the whole functionality of the application. Starting from the top there are: the start-acquisition button, the user-profile button, the send-acquisition button, and finally the exit button.

During data collection, the placement of the device reveals to be one of the main issue that we encountered during the definition of the applications' functionalities, especially for motor-free wheelchair users. In fact, it would be impractical for users with motor disabilities to hold it while moving. The installation of the device on a physical support fixed on the wheelchair, represents a possible solution, leaving the user the possibility to freely move while tracking its current path.

However, this simple solution introduces a computational problem: algorithms usually adopted for step detection and movement, cannot be applied anymore, and new data processing methods have to be adopted.

(a) MEP-Traces

(b) Welcome



(c) Main Menú

Figure 3.1: MEP-Traces Activities. From left to right: the MEP-Traces Splash screen, displayed at the application launch; the Welcome screen of the application after the login; the Main menu with the application's features buttons.

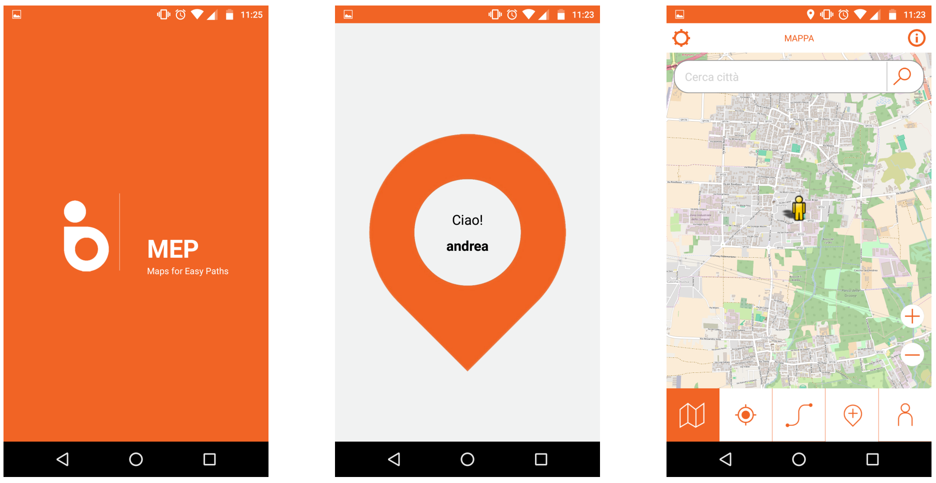
By definition, we classify the MEP-Traces routes as accessible. This anyway does not represent the absolute accessibility degree for an area, and users considerations could modify its definition by reporting the presence of obstacles/barriers along the recorded path.

In order reconstruct the path as close as possible to the real one, MEP-Traces records data coming out from multiple sensors(i.e., GPS, gyroscope accelerometer and magnetometer), required for a further computation of a particular sensor fusion technique(described in Chapter 3.2). The joint contribution of all the different recorded paths builds the MEP implicit data collection.

Path information alone would not be enough for the purpose of building accessibility maps, so another type of data has to be considered. This other data type is the user's explicit information, supplied during the mapping process. For this particular purpose, MEP-Traces allows to record timestamped[1] and geo-located pictures of the obstacles encountered during the mapping process. Those data are immediately uploaded on the server and they become available to the whole users community.

Implicit and explicit data jointly contribute to the establishment of the

[1]Timestamps[23] are sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second.



(a) MEP-APP initial screen. (b) Welcome. (c) Maps Visualization.

Figure 3.2: MEP-APP activities. From left to right: the initial screen with the MEP logo; the welcome screen; the map which can be consulted by the user.

"MEP Collection", which represents the starting point of our knowledge discovery process.

While sensors acquisition is totally automatic and user-transparent, the obstacle' position has to be explicitly reported by the user who encounter it along his/her route. Sensors and obstacles notifications reach the server which in turns proceeds with the update of the database with new available information.

Visualization of data stored in the MEP database is available in any moment to registered users through the MEP-APP application. It offers the possibility to retrieve the best possible route based on the user's needs taking into account his/her physical disease. The MEP-APP, exactly as the MEP-Traces application, is very intuitive.

Figure 3.2 represents the MEP-APP elements that the users encounter in the application. The application immediately localizes the position of the user over the map(Figure 3.2(c)). By using the search bar at the top of the application, the user can visualize information and notifications about the interested city.

The back-end server outperforms the exchange of information between the two applications: it receives, elaborate, and supplies data from/to many different sources. The data flow behind MEP-Traces and MEP-APP shows up in Figure 3.3. Green elements refer to the MEP-Traces data flow, while the orange ones represent the MEP-APP flow. The back-end server offers

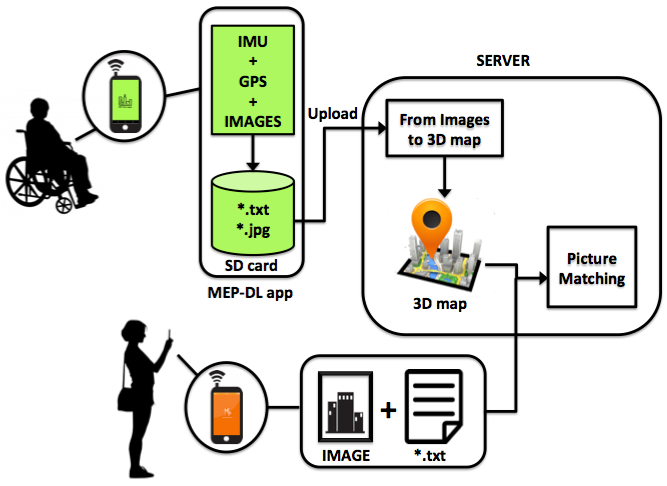


Figure 3.3: Implicit Data Collection general schema.

the set of tools which make the two mobile applications work properly.

3.1.1 Implicit Data Collection

As introduced in Section 3.1, the MEP data collection is made up by two types of data, i.e., "implicit" and "explicit". The concept of implicit data collection will be explained in detail throughout this section, while the treatment of the concept of explicit data is left for Section 3.1.2.

MEP-Traces (the upper green part in Figure 3.3) collects implicit data in a total automatic procedure which does not require any user intervention. For wheelchair users the device is assumed to be fixed upon the vehicle: in terms of acquired data, a fixed smartphone or tablet is supposed to produce an higher quality sensors data collection with respect to devices kept into pockets or bags.

The user records the GPS and IMU[2] information on the internal memory or into the SD-card[3] of the device where MEP-Traces has been installed on. Data coming from real-world GPS sensor demonstrate not to be accurate enough for our purposes. We then introduce a method which corrects the GPS coordinates adopting the IMU information, and produces a noticeable improvement of the localizations. Further details of this technique can be found in Section 3.2 which describes the MEP-Fusion engine in all its relevant aspects.

The data store into text files, dividing into:

[2]Inertial Measurement Units

[3]Secure Digital memory card

- *MEP_POSITION_INFO.txt* stores GPS data sampled every second.
- *MEP_MOTION_INFO.txt* stores data coming out from the sensors.
- *MEP_NMEA_INFO.txt* stores GPS data in the NMEA[4] format [34].
- *MEP_DEVICE_INFO.txt* collects some useful information about the device characteristics and battery consumption.

Implicit data have the characteristic to be completely transparent to the users when they are generated. Once the user collects implicit data, he/she can upload them on the server using MEP-Traces. The back-end server immediately starts elaborating data as soon as the upload process stops, and data are completely available (further details in Section 3.6). The first step of the elaboration employs a fusion algorithm to find an approximate path representation of the user experience along his/her route. A second step adopts a simple technique which rigidly transports points located over buildings on more appropriate places. Finally a clustering technique tries to group points coming from different paths into pre-specified definitions of accessibility. The server response is the result of the application of several methods and techniques coming from different fields, in order to suggest the best accessible path according the user's needs.

At the end of the elaboration of "implicit" data, the accessibility will be a kind of available and usable knowledge about the areas' accessibility around the space. Of course the creation of this knowledge strictly relies on the mapping process: not covered/mapped areas do not have any accessibility definition.

Implicit data alone only brings positive information about the accessibility of an area, but anything about its unattainability. For this reason the collection of explicit data, along with implicit ones, has to be adopted in order to produce complete results.

3.1.2 Explicit Data Collection

As discussed in Section 3.1.1, the server employs a set of implicit data to reconstruct the best accessible path for a given user. Considerations about implicit data alone are not enough to obtain a complete information about the paths' accessibility structures within a city. Implicit data, integrated

[4]NMEA format is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the National Marine Electronics Association.

with explicit information, produce an additional knowledge about different accessibility levels of the paths. Both applications allow users to send notifications about barriers or obstacles that they encounter during the travel. Notifications take place in the following way:

- Collecting at most three pictures of the obstacle from three different point of view.
- Evaluating the seriousness of a barrier can be expressed in a range from one to three.
- Inserting a short description to give further information.

The back-end server receives each user-filled formatted notification. All explicit information join together to produce an update of the obstacles list contained into the MEP database ("*public.obstacle*" table).

Users' notifications express fundamental information about the paths' accessibility which cannot be extracted directly from the implicit data collection conveyed by the MEP-Traces application. The completeness and soundness of information drifts directly from this additional data: once the condition of the area is known, i.e., the ground, the position of the steps and/or the poles, etc., it is possible to decide how to classify the given area in an accessibility scale. Using such kind of information, the users are served with the best possible path based on their disability.

The entire knowledge discovery process relies on this two kind of data collections, i.e., "implicit" and "explicit", representing the minimum fragment of information to reconstruct cities' accessibility levels.

3.1.3 Issue and Requirements

The MEP project innovativeness resides on the method used to collect data which the project purposes require for: no adoption of dedicated nor specific devices, but only commonly spread smartphones and tablets instead.

Mobile devices' spread and diffusion during the last few years reported a substantial increase, and, at the same time, the increased computing power of such kind of devices gives the MEP project a chance to be very effective all over the world.

Nowadays these devices come with a huge amount of low cost sensors chips marked by an high frequency rate and able to perform activities previously found only on fully equipped desktop operating systems. The growth of technology offers new possibilities to users and researchers which have to be investigated in dept.

MEP-Traces and MEP-APP demand to some minimum devices' characteristics, which can be summarized as:

- Satisfy the minimum requirements for the system to operate in the best condition, i.e., we expect at least a device to mount Android 4.4.4 KitKat or higher, providing at least accelerometer and magnetometer sensors and optionally the gyroscope sensor.
- Since the operations on the sensors are not just a matter of acquisition but it is also required a tuning of the parameters within them, a certain amount of flexibility of the hardware is required for the applications to work (dual core CPU or higher).
- Elaboration of the data collected by smartphones and tablets should be saved into a text format, therefore we require at least a minimum space of the SD-card.

3.1.4 Android Sensor Framework

Almost all Android devices come with a set of built-in sensors that measure motion, orientation, and a list of various environmental conditions in order to implement a series of function with the aim of improving the users' experience. Each sensor provides raw data characterized by high precision and accuracy capable to monitor three-dimensional device's movement or positioning as well as changes in the environmental ambient near the device itself.

In particular the Android platform supports three broad categories of sensors:

- *Motion sensors*: they measure acceleration and rotational forces along three main axes. Among this category there are accelerometers, gravity, gyroscopes, and rotational vector sensors.
- *Environmental sensors*: they measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. Among them there are barometers, photometers, and thermometers.
- *Position sensors*: they measure the physical position of a device. This category includes orientation and magnetometer senors.

The access to the device' sensors for the acquisition of the raw data requires the adoption of the Android sensor framework, which provides several

Table 3.1: Complete sensor overview of the Android platform.

Sensor	Type	Common Uses
TYPE_ACCELEROMETER	Hardware	Motion detection.
TYPE_AMBIENT_TEMPERATURE	Hardware	Monitoring air temperatures.
TYPE_GRAVITY	Software or Hardware	Motion detection.
TYPE_GYROSCOPE	Hardware	Rotation detection.
TYPE_LIGHT	Hardware	Controlling screen brightness.
TYPE_LINEAR_ACCELERATION	Software or Hardware	Monitoring acceleration along a single axis.
TYPE_MAGNETIC_FIELD	Hardware	Creating a compass.
TYPE_ORIENTATION	Software	Determining device position.
TYPE_PRESSURE	Hardware	Monitoring air pressure changes.
TYPE_PROXIMITY	Hardware	Phone position during a call.
TYPE_RELATIVE_HUMIDITY	Hardware	Monitoring dewpoint, absolute, and relative humidity.
TYPE_ROTATION_VECTOR	Software or Hardware	Motion detection and rotation detection.
TYPE_TEMPERATURE	Hardware	Monitoring temperatures.

classes and interfaces for the development of a series of sensor-related tasks. Two general class of sensors arise from the Android sensor framework, i.e. hardware-based and software-based sensors.

Hardware-based sensors are physical components built onto the device itself. Their data come directly from the measurement of specific environmental properties, such as the acceleration, geomagnetic field strength or angular change.

On the other hand, the software-based sensors do not refer to physical devices, although they mimic hardware-based sensors. Software sensors collect data from one or more hardware-based sensors denoting as virtual or synthetic sensors. Linear acceleration and gravity sensors reside among software-sensor category.

Table 3.1 presents all the possible sensor type commonly present on Android device. It also presents the sensor's classification into the hardware/software categories and a common use of the sensor's data type.

3.2 The MEP-Fusion Engine

The MEP-Fusion engine is a set of tools whose aim is the perfect reconstruction of the users' traveled paths, using implicit data collected through MEP-Traces.

Multi-sensor fusion techniques are required for many reasons. First of all the sensors' measurements quality: the IMU[5] measurements (represented by the joint contribution of accelerometer, gyroscope, and magnetometer) reveal to be corrupted by an intrinsic error which can be represented as an unpredictable white noise. On the other hands, the GPS sensor provides

[5]Inertial Measurements Unit

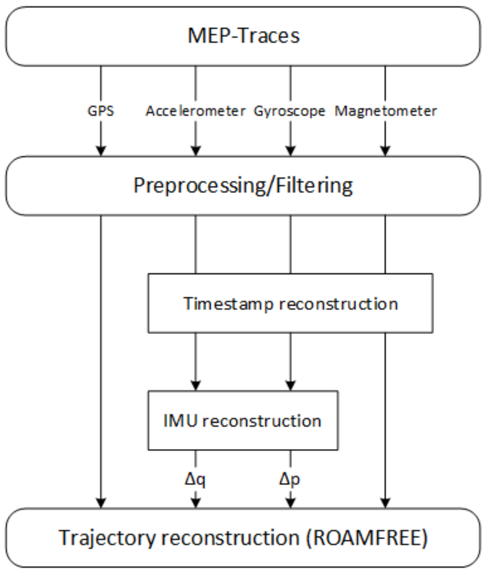


Figure 3.4: General processing schema applied to data coming from MEP-Traces in order to perform the trajectories reconstruction.

the absolute position of the device and it can displays errors in the order of meters or it can also be completely unavailable.

Another reason can be found in the a priori assumption of the target users: people with motor impairments are supposed to move using wheelchairs. This implies that traditional method, commonly adopted to localize pedestrians, based on step detection, appears to be ineffective for the MEP project purposes.

The adoption of the ROAMFREE[7][6] sensor fusion library presented in [2] offers a solution to the two explained problems. In particular MEP-Fusion engine employs a modified and enhanced form of the ROAMFREE library, accommodating to sensors commonly spread among mobile devices.

Figure 3.4 presents the MEP-Fusion engine general operative schema in the processing of MEP-Traces' collected data. The MEP-Fusion engine adopts splines in the Pre-processing/Filtering step to perform a fixed frequency sampling. This is required by the ROAMFREE framework which needs measurements collected at a fixed frequency, whereas the Android operating system adopts an event-based sampling schema while recording the sensors' measurements. Once the splines have been created, they are sampled at a frequency equal to the nominal frequency of the mobile sensors.

[6]ROAMFREE (Robust Odometry Applying Multisensor Fusion to Reduce Estimation Errors)

Because timestamps collected by the Android operating system according to the standard "wall" clock (i.e. time-date) are expressed in milliseconds from January 1, 1970 00:00:00.0 UTC[7], the correspondence with the real-world dates and times can be performed uniquely. Through the use of a reconstruction module, the IMU can be reconstructed according to UTC format.

Measurements coming from gyroscope and accelerometer cannot be used directly into the computation. A fundamental element of the ROAMFREE library, i.e., the *IMUhandler* is responsible for the combination of the inertial measurements, derived from the variations of translation and rotation. As result, fusing the obtained data together with the GPS measurements, produces an improvement of the path-noise cleaning process. Further descriptions of how sensors fuse and details about the ROAMFREE library are left for Section 3.2.1.

3.2.1 The ROAMFREE Library

The ROAMFREE library [2] is a framework developed by the "Artificial Intelligence and Robotics lab of Politecnico di Milano". It offers a library of sensor measurements models, a tracking module, and a calibration suite that allows the estimation of unknown sensor parameters.

Originally this set of tools where not specifically designed to solve the specific problem of path reconstruction for the MEP-Traces application, but its application on the MEP project can reveal an interesting twist.

The original purpose of the ROAMFREE library is the fusion of an arbitrary sensors number, in order to determine the pose[8] of a mobile robot. Even if this is not the case of the MEP-project, which presents a different quality and heterogeneity of data instead, it offers the fundamentals for the development of a specific and dedicated algorithm.

One of the main issue is that each device presents slightly different hardware characteristics which make impossible the a priori sensors' calibration. Through an abstraction on logical sensors, ROAMFREE performs a categorization of the sensors into different categories:

- Absolute position and/or velocity;
- Angular and linear speed;
- Acceleration and vector field (e.g. magnetic field and gravitational acceleration).

[7]Universal Time Coordinated

[8]Pose is the combination of position and orientation

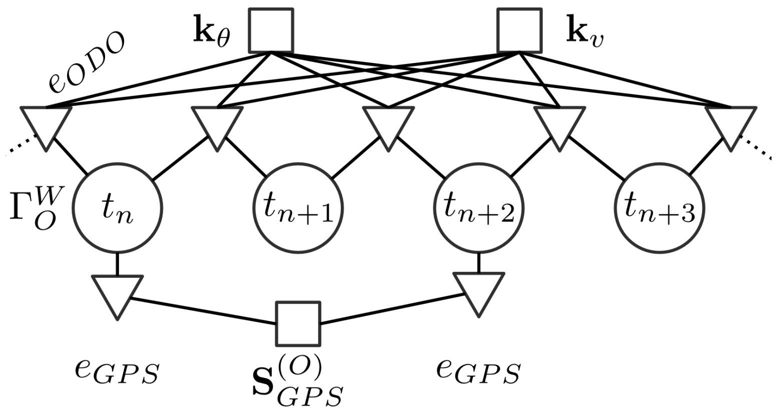


Figure 3.5: Instance of hyper-graph for pose tracking and self-calibration with four pose nodes.

A specific error model can be obtained for each sensor, by explaining the relation between the pose and the real measured data. Such explanation is obtained by taking into account the source of distortion, the bias and the noise which affect real-world measurements.

The ROAMFREE library employs three different reference frame type:

- W is the fixed world reference frame, taken into a known geo-referenced location;
- O identifies the moving reference frame placed at the center of the device to localize;
- S_i represents the i-th sensor frame, whose origin and orientation are defined with respect to O.

Through the use of the three different reference frames, ROMFREE is able, at time t, to track the position and orientation of the moving reference frame O with respect to global geo-referenced frame W, fusing the measurements in the location sensor frames S_i. Such an estimate is represented with the notation $\Gamma_O^W(t)$ including both the position and the orientation of the device.

Employing a master sensor, characterized through an high frequency, in order to predict $\Gamma_O^W(t + \Delta t)$ given the last pose estimate $\Gamma_O^W(t)$ and its measurements $z(t)$, the algorithm is able to produce a graph.

Each time a new reading for the master sensor is available, ROAMFREE instantiates a new node $\Gamma_O^W(t + \Delta t)$ using the last pose estimate available, $\Gamma_O^W(t)$, and $z(t)$ to compute an initial guess for the node. The initialization of the so called odometry edge[9] between poses at time t and $t + \Delta t$ requires the sensor's measurements $z(t)$.

Each time new measurements from non master sensors become available, the corresponding hyper-edges are inserted into the graph between the nodes with the nearest timestamp.

Figure 3.5 represents an example of possible hyper-graph where two types of hyper-edges are present (odometry hyper-edges, e_{ODO}, and global positioning systems hyper-edges, e_{GPS})

3.3 Geodetic System

Geodetic systems[10] apply for different purpose such as geodesy, navigation, surveying by cartographers, and satellite navigation systems.

The Earth is not a perfect sphere, and geodetic systems tackle the problem of obtaining the real position on the Earth in a precise mathematical way. The difference in coordinates between data, commonly addressed as "datum shift", can vary from one place to Another even within a country or region. The datum shift can be a value in between zero and hundreds of meters, which may represents an unacceptable error depending on the application.

The absence of a common surveying reference point before the advent of the GPS positioning system leads to the creation of nationals maps, usually not combining each other. As result, the lack of a common reference point produces a fragmented scenario for the data representation.

A datum can be defined as a reference point or surface against which position measurements are obtained. This is associated with a model of the shape of the Earth, and can be adopted in order to compute the exact position. While Horizontal datums describe a point over the Earth's surface, expressed in latitude and longitude, vertical datums measure elevations or underwater depths.

- **Horizontal datum**: represents the model to measure the positions on the earth. A specific point on the Earth assumes multiple different coordinates representing the different measurements adopted to represent the point itself. Despite several local horizontal datums have been applied in the past to reference some convenient reference point, the spread of recent technology produces precise heart's shape measure-

[9]Odometry edge is an hyper-edge representing a displacement

[10]http://wiki.gis.com/wiki/index.php/Geodetic_system

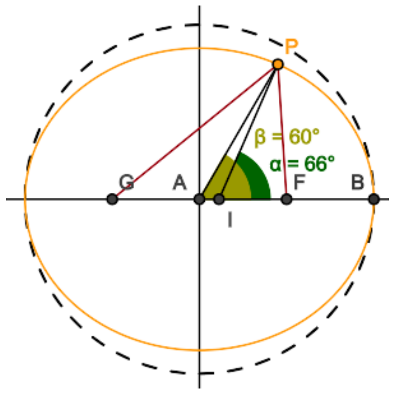


Figure 3.6: Ellipsoid approximation of the Earth

ments, allowing the adoption of datums spread over larger surfaces. The WGS 84 is now a commonly adopted standard datum.

- **Vertical datum**: represents the unit for the points' elevation measurement on the heart's surface. In common usage, elevations refers to the height above the sea level. This may introduce some issue because the sea levels are non-constant. For this reason we reasonably consider a specific zero point, computing the elevation based on the geodetic model being used, with no further reference to sea levels.

3.3.1 Geodetic Coordinates

An ellipsoid approximates the Earth's surface, and characterizes locations near the surface in terms of latitude (ϕ), longitude (λ) and height (h). This ellipsoid is completely parametrized upon the semi-major axis a and the flattening f.

It is important to notice that geodetic latitude (ϕ) is different from geocentric latitude (ϕ'): geodetic latitude corresponds to the angle between the normal of the spheroid and the plane of the equator, while geocentric latitude is determined around the center.

Figure 3.6 clearly shows that the same position on a spheroid may have different latitude's angle, depending on whether the angle itself measures the normal (angle α) or the center (angle β). The flatness of the spheroid (in orange) is greater with respect to the one of the Earth, resulting in an exaggerated difference between "geodetic" and "geocentric" latitudes. Table 3.2 represents the geodetic parameters required for the ellipsoid definition and construction.

From a and f we derive the semi-minor axis b, first eccentricity e and

Table 3.2: Geodetic defining parameters.

Parameter	*Symbol*
Semi-major axis	a
Reciprocal of flattening	$1/f$

Table 3.3: Geodetic derived geometric constants.

Parameter	*Symbol*
Semi-minor axis	$b = a(1-f)$
Reciprocal of flattening	$e^2, = 1 - b^2/a^2, = 2f - f^2$
Second eccentricity	$e'^2 = a^2/b^2 - 1 = f(2-f)/(1-f)^2$

second eccentricity e' of the ellipsoid. Table 3.3 reports the derived parameters.

The Earth-centered Earth-fixed (ECEF), or conventional terrestrial coordinate system, rotates with the Earth and has its origin at the center of the Earth itself.

- The X axis passes through the equator at the prime meridian.
- The Z axis passes through the north pole.
- The Y axis can be determined by the right-hand rule to be passing through the equator at 90° longitude.

Figure 3.7 represents the ENU's reference system principal elements. Many applications adopt the local East, North, Up (ENU) Cartesian coordinate system in which the local coordinates form a plane tangent to the

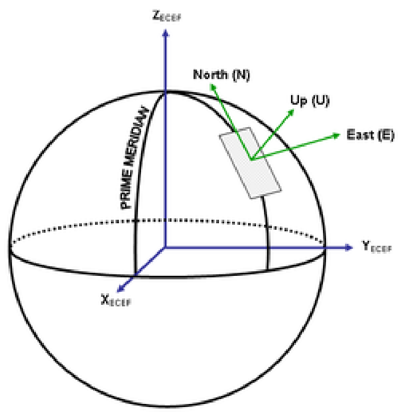


Figure 3.7: ENU reference system

Earth's surface fixed to a specific location. By convention the east axis, the north and the up corresponds to x, y and z respectively.

3.3.2 Conversion

Conversion from geodetic to local ENU coordinates involves a two stage process: first the conversion from geodetic to ECEF coordinates and finally the conversion from ECEF to local ENU coordinates.

In turn, the conversion from geodetic (latitude ϕ, longitude λ, height h) to ECEF coordinates requires the following operations:

$$\begin{aligned} X &= \left(\frac{a}{\chi} + h\right) cos(\phi)cos(\lambda) \\ Y &= \left(\frac{a}{\chi} + h\right) cos(\phi)sin(\lambda) \\ Z &= \left(\frac{a(1-e^2)}{\chi} + h\right) sin(\phi) \end{aligned} \tag{3.1}$$

where $\chi = \sqrt{1 - e^2 sin^2(\phi)}$. The parameters a and e^2 represent respectively the semi-major axis and the square of the first numerical eccentricity of the ellipsoid according to Tables 3.2 and 3.3. The parameter $\frac{a}{\chi}$ (denoted as *Normal*) represents the distance from the surface to the Z-axis along the ellipsoid normal.

ECEF to local ENU coordinates transformation requires a reference point: assuming the reference point located at $\{X_r, Y_r, Z_r\}$ and the interested element at $\{X_p, Y_p, Z_p\}$, the vector pointing from the reference to the interested element in ENU reference system is:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -sin(\lambda) & cos(\lambda) & 0 \\ -sin(\phi)cos(\lambda) & -sin(\phi)sin(\lambda) & cos(\phi) \\ cos(\phi)cos(\lambda) & cos(\phi)sin(\lambda) & sin(\phi) \end{bmatrix} \begin{bmatrix} X_p - X_r \\ Y_p - Y_r \\ Z_p - Z_r \end{bmatrix} \tag{3.2}$$

The conversion from local ENU to geodetic coordinates requires first a datum conversion from local ENU to ECEF and finally ECEF transform to geodetic. The inversion from ECEF to ENU coordinates is performed as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -sin(\lambda) & -sin(\lambda)cos(\lambda) & cos(\phi)cos(\lambda) \\ cos(\lambda) & -sin(\phi)sin(\lambda) & cos(\phi)sin(\lambda) \\ 0 & cos(\phi) & sin(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} \tag{3.3}$$

Despite the simplicity of this initial conversion, the conversion from ECEF to geodetic is a much more complicated problem, involving the following 15 steps, which require to know in advance the parameters $\{a, b, e, e'\}$:

$$
\begin{aligned}
&r = \sqrt{X^2 + Y^2} \\
&E^2 = a^2 - b^2 \\
&F = 54b^2Z^2 \\
&G = r^2 + (1 - e^2)Z^2 - e^2E^2 \\
&C = \frac{e^4Fr^2}{G^3} \\
&S = \sqrt{3}1 + C + \sqrt{C^2 + 2C} \\
&P = \frac{F}{3(S + \frac{1}{S} + 1)^2G^2} \\
&Q = \sqrt{1 + 2e^4P} \\
&r_0 = \frac{-(Pe^2r)}{1 + Q} + \sqrt{\frac{1}{2}a^2(1 + 1/Q) - \frac{P(1 - e^2)Z^2}{Q(1 + Q)} - \frac{1}{2}Pr^2} \\
&U = \sqrt{(r - e^2r_0)^2 + Z^2} \\
&V = \sqrt{(r - e^2r_0)^2 + (1 - e^2)Z^2} \\
&Z_0 = \frac{b^2Z}{aV} \\
&h = U\left(1 - \frac{b^2}{aV}\right) \\
&\phi = arctan\left[\frac{Z + e'^2Z_0}{r}\right] \\
&\lambda = arctan2\,[Y, X]
\end{aligned}
\tag{3.4}
$$

where $arctan2[X, Y]$ is the four quadrant inverse tangent function.

Expressing a set of measurements according to a reference datum allows to tune the level of accuracy when processing geographic data, focusing on the map's area relevant for the computation.

3.3.3 Cartographic Correction of Global Navigation Satellite System Trajectories From Low Cost Devices

Even though the MEP-Fusion engine operates a trajectories' cleaning (Section 3.2) they can still present some intrinsic error which cannot be removed without accessing to further information, such as a perfect superposition with buildings. Fusion of static information (such as the ones contained into the OpenStreetMap database) leads to interesting improvements.

The Global Navigation Satellite System (GNSS) trajectory-building superposition correction[22], developed by the "Dipartimento di Ingegneria

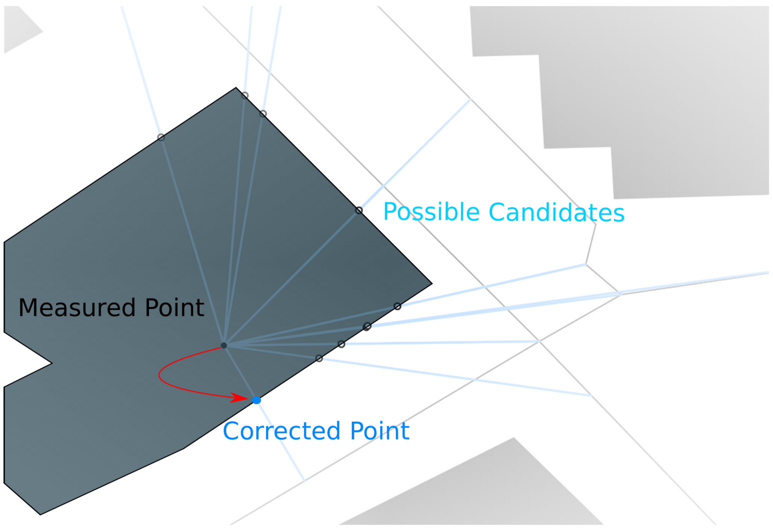
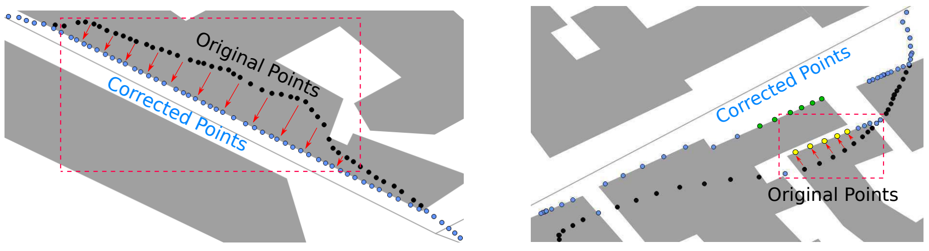


Figure 3.8: "Bring-outside" correction produced by the GNSS algorithm from the DICA.



(a) Real case example "bring-outside" correction.

(b) Wrong results example "bring-outside" correction.

Figure 3.9: On the left a real case result after the application of the GNSS correction. On the right there is an example in which the GNSS correction results in an error.

Civile e Ambientale" in Politecnico di Milano, employs a real time triggering technique over the GNSS points table. A dedicated trigger activates when new points appear into the database, performing a simple check of the point's coordinates superposition with respect to the coordinates of the buildings around it. If the trigger detects the occurrence of a complete overlapping, it rigidly projects the point outside the building's area considering the closest "road pipe" as reference.

Figure 3.8 shows the initial situation in which the trigger detects a new point (in black) pending over a building. In this case the point has a number of possible candidate points (displayed in white) at the building's boundary. The final point's coordinates (the blue point) correspond to the ones reflecting the closest road pipe.

Considering a real path example, composed by multiple points, this method produces results as in Figure 3.9(a). All the points occurring over a building (the black points) shifts toward new positions (the blue points). This example clearly shows that original measured points can even occur over large portions of buildings, due to large errors in the surveys.

Some critical situations can lead to a completely unwanted result such the one represented in Figure 3.9(b): large measurements errors produce the complete misleading of the building's side over which the points are projected (the yellow points), whereas their real correct position should be a completely different one (the green points).

3.4 PostgreSQL and PostGIS

PostgreSQL is a powerful, open source object-relational database system with more than 15 years of active development, and a proven architecture that earned a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems and it is fully ACID[11] compliant, with a full support for foreign keys, joins, views, triggers, and store procedures.

As an enterprise class database, PostgreSQL boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, table-spaces, asynchronous replication, nested transactions (savepoints), on-line/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance. It supports international character sets, multi-byte character encodings(Unicode), and it is locale-aware for sorting, case-sensitivity, and formatting. It is highly scalable both for the amount of data that it can manage and for the number of concurrent users that it can accommodate. Table 3.4 reports some of the main feature of the postgreSQL database system.

Alone it is not able to manage geographic referenced data. What makes the PostgreSQL able to accept geographic data is the PostGIS extension. PostGIS is a spatial database extender for the PostgreSQL object-relation which adds the support for geographic objects allowing location queries executable in SQL. Moreover PostGIS also offers the support for GiST-based R-Tree spatial indexes, and functions for analysis/processing of GIS objects. The PostGIS geography type provides a native support for spatial features represented on "geographic" coordinates, i.e., the geodetic coordinates.

The plane is the basis for the PostGIS geometry type, where the shortest path between two points can be approximated as a straight line. Operations

[11]Atomicity, Consistency, Isolation, e Durability

Limit	**Value**
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

Table 3.4: Database Features

on geometries take place through Cartesian mathematics, and straight line vectors, by which we can extract areas, distances, lengths, intersections, etc. Contrary the basis for the PostGIS geographic type is the sphere, where the shortest path between two points corresponds to a great circle arc. On the other hands, operations performed on geographies, such as areas, distances, lengths, intersections, etc, adopt a more complicated maths.

The GIS objects supported by PostGIS are a superset of the "Simple Features" defined by the OpenGIS Consortium(OGC). PostGIS extends the standard with support for 3DZ,3DM, and 4D coordinates. In particular the spatial objects assume the geometric forms[12]:

- **POINT**: it represents a single location on the Earth, represented by a single coordinate(including either 2-,3- or 4-dimensions). Points describe objects whose shape is not a relevant information for the project purpose.

- **LINESTRING**: it simply represents a path connecting locations taking the form of an ordered series of two or more points joined together.

- **POLYGON**: it describes an area. The outer boundary of the polygon assumes the form of a ring. The ring is a line-string both closed and simple. Holes within the polygon are rings exactly as the outer ones.

Features expressed in Table 3.4 in conjunction with the PostGIS extension make the PostgreSQL service the perfect base tools set for the MEP project. The ability to manage geographic data with ease and flexibility represents a uniqueness in the geographic big data field, no longer encountered in other database system.

[12] http://revenant.ca/www/postgis/workshop/geometries.html

3.5 Python Programming Language & Libraries

Python is a dynamic, object oriented language adopted in many software development area, and it offers a support for the integration with other programming language. Its design philosophy emphasizes code readability, and its simple syntax allows the concepts expressions in much fewer line of code with respect to other languages, such as Java or C++. This language provides constructs to enable clear programs writing, on both small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative, and functional programming as well as procedural styles. It includes a dynamic type system, and automatic memory management together with a large and comprehensive standard library. Python interpreters accommodates for the major operative systems, allowing code exchangeability between different systems.

Two Python's versions are particularly diffused in scientific research area, i.e., the 2.7 and the 3.5 version; this two versions have minimal differences between each other. Regarding the realization of the MEP project server side, we adopt the 3.5 version.

The *"Psycopg PostgreSQL adapter"* for Python language accounts for heavily multi-threaded applications, able to execute large numbers of concurrent INSERTs or UPDATEs. It is a wrapper for the libq[13] library which is the official PostgreSQL client library, representing a bridge between the Python's data elaboration and the PostgreSQL database system.

Python handles the management of large data collections through the *"Pandas library"*, a set of tools devised for an efficient and fast data manipulation. It is an open source, BSD-licensed library providing high-performance, easy-to-use data structures specialized for the Python programming language. A fast and efficient DataFrame object for data manipulation with integrated indexing, providing instruments for reading and writing data into different file formats, such as text files, CSV, Microsoft Excel, SQL databases, and the fast HDF5 format.

It also introduces an intelligent data alignment, and integrated handling of missing data, along with functions for table reshaping, slicing, fancy indexing, and sub-setting. Columns insertion and deletion is very easy. The traditional database functions for the data aggregation represent an interesting and powerful engine ready to use. All this features joined with the performance optimization, with critical code paths written in Cython[14] or

[13] https://www.postgresql.org/docs/current/static/libpq.html

[14] Cython is an optimizing static compiler for both the Python programming language and the extended Cython programming language. It makes writing C extensions for

C, makes the Pandas framework the ideal tool set for data management required by our project.

In computer science field, the thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, a part of the operating system. Each system differs from each other regarding the threads' implementation, but in most cases a thread can be regarded as a component of a process. Many threads can exist within a single process, hence their execution takes place concurrently. Unlike processes, threads share resources like memory[21].

The Python language, as most of the newest programming language, includes a threading library providing a set of tools for the multi-threading management.

The *Watchdog* library monitors file system's events triggered when a change occurs.

Python's incredible flexibility and spread leads to the theoretical possibility to model any kind of algorithm coming from many different fields. It comes with an interesting tools set allowing the prototyping of articulate problems, giving the programmer the possibility to express in details all the relevant information. The use of the Python programming language in computational neuroscience has been growing steadily over the past few years. The maturation of two important open source projects, i.e., the scientific libraries "*NumPy*" and "*SciPy*", provides access to a large collection of scientific functions.

"*Modular toolkit for Data Processing(MDP)*" is a Python data processing framework, that offers a collection of supervised and unsupervised learning algorithms, and data processing units. Single units can be combined together producing a data processing flow or more complex structures such as feed-forward network architectures. Its modular paradigm offers the positive basis for the implementation of new algorithms. At the same time the base of available algorithms in the library gets continuously updated and expanded, including in particular:

- Signal processing methods such as Principal Component Analysis, Independent Component Analysis, Slow Feature Analysis.
- A manifold learning methods such as (Hessian) Locally Linear Embedding.
- Several classifiers.

Python as easy as Python itself. http://cython.org/

- Probabilistic methods.

- Data pre-processing methods.

Given a set of input data, MDP takes care of training and executing all network's nodes in the correct order, and pass the intermediate data between them. In such a way, the definition of complex algorithm as a series of simpler data processing steps becomes more reliable. Efficient computations in terms of speed and memory consumption comes from the elaboration of data batches, whose computation can be also parallelized using the specifically designed parallel sub-package, a parallel implementation of the concepts of nodes and flows.

3.6 The MEP Project Workflow

MEP project's basic implementation requires the development of methods and procedures coming from different computer science fields, in particular the ones described throughout this chapter.

The definition of the proper workflow has been one of the crucial points we encountered during the development of the project's block structure. An articulate workflow precisely defines the sequence of the different element into the project and it combines them together in order to reach the final purpose of building accessibility maps of the different cities.

One of commonly adopted software engineering's design principle is the **Separation of Concerns(SoC)** principle. The idea behind the SoC principle is to split an application into distinct sections, where each section addresses a separate concern. The Model-View-Controller (MVC) pattern is nothing more than the SoC principle applied to Object Oriented Programming(OOP). The name of this pattern comes from the three main components used to split a software application: the *model*, the *view*, and the *controller*. MVC better classifies under "architectural" rather than "design" patterns, whose differences refers only to the broader scope of the latter with respect to the former.

The *model* represents the core element, containing the whole knowledge represented by logic, data, state, and rules of a specific application. The *view* builds a visual representation of the *model* instead. It only displays the data without handle them. The task of connecting the *view* to the *model* relies on the *controller* which operates a link between them, performing all the connection operations[20]. This particular framework is widely adopted in the development of many applications due to its high flexibility

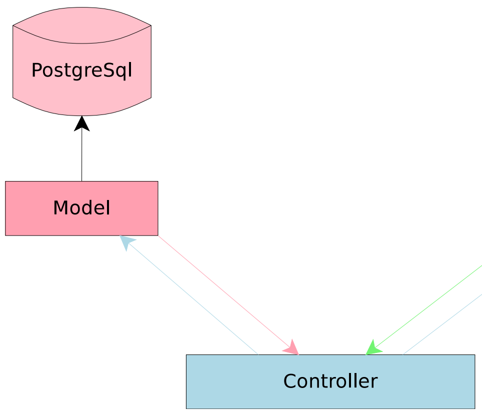


Figure 3.10: The MVC design pattern general block schema.

and scalability. Figure 3.10 represents the MVC design pattern in a basic and intuitive block schema, representing its main elements.

MVC is a generic and useful design pattern which brings large benefits such as:

- The separation between the view and the model allows to build the user interface (UI) without interfering with other parts of the program. Despite this is not the case of the MEP project, this represents an important code design strategy.
- Because of the decoupling between view and model, each part can be easily modified/extended without affecting the others.
- The maintenance of each part is easier due to the clear definitions of roles and responsibilities.

The MEP Project workflow starts by instantiating the *controller*, which implements the communication between the different elements of the program. Implicit and explicit elaboration of data conveyed by the two applications can be considered to start exactly at this moment. The *controller* further subdivides into three elements:

1. An *observer*, whose aim is the discovery of new elements added to the MEP-Traces acquisitions' space.
2. A *cluster* manager, whose principal purpose is the clustering of the user-uploaded paths.
3. One or more *worker*(s), responsible for the sequential/parallel load into the MEP-Database of the paths recorded by the users.

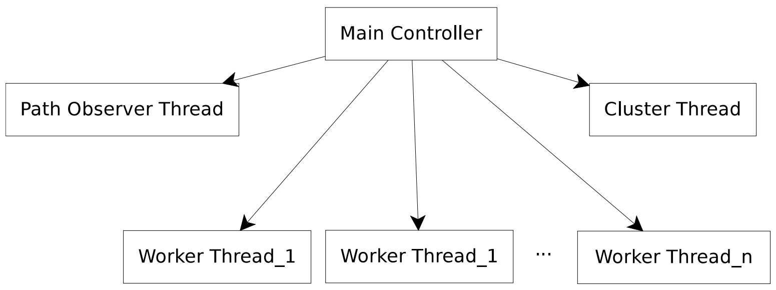


Figure 3.11: The Main Controller subdivision into the different elements cooperating into the scientific computation.

Figure 3.11 highlights the different elements cooperating into the project.

We develop the first two elements into two separate threads, kept alive for the whole program life. A Worker element only activates when the queue contains at least one element.

At the initialization of the *controller*, the program scans the storage and it pushes on the queue the elements discovered into the MEP-Server's memory: it preserves only those elements which have not been elaborated yet, while it neglects those that have already been processed. This task is accomplished through a database table ("*mep-folder-table*") containing a list of the elaborated folder. This shrewdness allows to keep track of the folders already computed. A table represents a smarter solution with respect to the backup files because of its persistence, easy management, and update facility. The *mep-folder-table* combines together the folder name, the user, and the date in which the algorithm processed it.

After the initial scan, the *controller* launches the *observer*. It monitors the directory where the users upload their acquisitions through the MEP-Traces application. If a change occurs, it provides an immediate response pushing the new element onto the processing queue. Each time a *.zip* appears into the "observed" directory, the *observer* updates the queue with this new element.

The *controller* also launches the *worker* element(s). If the queue contains at least one element, the *controller* pops the first one and it assigns the new item to a *worker* which immediately begins the elaboration. The general processing schema can be found in Figure 3.12. It represents the logical sequence of operations leading to the final accessibility definition of the areas.

The *observer* pushes new elements as soon as they appear into the server's folder, without waiting for their complete upload. This forces us to perform a check of the element's completeness before proceeding. The

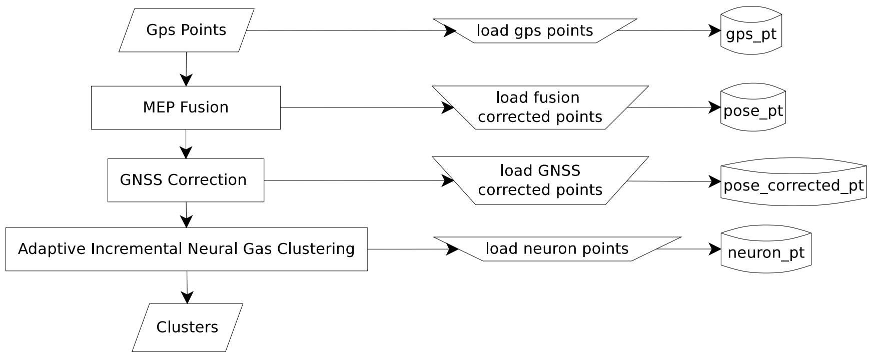


Figure 3.12: Worker general process tasks.

check takes place by looking at the element's dimension according to a fixed time lapse every 2 seconds: if no changes occur in the meanwhile, the file can be considered completely uploaded. Another check verifies the correctness of the *.zip* file in order to avoid to waste time and computational effort on corrupted or useless items. Bad structured elements remain in the central memory, but do not participate to the algorithm's computations. Once again the *mep-folder-table* keeps track of those elements by modifying the corresponding flag.

The process starts popping an item from the queue. The *controller* assigns this new element to a *worker*, whose identifier corresponds to the timestamp referred to the instant in which the *worker* has been created. This temporal mark is important in order to distinguish one *worker* from another.

At the beginning, each *worker* dumps data concerning the building and highway of previous operations. This solution has been adopted to make the GNSS correction faster, because too many rows in these tables stuck the machine. For this reason the time lapse between the moment in which the *worker* loads data into the database, and the one in which it use them for the GNSS correction must be serialized, in order to prevent other *workers* to flush information that are still valid. A simple lock mechanism prevents the dump of these data before they are actually used preventing the loss of information.

3.6.1 Decompression Phase

The decompression of the *.zip* file produces four files:

1. *MEP_DEVICE_INFO.txt*

2. *MEP_MOTION_INFO.txt*

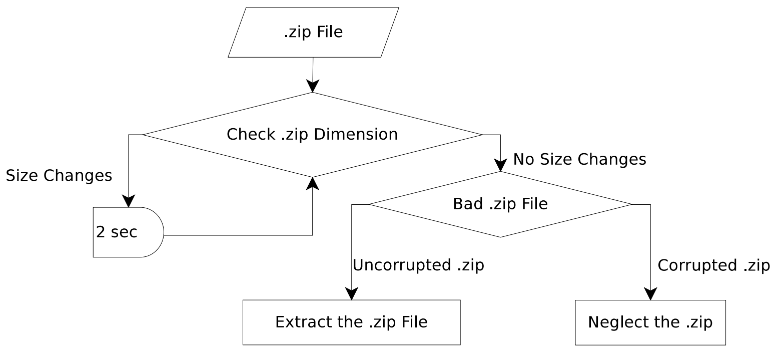


Figure 3.13: The .zip file check procedure.

3. *MEP_NMEA_INFO.txt*

4. *MEP_POSITION_INFO.txt*

presented in Section 3.1.

The *.zips*' files corruptions occur for several reasons, i.e., a network lack or the occurrence of a client-side compression's failure. Figure 3.13 presents the block procedure of the *.zip* file check. Each file, contained into the original *.zip* folder, refers to a particular step of the processing and carries out a precise task.

We get a rough approximation of path's behavior by loading the points contained into the *MEP_POSITION_INFO* file. It collects data coming from the direct acquisition of the GPS sensor without further processing. This initial data collection represents the scale over which we can evaluate improvements produced throughout the different cleaning steps.

The *MEP_DEVICE_INFO* file contains the set of information related to the device which supplied the acquisition. It collects the device model, the Android version name, the kernel signature, and other useful facts. The MEP project adopts those information in order to perform statistics about MEP-Traces', i.e., memory and battery consumed during the acquisition, the location and duration of the recording, etc.

3.6.2 Building and Highway Tables Population Phase

The GNSS correction in Section 3.3.3 requires the *MEP_NMEA_INFO* file as input. This file contains the relevant satellite's position information regarding the user's path.

Figure 3.14 highlights the fundamental steps of the processing. Each point builds a series of non-overlapping square windows, used by the script to

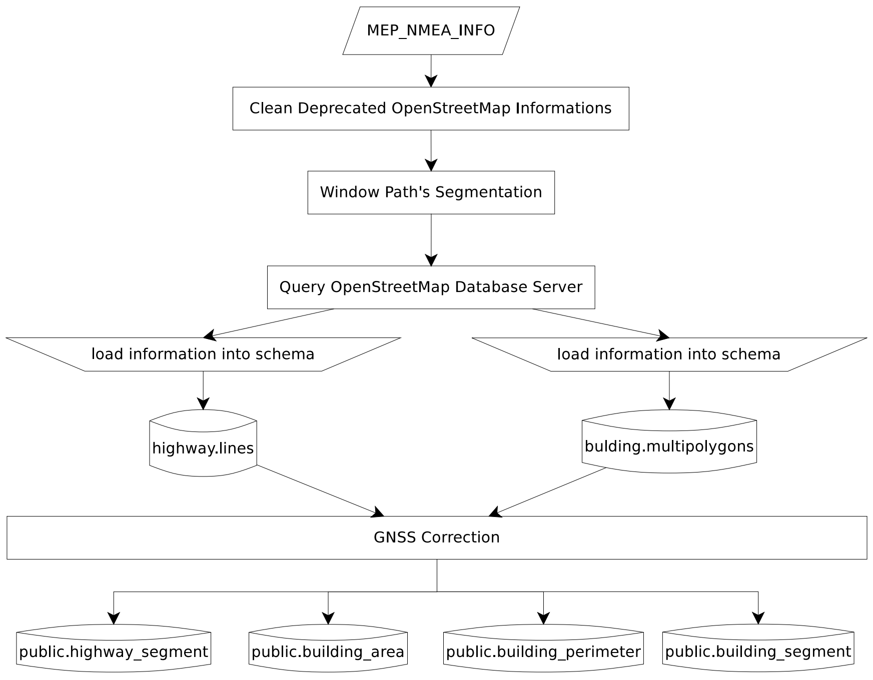

MEP_NMEA_INFO
Clean Deprecated OpenStreetMap Informations
Window Path's Segmentation
Query OpenStreetMap Database Server
load information into schema
load information into schema
highway.lines
bulding.multipolygons
GNSS Correction
public.highway_segment
public.building_area
public.building_perimeter
public.building_segment

Figure 3.14: Population of the Building and Highway tables required by the GNSS correction to operate. The MEP_NMEA_INFO file contains the points' sequence corresponding to the recorded path. This can be used to create a series of overlapping windows. Each window corresponds to a query to the OSM server to retrieve building and highway falling within its area. The server's response contains all the demanded information, which can then be loaded locally into the related MEP-Server schema. Subsequently to the schema, also the related tables have to be updated.

query the OpenStreetMap database through a request message. The server replies with an *.xml* file that contains information about the interested area.

The OGR tool perform the task of loading the OSM responses onto the MEP-Database, inserting the newest buildings and highway information, through the "*ogr2ogr*" command, contained into the GDAL library. OGR is a toolkit for the manipulation of spatial data, comprehensive of a wide set of features, under the Geospatial Data Abstraction Library (GDAL)[15].

3.6.3 MEP-Fusion Correction Phase

The *MEP_MOTION_INFO* and *MEP_NMEA_INFO* files jointly provide the input to the MEP-Fusion correction algorithm(presented in Section 3.2).

Even though the Android system provides common functionalities for the sensors' management within applications through the Android Sensor Framework (depicted throughout Section 3.1.4), each device presents a different combination of chips, which results into slightly changing sensors' acquisitions. For this reason, we introduced strategies to standardize results even though inputs data exhibit heterogeneous characteristics.

In particular recent devices, such as the Huawei Nexus 6P, mount sensors characterized by an high sampling rate frequencies in the order of KHz[31]. Computations of accelerometer, gyroscope, and magnetometer's data coming from this type of devices is not trivial due to the high quantity of rows produced during the acquisition. The Android Sensor Framework records the most recent event occurrence into the a dedicated buffer, that will be definitely written down into the MEP_MOTION_INFO file as soon as the system frees the necessary resources to perform the task. The system writes samples row by row marking them with the precise instant, i.e., the timestamp, in which the code function occurs identifying the sensors' evolution during time.

Devices characterized by an high frequencies fill the buffer with a huge number of records which could also assume the same temporal marker at the moment of writing into the *MEP_MOTION_INFO* file. In other words, this causes two or more samples to assume coincident timestamp. This is an undesired situation which may lead to computational problems.

A proper "down-sample" operation takes the situation back to a more reasonable one for the algorithm, so that it can handle more easily sensors'

[15]GDAL is a computer software library for the read and write of raster and vector geospatial data formats. It presents a single abstract data model to the calling application for several supported formats and also implements a variety of command line interface utilities for data translation and processing. OGR works under the GDAL library providing additional tools for simple features vector graphics data[24]

data, avoiding errors at the moment to reconstruct the paths.

Other recent devices, such as the Motorola Moto X Play Edition(XT1562) or the Motorola Moto G $3rd$ Generation(XT1541), do not come with the gyroscope included into the sensors set. Despite the gyroscope's lack does not influence the user experience in every day usage, it represents a serious issue from the MEP-Fusion point of view. Even if the orientation's information can be extracted from other sensors, the Fusion algorithm explicitly demands the gyroscope sensor's data.

Samsung devices shows a bias affecting the magnetometer sensor instead. This particular bias demands for a dedicated procedure: retrieved data have to be conformed to other devices present on the market.

All this problems require the adoption of some practical solutions in order to make the MEP-Fusion algorithm operate in suitable conditions on heterogeneous data.

The process begins extracting from the *MEP_MOTION_INFO* file the different sensors' records, grouping them according to their types:

- *TYPE_GYROSCOPE*
- *TYPE_GYROSCOPE_UNCALIBRATED*
- *TYPE_ACCELEROMETER*
- *TYPE_MAGNETOMETER*
- *TYPE_MAGNETOMETER_UNCALIBRATED*

The introduction of a fake artificial gyroscope addresses the problems related with the absence of such sensor's type. Fake data get mixed with the real ones reproducing plausible temporal marker positions.

The separation of the *MEP_MOTION_INFO* file, grouping the sensors according to their type, leaves the possibility to handle missing values.

Once we get different data frame containing the related sensor's data, procedures of missing values handling operate the first simplest cleaning of the data collection.

Samples falling on the same timestamp are initially spread over the range between the last available and the current timestamp. If the explained procedure does not solve the problem then we can fuse the rows performing the mean between them in order to reach the ideal situation of non overlapping timestamps. The whole process depicts in Figure 3.15, which represents the block schema of the pre-processing adaptations required by the MEP-Fusion tool.

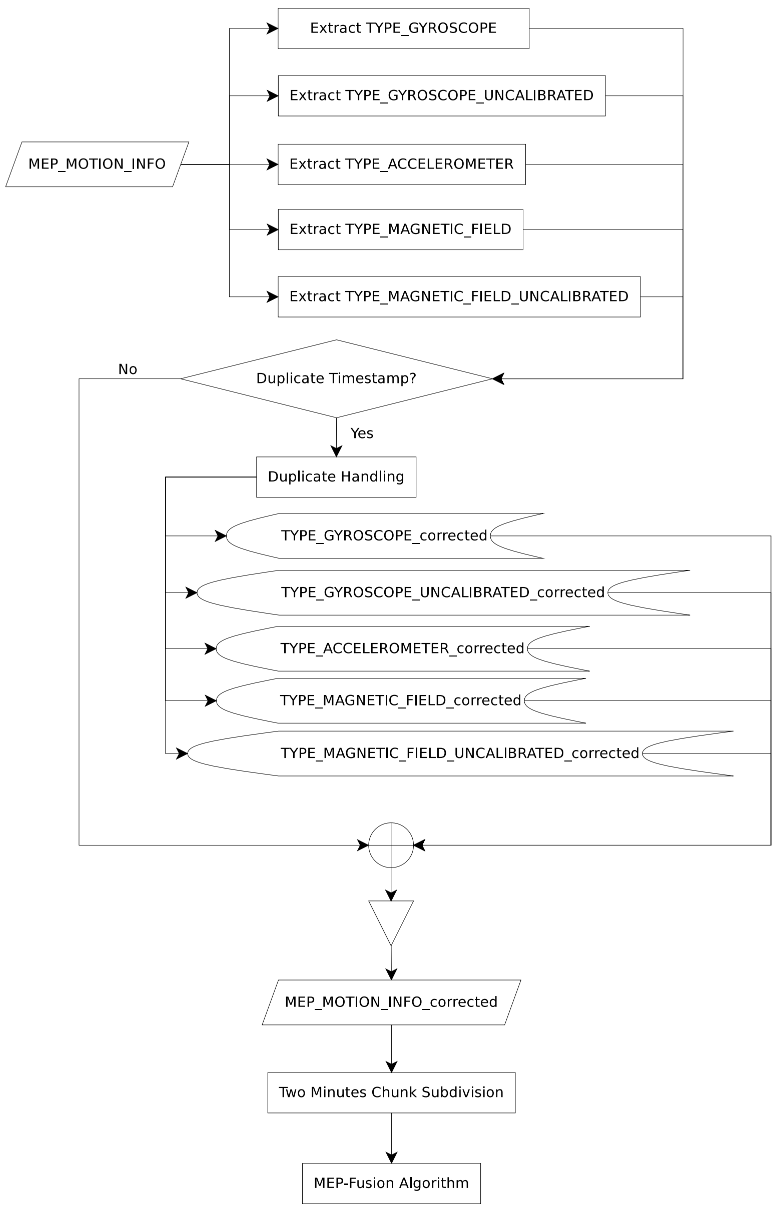

Extract TYPE_GYROSCOPE
Extract TYPE_GYROSCOPE_UNCALIBRATED
MEP_MOTION_INFO
Extract TYPE_ACCELEROMETER
Extract TYPE_MAGNETIC_FIELD
Extract TYPE_MAGNETIC_FIELD_UNCALIBRATED
No
Duplicate Timestamp?
Yes
Duplicate Handling
TYPE_GYROSCOPE_corrected
TYPE_GYROSCOPE_UNCALIBRATED_corrected
TYPE_ACCELEROMETER_corrected
TYPE_MAGNETIC_FIELD_corrected
TYPE_MAGNETIC_FIELD_UNCALIBRATED_corrected
MEP_MOTION_INFO_corrected
Two Minutes Chunk Subdivision
MEP-Fusion Algorithm

Figure 3.15: The MEP-Fusion pre-processing schema. This schema tackle many different problems, focusing on the purpose to get heterogeneous data as input to the MEP-Clusterpath algorithm. The results of this step get loaded into the MEP-Database in the pose_pt table and activate the GNSS correction trigger.

The final step of the pre-cleaning processing provides the rejoin of the different sensor's subsets into a single data frame, which subsequently divides into two minutes length data collection. Dividing the complete collection into a smaller sub-data set makes the elaboration of the MEP-Fusion algorithm faster and easier.

The final results obtained by the application of the sensors' fusion algorithm over the different windows collect together into a single output file, i.e., "*TPoseSE3(W)*". The timestamps perfectly indicate uniquely the row position with respect to the others, thus the total path reconstruction it's just a trivial operation. The *TPoseSE3(W)* file is also used for the upload of the newly obtained information onto the *pose_pt* table in the MEP-Database, in order to keep track of this cleaning step.

3.6.4 GNSS Correction Phase

The insertion of the Fusion corrected points makes the GNSS correction trigger start the correction. At this moment, all the information about the building and highway positions (obtained addressing the OSM database) cooperates to project building-overlapping points into more a reasonable location along the proximity of the building's perimeter.

The general operative workflow of the GNSS correction is outlined in Figure 3.16. It shows the sequence of the steps adapted specifically for the MEP-Workflow, and the operations required to correct tables within the MEP-Database. Those tables will be later supplied to the GNSS algorithm to actuate the correction. The different tables keeps track of all the different elements according to the geometric types defined by the PostGis framework, explained in details in Section 3.4.

MEP-Clusterpath algorithm elaborations take advantage of the cleaning operations giving a chance to operate in suitable conditions. The quality level of results strictly depends on the data collection. For this reason avoiding, or at least reducing, the effect of outliers and missing values represent a benefit for the algorithm computation. The MEP-Clusterpath algorithm will be described in details in Chapter 4.

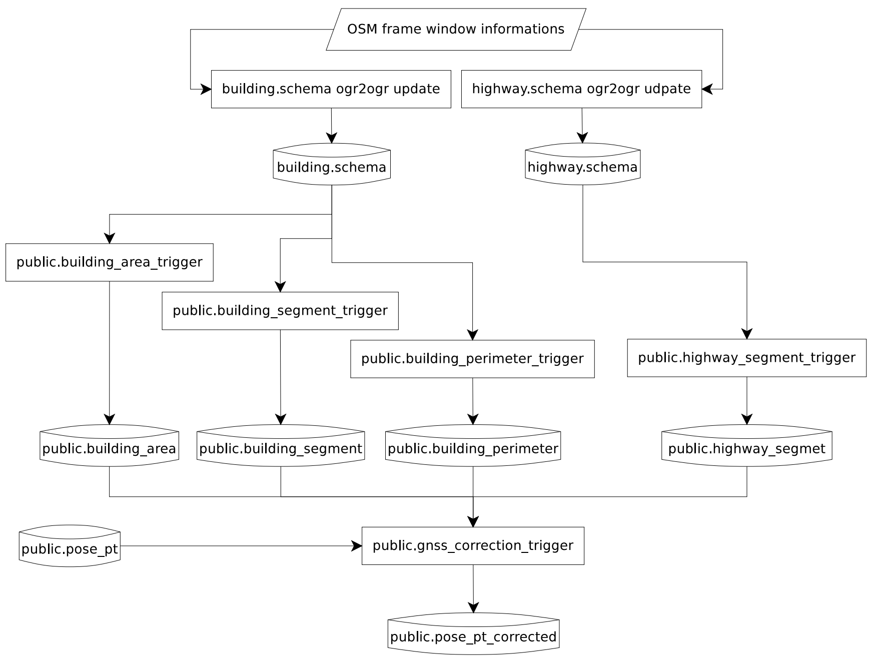

OSM frame window informations
building.schema ogr2ogr update
highway.schema ogr2ogr udpate
building.schema
highway.schema
public.building_area_trigger
public.building_segment_trigger
public.building_perimeter_trigger
public.highway_segment_trigger
public.building_area
public.building_segment
public.building_perimeter
public.highway_segmet
public.pose_pt
public.gnss_correction_trigger
public.pose_pt_corrected

Figure 3.16: GNSS correction workflow adopted in order to keep updated the tables public.building_area, public.building_perimeter, public.building_segment, public.highway_segment. These tables contains all the data required for the application of the GNSS "outside-building" projection. We decide here to represent also the population of the four tables underlying the GNSS correction although the correction take place when the new points from the Fusion correction come into the MEP-Database.

Chapter 4

The MEP-Clusterpath Algorithm

"Camminare è un mezzo, non un fine."

Angela Gambirasio

4.1 Algorithm Preparation and Definitions

The MEP-Clusterpath algorithm has the main purpose in the perfect integration with the different elements of the MEP project workflow. Computational and memory resources are shared between different algorithms. Hence each element has to interact with each other, and the complete tool set must not exceed the machine's possibilities. A suitable trade off between results' accuracy and lightweight computations represents the final purpose of the optimization process. Optimal policies' definition and tuning of parameters must take care of all the different aspects allowing to free resources as soon as they are no more required.

The *cluster* thread is activated at the beginning of the program and it is maintained alive for the whole program life, cooperating with the database which stores the queue of the cities that have to be processed. The MEP-Database contains a properly designed "clustering queue" table (precisely the *cluster_queue* table) in order to keep track of the cities already clustered, and the ones over which the clustering has to be repeated. This happens when newly uploaded points change accessibility definition of a particular area introducing new data to the collection. A priority index between the different cities contained into the MEP-Database describes the level of relevance that we want to attribute to a particular element. Precisely, it resides into a column of the *cluster_queue* table.

The thread synchronizes a local queue at its initialization and periodically updates it by monitoring changes of the corresponding table. The table get updated every time a user uploads a new acquisition. This process is totally automatic thanks to a trigger on the *raw-gps-point* table:

- If the acquisition's location already constitutes a row into the *cluster_queue* table and it has already been processed, then the boolean *cluster-repeat* flag gets the *TRUE* value, by meaning that the clustering process must be repeated as soon as possible, including the newer data.

- If the acquisition's location does not corresponds to any row of the *cluster_queue* table, then it gets included within the queue with the boolean *cluster* parameter equal to *FALSE*, and the *cluster-repeat* flag equal to *TRUE*.

The queue collects the complete information's set of the processing phase. On the other hand the local queue stores only those elements whose accessibility level has not been defined yet or the ones that display outdated information due to new uploads modifying the accessibility of a particular area.

The composition of the data set starts with the pop of an item from the local queue: this new element goes into a query that retrieves all the points related to a city from the MEP-Database. Results that come out from the query build the complete data set related to that city, and they summarize the most updated information available for the computation.

At this point two possible choices rise up:

- Randomize the data set, i.e, shuffling the collection.

- Leave the data set as is to reflect the order relations which characterize the paths' points.

MEP-Clusterpath strictly relies on the Adaptive Incremental Growing Neural Gas Network(*AING*) algorithm discussed in Section 2.5.2, with slightly modifications in order to adapt to geospatial/geographic datum. The choice of the AING algorithm, as starting point of the development of the MEP-Clusterpath algorithm, arises from its high flexibility and degree of adaptation upon many different classes of data.

4.1.1 Nodes, Arc and Graph

We expect the learning process to produce a graph. We know from graph theory that a graph is a structure amounting to a set of objects in which some pairs are related according to a particular characteristic. In particular these objects corresponds to mathematical abstractions, i.e., nodes, and elements' pairs connection denoted through the term edge [35].

Python's *MDP* library contains basic definitions and implementations of the concept of *Node*, *Edge*, and *Graph*. For the MEP-Clusterpath implementation we devise two extensions for the *Node*'s concept producing two nodes' type:

- *Neuron*: contains in-coming and out-coming *arcs* to neighboring neurons and out-coming *arcs* to samples. They rise as the meaningful representative points for the accessibility representation of a given area.
- *Sample*: contains only one in-coming edge from the neuron they are related to. As opposite to *Neuron*, they cannot display any out-going *arc*.

Two more *arc*'s types can be defined:

- *NeuronEdge*: they realize the neuron-to-neuron link, where the corresponding age parameter keeps track of the edge's seniority.
- *SampleEdge*: they realize the neuron-to-sample connection. As opposite to *NeuronEdges*, they do not contain the concept of edge's seniority, but they are only adopted to keep track of the samples referring to a particular node.

We adopt this separate edges' specialization into different objects in order to take advantage of the two elements' different characteristics. Within the graph, nodes can be connected to each other relating different groups of samples with neighboring ones. During the training phase the graph representing the accessibility topology keeps updated through the merging process, responsible for the removal of old-fashioned links. In particular, the algorithm frequently drops and creates *NeuronEdges* throughout the different iteration, while it preserves the *SampleEdges* instead. In fact, they can only be caught to some other node during the merging procedure, but the total number of *Samples* supplied to the algorithm equals the number of *Samples* contained into the final graph. As a consequence of the specialization of this two new concepts, also the graph's one must be updated and expanded accordingly in order to contain these new elements, producing the *ExtendedGraph* object.

4.2 The MEP-Clusterpath Algorithm

4.2.1 Initialization

Algorithm's initialization starts by defining an empty graph g. The graph can be initialized with two points randomly chosen from the input data frame that corresponds to the first two nodes of the g. When no point's pair constitutes the initialization, two random defined nodes join g. Other parameters may optionally participate to the algorithm's initial phase and their definition is fundamental to reach a suitable operative point in reasonable time. If they are not externally assigned, they will assume default values.

In particular they are:

- max_{age}, i.e, the maximum age that an edge can assume, before the graph merging/reshaping procedure eliminates it.
- max_{nodes}, i.e., the maximum number of nodes characterizing the built of the output topology.
- d, i.e., the mean distance estimate of all the existing neurons with respect to the center of-mass of the observed data-points.
- d_{min}, i.e., the minimum distance which can be tolerated between two different nodes.

Tuning of max_{age} parameter reflects the desired relevance which we want to attribute to younger edges with respect to elder ones: each time an input point hooks to a neuron, the edges emanating from the neuron itself increase the corresponding age value. The age can increase up to a maximum value, and when it overpasses this superimposed threshold, the network gets reshaped removing the edge which gets through max_{age}. This expedient makes the network always preferring newer information with respect to old-fashioned ones.

The d parameter reflects the mean distance of the existing neurons with respect to the center of-mass of the observed data-points and affects the initial radius for the merging process to fuse neighboring neurons. Each time the algorithm exceeds the maximum number of neurons, the parameter k is updated incrementally by summing d. Once k has been incremented, the merging process can be repeated. In such a way the probability to fuse two nearest neurons becomes more reliable, and the neuron's number can level off to suitable values. Tuning the d parameter is crucial: low values correspond to slow merge between item pairs, while high ones produce too rapid merging between elements' pairs. A suitable trade-off can be found considering the half length of two successive points into a path as initial value for d, so that it assumes values strictly based on the real data set.

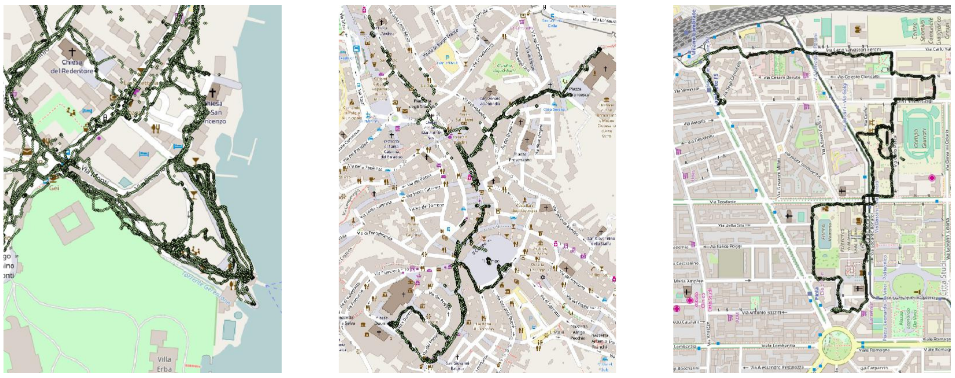
The max_{nodes} parameter fixes the maximum number of nodes that the algorithm can employ to produce the output topology. Tuning this value is not trivial since it fixes the learning's "deepness" that we want to obtain in order to fit memory constraints or speed up computation. According to 2.5.2 the definition of the max_{nodes} cannot be easily found a priori through a simple formula.

Our adaptation of the AING algorithm over geospatial data mitigates the effect of max_{nodes} by introducing a new parameter, i.e., d_{min}. It prevents the creation of neurons which exhibit too close coordinates with respect to an existing one, i.e., its node-to-node distance is below an externally defined threshold represented by the parameter d_{min}. We use it to tune the minimum neurons' density that we want to obtain for the final network's representation. Some considerations about the distance between two sequential samples within a path help us to set this newly introduced parameter. Our purpose is to discover the general accessibility topology of a city adopting as input the users' path, which exhibit an elongated structure by definition. Graphs with nodes containing a single sample produce no knowledge. Our work concentrates on the extraction of meaningful representative points describing the users' collected routes. An approximate number for the d_{min} parameter can be computed as twice the distance between two sequential samples. With this adaptation, the topology's representation exhibits a minimum density or granularity for the displacement of the neurons over the city.

4.2.2 The MEP-Clusterpath Input Set

The input set for the MEP-Clusterpath algorithm is composed by a two columns matrix containing latitude and longitude information. Elements' pairs represent the coordinates in space of the GPS points.

In particular those points comes out from the different cleaning processes depicted throughout Chapter 3 and they represent the suitable collection for the MEP-Clusterpath. We can refer to these points as the minimum data-block information, while others, such as timestamp, user name or email, do not add further details. Final results are user-independent: they fuse data coming from multiple users, acquisitions and days. Fused information turn into a loss of the possibility to link the single point to the user who recorded it.


(a) Input collection obtained after the GNSS correction for the city of Cernobbio.

(b) Input collection obtained after the GNSS correction for the city of Siena.

(c) Input collection obtained after the GNSS correction for the city of Milano.

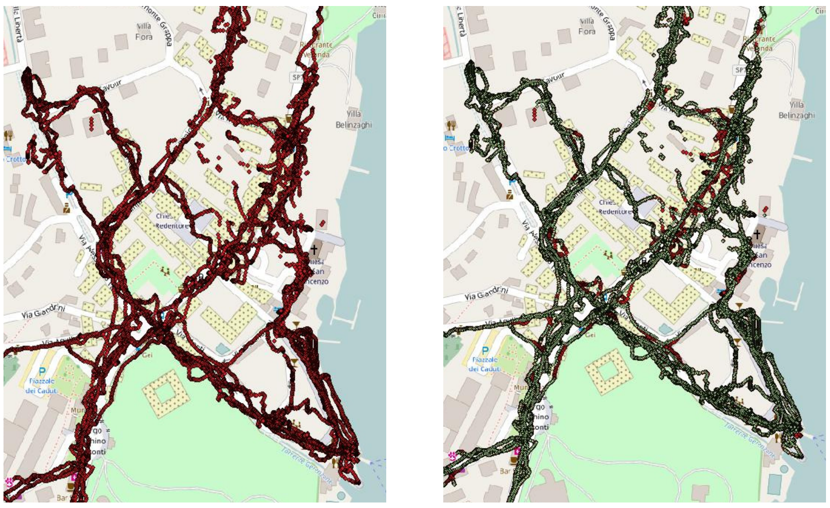
Figure 4.1: Three different real data collection referred to three different cities, i.e., Cernobbio, Siena and Milano respectively.

A further subdivision of the training set into smaller chunks makes the elaboration easier, allowing incremental and multiple training phases. This particular feature comes from the *MDP* python library presented in Section 3.5. This subsets collect two thousand points from the original collection and they represent the input training set of the MEP-Clusterpath algorithm.

An example of input collection for the MEP-Clusterpath algorithm appears in Figures 4.1(a), 4.1(b) and 4.1(c). As we can see, the points collections over different cities are highly different between each others regarding data heterogeneity and homogeneity. The algorithm has to be then perfectly adapted over different accessibility models, in order to characterize the user's experience along the recorded paths. Data quality also depends on the device which perform the acquisition. For this reason we expect that paths collected over different devices also exhibit different characteristics.

Further disturbing elements for the acquisition of GPS data are the urban canyon. Many different ancient cities exhibit narrow buildings and tight streets for military defense reasons. Nowadays we have other structures, i.e., skyscrapers, characterized by huge glass facade, which constitutes urban canyons. Urban canyons are the base unit through which the complex urban morphology can be described and they appear to be particularly difficult for the GPS sensors due to the inadequate satellite's exposure.

Last but not least, we do also expect that not all users care about the quality of data that they acquire. The project's requirements demand for


(a) Raw data referred to Cernobbio.

(b) GNSS correction over raw data referred to Cernobbio.

Figure 4.2: Real examples of raw and cleaned data representation of Cernobbio coming from the MEP-Database.

a continuous movement while the user records the GPS and sensors' data. When a user stops during the acquisition, we see that a measurement noise affects the GPS samples which start turning around a position, producing then a mass of points. This phenomena leads to a sub-optimal elaboration.

Figure 4.2(a) shows the data as they come from the MEP-Traces application with no further processing, while Figure 4.2(b) clearly represents how the GNSS process cleans the input data. In such a way the could become an acceptable training set for the MEP-Clusterpath algorithm.

4.2.3 The MEP-Clusterpath Training Phase

The training phase of the algorithm loops on each chunk inspecting element by element the complete collection: for each point x, the algorithm scans the graph g and extracts the first and the second nearest nodes, with respect to the x's coordinates. Each node stores locally its adaptive threshold obtained in past steps. The thresholds' computation relies on the set of samples (X_y) and the set of neighboring nodes (V_y), according to the formula:

$$T_y = \frac{\sum_{e \in X_y} dist(y,e) + \sum_{e \in V_y} |X_e| \times dist(y,e)}{|X_y| + \sum_{e \in V_y} |X_e|} \tag{4.1}$$

Graphical representation of threshold's concept can be found in Figure 4.3. Thresholds' value represent a trade-off between information locally

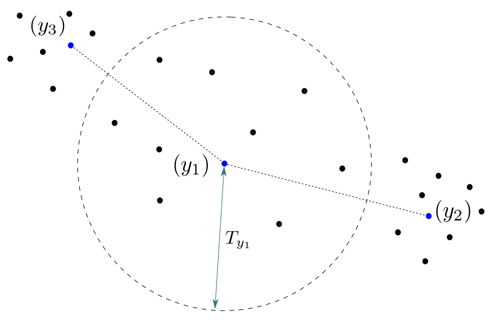


Figure 4.3: MEP-Clusterpath threshold definition reflects the one of the base AING algorithm.

assigned to a neuron with respect to information contained into neighboring neurons. Its design reflects a dynamic evolution of the network representing the data supplied to the algorithm and the topology of the neurons.

Considering the thresholds corresponding to the two closest nodes, three possible cases arise considering the coordinates of the input point with respect to the training sample (presented in 2.5.2):

- 1st *case.* The input point x is far enough from n_0, i.e., $dist(x, n_0) > Tn_0$, and $dist(x, n_0) > d_{min}$: a new neuron joins the graph g at the sample's coordinates, and a new sample joins the new neuron's sample-list. Finally the thresholds of the new node is updated accordingly.
- 2nd *case.* The input point x is close enough to n_0 but far from n_1, i.e., $dist(x, n_1) > Tn_1$, and $dist(x, n_0) > d_{min}$: a new neuron joins the graph g at the sample's coordinates, and a new link connects the new node to n_0. Finally the sample gets into the new neuron's sample-list, and the threshold of the node n_0 is updated accordingly.
- 3rd *case.* The input point x is close enough to both n_0 and n_1 nodes, i.e., $dist(x, n_1) < Tn_1, Tn_0$: move n_0 and its neighboring neurons toward the input point x. Increase the age of n_0's emanating edges. Then connect n_0 and n_1 with a new edge whose age is equal to 0. Finally remove old edges from the graph if any, and update both n_0 and n_1's thresholds accordingly.

Figure 4.4 graphically explores the three possible situations which arise considering the position of the input point x. Algorithm 1 presents the pseudo-code of the training step of the MEP-Clusterpath algorithm.

The base version of the AING algorithm (presented in Section 2.5.2) avoids an indefinite growth of the graph's dimension by shrinking the nodes'

Algorithm 1: MEP-Clusterpath Training Algorithm (max_{nodes}, d, d_{min})

```
Initialize graph $g$ with the two points input vector;
$k = 0$;
while Some input data-point $x$ remain unread do
    Get current data-point $x$;
    Let $n_0$ and $n_1$ be the two nearest neurons from $x$ in $g$;
    Retrieve the thresholds $Tn_0$ and $Tn_1$ stored into the neuron's
      local space;
    if $dist(x, n_0) > Tn_0 \wedge dist(x, n_0) > d_{min}$ then
        $g \leftarrow g \cup \{n_{new}/w_{n_{new}} = x\}$;
        Add the sample to the $y_{new}$ samples' space;
    else
        if $dist(x, n_1) > Tn_1 \wedge dist(x, n_0) > d_{min}$ then
            $g \leftarrow g \cup \{n_{new}/w_{n_{new}} = x\}$;
            Connect $n_{new}$ to $n_0$ by an edge of age 0;
            Add the sample to the $n_{new}$ samples' space;
        else
            Increase the age of edges emanating from $n_0$;
            Let $\epsilon_b = \frac{1}{|X_{n_0}|}$, $\epsilon_m = \frac{1}{100 \times |X_{n_0}|}$ ;
            $w_{n_0} + = \epsilon_b \times (x - w_{n_0})$;
            $w_{n_n} + = \epsilon_n \times (x - w_{n_n})$, $\forall n_n \in V_{n_0}$;
            Connect $n_0$ to $n_1$ by an edge of age 0, reset if it already
              exists;
            Remove old edges from $g$ if any;
        end
        while Number of neurons in $g > max\_neurons$ do
            $g \leftarrow Merging(k, g)$;
            $k = k + d$;
        end
    end
end
```

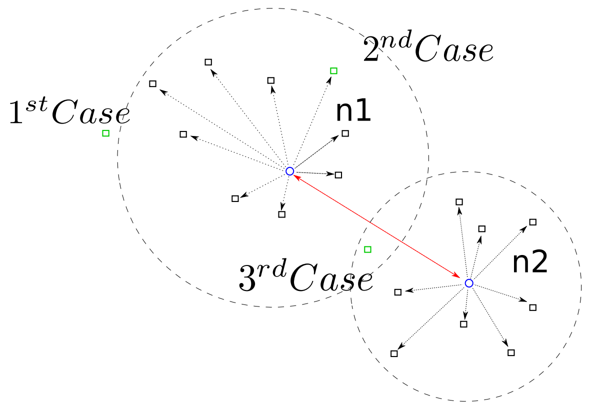


Figure 4.4: Green square points correspond to the three possible cases based on the distance of the point x with respect to first and second nearest nodes. The black squares are the samples already assigned to a neuron node (blue circle points).

number within the bound expressed through the max_{nodes} parameter. When the number of nodes overpass this threshold, a sub-call to the merging procedure reshapes the graph itself fusing nodes together according to a specifically design probability (Section 4.2.4).

The adaptation presented in Algorithm 1 works under the assumption that neurons' geographical coordinates should not display an arbitrary closeness between each other. New neurons born if and only if their coordinates preserve the minimum required distance imposed to the training algorithm. Adopting this changes, our procedure rarely overpasses the max_{nodes} bound, reducing then also the merging procedure sub-algorithm calls. Reduction of merging invocations, turns at the same time into a lighter computation of the final output topology and realizes also the desire to perform a dimensional scaling. Compact representations of the graph produce more interpretable models while describing all the relevant aspects through meaningful parameters.

We evaluate a suitable value for the parameter d_{min} estimating the distance between two succeeding points within the same path, obtaining the order of magnitude of this new parameter. Our considerations of this parameter rely upon the fact that too close neurons represent the accessibility of almost the same area. Validation of the results aims to get representations of wide areas through simple points structures collecting all the accessibility information conveyed by the inputs.

MEP-Clusterpath adaptation of the original AING algorithm prevents the birth of nodes' groups toward restricted areas, mitigating the negative effects produced on the learning by the overfitting phenomena. In this way the desired output topology's granularity can be tuned modifying the d_{min} parameter in order to obtain densities within an imposed range.

4.2.4 The Merging Procedure

The merging process starts by creating a new empty graph $\tilde{g}$. Two randomly picked nodes get out from g's neuron list and come into the new graph $\tilde{g}$. These two nodes accomplish to the same role of the initialization vector supplied as input to the AING algorithm. Successively a new link in $\tilde{g}$ connects the new nodes with an age equal to 0.

In order to keep track of the node's generation process throughout the different reshaping sub-procedure, when a node moves from g to $\tilde{g}$, also its samples have to migrate with him. In such a way, the number of samples within the output topology is equal to the number of inputs supplied to the algorithm.

The merging algorithm threats the g's neurons list as it were an input data set: after the initialization, it explores the nodes $y \in g$ and retrieves the first and second nearest neurons in $\tilde{g}$, corresponding to $\tilde{n}_0$ and $\tilde{n}_1$. Each node y contains the set of related samples (i.e., X_y) and the list of the edges to neighboring neurons (i.e., V_y). A neuron y is considered to be far enough from its nearest neuron $\tilde{n}_0$ when the probability of the nodes y y not to be assigned to $\tilde{n}_0$ is large enough. This probability can be expressed according to the formula:

$$P_{y,\tilde{g}} = \frac{|X_y| \times dist(y, \tilde{y})}{k} \tag{4.2}$$

Equation 4.2 computes the probabilities of the two nearest nodes as:

$$\begin{aligned} P_{y,\tilde{n}_0} &= \tfrac{|X_y| \times dist(y,\tilde{n}_0)}{k} \\ P_{y,\tilde{n}_1} &= \tfrac{|X_y| \times dist(y,\tilde{n}_1)}{k}. \end{aligned} \tag{4.3}$$

The merging algorithm picks up a random number from a uniform distribution in 0 and 1. Once again there are three possibilities:

1. If the random number is lesser or equal than $P_{y,\tilde{n}_0}$: add the node coming from g to $\tilde{g}$, migrate the samples to the new node in $\tilde{g}$ and update its threshold.

2. If the random number is lesser or equal than $P_{y,\tilde{n}_1}$: add the node coming from g to $\tilde{g}$ and link the new node with $\tilde{n}_0$. Migrate the samples from the node in g to the new one in $\tilde{g}$ and update both thresholds of the new node and $\tilde{n}_0$.

3. If both two previous conditions are not satisfied, y and its related samples are included into the samples' space of $\tilde{n}_0$. Age of the edges emanating from $\tilde{n}_0$ increments, and its neighborhood is adapted toward y. A new/reset link connects $\tilde{n}_0$ to $\tilde{n}_1$. Finally outdated edges get removed.

When neurons' number starts to increase, the merging process immediately reshapes the graph by fusing nodes which are considered to be close enough each other. The final purpose of the merging procedure is the representation of large areas identified by similar level of accessibility with a small number of meaningful elements.

High costs comes from the necessity to build a new graph (i.e., $\tilde{g}$) starting from scratch at each sub-call to the merging procedure. Even if the merging process is optional and can be avoided by setting the *max_neurons* parameter to $+\infty$, without merging, the graph's dimension would exhibit larger values. For this reason the *max_neurons* parameter fixes the upper bound on the elements' number of the final topology in order to match the memory constraints.

Once the merging process ends, $\tilde{g}$ substitutes to the outdated graph g. The sub-call returns to the training which can continue the elaboration over the input samples. The merging procedure pseudo-code can be found in Algorithm 2.

Particularly relevant tuning parameters are the max_{nodes} and the d parameters. Higher values for max_{nodes} make the merging process more sporadic, i.e., invocations to the merging subprocess will becomes fewer with respect to cases in which max_{nodes} assumes higher values. Avoiding the merging process would produce lighter computations and an easier resource management.

On the other hands, too low values for max_{nodes} parameter make the merging process works harder on the graph trying to fuse together an high number of nodes, in order satisfy dimensionality constraints. This intensive operation demands for a huge amount of time and resources in terms of power and memory consumption.

The definition of the d parameter is fundamental because it affects the computation of the probabilities over which the merging process fuses together different neurons. Before the invocation of the merging algorithm, the parameter k updates through the formula:

$$k = k + d, \tag{4.4}$$

affecting the probability $P_{y,\tilde{y}} = \frac{|X_y| \times dist(y,\tilde{y})}{k}$. The assignment of the d parameter has to consider a trade-off between too long merging times and a

Algorithm 2: AING Merging Algorithm (k, g)

```
Initialize graph $\tilde{g}$ with two neurons chosen randomly from $g$;
while Some neuron $y$ in the $g$'s neurons list remains unread; do
    Get current neuron node $y$;
    Let $\tilde{n}_0$ and $\tilde{n}_1$ be the two nearest neurons from $y$ in $\tilde{g}$;
    Let $d_1 = dist(w_y, w_{\tilde{n}_0})$ and $d_2 = dist(w_y, w_{\tilde{n}_1})$;
    Retrieve $X_{\tilde{n}_0}$ and $V_{\tilde{n}_0}$;
    if $random_{uniform}([0,1]) < min\left(\frac{n_y \times d_1}{k}, 1\right)$ then
        $\tilde{g} \leftarrow \tilde{g} \cup \{n_{new}/w_{n_{new}} = x\}$;
        Migrate the samples from the old node in $g$ to the new node in $\tilde{g}$;
    else
        if $random_{uniform}([0,1]) < min\left(\frac{n_y \times d_2}{k}, 1\right)$ then
            $\tilde{g} \leftarrow \tilde{g} \cup \{n_{new}/w_{n_{new}} = y\}$;
            Connect $n_{new}$ to $\tilde{n}_0$ by an edge of age 0;
            Migrate the samples from the old node in $g$ to the new node in $\tilde{g}$;
        else
            Migrate the samples from the old node in $g$ to the new node in $\tilde{g}$;
            Increase the age of edges emanating from $\tilde{n}_0$;
            Let $\epsilon_b = \frac{1}{|X_{\tilde{n}_0}|}$, $\epsilon_n = \frac{1}{100 \times |X_{\tilde{n}_0}|}$ ;
            $w_{\tilde{n}_0} += \epsilon_b \times (x - w_{\tilde{n}_0})$;
            $w_{\tilde{n}_n} += \epsilon_n \times (x - w_{\tilde{n}_n})$, $\forall n_n \in V_{\tilde{n}_0}$;
            Connect $\tilde{n}_0$ to $\tilde{n}_1$ by an edge of age 0, reset if it already exists;
            Remove old edges from $\tilde{g}$ if any;
        end
    end
end
return $\tilde{g}$;
```

trivial underfitting of the network's topology.

4.3 Neurons' Heat Maps Representation

Heat maps are intuitive graphical data representations, where individual values contained into a matrix can be described with colors. They've been adopted in many different fields and their huge spread comes from their incredible readability[37].

By representing each neurons' samples-set as a Gaussian shaped curve, we can extrapolate heat maps containing information about the cities' accessibility.

In mathematics, a Gaussian function is a function of the form:

$$f(x) = a \cdot e^{-\frac{(x-b)^2}{2c^2}}; \tag{4.5}$$

with arbitrary real constants a, b and c. The graph of a Gaussian exhibits a characteristic symmetric "bell curve" shape. The parameter a defines the height of the curve's peak, b is the center position of the peak and c (the standard deviation) controls the width of the "bell". Gaussian functions are widely adopted in different fields such as statistics (the normal distributions), signal processing (Gaussian filters), image processing (two-dimensional Gaussian performs Gaussian blurs), and in mathematics (solve heat equations and diffusion equations and to define the Weierstrass transform). Gaussian functions arise by composing the exponential function with a concave quadratic function. In two dimensions, the exponential e power assumes any negative-definite quadratic form. Consequently, the level sets of the Gaussian will always be ellipses. In particular a two-dimensional Gaussian function can assume the form

$$f(x,y) = A \cdot exp\left(-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(x-y_0)^2}{2\sigma_y^2}\right)\right). \tag{4.6}$$

The coefficient A refers to the peak's amplitude. The parameters x_0, y_0 correspond to the peak's position, and σ_x,σ_y to the x and y blob's spreads. The plot of Expression 4.6 assumes the shape observed in Figure 4.5.

In general, a two dimensional elliptical Gaussian function assumes the form:

$$f(x,y) = A \cdot exp(-(a(x-x_0)^2 - 2b(x-x_0)(y-y_0) + c(y-y_0)^2)) \tag{4.7}$$

where the matrix

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} \tag{4.8}$$

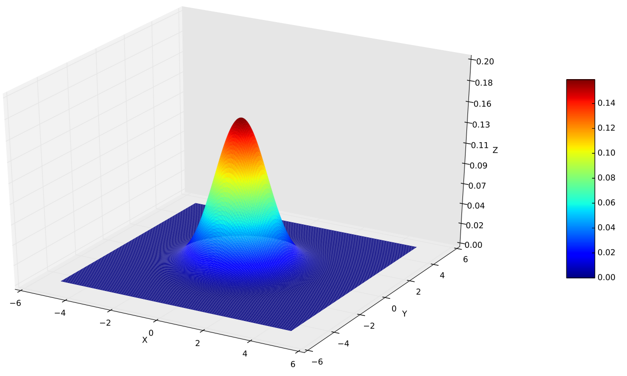


Figure 4.5: Two-dimensional representation of a generic Gaussian function.

is positive-definite.

For the general form of the Equation 4.7 we have to set parameter a, b and c as:

$$a = \frac{cos^2(\theta)}{2\sigma_x^2} + \frac{sin^2(\theta)}{2\sigma_y^2}. \tag{4.9}$$

$$b = -\frac{sin(2\theta)}{4\sigma_x^2} + \frac{sin(2\theta)}{4\sigma_y^2}. \tag{4.10}$$

$$c = \frac{sin^2(\theta)}{2\sigma_x^2} + \frac{cos^2(\theta)}{2\sigma_y^2}. \tag{4.11}$$

in order to effectively produce a rotation within the blobs.

The computation of the orientation of the blobs is not trivial. We decide to adopt the simple linear regression method to estimate the slope of the straight line representing the main blob's orientation. Each neuron stores locally the list of the samples that generated a node. This set can be used as input for the simple linear regression method.

Using such definitions the different areas can be represented through blob-smoothed elements whose connection gives us the general accessibility level of a city.

Algorithm 3 presents the pseudo code of algorithm adopted for the computation of the complete heat maps layer of a city. It receives as input the lists of neurons and obstacles. In particular, the neurons' list contains information related to the blobs' position and orientation, while obstacles' one collects the coordinates and the level of inaccessibility reported by the

Algorithm 3: HeatMapGridBuilder(*neurons, obstacles*)

```
Compute the boundary of the grid;
mlt = the minimum between neurons and obstacles' latitude;
Mlt = the maximum between neurons and obstacles' latitude;
mlg = the minimum between neurons and obstacles' longitude;
Mlg = the maximum between neurons and obstacles' longitude;
lat axis = linspace(mlt, Mlt, resolution);
long axis = linspace(mlg, Mlg, resolution);
neuron layer = len(lat axis)×len(long axis) grid of zero elements ;
obstacle layer = len(lat axis)×len(long axis) grid of zero elements ;
while Some neuron remains unread do
    if $\sigma_{lat}! = 0$ and $\sigma_{long}! = 0$ then
        Compute the Gaussian function: gauss(long axis, lat axis,
          $\mu_{long}$, $\mu_{lat}$, $\sigma_{long}$, $\sigma_{lat}$, $\theta$, accessibility level);
    end
    Sum the current Gaussian to the neuron layer;
end
while Some obstacle remains unread do
    Compute the Gaussian function: gauss(long axis, lat axis, $\mu_{long}$,
      $\mu_{lat}$, $\sigma_{long}$, $\sigma_{lat}$, $\theta$, accessibility level);
    Sum the current Gaussian to the obstacle layer;
end
elevation = merge the two layer, i.e., neuron layer, obstacle layer;
elevation = Radial Basis Function Interpolation of the elevation layer;
return the complete elevation grid;
```

user. As output, the algorithm produces a grid representing the accessibility surface of the desired city.

Chapter 5

Experimental Results and Validation

"Cè chi nella vita si fa strada a pugni e chi a parole: a volte è difficile capire la differenza."

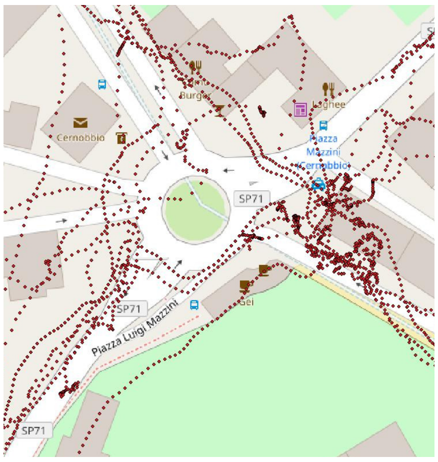
Angela Gambirasio

In this chapter we summarize the results obtained from the MEP-Clusterpath algorithm using the data collection conveyed by MEP-App and MEP-Traces. As depicted throughout Chapters 3 and 4, the MEP-Clusterpath performs an elaboration of the implicit and explicit data contained into the MEP-Database server. Starting from raw data, the algorithm adopts different cleaning techniques improving the quality of data supplied as input of the MEP-Clusterpath algorithm.

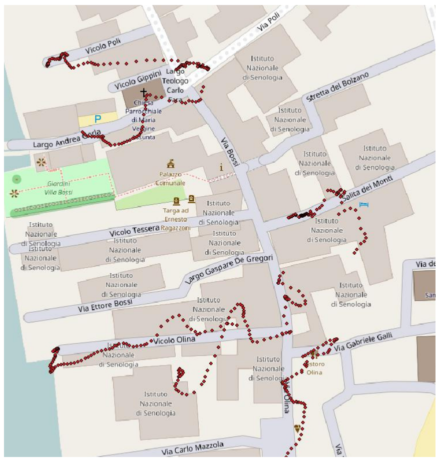
Looking at Figure 5.1 its easy to see two possible patterns coming from Cernobbio (Figure 5.1(a)) and Orta San Giulio (Figure 5.1(b)), over which we decide to perform several experimentations.

Many clustering algorithms reveal to be very sensitive to the presence of outliers within the training set. For this reasons we decide to apply a variety of cleaning tasks, described throughout Chapter 3 in order to reach a suitable data set. The reduction of the effects produced by the presence of such disturbing elements is a fundamental step to obtain heterogeneous and relevant results. MEP-Clusterpath algorithm (described in Chapter 4), as well as AING(Section 2.5.2), is very sensitive to the presence of outliers which would lead the algorithm to produce neurons over coordinates which have not been ever recorded.

In Figures 5.2(a) and 5.2(b) the red points correspond to the raw GPS data collected by users, while in green we represent the points corrected

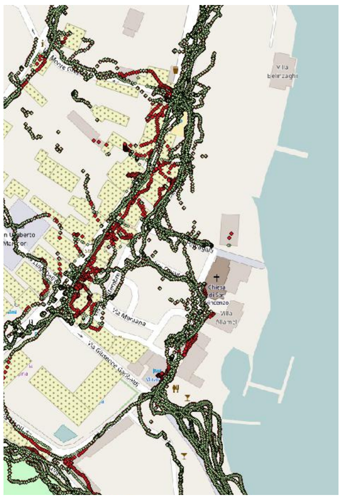


(a) Raw data(red points) referred to Cernobbio.

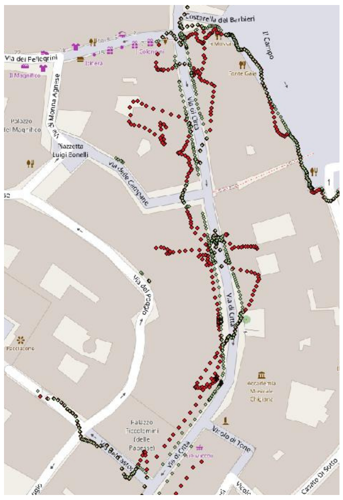


(b) Raw data(red points) referred to Orta San Giulio.

Figure 5.1: Real examples of raw and cleaned data representation of Cernobbio and Orta San Giulio coming from the MEP-Database.



(a) Raw (red points) and GNSS-corrected(green points) data comparison from Cernobbio's urban canyon.



(b) Raw (red points) and GNSS-corrected(green points) data comparison from Siena's urban canyon.

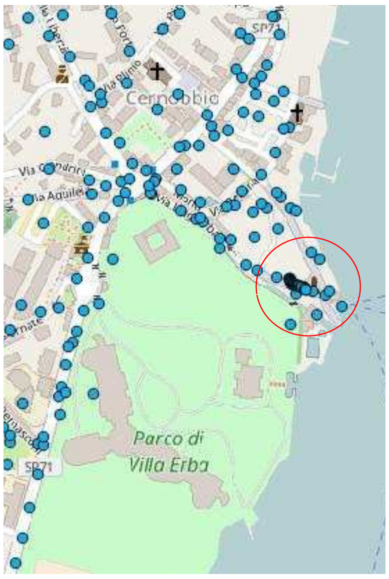
Figure 5.2: The urban canyon problem around cities. In panel 5.2(a) Cernobbio recorded data; in panel 5.2(b) an information' slice from the city of Siena.

through the GNSS procedure. From these two Figures we can see how the urban canyons negatively affect the recording of GPS information due to the bouncing of the signal from one wall to another. Without a cleaning preprocessing, the application of the MEP-Clusterpath algorithm would lead to totally undesired results. Clusters' resulting definitions based on data affected by large measurements errors lead to detrimental results, because they do not reflect the real path traveled by the user. Under the hypothesis that implicit collections represent the accessible paths within a city, the presence of outliers would brings neurons to move toward positions that do not match the real accessibility of an area.

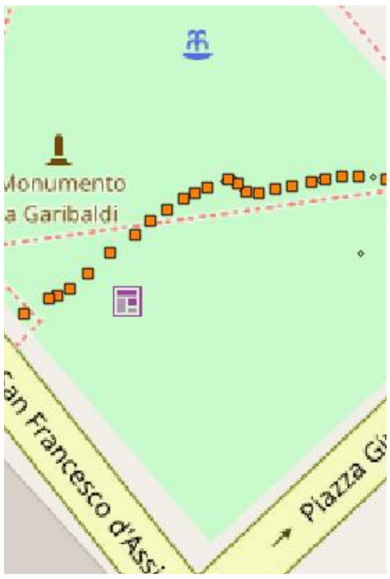
Considering all this aspects together, we adapt the MEP-Clusterpath algorithm over different topological situations. A suitable learning over problematic areas can be obtained through the proper tuning of the algorithm's parameter. Our procedure reveal to be resilient over urban canyon, mitigating the effects of different issues encountered during the development of the thesis. The tuning of parameters should be also performed over different cities in order to validate the set up.

The application of the base version of the AING algorithm leads to results in Figure 5.3. The blue points represent the neurons displacement over the area of Cernobbio. After the elaboration, we notice that some nodes tend to group towards restricted areas. This neurons' concentration is somehow related to the overfitting phenomena, because more than one neuron brings similar information with respect to neighboring ones. Wider areas are then represented with a single point describing the level of accessibility. In particular, the red circled area (Figure 5.3(a)) highlights the overcoming of this undesired phenomena. Again in Figure 5.3(b) the AING algorithm identifies each input point as representative for the underlying area. This method explains too closely the input samples bringing to the overfitting phenomena (Section 2.2). In those cases it works too hard on the supplied data set, trying to extract information which are not really present inside the collection: the method stops to fit the real model and starts to explain the intrinsic error of the training set instead. In our case the nodes will contain no samples, except for the one that generated the neuron itself. Representing each input point with a node is detrimental from the dimensional scaling point of view: in those cases the algorithm does not perform dimensional scaling at all.

The MEP-Clusterpath's design and tuning tackles this problems in order to avoid concentrations of neurons within tight areas. During the training phase, the algorithm creates new nodes if and only if they preserve the minimum required granularity for the output topology as described in Section

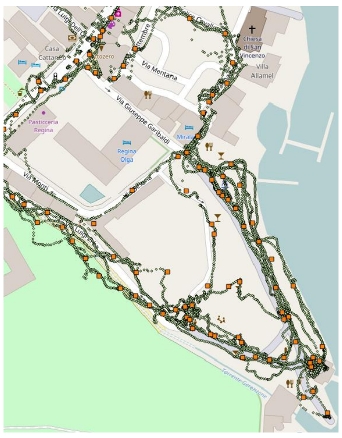


(a) Slice coming from Cernobbio where the base version of the AING algorithm produces groups of undesired neurons within restricted areas.



(b) Slice coming the clustering results obtained from the application of the AING algorithm over inputs of Novara.

Figure 5.3: Both slices represent two cases in which the AING algorithm produced sub-optimal results. Into the left panel, blue points show the displacement of the neurons Cernobbio. In particular, the red circle points out the area in which the AING algorithm produced sub-optimal results. Into the right panel orange squares indicate the neurons' position over Novara. Once again the algorithm fails to fuse information while creating the accessibility topology of the city.


(a) Green points refers to the raw GPS data collected by users along their routes. Orange points represent the neurons' displacement over the area of Cernobbio.

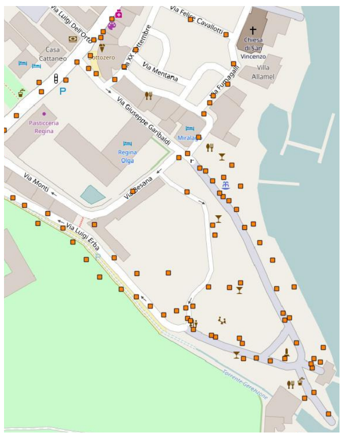

(b) The orange points offer a representation of the neurons coordinates.

Figure 5.4: Neurons' displacement in Cernobbio.

4.2.3. On the other hand the intervention of the merging procedure operates a reshape of the graph, fusing together nodes when their number exceeds the maximum granted to the algorithm to build the output topology. These two different competing forces operate a trade-off between the accuracy of the model an the computational effort required to produce it.

Figure 5.4 presents an example of how the fix of the minimum distance parameter affects the final topology of a city. In Figure 5.3(a), we can see how neurons (the orange square points) spread over Cernobbio city. This is totally different with respect to displacement in Figure 5.3. The underlying green points correspond to the training input supplied to the algorithm in order to perform the learning of the accessibility's topology, while the orange points are the nodes obtained as output of the MEP-Clusterpath algorithm. Thanks to these representations, we can see how the neurons effectively learn the different routes collected by users.

Into Section 5.1 we will compare the performances obtained by testing different setup of the algorithm offering also a detailed comparison between the MEP-Clusterpath and the AING base version of the method.

5.1 Algorithms' Performances

In this Section we present the analysis of the AING and the MEP-Clusterpath algorithms performances, recorded during our tests. Several runs of the algorithm with different settings produce results collected in Tables 5.1 and 5.2. In particular, we decide to focus our tests over a randomly extracted subset of the complete collection related to Cernobbio. Our data set is composed by 31432 points and joins data that come out from different users and devices, cleaned through the Fusion and GNSS algorithms.

Table 5.1: Results obtained by testing all the possible settings configuration. Each setting can be uniquely identified through a corresponding identification (id) number. The column time refers to the time laps spent by the algorithm to produce the results. The d parameter indicates the mean distance from all the existing neurons with respect to the center of-mass of the observed data-points. The d_{min} parameter collects the different values for the minimum granularity tested with our algorithm. The max_{age} parameter fixes an upper bound seniority for the nodes links; by setting it to $+\infty$, the algorithm will never eliminate outdated links. Nodes' number column collects the final number of neurons produced by the algorithm.

id	time	d	max_{nodes}	max_{age}	neurons num
1	19:17	5.00e-06	400	∞	385
2	30:23	5.00e-06	300	∞	293
3	27:00	5.00e-06	500	∞	469
4	06:51	5.00e-05	300	∞	237
5	07:52	5.00e-05	400	∞	374
6	08:12	5.00e-05	500	∞	385
7	06:52	5.00e-05	300	10	231
8	06:45	5.00e-05	400	20	328
9	09:41	5.00e-05	500	50	396
10	19:38	5.00e-06	300	10	289
11	17:17	5.00e-06	400	20	388
12	19:35	5.00e-06	500	50	478
13	17:00	5.00e-06	400	100	383

Table 5.1 collects what we obtained by several runs of the AING algorithm over the sub data set. The max_{nodes} parameter calibrates the trade-off between memory constraints and computational speed. As explained in Section 2.5.2, its calibration is not trivial, and an automatic procedure to obtain this fundamental parameter does not exist. For this reason we decide to test three different values: 300, 400, and 500.

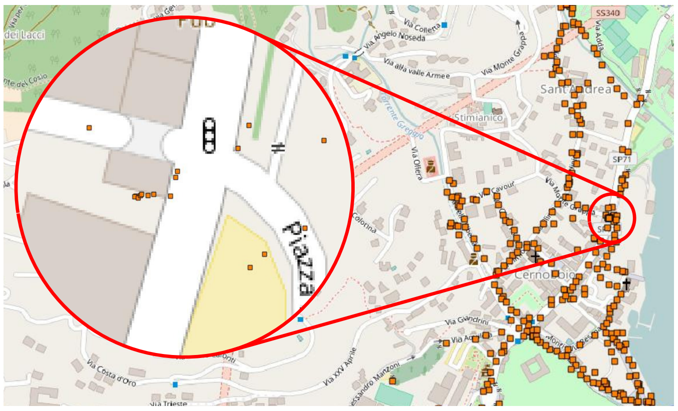


Figure 5.5: Detailed zoom of the AING's weakness. Resulting nodes that characterize a graph, may sometimes distribute over tight areas, leading to an undesired situation of overlapping information.

Each of this three possible values, produces different neurons' concentration over the area. Adopting 300, we force too many nodes to group together, producing overly complex shapes which cannot be easily represented through Gaussian blobs. On the other hand, too high values of max_{nodes} lead to the uprise of the phenomena described beforehand: the algorithm does not fuse enough information producing an unsatisfying topology. Values chose close to 500 produce a suitable compromise between shapes complexity and generalization, required for the subsequent heat map representation of the Gaussian blobs (Section 5.2).

Focusing on d values in Table 5.1, we can state that values near 5.00e−5 make the algorithm faster with respect to the ones close to 5.00e−6. This can be explained appealing to the fact that AING has to keep the nodes' number within a predefined threshold. When they overpass this threshold, the algorithm invokes the merging procedure to bring them back inside the limits. The d parameter affects the merging procedure modifying the probabilities that establish which nodes have to be fused together.

The max_age parameter seems not to have any influence onto the results. However we prefer to attribute more relevance to the latest samples coming from the input data set.

Figure 5.5 shows a zoom of the area interested by too dense nodes concentration. This situation appears in almost all settings gathered in Table 5.1 and lead to an undesired situation, solved with the MEP-Clusterpath algorithm.

In Table 5.2 we reported the results obtained by the application of the MEP-Clusterpath algorithm. Despite low computational times, rows with

Table 5.2: Collection of results obtained by varying the parameters set of the MEP-Clusterpath algorithm. We refer to each test through an id number which uniquely indicates the parameters set that we employ for the test itself. All the described features reflect the ones it Table 5.1 and they can be used to operate comparisons between different settings for each algorithm, or evaluations between tunings of the two different algorithms.

id	time	d	d_min	max_{nodes}	max_age	neuron's num
1	03:16	5.00e-05	5.00e-04	300	inf	287
2	01:36	5.00e-05	5.00e-04	400	inf	291
3	01:13	5.00e-05	5.00e-04	500	inf	292
4	04:11	5.00e-05	5.00e-04	500	inf	287
5	03:04	5.00e-05	5.00e-05	300	inf	297
6	18:02	5.00e-06	5.00e-05	300	inf	300
7	16:49	5.00e-06	5.00e-05	500	inf	500
8	21:03	5.00e-06	5.00e-05	400	inf	396
9	20:40	5.00e-06	5.00e-05	500	10	497
10	20:04	5.00e-06	5.00e-05	500	20	495
11	21:40	5.00e-06	5.00e-05	400	20	397
12	39:56	5.00e-06	5.00e-05	300	20	299
13	30:36	5.00e-06	5.00e-05	300	10	299
14	20:44	5.00e-06	5.00e-05	400	100	395

id between 1 to 5 exhibit results characterized by poor quality: with high values of d and d_{min} the algorithm fuses many neurons together which should be kept separated instead. From Figure 5.6(a) we can see the uprise of this problem, where each single neuron tries to represent a complex blob shape which cannot be easily described through mean and variance. Too complex geometrical representations produce no interesting results when we express them through simpler shapes (such as Gaussian blobs). To overcome this problem, we decide to concentrate upon topologies obtained with higher number of nodes, which are supposed to represent smaller and simpler areas with higher accuracy. With this expedient, we expect the algorithm to build elementary Gaussian blobs over the topology learned above a specific city.

Table 5.2 underlines how parameters setting influences the computational time required by the algorithm to produce the neurons' displacement over the map. Lower values of d increase the number of invocations to the merging algorithm, implying longer learning times. At each call to the merging subprocess, low d values make the algorithm fuse few nodes together per time. As a consequence, to preserve the maximum number of nodes imposed by max_{nodes}, the algorithm has to repeat the merging procedure more then once.

The introduction of the d_{min} parameter forces the algorithm to observe a minimum distance between neighboring neurons. Adopting too large values (d_{min} = 5.00e−4) produces no relevant results. For this reason we consider lower values (d_{min} = 5.00e−5) which produce the expected results from the topology's minimum granularity point of view.

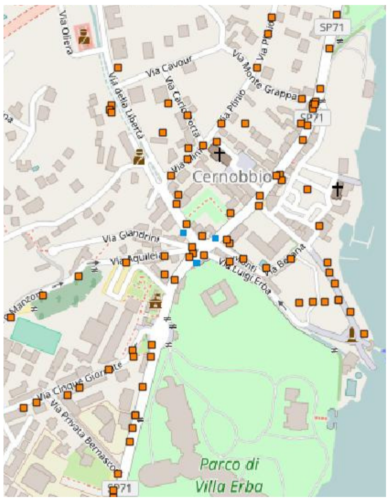
Regarding the max_{age} parameter we change it from $+\infty$ to a value between 10 to 100. We recall that the max_{age} parameter is responsible for the trade-off between newly arrived information and the previous one. With a value equal to $+\infty$, the algorithm includes all the information into the computation excluding the aged-edges removal procedure. On the other hand integer values different from $+\infty$ make the algorithm to forget the outdated links. Considering the topologies obtained from different tests, we noticed that different values assigned to the max_{age} parameter do no produce relevant changes from one output to another. We decide anyway to give more relevance to newer information, and, for this reason, we define it equal to 10 in order to reflect this intention.

Pictures in Figure 5.6 display some results obtained with different settings of the parameters according to Table 5.2. We can see how the variation of the parameters setting corresponds also to a different neurons' displacement all over the examined area. In Images 5.6(c) and 5.6(d), we decide to leave also the GNSS corrected data as reference of the input data set, in order to compare results between different settings. In particular image 5.6(a) represents a situation in which the quality of the results is not satisfying our expectations. We prefer displacements as the ones obtained in Figures 5.6(b), 5.6(c) and 5.6(d) instead.

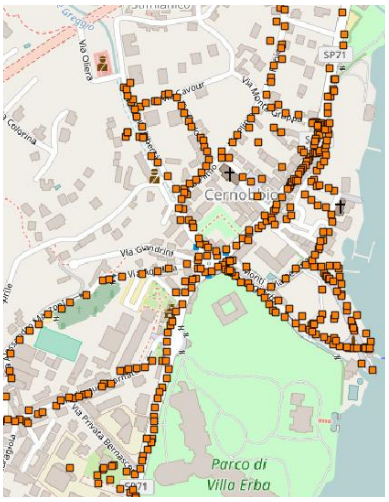
Our aim is to obtain a suitable nodes disposition over the city's area, because it heavily affects the final heat map images produced by the algorithm, starting from the output topology. The hard work performed on the parameter set definition would be totally useless, unless the final images satisfy the desired quality and readability criteria expected from heat maps. We left the description of the final heat map images construction in Section 5.2, where we explain in details how the expected results are produced.

5.2 The MEP Heat Maps Representations

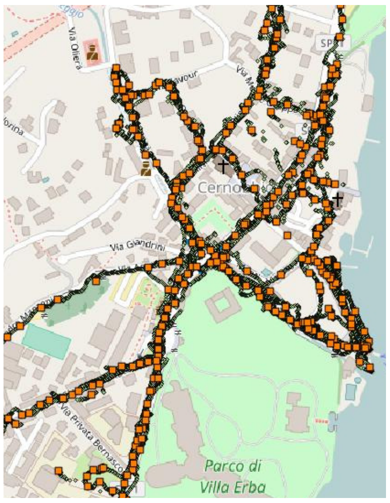
Starting from topologies obtained by the application of the MEP-Clusterpath algorithm, we can graphically define a specific accessibility representation



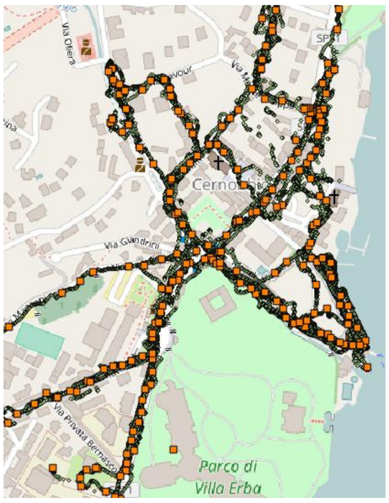
(a) Results obtained with parameter set corresponding to id=2.



(b) Results obtained with parameter set corresponding to id=4.



(c) Results obtained with parameter set corresponding to id=8.



(d) Results obtained with parameter set corresponding to id=12.

Figure 5.6: Each panel presents the final topology arising with a different set of parameter. The "id" links the panel to the corresponding parameter set in Table 5.2. In such a way its easy to compare the results obtained from different tunings.

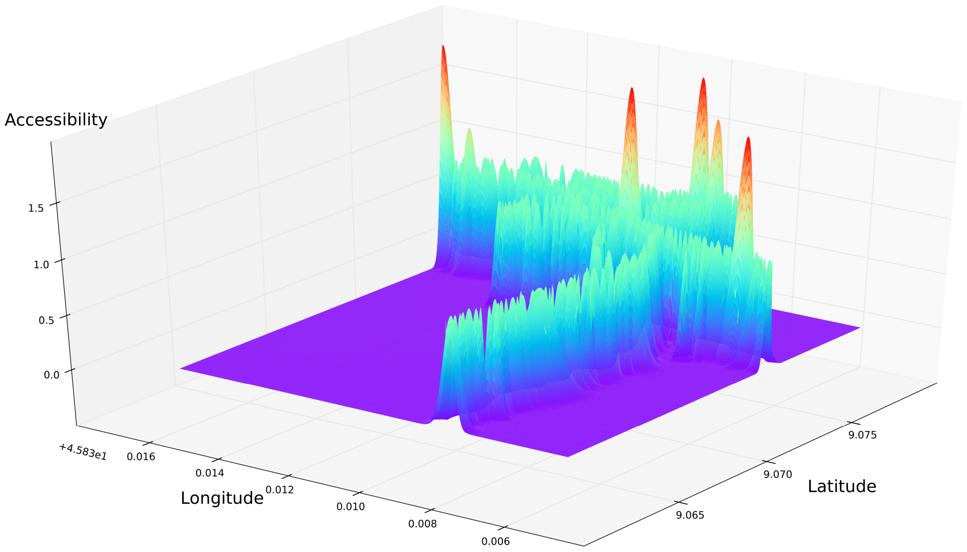
for each city in order to build the related heat map. We design each node within the graph to contain samples describing its neighboring area. Thanks to this shrewdness, we are able to obtain the shape of the blob corresponding to each node, according to what we described in Section 4.3, by simply retrieving the associated samples.

Initially we present heat maps as result of the application of the base version of the AING algorithm on the data recorded on Cernobbio, in order to obtain some reference images over which we can evaluate the effectiveness of our improvements. Figures 5.7(a) and 5.7(b) display heat maps obtained from the same topology adopting a parameters setting corresponding to row with $id = 1$, contained into Table 5.1. In details, Figure 5.7(a) offers a three dimensional view of Cernobbio, while Figure 5.7(b) a simple two dimensional one. These two images are particularly relevant because they highlights how the blobs interacts each other, achieving the final goal to produce heat map images which describes the accessibility of a given city. Blobs' magnitude in Figure 5.7(a) uniquely identifies the accessibility level discovered during the learning process which can be successfully adopted to color the map. The accessibility scale valid for both figures can be found in Figure 5.7(b). Purple areas denote those which have not been covered by the mapping process, and the transition flows from green (which marks accessible areas) to red (indicating the presence of a critical issue coming from an obstacle or barrier). The purple layer indicates the base unrecorded level. Higher red peaks reflect the inaccessibility level experienced and subsequently noticed by users.

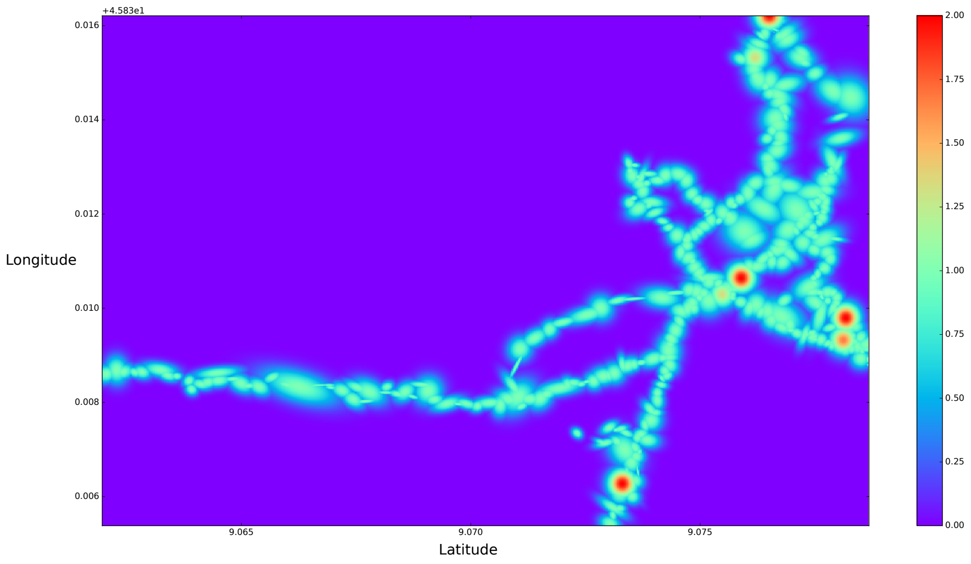
We apply the MEP-Clusterpath over the same dataset producing results depicted in panels in Figure 5.8. On the left-hand side, Figure 5.8(a) displays a three dimensional representation of Cernobbio surface according to the accessibility levels obtained by fusing the learned topology with the explicit information conveyed by the users with MEP-Traces. On the right-hand side, Figure 5.8(b) represents a two dimensional heat map of the same area offering an interesting perspective of the whole surface of the city.

Comparing results obtained from the application of AING and MEP-Clusterpath algorithms (presented in Figures 5.7 and 5.8), we can evaluate virtues and vices of both.

We described the uprising of the phenomena in which the AING creates uncontrolled neurons concentrations over tight areas. Even though from the topology point of view they do not represent a critical issue, nevertheless they could create a problem when we want to represent them through an heat map. We notice that in those cases, the algorithm somehow fails to fuse information regarding a specific area, and the neurons ends up con-

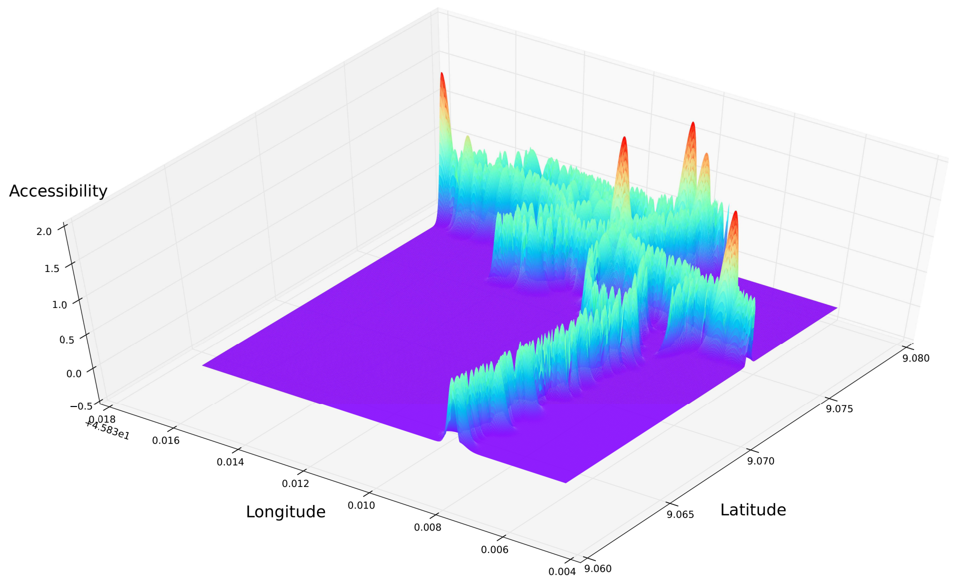


(a) Three dimensional representation of the accessibility topology obtained through the AING algorithm over Cernobbio. The base layer collects the longitude and latitude of the neurons over the surface, while the third axis expresses the level of accessibility of an area.

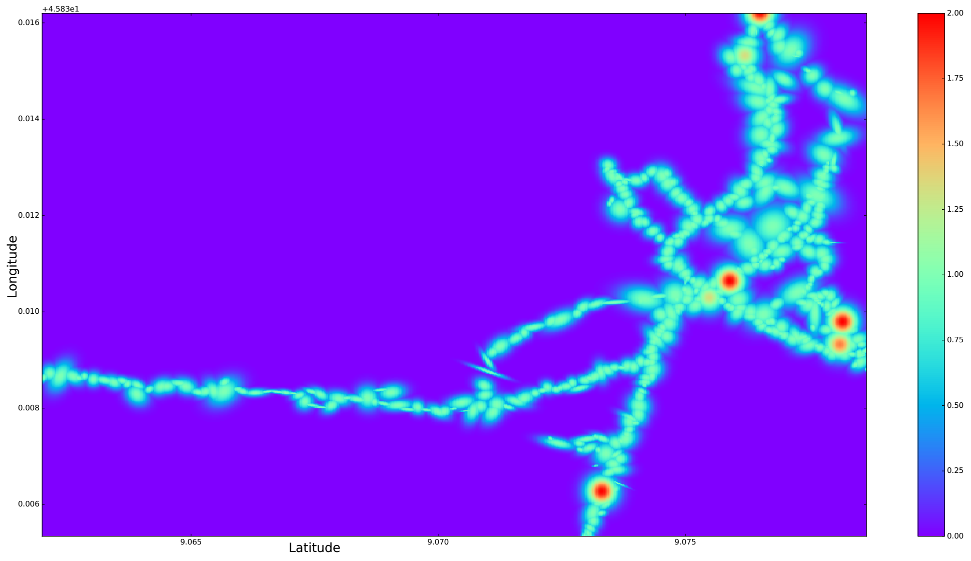


(b) Two dimensional representation of the accessibility topology obtained through the AING algorithm over Cernobbio. The two axis locate the longitude and latitude of the nodes, while the coloring highlights the different accessibility levels.

Figure 5.7: Two and three dimensional heat maps representations of the accessibility obtained by the application of the AING algorithm on the data contained into the MEP-Database related to Cernobbio. Both panels represent the same input topology from a different point of view. In particular, the accessibility scale valid for both figures appears in Figure 5.7(b). Starting from 0 (unrecorded areas) the scale describes values up to 4.



(a) Three dimensional representation of MEP-Clusterpath algorithm results over Cernobbio. The base layer collects longitude and latitude of the neurons over the surface, while third axis expresses the level of accessibility of an area.



(b) Two dimensional representation of the MEP-Clusterpath algorithm results over Cernobbio. The two axis locate the longitude and latitude of the nodes, while the coloring highlights the different accessibility levels.

Figure 5.8: Heat maps referred to Cernobbio. In both figures the green blobs refer to the accessible areas while the red peaks correspond to the places where an obstacle or an inaccessible barrier has been encountered by a user. The lowest purple level instead marks the areas with no reference path recorded by users.

taining just a single sample. With just one element, the samples variance will be equal to 0 producing no blob at all. Those areas will get no color representation for the accessibility.

The MEP-Clusterpath algorithm solves this problem by fixing a minimum distance between nodes that forces the algorithm to fuse a minimum amount of information. In this way there will be no neuron containing just a single element, and it can be represented with ease through blob-shape element.

5.3 Results Validation

As we describe in Section 2.2.1, the aims of unsupervised learning methods is to perform a model's inference describing the hidden structure from an "unlabeled" set of data. This means that the training of this kind of methods does not rely on any error signal back-propagating through the method in order to improve the quality of the obtained results.

Classical examples of unsupervised learning, in both natural and artificial neural networks, usually rely on the Donald Hebb's principle: neurons that fire together wire together. Within Hebbian learning theory, connection reinforces irrespectively to a signal error. Specifically, they are the product of the coincidence between action potentials of two different neurons. Hebbian Learning underlies a range of cognitive functions, such as pattern recognition and experiential learning. Among many neural network models, the self-organizing maps (SOM)[1] represents commonly adopted unsupervised learning algorithms.

MEP-Clusterpath, according to unsupervised learning definition, does not represent an exception: MEP-Database collects implicit and explicit information coming from MEP-Traces and MEP-APP, but no a priori information about the accessibility-level of a given area. As consequence, also the nodes position can't be identified a priori and used for the algorithm training. To build heat maps evaluating the local accessibility of a given area, we compare our results according to how good and intuitive appear their representations from a user perspective. First of heat maps must offer an easy readable information to any kind of user. Even though we evaluates some computational related parameter in terms of speed and memory consumption, the final setup must satisfies the real users' requirements.

Evaluation of the final results can be graphically performed tanks to

[1]As we describe in Section 2.5 self-organizing maps(SOM) are topographic organizations in neighboring locations within the map groups inputs with similar properties.

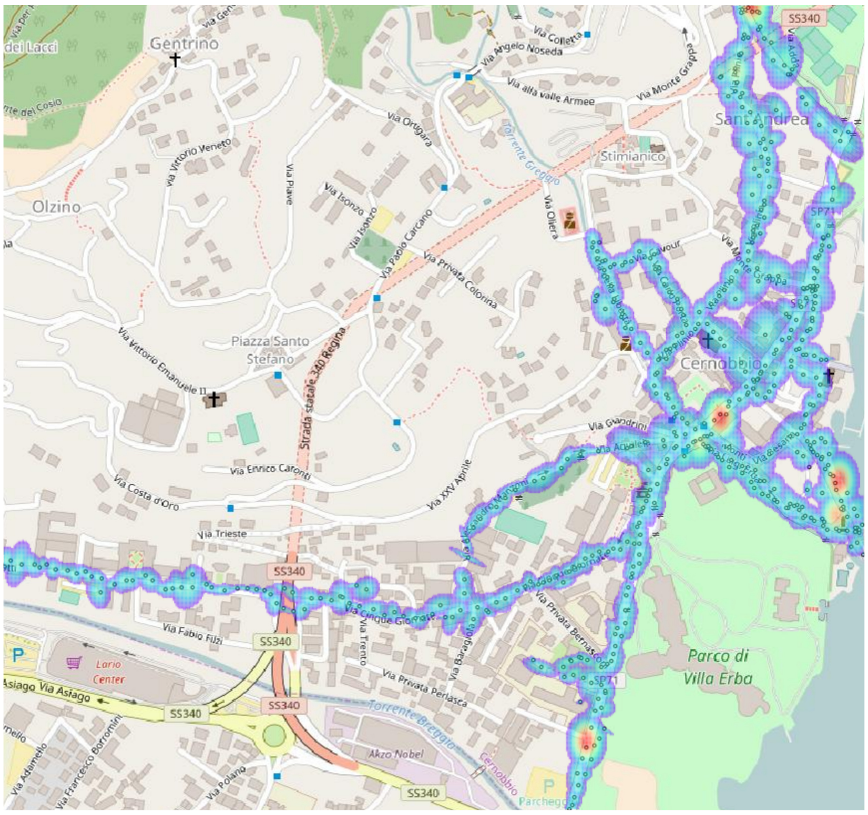


Figure 5.9: Resulting heat map obtained over the city of Cernobbio. Green shaded areas locate the accessible areas within the city's surface while the red blobs precisely locate the inaccessibilities descending by the presence of an obstacle.

Figure 5.9. The overlap between cartographic and heat map obtained according to the MEP-Clusterpath algorithm offers the desired representation of the different areas within Cernobbio. In this case, we decide to remove the purple color marking the unrecorded areas obtaining the visualization of clear and intuitive maps. Users sampling process updates the definition attributed to a given area and changes its level of accessibility.

Soundness of the results cannot be easily obtained due to the fact that the MEP-Clusterpath algorithm is an unsupervised learning method. Validation of the results is then left for future works and it represent a limit of this kind of approach: in fact, it is not possible to define an a priori definition of the accessibility level of an area. Local considerations can be performed through a simple visual inspection of the map, considering areas whose accessibility level can be verified personally.

5.4 The MEP APIs

Results obtained with the MEP-Clusterpath algorithm can be visualized using MEP-APP, offering the accessibility maps to any user who requires them. This application has been developed with the purpose to offer an interactive two dimensional maps visualization of the information contained into the MEP-Database, with no exception for the results obtained with the MEP-Clusterpath algorithm. In Section 3.1 we defined how the users can interact with the application, but we will now describe how we exchange the information through the MEP-APIs.

Once the user demands information of a given city, MEP-APP performs an *http* request to the MEP back-end server defining the boundary and all the parameters required for the composition of the image that will constitute the reply to the request. We address this specialized subprogram as MEP-Heatmap-Visualizer: this piece of software gets into the APIs tool set, with the aim of make available the information produced by the execution of the MEP-Clusterpath algorithm.

We consider as a necessary constraint the fact that exists the possibility in future that many different applications can use the MEP Apis to connect with the MEP-Server in order to retrieve accessibilities information about cities around the world. For this reason we imagine them as general as possible in order to be robust over time. MEP-Heatmap-Visualizer does also offer a three dimensional plot of the topologies discovered during the learning process replaying to the different requests reaching the server.

The MEP's APIs conclude the life cycle of the MEP-Clusterpath algorithm which, starting from the input data coming from the two applications, produces the results for the applications themselves.

Chapter 6

Conclusions

"Non chiederti solo cosa tu possa fare per il disabile, ma anche cosa il disabile possa fare per te."

Angela Gambirasio

According to World Healt Organization statistics, about 15% of the world's population, which represents over a billion people, presents some form of physical disability or mobility impairments. This works represents an initial step in the direction of building accessibility maps, i.e. cartographic maps containing information about the accessibility of the cities around the world. In particular the main purpose of the thesis is to develop a totally automatic procedure to build heat maps, starting from the implicit and explicit data conveyed by MEP-Traces and MEP-APP.

To reach the final goal we adopt different cleaning procedure to mitigate the error which affects any existing sensor. For the data pre-processing, we experiment two methods: the MEP-Fusion algorithm (Section 3.2), and the GNSS trajectory correction (Section 3.3.3) algorithm. The first one performs a sensor based GPS cleaning. The data extracted from the gyroscope, magnetometer, and accelerometer sensors get fused together in order to improve the user's localization into the recorded paths. The GNSS trajectory correction shifts those points that overlap with a building area toward the nearest road. It uses the static information contained into the OpenStreetMap service in order to operate at best.

A neural network approach executes the learning of the topology. The method presented in this thesis, strictly relies on the Adaptive Incremental Neural Gas Network approach with some changes, in order to be able elaborate the geographical datum. We do also propose an adaptation which force the algorithm to suitably distribute the nodes over the surface that it is going to be learned.

Finally we build heat maps combining data produced by the MEP-Clusterpath algorithm, in order to visualize the obtained clustering results with the accessibility zones into a city. All the interested citizens can participate and offer their own contribution to the project, improving movements quality of people affected by physical impairments. MEP-APP also offers the access to the information the users contribute to, by visualizing the heat maps produced as result of the MEP-Clusterpath algorithm.

We perform several tests in order to obtain a suitable parameters set for our algorithm, and we devise an improvement of the base Adaptive Incremental Neural Gas Network (AING) algorithm. MEP-Clusterpath allows to define a minimum density for the nodes' distribution to perform the learning of a particular area. This avoids the undesired situation of neurons concentration within limited zone.

Our tests demonstrate how this problem must not be underestimated: resulting heat maps poorly describe the accessibility of the area interested by the phenomena.

Despite a rigorous validation approach is still missing, we obtain promising heat maps which demonstrate to be interpretable with ease by any kind of user. The lack of a validation approach represents the main limit of our approach, and we left it for future development as described in Section 6.1.

6.1 Future Works

The approach adopted to perform the task of topology learning falls into the unsupervised learning methods set. For this reason, as we described in Section 5.3, we were not able to validate the obtained results through a rigorous procedure.

The presented approach still lacks a validation procedure of the topologies obtained through the MEP-Clusterpath algorithm, presented into this work.

We suggest as a possible proposal the possibility to develop a crowdsourcing technique. Our idea is to leave this fundamental task to users who perform mapping around cities using MEP-Traces. This could bring to possibly conflicting information, that should not anyway trusted blindly. Of course users notifications should be mediated through specifically developed statistical method. In this way, it would be possible to validate heat maps produced by the MEP-Clusterpath adopting rigorous methods.

Other possible improvements can be obtained applying the Ramer-Douglas-Peucker (RDP) [29] [28] algorithm which reduces the points number into a segmented line. Given a curve composed by linear segments, the idea of the algorithm is to find a more compact representation of the curve (in this case denoted as Polyline) containing a subset of points which define the original path. We expect that this algorithm could produce a further cleaning of the input paths recorded with MEP-Traces.

Bibliography

[1] Hierarchical Clustering Algorithms. A tutorial on clustering algorithms. *Online][Cited: 24 04 2008.] http://home. dei. polimi. it/matteucc/Clustering/tutorial_html/hierarchical. html*, 2013.

[2] Gianluca Bardaro, Ava Vali, Sara Comai, and Matteo Matteucci. Accessible urban routes reconstruction by fusing mobile sensors data. In *Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia*, MoMM 2015, pages 84–92, New York, NY, USA, 2015. ACM.

[3] Mohamed-Rafik Bouguelia, Yolande Belad, and Abdel Belad. An adaptive incremental clustering method based on the growing neural gas algorithm. In *Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods*, pages 42–49, 2013.

[4] Kenneth P Burnham and David Anderson. Model selection and multi-model inference. *A Pratical informatio-theoric approch. Sringer*, 1229, 2003.

[5] Hong Chang and Dit-Yan Yeung. Robust path-based spectral clustering with application to image segmentation. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 278–285 Vol. 1, Oct 2005.

[6] Hong Chang and Dit-Yan Yeung. Robust path-based spectral clustering. *Pattern Recognition*, 41(1):191 – 203, 2008.

[7] Davide Antonio Cucci and Matteo Matteucci. On the development of a generic multi-sensor fusion framework for robust odometry estimation. *Journal of Software Engineering for Robotics*, 5(1):48–62, 2014.

[8] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[9] Bernd Fischer and Joachim M. Buhmann. Bagging for path-based clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(11):1411–1415, November 2003.

[10] Bernd Fischer, Thomas Zöller, and Joachim M. Buhmann. *Path Based Pairwise Data Clustering with Application to Texture Segmentation*, pages 235–250. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[11] Joel N Franklin. Matrix theory, 1993. *Mineola: Dover Publications Inc*, 1968.

[12] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[13] James E Gentle. *Matrix algebra: theory, computations, and applications in statistics.* Springer Science & Business Media, 2007.

[14] Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. *The elements of statistical learningFis : data mining, inference, and prediction.* Springer series in statistics. Springer, New York, 2009. Autres impressions : 2011 (corr.), 2013 (7e corr.).

[15] Christopher C Heyde. *Quasi-likelihood and its application: a general approach to optimal parameter estimation.* Springer Science & Business Media, 2008.

[16] Frederick S Hillier. *Introduction to operations research.* Tata McGraw-Hill Education, 2012.

[17] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[18] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.

[19] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2017-02-03].

[20] Sakis Kasampalis. *Mastering Python Design Patterns.* Packt Publishing, 2015.

[21] Leslie Lamport. How to make a mulitprocessor computer that correctly executes multiprocess programs. In *Readings in computer architecture*, pages 574–575. Morgan Kaufmann Publishers Inc., 2000.

[22] Marco Negretti Ludovico Biagi. Correzione cartografica di traiettorie gnss da periferiche a basso costo. feb 2016. Politecnico di Milano - DICA - Polo Territoriale di Como.

[23] Claudia Maria Bauzer Medeiros. *ADVANCED GEOGRAPHIC INFORMATION SYSTEMS-Volume I.* EOLSS Publications, 2009.

[24] Markus Neteler, Venkatesh Raghavan, et al. Advances in free software geographic information systems. *Journal of Informatics*, 3(2), 2006.

[25] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[26] Uyen T. V. Nguyen, Laurence A. F. Park, Liang Wang, and Kotagiri Ramamohanarao. *A Novel Path-Based Clustering Algorithm Using Multi-dimensional Scaling*, pages 280–290. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[27] Harun Pirim. *A Minimum Spanning Tree Based Clustering Algorithm for High Throughput Biological Data.* PhD thesis, Mississippi State, MS, USA, 2011. AAI3450335.

[28] D. K. Prasad, C. Quek, M. K. H. Leung, and S. Y. Cho. A parameter independent line fitting method. In *The First Asian Conference on Pattern Recognition*, pages 441–445, Nov 2011.

[29] Dilip K. Prasad, Maylor K.H. Leung, Chai Quek, and Siu-Yeung Cho. A novel framework for making dominant point detection methods non-parametric. *Image and Vision Computing*, 30(11):843 – 859, 2012.

[30] M. Matteucci F. Salice GOODTECHS '16 S. Comai, E. De Bernardi. Maps for easy paths (mep): Enriching maps with accessible paths using mep traces. *2nd EAI International Conference on Smart Objects and Technologies for Social Good, 30 novembre - 1 dicembre 2016, Venezia, Italy*, dec 2016.

[31] Bosch Sensortec. Bmi160, 2015. BST-BMI160-DS000-07.

[32] Noam Shental, Assaf Zomet, Tomer Hertz, and Yair Weiss. Pairwise clustering and graphical models, 2003.

[33] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.

[34] Paul Spivak. Gps controlled marine speedometer unit with multiple operational modes, March 5 2002. US Patent 6,353,781.

[35] Richard J Trudeau. Introduction to graph theory (corrected, enlarged republication. ed.), 1993.

[36] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[37] Leland Wilkinson and Michael Friendly. The history of the cluster heat map. *The American Statistician*, 63(2):179–184, 2009.