UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITECNICA SUPERIOR

BACHELOR IN AUDIOVISUAL ENGINEERING

PROYECTO FIN DE GRADO

AUTOMATIC PLACES OF INTEREST DETECTION BY IMAGE PROCESSING AND DATA MINING

Author: Luis Herranz Mañas
Supervisor: Ivan Gonzalez Diaz

February 24, 2015

Dedication

It has being five years since I first started this career. It feels like a long time ago when I decided to enroll in this university into a journey of wonder and hard work. I can no longer consider myself that eighteen years old, for good I have been modeled into a more organized and strict version of myself. I see the world in a new way, I could have never expected, when I first stepped in this university, that I could even began to understand how the digital world worked or what was the magic behind programs such as the famous Adobe Photoshop [8] or other computer sciences. I can now confront bigger challenges and I feel ready to take my studies to a new level, this has been just the beginning of my journey and there still is a long road ahead of me. We get easily a grab on those bad moments when finals and project deadlines may overwhelm us, but those times will come to an end and eventually we could look behind us and see what we have achieved. I personally look back and see a lot of people who have accompanied me on my journey.

First, I want to thank my family, starting by the ones who have truly shaped me into what I am. Ana Maas Antn and Pedro Herranz Pinto, you two are the reason of why I am writing this thesis right now. I am grateful for who I have become and with you I will specially celebrate what I have achieved or will be able to do. Consider yours any success I am able to make, for this would not have been possible without you. There are two others who I am also grateful. Cristina and Blanca Herranz, dear sisters with you I can share my empathy, we are all in the way to different goals but we are all studying and we know how hard things can get when bad times hit. There is another pair of couples who have been a source of strength and inspiration. My grandparents Arturo Maas, Mary Cruz Antn, Carmen Pinto and Pedro Herranz this people have show me how to fight and face different situations. They told me how to stay true to your principles and go against the odds. It will be totally unfair to leave many other members of my family without a particular acknowledgment, but unfortunately I have a very extensive family and limited space, but know that you all have help me reach this point and I'm very thankful for it.

I must make a mention to my second family, my closest friends. Some of you understand the burden of the engineer even more than I do but we all have been sharing this changes in our lives. For this I thank Miguel Angel Gonzalez, Ignacio Gil, Ignacio Castellano, Alberto Vela, Fernando Sanchiz y Javier Morell.

There have been many who have accompanied me in this journey. I really feel that this would not have been possible without you. I have to mention Patricia Rodriguez Lluch, Juan Luis Sanz Moreno y Carmen Alonso Martinez as those who have kept me running, these people with their individual efforts have kept me from wreck my career and my whole live. The same applies to the following honorable mentions: Estefania, Alvaro, Jorge Lavn, Rubn, Mur, Almudena, Javi, Alvaro de la Serna, Cristina and Alejandro. You have all make this degree a special journey and I am

very thankful for it.

Finally, and most certainly not least, to the professors at UC3M that have helped me through my degree and have taught me more than knowledge. Thank you Julio Villena, Raquel Crespo, Aberlardo Pardo, Emilio Parrado , María Celeste Campo, Fernando P. Cruz, Jeronimo Arenas, Luis Antonio Azpicueta and Luis Fernando de Inclan. My last mention goes to Ivan Gonzalez who is also the supervisor of this project, thank you for all those subjects and for helping me make the best project I could do.

Abstract

Computer science has experienced a continuous exponential development over this past decades. Almost everyday we can find news about a new algorithms being developed or plug-ins being released. Each is a step forward in this science. The World has shown us that there are multiple ways to do the same thing. Having the same outcome by following very different ways with not necessarily having a best way to do it. There is probably a way that becomes the most optimal for us depending of the priority we give to parameters such as running time, external permission, portability or compatibility. Fulfilling some objectives may have more priority than fulfilling others. This will help us find which way we are going to do it.

With the rapidly growing use of the Internet, and public Open-source code of complicated algorithms everyday becomes easier to develop new tools. In this case we have develop a tool which can generate a list of the points interest in a city or town. The application is based on Flickr's database. Flickr [9] is a web application able to store pictures taken by their users, very similar to other known social networks such as Facebook . Once those pictures are stored some information like GPS location or the date is embedded within picture. Flickr is able to extract that information and is allowed to distribute it. By means of a query we are able to communicate and ask for a database of pictures given some search tags so we are able to philter those pictures that we are interested in.

Now what we are given is a large number of coordinates associated to a certain tag, such as city name Berlin. We can expect that most of those points are positioned around that place, which it is mostly true apart from a few exceptions. Now we are able to generate our own results by passing this data through our code. We will firstly be able to generate a new database based on the inputs, this process can be done automatically but it is highly recommended to follow under user supervision using the user interface. An User Interface has been developed for this matter, easing the process of database creation by taken away the mayor part of its tediousness. This process has been verified through multiples experiments and can guarantee an above 50% precision, and yet there are multiple ways we can improve our results many of which we will discuss during the Future Works subsection.

The process will be complimented by a visual clustering process which allows to associate images and labels, we also got from Flickr, to the clusters we have already found in previous steps. So we can implement programs like a personalized visual guide or a plugin to automatically generate tourist landmarks for a city.

Keywords: Clustering Data, Flickr, Cluster Analysis, Visual Clustering, Data Mining, DBscan analysis, GIST.

Contents

List of Figures

Chapter 1

Introduction and Socioeconomic Context

Over the past decades, the growth and widespread availability of the Internet has led to a great number of changes for all sectors of society, transforming many elements in our everyday lifework. The way content is now created and instantly shared, is leading us to a new era of communications. The internet has become the greatest source of wisdom and learning. All kinds of users are now capable of developing beginner or advanced skills to use in professional occupation. As an example, self-learning of a wide variety of contents is now possible by using the web. We can easily learn many new different skills by just following guides, videos or tutorials.

Eleven years ago, back in 2004, 14 out of the 43 million Spaniards had Internet access at home [1]. Although this percentage was low, it has grown up to the 62% of Spain's population in 2013. With the Internet being used as a daily, worldwide tool, we have experienced a 'Boom' in web applications. It can be said that there is an application for everything out there. Sadly, many content creators may feel very frustrated when trying to find new unimplemented ideas. This goal can be particularly complicated for small independent developers when large companies, such as Google [7] and Apple [6], are constantly fighting the innovative race to get the market supremacy. So the question is: can a small developer compete against these 'giants' which are constantly investing thousands of millions of dollars in human and technology resources?

In search of new undeveloped areas, social networks can be considered as a potential source of unmet needs. With so many people connected to the Internet we can assume that these user-content-generated web applications are frequently used. The need of the continuous update and renewal of contents can be considered as a source of opportunities for any web developer. It is very frequent that new applications emerge from giving a new vision of old-fashioned, well-stated concepts. However, any idea must be translated into a potential solution to practical issues.

The idea behind this project is to give an answer to the following question: is there already at tool that can help a tourist finding the historical places in a town? We estimate that there are plenty of possible answers. But which of those tools can also tell him where to go if he is interested in 'Nightlife'? And what about only those which are adapted to handicapped people? Well, we believe this will be an interesting service to offer and this is what leads us to start

out this project. While contemplating some of the already existing options we see that most of these applications are based on pre-computed databases. Most of these databases are exclusively present in large cities. This means that such applications are not be able to find key locations in remote places or even in cities located in lesser developed countries.

1.1 Motivation

There are several factors that have contributed to the motivation behind this project. One of reasons has been the possibility of designing a new application; a system based on the content of web applications and social networks. This content is created and shared by its own users, such as Flickr. In this platform we can get lots of information by analyzing the stored meta-data of its pictures. The advantage is that we can design a filtering system by using the tags stored in the meta-data. This means that we will have a personalized search engine able to decide which locations we want to find by using multiple statements. There are numerous applications, an example would be a personalized guide creator. Users will insert the tags and the name of the city, and then make use of the automaticly-generated map.

We found that it was possible to develop a new way to determine where the interesting places are. Since these locations are frequently visited, we can map them using the content created by users. This process would neither be created nor biased by third parties software or manually obtained information. We must remember that social networks such as Flickr are fueled by content generated by users themselves. So, by using our application they would be able to generate maps without manually entering registers unlike other map-based applications. In a few words, by using this method some content-generated-by-users applications could turn into content providers and be able to allow a new set of services. The results will be an estimation, probably less accurate than other map based content providers such as Google Maps [2] or Places [4]. These two are already well-known and widely spread services. Our advantage is that our content is generated automatically and on-demand, avoiding tedious manual annotations.

The automatic generation of content is possible whenever we have enough pictures to make it possible. If users uploaded pictures of remote landscapes, even the these places could be easily mappable.

This application may find its limits in those places that cannot be accessed, and subsequently photographed by tourist.

1.2 Objectives

In this section you will find a quick overview of the target of this project.

The main objective is to create an application based on Flickr. It aims to create a map by selecting the tagged elements and locations that the user has demanded. The process will be fully automatic and will only require that the users sets a pair of tags by inserting multiple strings. After that, the application will follow a chain of steps, one after another, and will end giving the coordinates of several places of interest that are relevant for the user query. Then a second

phase begins, the process will get a set of pictures for each element in the list of coordinates we got from the previous step. The pictures are acquired by using a second query to Flickr. Now, the objective will be to have those pictures associated to those GPS locations, got from another Flickr query, and a selected set of tags for each.

Since the validity of our outcomes depends directly on the public content that users post on and not to the direct sources of verified databases of content providers such as Google Maps. We must test the validity of our application by running multiple experiments. We must have a minimum precision and recall values.

Next, we list a set of particular goals to be fulfilled during the project:

1. Design a new clustering scheme able to detect and locate those places that have some special interest to the public. This will be done by using GPS coordinates associated to pictures got by Flickr.
2. Quantitatively evaluate the clustering process used over the samples that assess performance novel and determine changes to make accordingly.
3. Design a way to visually determine the images that best represents each location of interest, and also to tag them appropriately.
4. Evaluate qualitatively the visual processing module of the precious point.
5. Design and create a fully functional application using Matlab which allows to control the system.

Chapter 2

Related Technologies and State of Art

2.1 State of Art on Automatic Systems for Detection of places of interest

Ever since the Geo-localized multimedia content appeared as databases such as Flickr [9], Panoramio [10] or Picasa [16], the research community has been trying to develop new tools that use the underlying potential in this data.

Thereby, schemes such as the one proposed in [20], are based in the premise that cultural or touristic interest locations are photographed more frequently by the users of these content platforms. By taking a Geo-tagged database, this takes a clustering or grouping process in which the GPS coordinates are associated to the images themselves. The output of this step is a set of clusters that are associated with the landmarks or places of interest. Then, for each of the clusters, we obtain common visuals representations and the set of tags that comes with them by using visual and text processing.

Very similar proposals are explored in other projects like [29] and [34], where a large model of Geo-tagged database is proposed. The database includes 5319 monuments or interesting landmarks, located over 1259 cities in 144 different countries.

To perform automatic grouping of GPS coordinates and consequent discovery of locations of interest, most of the proposed techniques used clustering based on point density: examples are the Mean Shift algorithm [35], the DBscab algorithm [22] or its generalization in OPTICS [25].

Such techniques can be extended later to solve the problem of the geo-tagging of new contents, as proposed in [33], where the authors use discriminative models with latent variables (latent SVM) to solve concurrently a clustering problem or landmark discoveries and geographic tags of content (inference GPS coordinates).

However, due to the low percentage of geo-tagged images in databases such as Flickr (around 1%), the authors of [28] propose a system that combines GPS coordinates (if available) with other types of meta-data (points of capture, user visual content) and thus, work with partially labeled data. In particular, this paper proposes a way to get the geolocation when not available from other meta-data.

2.2 Related Technologies

During this section we will explain the main techniques and algorithms that we have used in the project.

2.2.1 Data Mining and Clustering Algorithms

Data Mining [30] [31], also called knowledge discovery in databases, is in computer science, the process of discovering interesting and useful patterns and relationships in large volumes of data. This field combines tools from statistics and artificial intelligence (such as neural networks and machine learning) with database management to analyse large digital collections, known as data sets.

Cluster search analysis is a primary method for database mining. It is either used as a stand-alone tool to get insight into the distribution of a data set (for example to focus on further analysis and data processing), or it may also be used as a pre-processing step for other algorithms operating on the detected clusters. Almost all of the well-known clustering algorithms require some input parameters. Those inputs are key to determine the outcome of the process. Some algorithms will also determine the center of the cluster, or set a reference point to represent the whole cluster, such is the nature of K-means. Experimentation through many executions using different parameters are advised, since the outputs may change drastically by using different inputs.

There are two basic types of clustering algorithms defined by Kaufman and Rousseeuw in 1990 [14], namely *partitioning* and *hierarchical* algorithms.

Partitioning algorithms construct a partition of a database D of n objects into a set of K clusters. Where K is an input parameter for these algorithms and requires users to have some domain of knowledge in order to set it accordingly. The partitioning algorithm typically starts with an initial partition of D and then uses an iterative control strategy to optimize an objective function. Each cluster is represented by the gravity center of the cluster such as K-means [32] algorithms or by one of the objects of the cluster located near its center as k-medoids [21] algorithms. Consequently, partitioning algorithms use a two-step procedure. Firstly, determine k representatives minimizing the objective function. Second, assign each object to the cluster with its representative closest to the considered object. K-medoids and later K-means are some of the first clusters-identification tools created.

Hierarchical algorithms create a hierarchical decomposition of the database (D). The hierarchical decomposition is represented by a dendrogram, a tree that iteratively splits D into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of

the tree represents a cluster of D. There are two different approaches to the tree generation. The dendrogram can either be created from the leaves up to the root (agglomerative approach) or from the root down to the leaves (divisive approach) by merging or dividing clusters at each step. In contrast to partitioning algorithms, hierarchical algorithms do not need k as an input. However, a termination condition has to be defined indicating when the merge or division process should be terminated. In other words we need to specify other parameters so that we can decide where the cluster ends and where the other begins. One example of a termination condition in the agglomerative approach is the critical distance Dmin between all the clusters of Q. The main disadvantage with this process is determining the value of those additional parameters. If we take the example before, Dmin (implying minimum distance), the key idea is that two points belong to the same cluster if you can walk from the first point to the second one by a sufficiently small step. To avoid this parameter, the literature offers other algorithms, such as one named Ejcluster [15], which follows a divisive approach. This particular algorithm does not require any parameter since this method will find it by itself. However the computational cost due to the distance calculation of each pair of points is greatly increased. This factor will be make applications such as character recognition for large databases a very costly process in time. Finally we also have explored the density based procedure. This approach allows to identify clusters in any k-dimensional point set. Results are usually expressed as histograms and non-overlapping cells. Density based clustering gets good results, finding any number of clusters with any shape.

2.2.2 K-means

The K-means [32] clustering algorithm is considered as one of the most simple. Nevertheless, it can be very useful when trying to determine a centroid for a given number of samples. K-means computes the numeric mean for a n by m dimensional matrix, where n are the dimensions and m indicates the number of samples. The function requires an additional parameter for the analysis. A parameter which indicates the number of clusters we are working with. This almost automatically excludes K-means as our main search engine tool. We can still use this algorithm if we are willing to work with a fixed number of clusters, or we are able to determine such thing by other means.

Since K-means algorithm is unable to automatically determine the number of clusters in a given set of points, the developer must determine which values should be optimal. This is a fragile parameter, and the best way to get different values of our evaluation output, is by a slow increment which will rise until an error is returned. The error must be caught in case that we want the execution not to be interrupted. Another fact is that K-means searches for spherical clusters, and also these clusters are expected to have similar number of components. None of these qualities are something we would see reflected in real cases. You will see that people will take more pictures of some places while almost ignoring others. And, that in many cases they do not follow a spherical pattern if they follow a pattern at all.

We found another possible application for K-means was in the matter of Visual clustering. Visual clustering, similar to Partitioning cluster algorithms, is based on finding the N centroids of an image descriptor. It is not directly applied over the images themselves but over their descriptor. If we wish to follow a quick Visual Clustering process without applying further calculations and extracting a fixed number values from the descriptor, K-means becomes idealfor this purpuse due to simplicity and its fast running time. K-means is able to find various cumulus

of the values of a set of long descriptors.

2.2.3 DBscan

DBSCAN is another clustering algorithm. Unlike K-means, it is based on the distance between the points that belong to the same cluster and the total number of points. K-means on the other hand, focuses on the distance to the centroid. DBscan selects the point in a cluster by 'jumping' from one to another. See this as a metaphor, we are only able to jump from one point to another if the distance that separates both is not too large [12], as seen in figure 2.2.

Since it is a density based clustering algorithm, it needs to have a minimum concentration of points to be considered as a cluster. This is an input parameter, Min_points, that allows us to be more or less permissive when deciding how dense our clusters be. The second parameter that this process requires is Max_distance. This one refers to the distance that can separate two points that belong to the same cluster. In technical terms, we can sum up a cluster as the set of points that are density connected with each other. The following figure: 2.2 represents these concepts, we provide additional definitions during the appendix of this document.

DBscan is well known for its advantages:

- DBscan can be applied to multiple dimensional spaces and has easy representations in 2 and 3 dimension.
- The algorithm can detect any point disposition, meaning that the figure that the cluster forms does not matter. We have an example in Figure 2.1. You can see that DBscan is able to identify clusters of all sizes and silhouettes.
- This cluster search process is able to differentiate and tag the points inside each cluster. After the execution, we can identify internal or peripheral cluster points and outliers as well. DBscan [23] starts tagging cluster by cluster, and the points that do not belong to the cluster are marked as outliers. In subsequent interactions, points labeled as outliers can be adopted by other clusters.

As we said, DBscan is the spine of this project. Knowing its advantages, we use this algorithm for a cluster search over large quantities of GPS samples. We do not know the number of clusters and we only can guess an approximation of how many points are the composed of. In addition, samples will probably be dispersed in non-regular patterns. These reasons make DBscan perfect for the process.

Choosing a Clustering Algorithm

At this point users know that the chosen algorithm has been DBscan, here we present the reasons behind our election. During early experiments, other two options were considered , K-means and the Optics [26] algorithm. In the 'Further Improvements' section we propose a theoretical improvement for the DBscan to increase the performance of the clustering module. It is time to enumerate the advantages that offers DBscan:

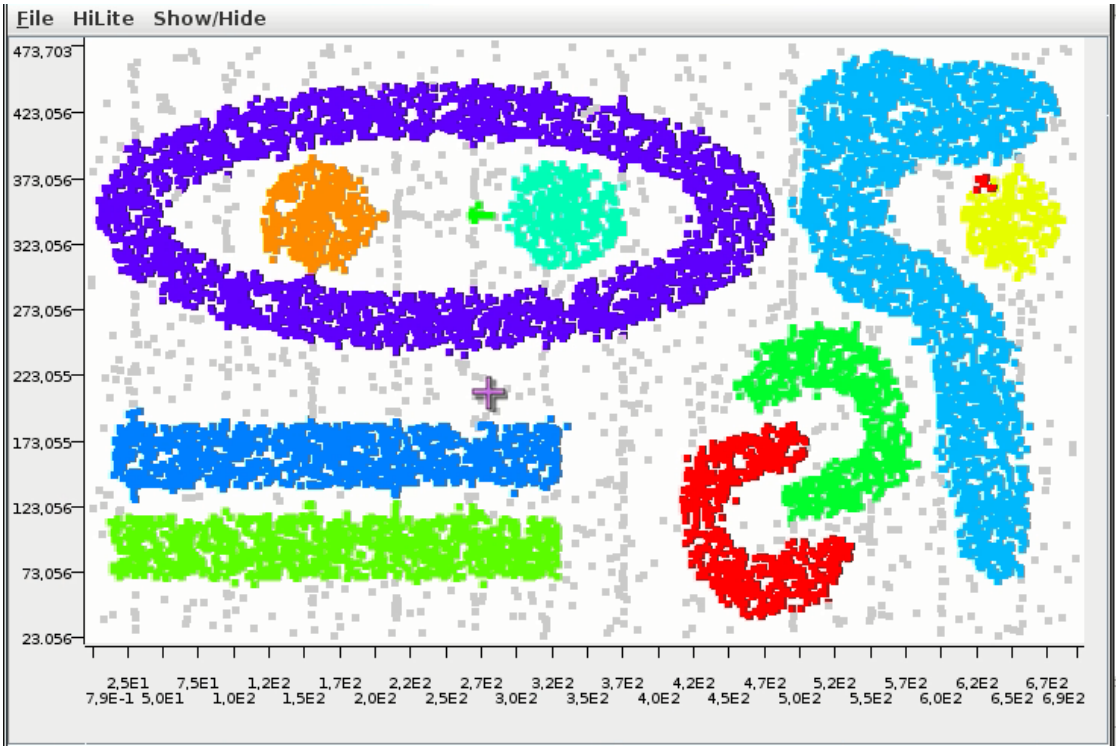

Figure 2.1: Image displaying different cluster silhouettes as seen in [18]

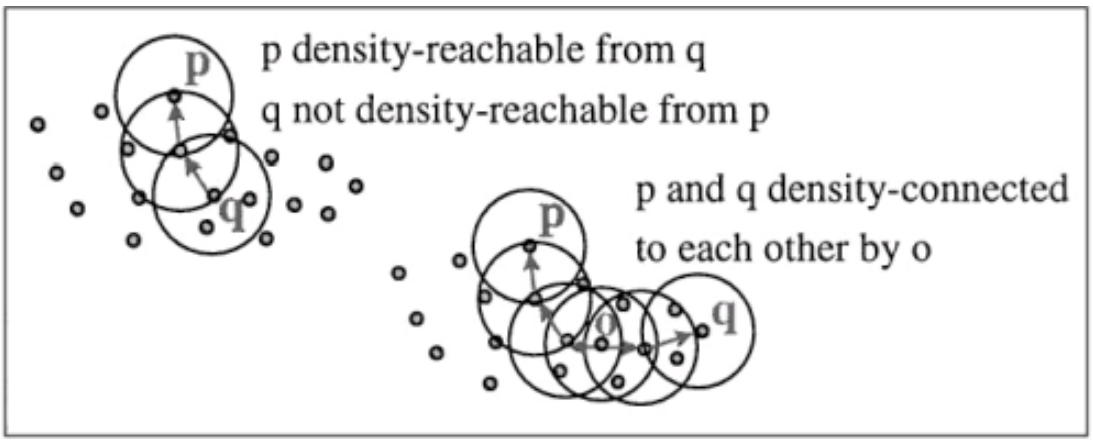

Figure 2.2: Example of reach-ability and cluster connections

- Clusters can take any form and have any size. This is exactly what we are looking for. This has a minor set-back, in a environment with a very high density of samples DBscan might be unable to differentiate from what should be two different adjacent clusters.
- DBscan has a relative small run-time compared to other clustering algorithms, Optics for example, as long as we do not work with a large amount of samples or a really high Min_Points parameter.
- DBscan's results are really easy to understand and is able to differentiate points that have not been grouped up as well as labeling the peripheral and inner points of a cluster.

It was fairly easy to discard K-means from our possibilities. K-means requires to know the number of clusters we are looking for. If we searched for the same number of clusters that were in the Ground Truth database, it would return an error.

On the other hand we had the Optics algorithm. This algorithm was able to partially solve the adjacent cluster separation problem. The way Optics is based in a density of reachable points makes it harder to include more points as we go further away from the center of the cluster. But still, additional problems must be confronted. For instance, by being more restrictive Optics is sometimes unable to detect the less populated clusters, especially those which are separated from the center. In addition, the complexity of the algorithm makes it hard to find a representation for the clusters. In the end, Optics offered similar benefits and was not significantly superior to what DBscan was offering. Due to its computational cost difference, *we finally favored DBscan as our clustering algorithm.*

2.2.4 GIST

GIST was originally proposed by Aude Oliva and Antonio Torralba [24]. The idea behind this algorithm is to develop a low dimensional representation of the scene, which does not require any form of segmentation. The idea is similar to what the human mind does. We can see a sight for a very short period of time and still be able to recognize what Torralba calls 'the essence' of a picture. This 'essence' is what our eyes identify first when staring at an image. We firstly identify the silhouette of a scene and the perspective of the objects in it. The relation between the contour of the main element and its properties is what we call the 'Spatial Envelope' and it is represented by the following properties described by:

- **Nature Degree:** The structure of a scene is very different between artificial and natural environments. In a city, horizontal straight lines and vertical constructions dominate, while most natural landscapes have wave-shaped contours and natural textures. Therefore, scenes with edges biased towards vertical and horizontal directions have a low degree of nature, while in scenes with high waves will have a high degree of nature.
- **Opening Degree:** The existence of the skyline and the lack of objects within the limits of the image give the scene a high degree of openness. The opening degree of a scene decreases when the number of elements on the edges increases.
- **Ruggedness Degree:** Depends on the size of items in each scaled space, its ability to build complex elements and the relationships between elements that belong to other structures.

The ruggedness is closely related to the fractal dimension of the scene and therefore its complexity.

- **Roughness Degree:** Roughness refers to the diversion of land toward horizon. Rough Environments produce oblique contours in the image and tend hide the horizon. Most artificial environments are built on a surface. Therefore, rough environments are mostly natural.

- **Expansion Degree:** The convergence of parallel lines gives depth perception in a two dimensional space. A front street view of a building has a low degree of expansion and a street with long drain lines has a high degree of expansion.

We divide the image into a 4-by-4 grid to study its properties. This division allows comparison between neighborhoods. Then, GIST uses a GABOR filter for border detection, GABOR represents through different frequencies and orientations a similar pattern to our human perceptive system. In the Fig: 2.3 there are some examples of GABOR filter representations over images, each emphasized in a different degree so we can identify the differences; a) had a high Opening degree, b) remarked Roughness, c) Ruggesness. The second row represent artificial sights: d) represents a high degree on Opening, e) expansion and f) on Ruggesness. This is an example of different spectrograms. In our project we do not study the concrete values of any of the properties above.

Based on the information extracted through the Spatial Envelope, GIST will generate a unique spectrogram for each image 2.5. Meaning that if we compare two or more pictures, the more properties they have in common, the more similar their GIST spectrogram will be. Here we have some examples of the spectrogram extracted from different images 2.4.

The GIST descriptor has already been assessed in some experiments (See some of Torralba's work [24]), and it is said to have a great performance, being able to extract a specific descriptor for an image while keeping the time reduced to a fashionable span. In terms of accuracy and time investment GIST is a very successful tool. According to experiments run during the spatial envelope project [27], GIST is able of analyzing images at a rate of 0.18 seconds per image (There might be a slight variation depending on the machine capabilities).

2.2.5 Flickr

Flickr [9] is an online social network that allows its users to share pictures and videos through Internet. Unlike Facebook, Flickr is populated by a community of photography enthusiasts, and pictures are usually uploaded with a larger resolution that makes it easier to appreciate details. A unique feature about Flickr is the possibility offered to users to tag their pictures.

Flickr has developed an associated Application Programming Interface (API) to be used by other developers in their applications. The API allows a large range of different queries. A developer can ask for a specific picture, any of its meta-data or a whole list of data based on GPS locations, tags, dates,... By using this query with the city name we are able to easily get a large amount of pictures up to the number we specify. The queries have been asked by using FlickCURL [17] which is a C library. It is one of the many possibilities we could have chosen to communicate with Flickr's servers. Even if Flickr has a moderation team, users are still free

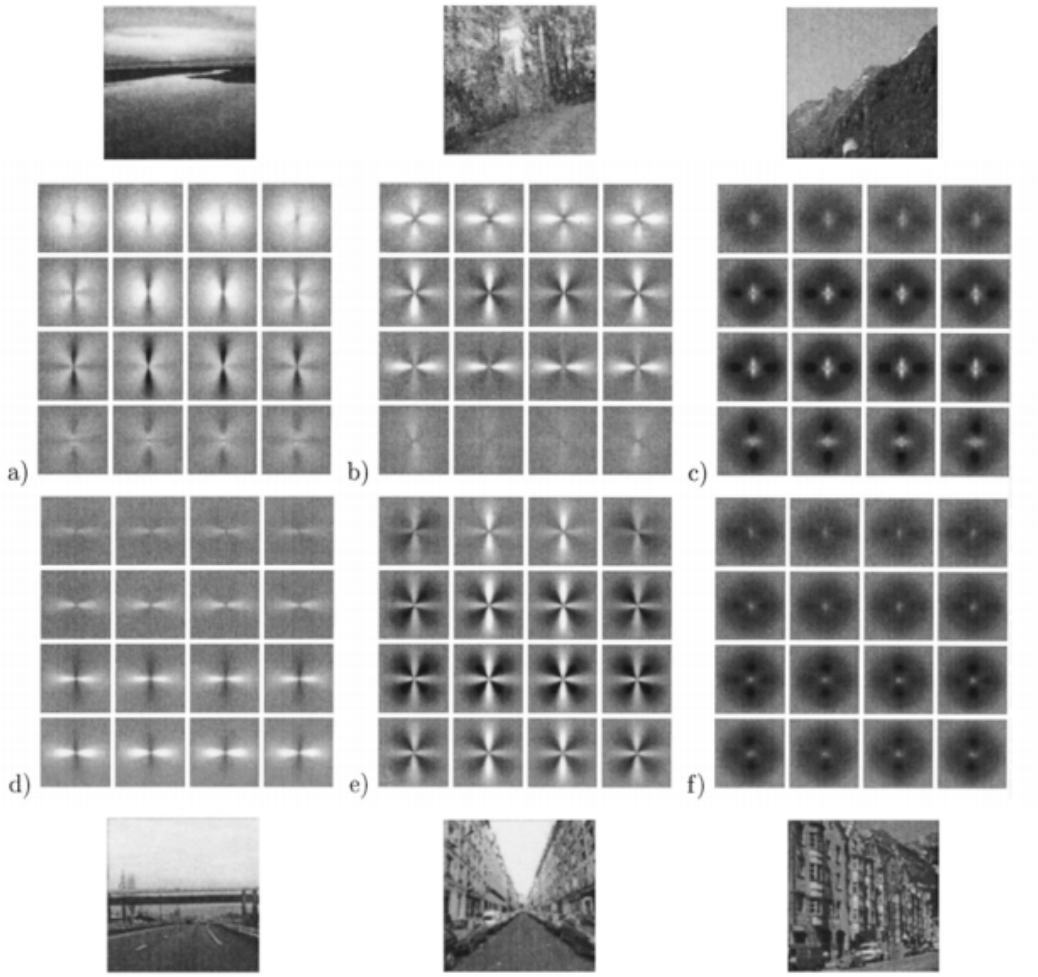

Figure 2.3: Examples of GIST descriptor's spectrograms extracted from pictures as in [24]

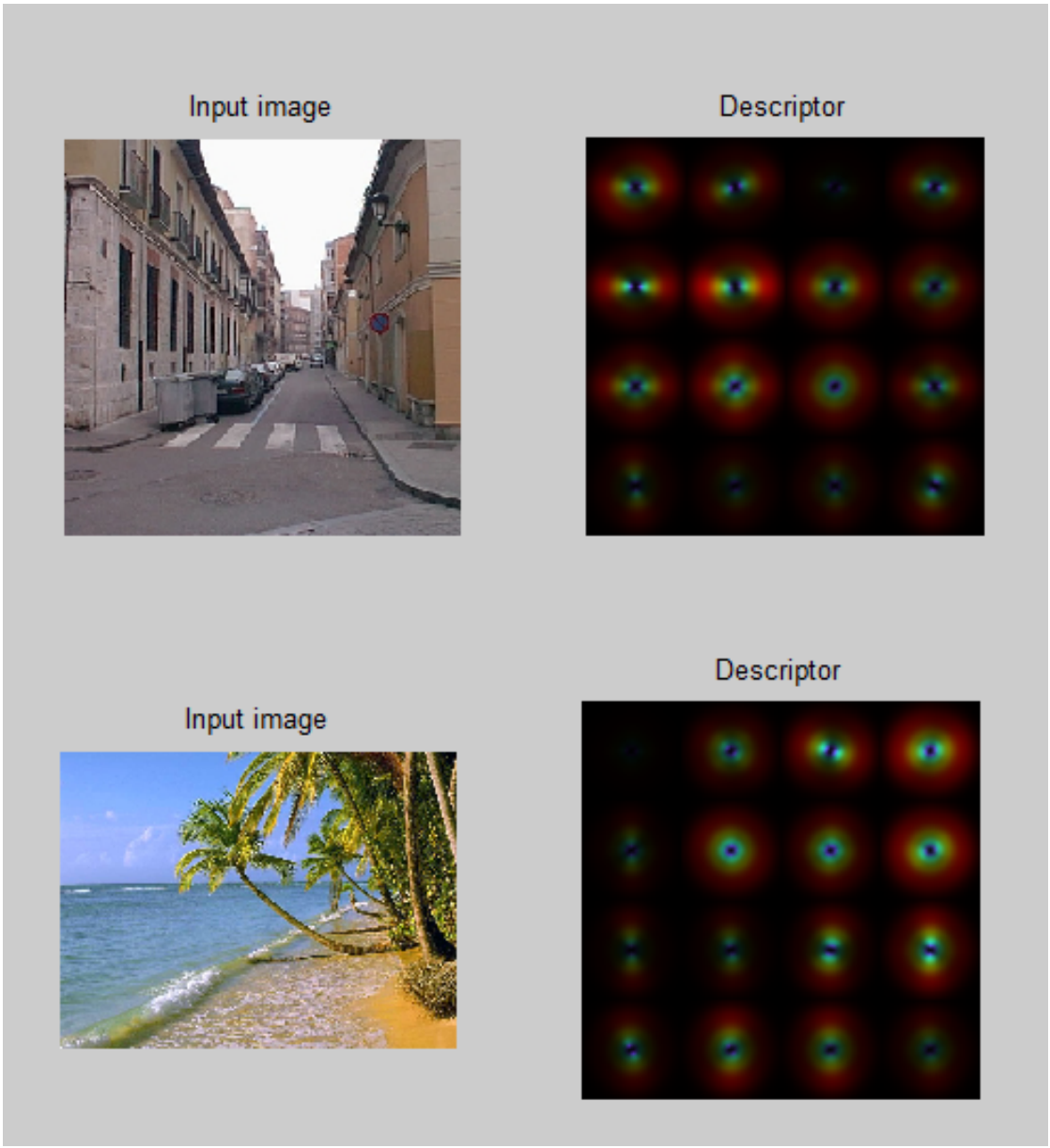

Figure 2.4: Examples of GIST descriptor extracted from samples images, taken from Torralba's work [24]

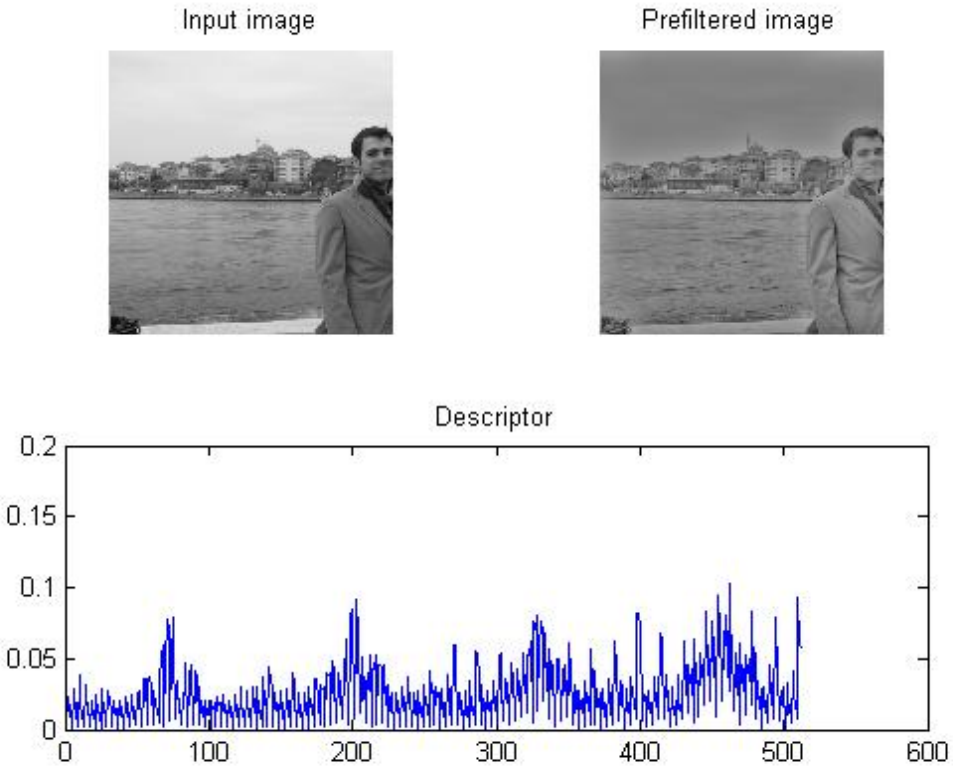

Figure 2.5: Extraction example of GIST descriptor and histogram representation as in [13]

to tag their images as they wish. This means that during a query there is no guarantee that it might return some incorrectly tagged pictures.

Chapter 3

A system for automatic generation of databases with landmarks and places of interest

During this chapter we are going to discuss the functionality of our system, as well as the functionality of every of its modules in the application. In Fig 3.1 we show a diagram depicting all the modules and inputs or outputs for each.

The process is started when the user inserts the name of the city or the corresponding coordinates of the region he is interested in. As an option, he may input a set of tags (e.g. nature, historical,...) that will narrow the search and the output. In the first module, given the name inserted before, a Query to Flickr will be made. As an output, this module will return the list of coordinates that are identified as the locations where Flickrs users took their photographs. In the next modules, given the list of coordinates we will apply clustering processes to get another list of GPS coordinates that will represent locations where FLickr users consistently take pictures. A second Query is now made an we will get a large number of images and labels associated to each of them. These imported pictures represent each of the locations from our previous list. To do that, we will associate the elements in the GPS list to a reduced number of images. The last module is an optional step. It is made to do small manual corrections for the coordinates, images or labels that represent each final location.

During the rest of this chapter we will do an in-depth study of each of the modules shown in the fig: 3.1.

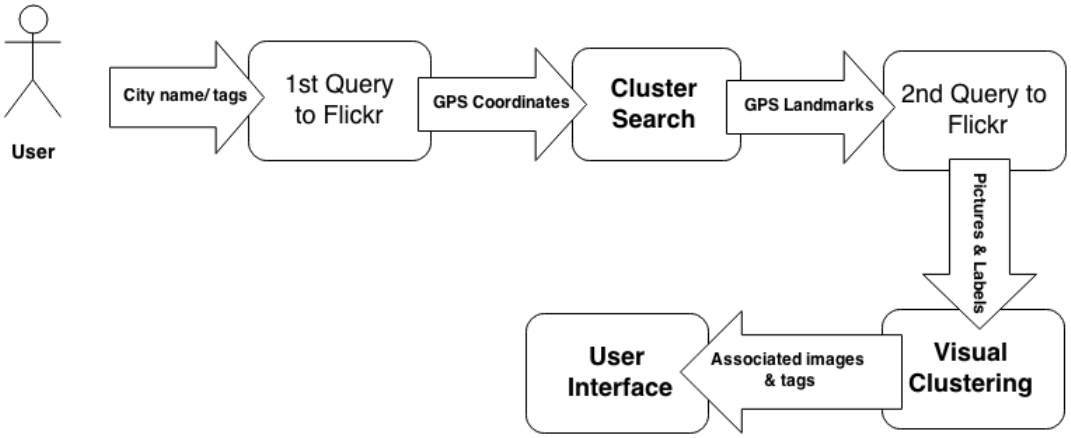

Figure 3.1: Diagram representing each module of the application

3.1 First Query to Flickr: getting a list of GPS coordinates

Any queries during our application or its previous experiments are using the libraries in C included in the official Flickr's API [11], FlickCURL to be more concrete.

This module will take the user's input, which will describe an area of interest (city, region, ...). The output of the module is a set of GPS coordinates corresponding to pictures taken by users within the geographical area defined by the user query. By using a simple query with the name or coordinates of a city we can easily get a large amount of pictures up to a number we specified (200000 in our case). Each coordinate represents the location where a picture was taken.

The exact same process is followed if we want to narrow search by using one or multiple tags. The difference is that we will only import those pictures tagged with all those descriptions. Meaning that the list of coordinates returned may be filled with the maximum number of samples we asked for a full list of samples if the tags are too restrictive. It is highly recommended that user searches for general terms such as 'night life' or 'entertainment' instead of using more colloquial expressions such as 'nice disco' for example. Spelling error might result in an empty list, or a completely different output than intended.

3.2 Spatial Cluster Search

This step can be considered as the keystone of the application. During this process the application uses the list of GPS coordinates we got from the previous Flickr Query to determine the GPS locations associated to places of interest; in other words, this module finds in which locations users consistently take pictures.

We will follow the diagram 3.4 structure and describe the process followed in this module.

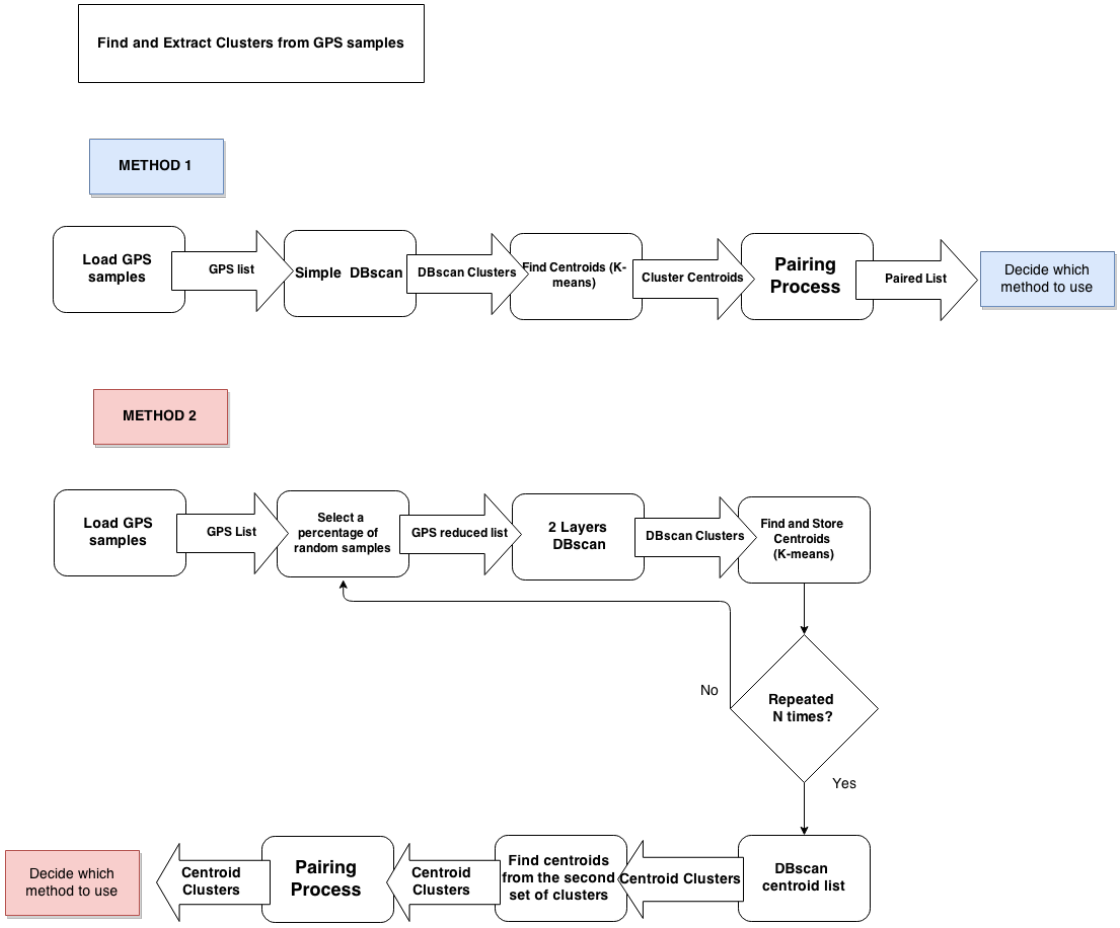

Figure 3.2: Diagram showing the different methods and their modules of the Cluster Search Process

Before describing the method, it is worth noting that we first convert the GPS coordinates into an Eculidean space. To do so, we first select an origin of coordinates located at the town center, and then compute the euclidean coordinates in meters corresponding with all the GPS locations. The GPS coordinates of the town center can be either retrieved from external datasets (Wikipedia, Google Maps) or computed as the mean of the available points. An in-depth discussion of these two alternatives is provided in the experimental section.

As one can see, we have considered two different strategies to perform this task. At the end of both methods, their outputs will be compared and the output with the largest amount of clusters will be chosen as a valid representation of the search.

For most cases, it is intended that method two will be chosen as the valid source of output. Yet, in some occasions, method one will be chosen if we face an unfavorable context. This way we will always have a reasonable output and our application will still work when facing problems.

Both methods share a common beginning. They will take and load the samples we got from

the Query and create a very large list of GPS points. As a first step both methods will transform this GPS coordinates into an Euclidean Plane measured as meters. The reference point is set as the previously set origin.

At this point our methods split and will end having different outputs. We will explain both in detail:

Method 1: Simple DBscan using all the samples. Here, we will take all of our converted samples (meters) and run a DBscan over all of them. Since we are working with a large database the DBscan parameters will be set max_dist=25 and min_points=220, considered as very restrictive parameters. Let us remember that, as we defined in section 2.2.3, max_dist stands for the distance that can separate two points belonging to the same cluster; whereas min_points defines the minimum number of samples that can be considered as a cluster. While 25 meters is a fixed variable, 220 points represents a small percentage of the total number of samples found (around 0.0015%). The rationale behind these values will be explained in detail during the experimentation chapter, but for now, let us simply note that they represent the values that provided the best results in our experiments.

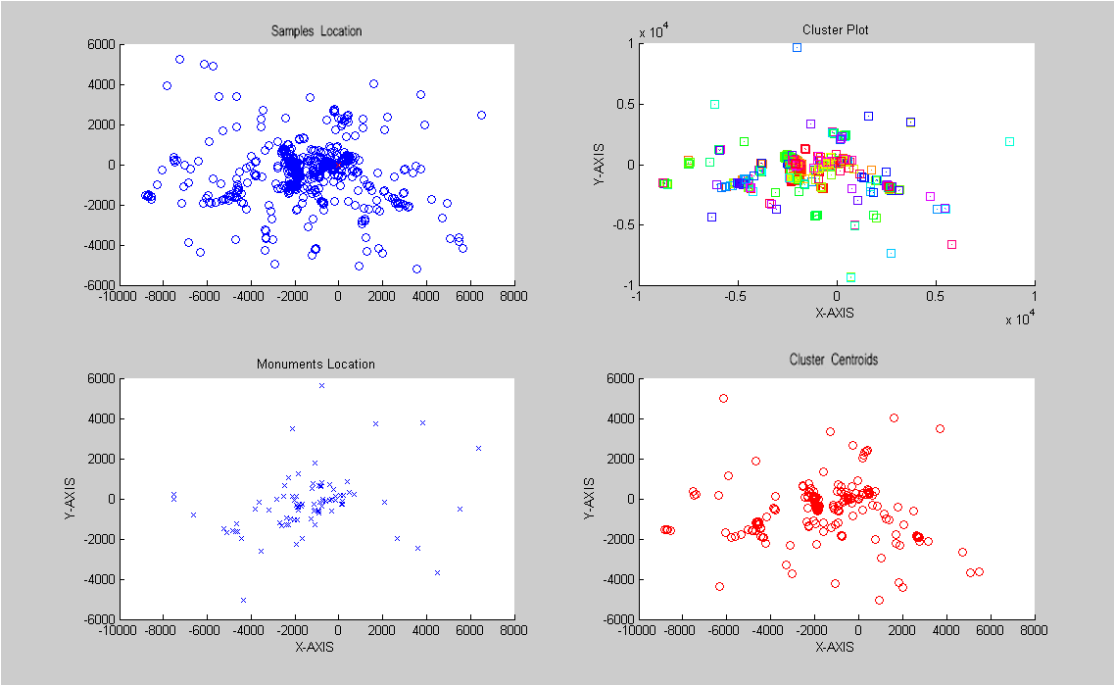

Figure 3.3: DBscan process and comparison

After finding the clusters and identifying which points they own, we now proceed to find a representative point for each cluster. The process requires to calculate the mean of all the cluster's points. The resulting coordinates will be considered as the reference point for the location of the landmark. At the end of this process, we will have a list of K points (Where K is the number of found clusters) that represent K landmarks. This will be the output offered by our first method. The figure 3.3 shows the steps followed up to this moment. In the top left, we see the representation of all samples taken from the Flickr Query (note that the axis are

measured in meters). To the right we find different set of points marked in many colors. This is the output after the DBscan, each color represents a cluster. In the bottom-left corner we find the locations of the landmarks retrieved from a Ground Truth database. We can compare the position of the real monuments with our estimations in the fourth representation.

Method 2: An in-depth statistical study using a reduced list. This method begins by randomly selecting a set of 30000 points from the query. We will apply a double layered DBscan to this set. When we refer to 'Double Layered' we are talking about two different executions of DBscan focusing on two different areas that we depict in Figure:3.4. One of the layers will apply a very restrictive DBscan to the inner region, is the one with the higher concentration of pictures taken. While the second DBscan, one more permissive, is to be applied to the outer region. The inner region is considered as the area inside a circumference with radius equal to 2 kilometers, centered at the origin (0,0). The rest of the area outside the circumference is considered as outer region, in this area we usually find parks and large surface landmarks which have a lesser density of samples.

It is worth mentioning that the parameters for the reduced DBscan were decided during the experiments too. For our inner DBscan we use min_points set to 120 and max_dist set to 40. For the other we can be more flexible, that's why we set our parameters to 55 meters and 100 points. Unlike the previous section this parameters are fixed for any number of samples (while larger than 30000 the output will be optimal).

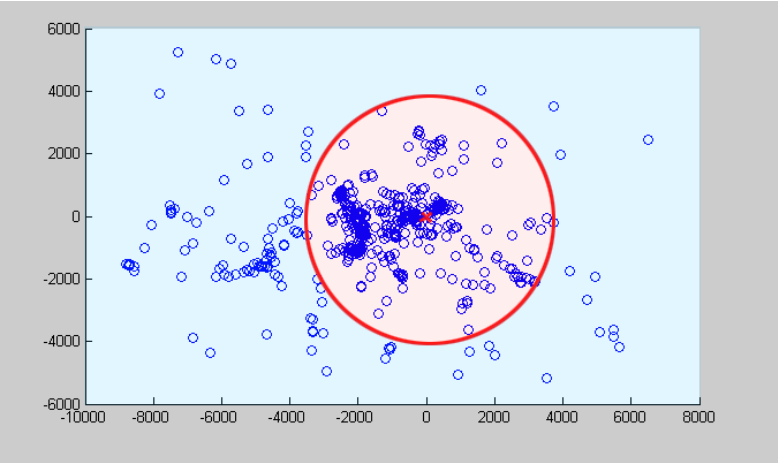

Figure 3.4: Representation of the two differentiated regions for Method two

Afterward we will find the representative points for both sets of clusters created during both DBscan executions. As in method 1, we use the mean for this task, following the exact same way as in previous occasions. Then, both lists are concatenated into a single one to facilitate the process.

So we have a set of points that represent the locations we are looking for. We are now taking a smaller set of clusters compared to the first method approach. By using less number of samples we find a lesser number of clusters. Also, since the samples are selected randomly, this method obtains different results among executions. Hence, in order to provide a robust list of places of interest which is not dependent on the algorithm initialization and randomness, we repeat

the process up to 20 times. Each time the results are stored in a list. At the end of the last execution we expect to have a list around 20 times larger than the previous which might also likely contain several repeated locations. Although this approach requires an additional step to remove the duplicities (see next paragraph), it helps to minimize the probability of missing locations of interest.

The next step is to run a third different DBscan over this larger centroid list. This clustering process will find which points consistently repeat themselves during different executions, avoiding missing clusters that by random chance may have not appeared in one of them. So now we can detect a cluster that might appear 1 of every 3 executions. Also, by using multiple clusters to determine a centroid, we are reducing the variation of a cluster when slightly shifting positions in different executions. By setting the min_points parameter to 20/3, we consider that a location is considered of interest if it appears on at least the 33% of the executions. Our list will be reduced to a slightly larger list than the one we would get from a single DBscan execution. This will be our output for this second method.

Comparison between both methods

There are a pair of advantages for using each of the methods:

By using the first one we will have a slightly faster running time. Additionally, if the previous modules are unable to find 30000 samples and we are forced to work with a smaller number, this method will adapt to this scenario, something that does not happen with the method 2. We simply need to modify the min_points for a more optimal DBscan execution.

Our second method is designed to optimize the precision of the returned list of locations of interest, it will find and separate clusters that the first method was merging into one. Even if we are not working with the full set of samples, its design will find every cumulus of pictures taken due to its statistical process.

As it was said before, during standard searches (cities or towns),in most cases the second method will generate better results. The first method mostly prevails when we do not reach the optimal number of samples to work with. As an example, it will determine the output when remote places being analyzed or when by using multiple tags we greatly reduced the samples found.

After all calculations have been done we reverse the meter conversion to get once again the corresponding GPS coordinates, which will be used to generate new queries for Flickr.

3.3 Second Query to Flickr: getting pictures for given coordinates

After we got a final list of GPS coordinates, we will ask Flickr to provide us with 100 pictures for each detected location in the list (always in case Flickr contains 100 available pictures). Each picture comes with its own set of tags or labels provided by the uploaders. These are the same tags that helped us to narrow our search during our first query.

3.4 Visual Clustering Process

This is considered as the final step needed to identify interest points in the location under our study. As inputs we receive, for each location of interest, both the pictures we got from our second query to Flickr, as well as the associated tags provided by the uploaders.

This module starts by computing the GIST descriptor for each downloaded image. Also we must define the parameters for the GIST parameter analysis. The GIST already has a function that allows to resize and rescale pictures. Default settings are images of 256x256 pixels with a division of a 4x4 grid to create the neighborhoods. And after the pre-filtering process (included in the GIST algorithm), we are left with a particular spectrogram that uniquely represents each picture. More in-depth details of this function are discussed during the GIST subsection in the Related Technologies section 2.2.4.

Once we got the descriptor based on the Spatial Envelope, the objective is to follow a visual clustering process to determine which images are representative of that location. Given that our image database is based on Flickr, it could happen that some of the images we got from the query can be wrongly tagged or allocated. Since some landmarks may be identified by multiple sights, we have chosen to represent each location by using 3 different images. By using K-means we can separate into N different cumulus, where N is the number of key pictures we want to obtain. So with N=3, K-mean will separate 3 different centroids by looking at the GIST spectrograms. Next, we will store all 3 and do a one-by-one comparative with each of the hundred spectrograms. The process ends when we have decided which image's descriptor is the closest to each of the 3 K-means selections. In this figure 3.5, we can see the spectrograms of both the K-means selection and the descriptor of the image chosen as the representative with the actual image below it.

Later in this document (chapter 4), we will explain why choose 3 images and not a different number.

3.5 User Interface

The outcome of our clusters might not be as accurate as we had hoped for. For small corrections and adding details we have designed the User Interface.

This is considered as an optional module. The user interface receives the outputs from both visual and spatial clustering processes. It takes the list of GPS coordinates symbolizing the monuments. At this point each landmark has 3 images and some labels associated to it so we can have a visual and textual representation of the location. This interface does small readjustments or helps taking decisions. It is designed to be able to create the customized map that meets the users expectations.

The Interface 3.6 is composed of a map (got from Google Maps Static API [3]) that gets centered over the location under analysis and a set of buttons that allow to control the longitude and latitude adjustments. Also it has an image selection panel which allows to pick or discard the images to represent the location.

One of the possible readjustments is to slightly move the GPS locations to get closer to the

location we wish. We can not do adjustments larger than 50 meters distance, else we might compromise the integrity of the process.

After we get the visual clustering process's outcome, we can judge if any of the chosen images does a poor representation of the location, and discard it by clicking a button.

Once we have chosen which images belong to the location, we use their labels to make a small description of the place of interest. This module has been previously developed by a research group [19].

This marks the end of the application. In the end the cluster saves whatever options were changed and discards previous values. The outputs remain with the format they came into the User Interface: a GPS coordinate associated with a single or multiple pictures and labeled by tags.

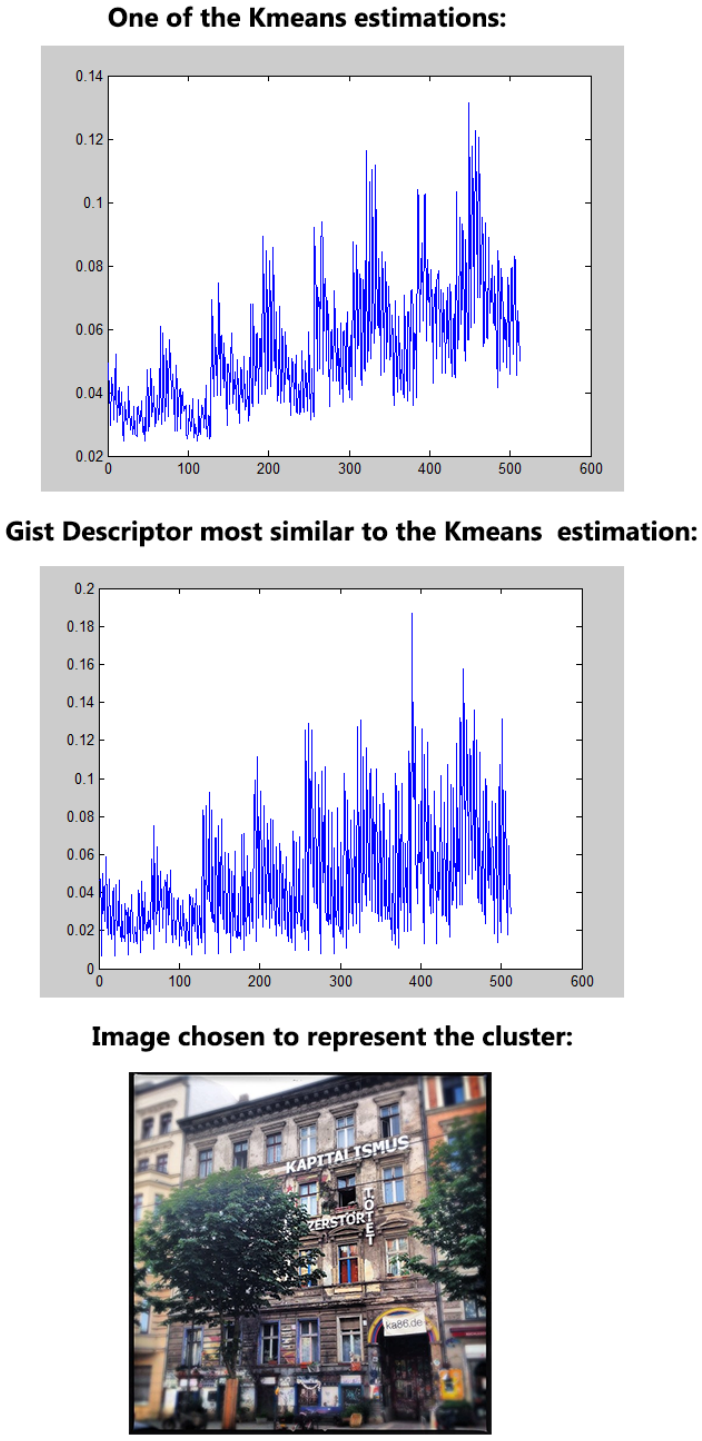

Figure 3.5: Selection of an Image using Kmeans over the GIST descriptor

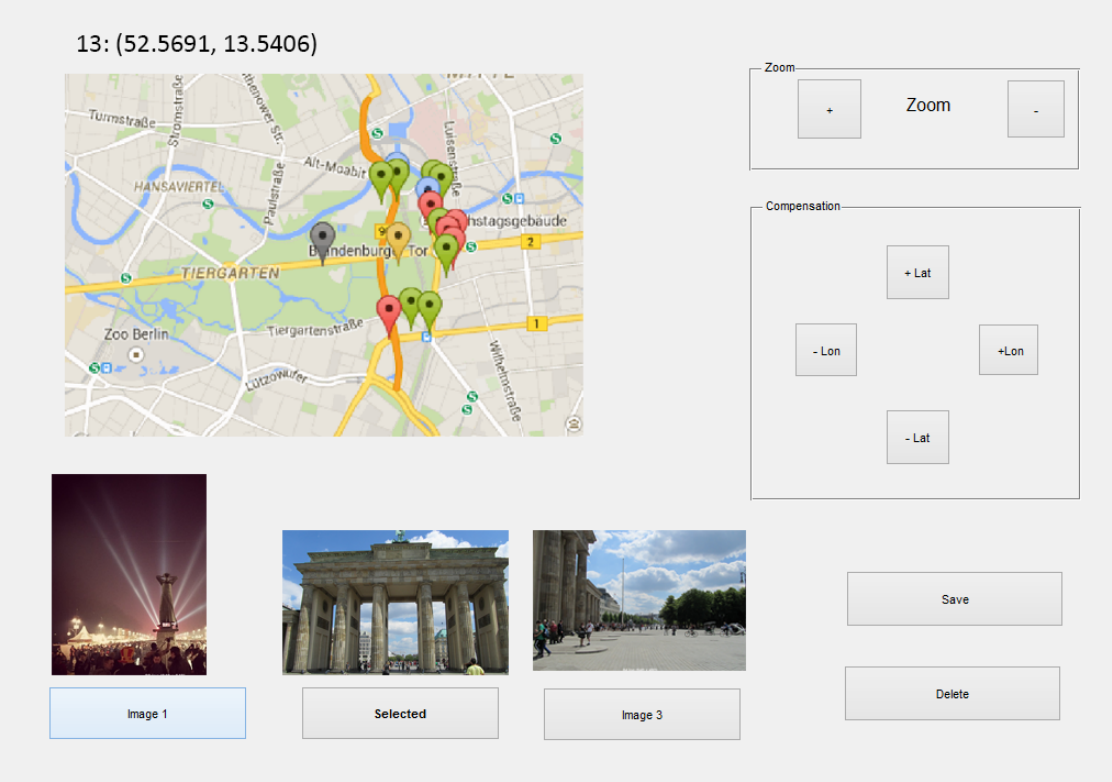

Figure 3.6: Picture of the UI implemented with GUI

Chapter 4

Experiments and Error Analysis

This chapter is dedicated to assess the different modules identified in our processing pipeline. The first part is related to justify the selection of a specific parameter set, and the use of certain functions.

The second part will focus on the explanation of possible errors and to provide solutions. Let us note that we are interested not only on the system performance, but also in its efficiency, thus looking for the shortest execution time at a given quality.

4.1 Experiments

We will now review all the project phases, as shown in Fig.4.1 It has to be remembered that our objective is to confirm if the obtained results were accurate enough.

Although the intended final version of the application should be able to find any tags for any town, during the early stages of development we decided to focus on a single city, Berlin. We have chosen Berlin because of its familiarity and its high number of historical and cultural places.

4.1.1 Ground Truth and Experimental Setup

Firstly we need a string input to begin our experiments. Since we focused on Berlin, the first step was to manually make a database, including all Wikipedia-databased historical landmarks [5]. We required this Ground Truth database to evaluate our results, in order to compare them to the locations shown in the final application. Our evaluation was based in the values of both Recall and Precision, and, in particular, on the the F-measure, which combines both of them into a single metric. We think that it is not necessary to apply the same process on more cities to assess the validity of this application.

Hence forth, the output for this module was based on the GPS coordinates of the whole

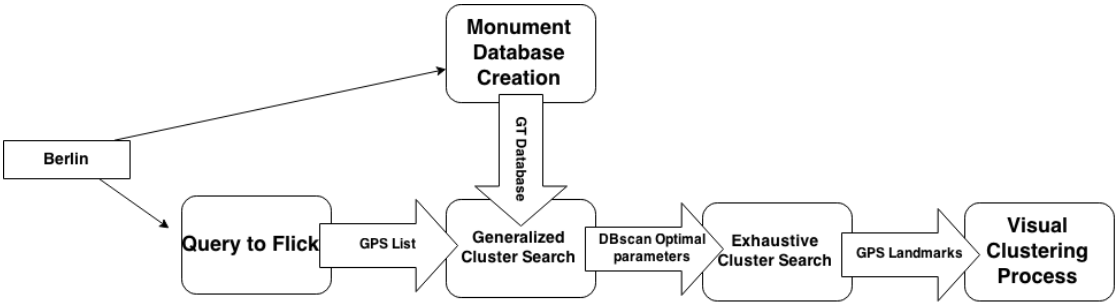

Figure 4.1: Diagram showing the different modules developed during the early stages of the project

list of the selected locations within the city. The Ground Truth List was composed by 112 GPS imported landmarks. It could be argued that some of these place are not relevant tourist attractions. We have still included them in the list. It has to be taken into account that we are designing an automated process, we are not going to do any manual selection.

4.1.2 Experiments with the GPS clustering module:

Before explaining the following steps, some decision-making issues had to be introduced:

Estimate the city center

To maintain this process as automated as possible, it is necessary to develop an strategy to determine the relative position around which all of our measures will organize themselves. We planned that the user could just simply introduce the city or town name and the script alone would show a set of clusters representing locations of interest. At this point, we needed to determine a reference point, so we studied two possibilities:

- Our firs option was to find an external content provider which could give us the coordinates of the geographical center of a population by a simple query. This is the chosen method for the present experiment. However, if we want to keep ourselves independent from third companies this might be a step-back.

- The other possibility was to determine and estimate a new point of reference as our reference point. The first possibility that comes to mind is to set our reference point as the physical mean point of all our samples. In our opinion, this is a good choice and in most cases would be an accurate approximation to the real city center. In fewer words, our reference point would be located where the pictures were taken. This fact is believed to help out the cluster search process, specially for our Exhaustive Search (method 2 of our cluster search module), which will be allowed to cover more concentrated areas with its radius. Another method to establish the city center would be by using more complex density-based algorithms, but it could bring in the risk that a very high density location could become

the city center itself. There should be future tests focused on this subject, but up to now we have got good feedback by using this process in Berlin.

Figures 4.2 are shown as an example. Both circles represent the radius in which the more precise DBscan is applied during the Specific Search. The red one represents the point returned from an external database. Below, in green, the reference point is represented as if it was established as the mean point of all samples. The distance that separates both points is not larger than 2 kilometers.

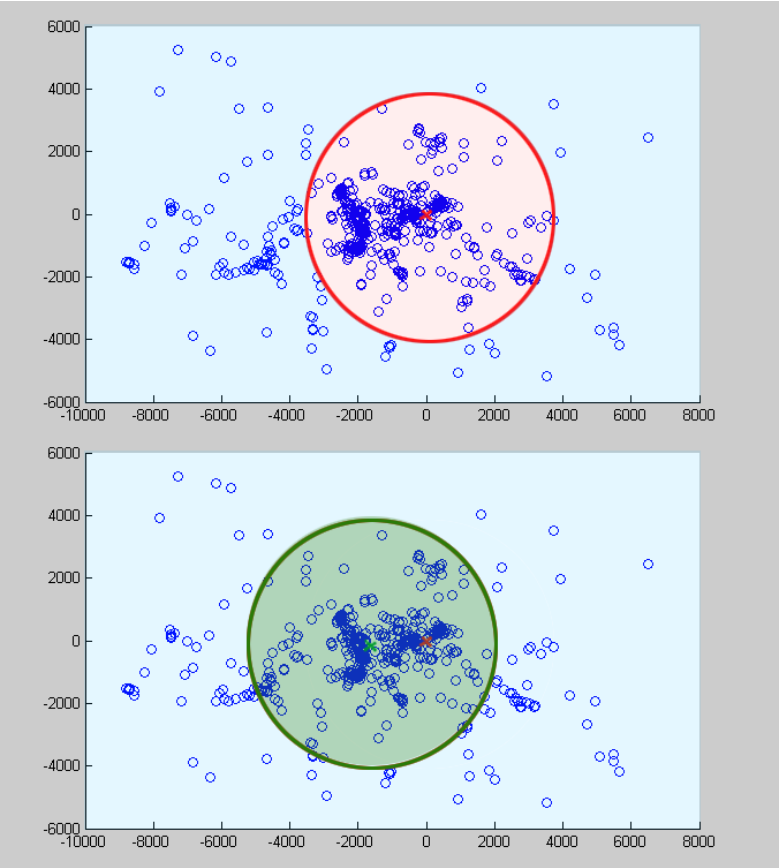

Figure 4.2: Cluster Search Comparison given 2 different reference points

Determining the representative point for a cluster

A disadvantage of using DBscan is that we do not get a simple representation of the cluster. Instead, we simply get the list of data that corresponds with each cluster. It is true that DBscan is capable of differentiating between the internal and peripheral points of a cluster, but that is far from being represented by a single point.

Hence, in order to provide a unified representation of a cluster (the GPS coordinates of the location of interest), we have simply computed the mean of all the points that belong to the

cluster.

In the figures: 4.3, 4.4, 4.5, 4.6; we show some examples of the location of the centroids after determining its representative point. We have represented some unconventional silhouettes to represent its behavior in extreme cases.

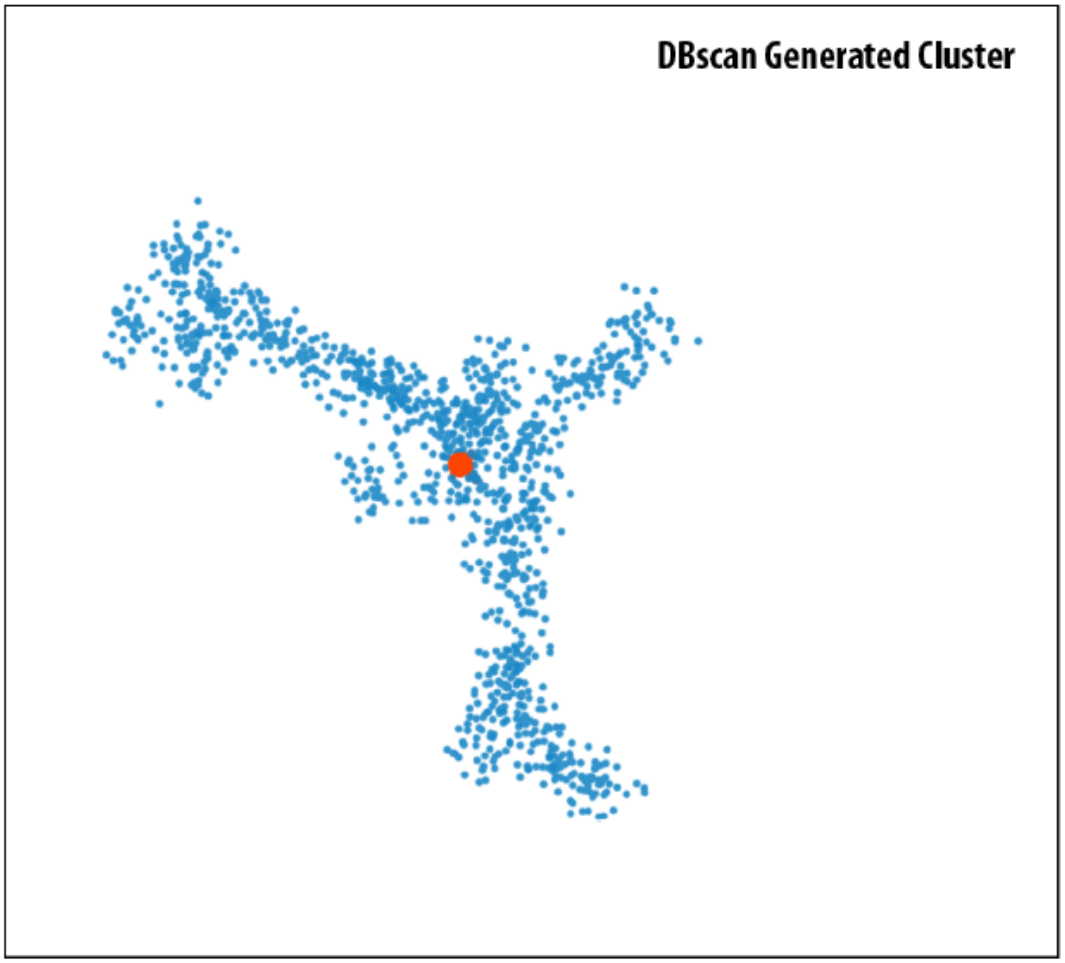

Figure 4.3: Centroid of a DBscan generated cluster

Around the 70% of the clusters could be considered as circular silhouettes (figure 4.6), and their center is usually occupied by a large cumulus of points. This high concentration will be helpful during the visual clustering stage, where we will gather pictures in a small area around the centroid. If our application can find the number images it needs without going far away from the center, then there is a higher chance that most of the gathered images point to the same sight, greatly easing the visual clustering process. Other clusters have more singular silhouettes: figures4.3, 4.4. By using the mean we are locating the representation of the cluster in areas with a high density of pictures. There are exceptions, see figure 4.5, in this case the reference is located outside the cluster. This may cause that import the wrong pictures when executing our second query to Flickr.

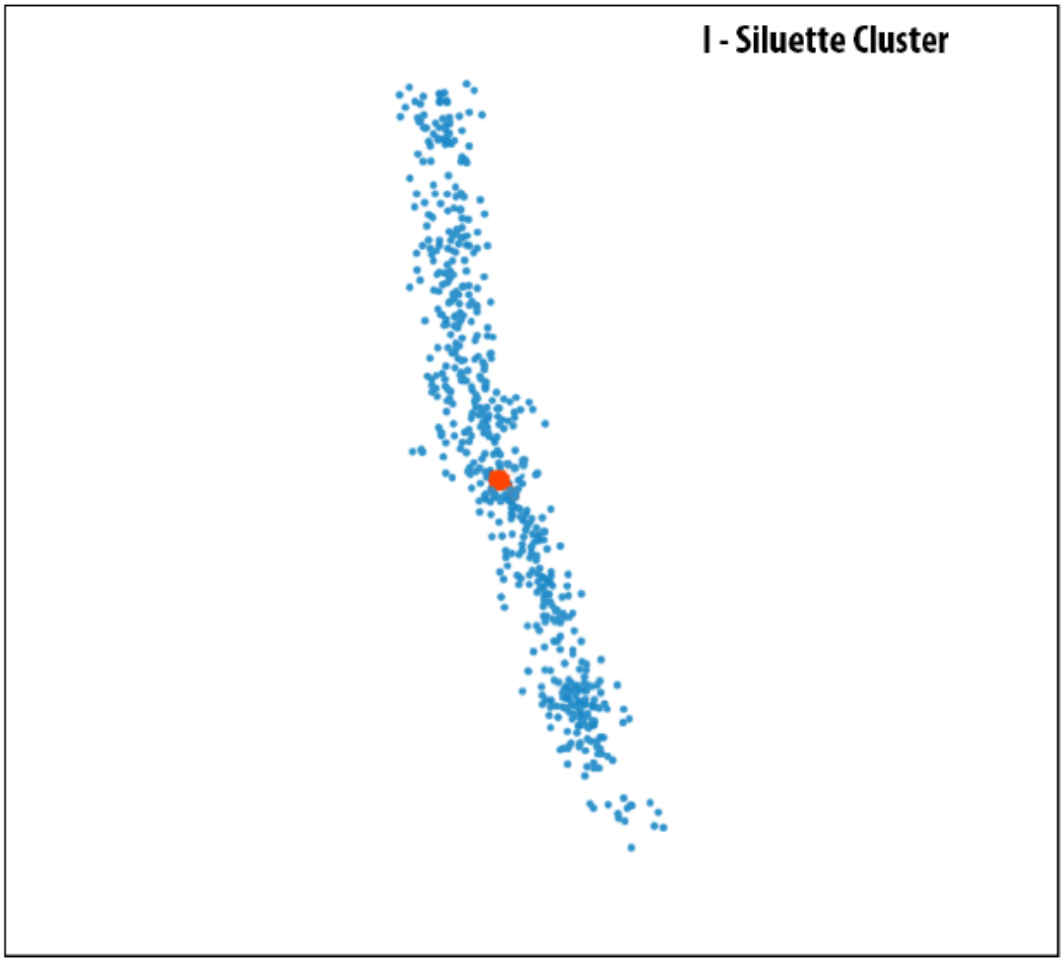

Figure 4.4: Centroid of a DBscan generated cluster I-silohuette

Validation of DBscan parameters

The process designed to find the optimal parameters of DBscan is depicted in Figure 4.7. It begins by parsing Cartesian coordinates into an Euclidean plane with the city center as the reference point (0,0) or origin. Meters is an unit which is easier and more intuitive to manipulate. The process only parses a percentage of the whole number of samples.

Trying to keep the computational complexity bounded, we start by randomly select a subset of samples (GPS coordinates) to be used in our experiments. Then we proceed to do our DBscan analysis. In order to find the optimal set of parameters, a two dimensional array of (max_dist,min_points) values will be evaluated. For each pair of values, a list of GPS coordinates is obtained using DBscan. Afterwards we define which points belong to which cluster and then we find the mean point for this set to find a reference point to represent the whole cluster.

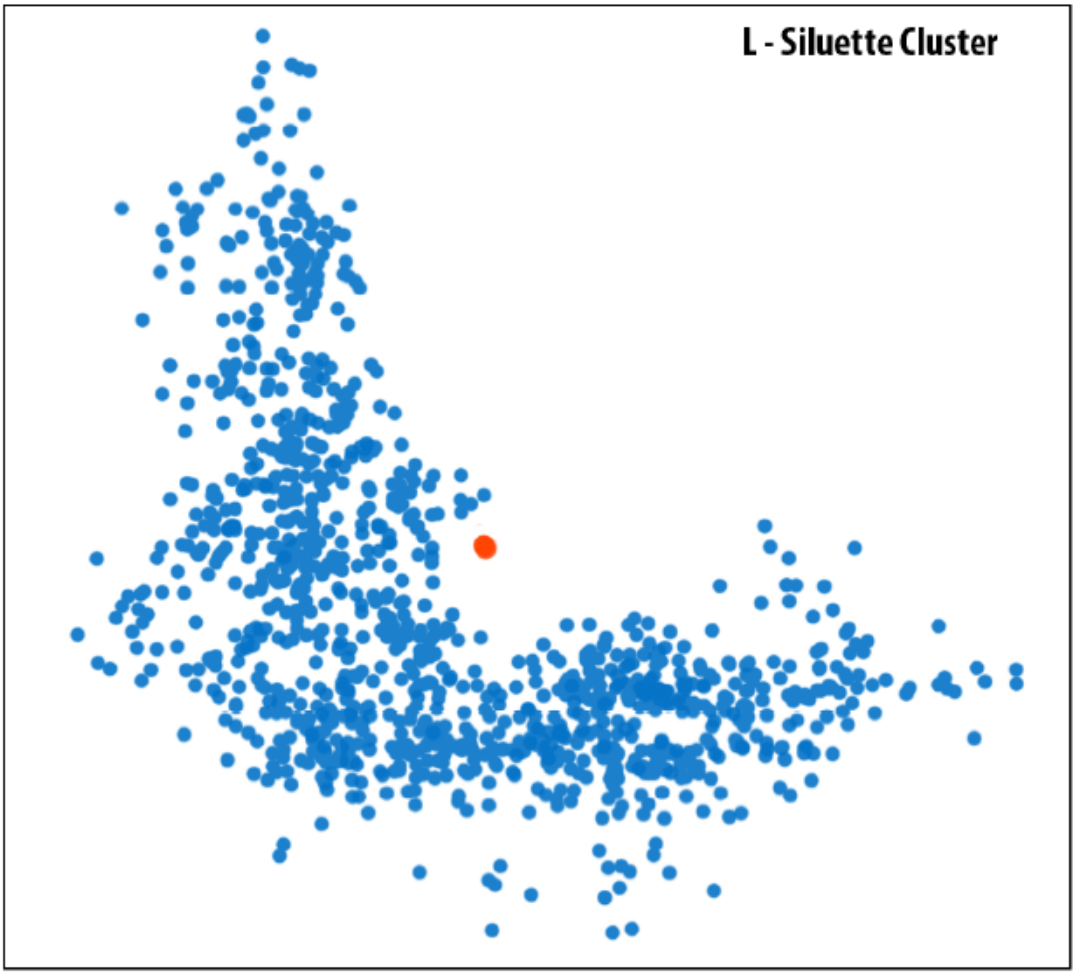

Figure 4.5: Centroid of a DBscan generated cluster L-silohuette

Before we commence deciding which monument should be represented by which cluster we must set a hierarchy. The reason is that when we try to sort out the pairs, some cluster may choose the same Ground Truth coordinate. The order that the clusters will follow to be paired with Ground Truth elements is the number of points they are conformed of. This means that the larger clusters will be the first ones when choosing a monument from the list.

In order to provide a quantitative evaluation for each pair (max_dist,min_points), once a list of GPS coordinates is provided by DBscan, we need to follow an alignment process between this list and the ground truth list described in section 4.1.1. For that end, we created a pairing module.

The pairing module is quite simple. A cluster and an element from the Ground Truth list become paired when the distance between both is smaller than the threshold that we are studying in that moment (by default we work with four different thresholds). When making a couple

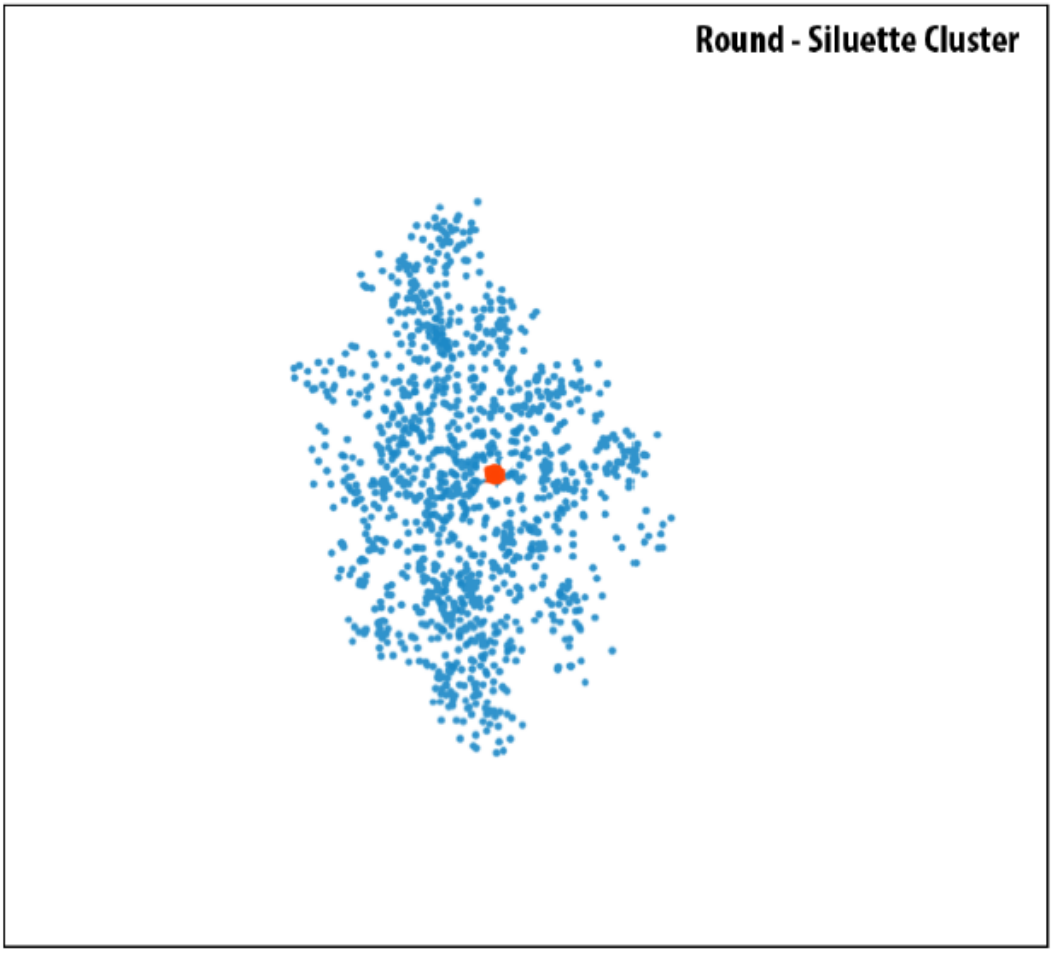

Figure 4.6: Centroid of a DBscan generated cluster round-silohuette

Monument-cluster we must consider some requisites:

- The monument must not have been paired before. To follow this rule, when a monument has been paired it will be automatically removed from the list and will not be available during the rest of the pairing process.
- Clusters must follow an order to decide which gets to choose first. The order is very simple, clusters which contain more samples are considered more relevant, and therefore, they get to choose first.

Each time a pair is found, the points conforming it are stored in the Pairing Matrix. To increase the number of data we are extracting from each DBscan, we do four different pairing processes, one for each threshold value: 50, 100, 200 and 300 meters. So after each DBscan execution, there will be four different 'Pairing matrices' for each combination of DBscan parameters.

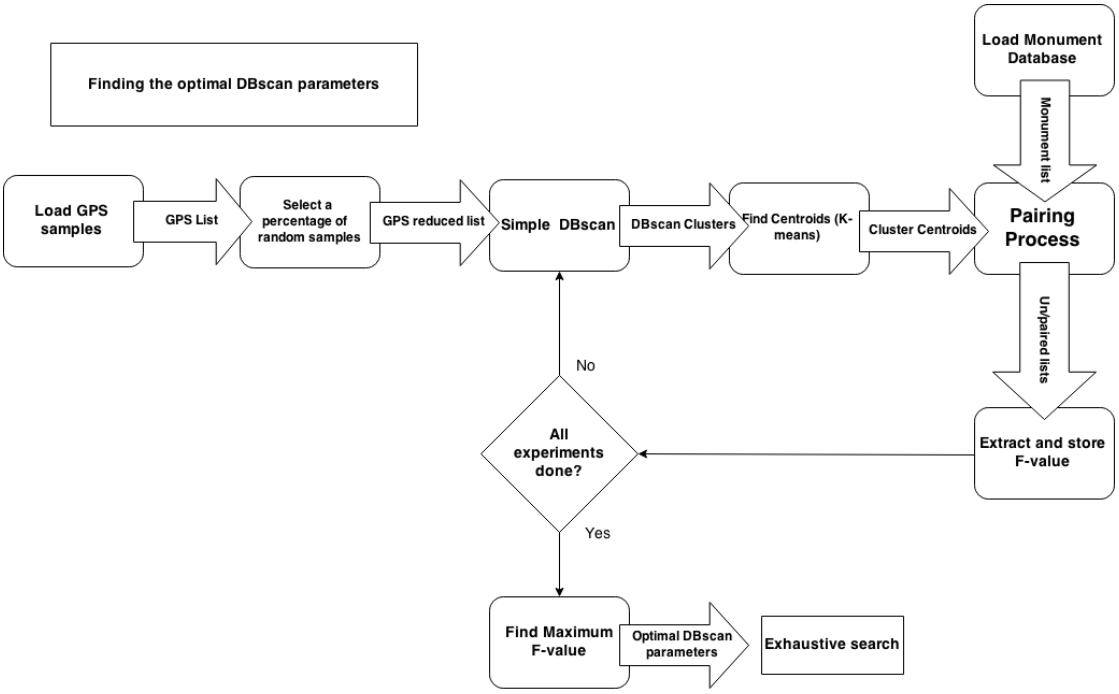

Figure 4.7: Diagram showing the process of finding the optimal parameters

These matrices store the pairs while the unpaired elements (clusters and monuments) remain in the original matrices. This way we can easily determine which have been detected, which are false alarms and which cases have we missed; it is of great help for the F-value extraction.

F-value extraction and result evaluation

Once we have generated the list and we know the situation for every monument and cluster, we evaluate the results using the F-value which is based on precision and recall values. To compute the F-value we require to know:

- True Positives: refers to those clusters (or monuments) that have successfully found their pair in the Ground Truth list.
- False Positives: the equivalent to False Alarms, it refers to those clusters that have been left unpaired, meaning that no Ground Truth coordinates have been asigned.
- True Negatives: they are irrelevant to our calculations and they do not have a representation in our process.
- False Negatives: these ones are a reference to those GT monuments that have not be assigned to any automatically computed cluster.

Remember that this process is repeated for every value of the pairing threshold and that we find the F-value for each of the different DBscans databases that were made in the previous step.

At the end we will have a N by M table where in N are represented the different values of Min_Points and so does M with the Max_Dist parameters of the DBscan step. We name this table the F-value matrix, and we will find four different versions as there are four thresholds to evaluate. This table is represented with a mesh, as in figure 4.8. The mesh can display how changing any of the parameters can alter the outcome of F. We can easily determine which set gives the best results by finding the maximum values of F. This set of parameters will probably change slightly during different executions but will have a bigger difference when changing towns or cities.

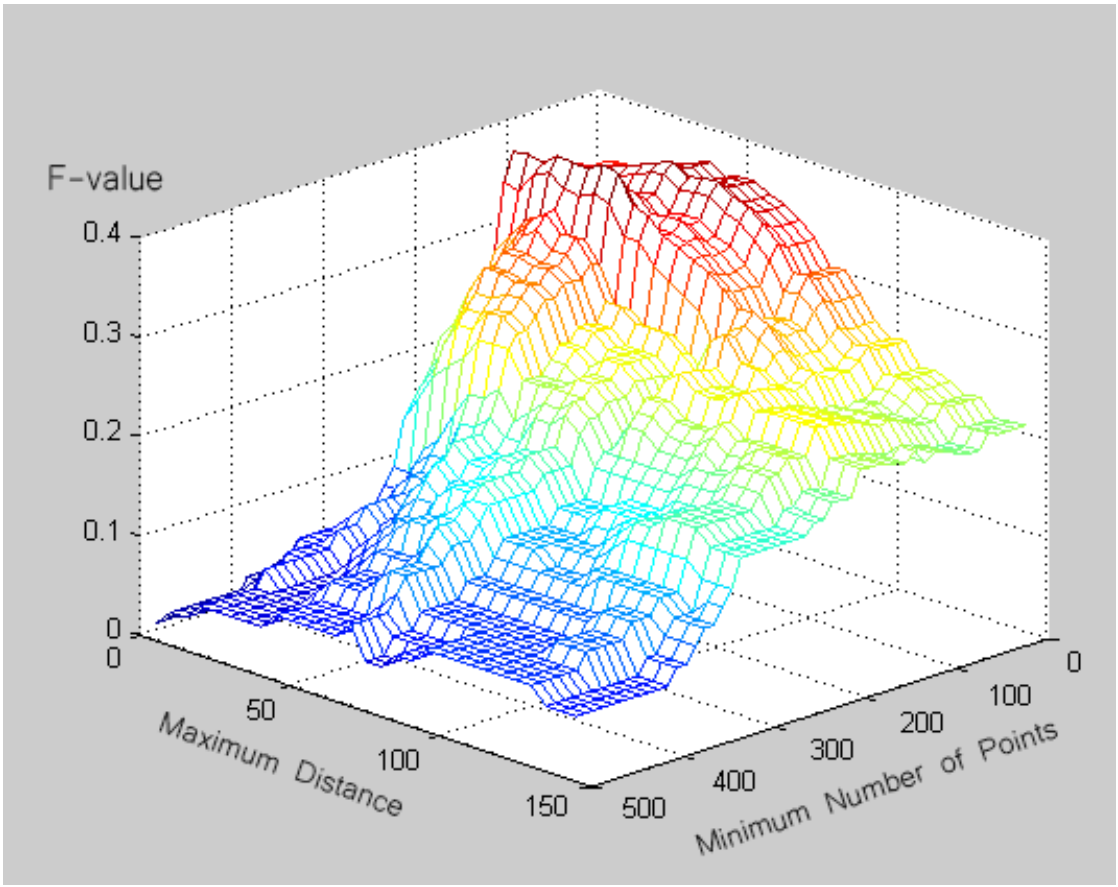

Figure 4.8: Display of a mesh representation of the F-value using 30000 samples and Threshold 200 m

This process will be repeated for 1372 times in a standard analysis (if we have not modified the DBscan parameters) trying all possible combinations in our F-value matrix. This means that we will do around 5000 calculations of F-values by default. At the end, our F-value matrix will have N-by-M-by-4 dimensions.

Having all the possible F-values, we find the highest F-value of a specific pairing threshold and determine the associated DBscan parameters. In the past, we experimented by finding the highest F-value of all Thresholds, but the calculations did not point to a more accurate solution than choosing an intermediate threshold (100 or 200 meters). The optimal values we extracted

followed the largest F-value for the 200 meters threshold. The values were min_points=120 and max_dist=40. These are the values we use during our cluster search in the application and also in the incoming experiments.

Let us note that the larger the Threshold is the larger the F-value will be, meaning that 300 meters will always have a larger F-value than 200 meters, as a consequence of being more permissive. The values we extract from here will serve us for both the next experimental step and the official application. It is also noteworthy that, for most cases, the number of clusters has a positive effect on the size of the F-value, the more clusters we find, the larger F-value will be.

Optimal Max_dist and Min_Points

As we made some experiments, we realized as it was predictable, that there was a relation between the chosen number of samples and the optimal value of Min Points we should use to maximize the F-value. In few words, the more samples we use, the larger the DBscan parameter should be.

After some experiments (see table: 4.1), we are able to estimate the DBscan parameter relation. Their behavior can be seen in the figure 4.9. We can deduce that the parameter Min_points has a direct relation with the number of samples. And in the other hand Max_Distance does not really follow a clear increase or decrease as we use more samples. It seems that the optimal value trends to be set between the 30 and 40 meters with slight variations. We store some values during our experiments in order to avoid doing the 'DBscan parameter validation' multiple times. In the previous table 4.1 we can also see the time each DBscan execution took when using different numbers of samples. The results are reflected in the figure 4.10. This process has a very high computational cost, taking around 10 hours to process for 30000 samples.

N Samples	Selected Min_Points	Selected Max_Dist [m]	Execution Time
1000	10	70	219
5000	20	35	3246
10000	40	35	8871
25000	90	25	30704
30000	110	40	72413

Table 4.1: Table displaying the Optimal values and executing time for a N-samples execution

By using this estimation we can find the value that should optimize the second phase search process. And for future executions, we can avoid to spend large amounts of time in discovering with DBscan parameters to use by simply looking ate the figure 4.9 and taking such values for our phase two.

Comparison of clustering strategies

This module, represented in figure 4.11, is focused in getting the best possible representation of the interest points of a landscape. It gets the maximum possible F-value out of a set of optimal parameters. These are the same parameters we got from the DBscan Parameter Validation process, section 4.1.2.3. Our results here the capacity of finding through cluster search that this

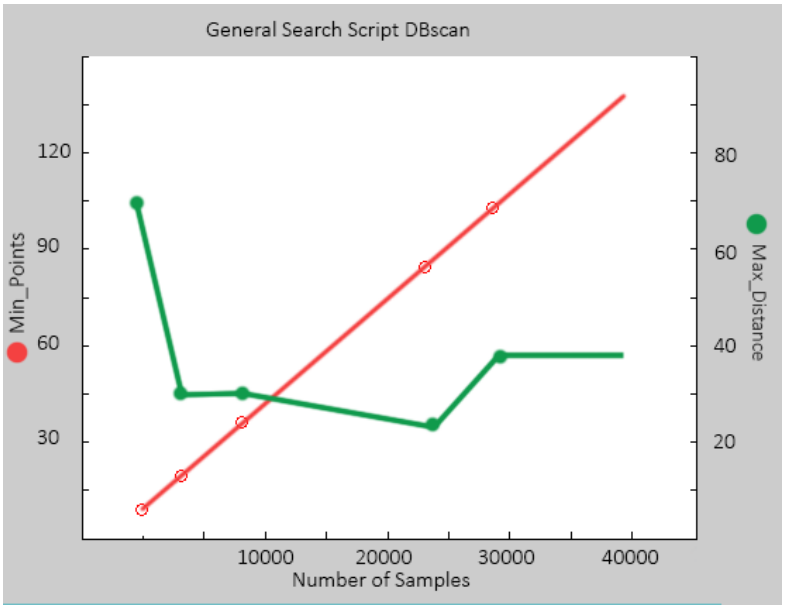

Figure 4.9: DBscan parameter relation

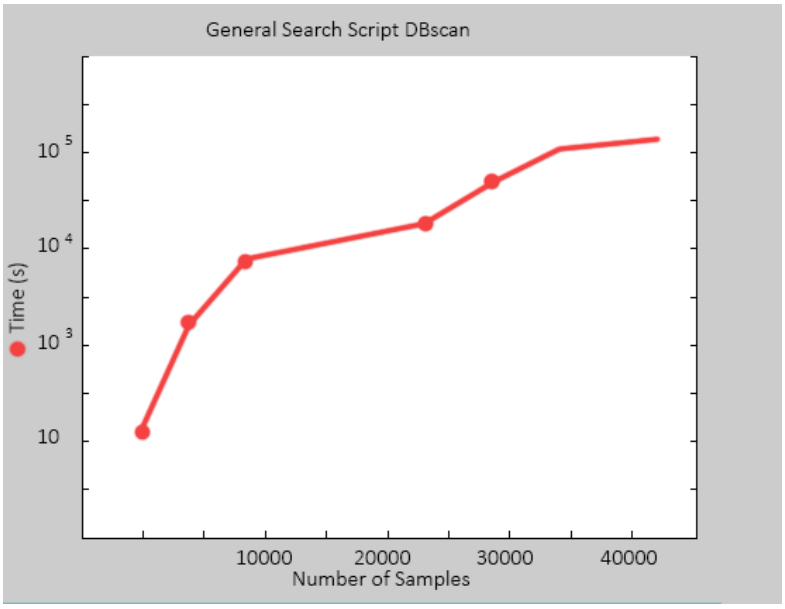

Figure 4.10: Graph sampling the time and number of samples relation

project has. As a second task, in this section we will determine the effect caused by statistical variation or the impact when using a reduced number of samples.

As you may see this process holds 2 different procedures.

- **Method 1: All samples DBscan.**
- **Method 2: An in-depth statistical study using a reduced list.**

Both share many similarities with the Spatial Cluster Search analysis in section 3.2. The difference is that in this section we are focused in result analysis. Extracting and comparing different F-values will help us determine the success of the procedure, proving the validity of our Spatial Cluster Search module.

Multiple 30000 Sample DBscan compared to 200000 Sample DBscan. also known as comparation between methods 1 and 2. Note that we were unable to experimentally find optimal values for the second method. Validating the parameters of a 200000-samples experiment will take days, so we decided to be permisive given our estimation figure 4.9. Parameters and results can be examined in the table 4.2. We notice that, even if we considered the second method as a superior searching tool, their F-values a very similar. The reason behind is that the F-decreases when more clusters are left unpaired. However method one was able to find and pair more of the clusters, this is what we are trying to do in this project and justifies choosing method two over method one.

	Samples	MinPoints/MaxDist	Clusters Found	Clusters Paired	F (TH=100)	F (TH=200)
Mthd. 2	30000 (20)	120/35	137	58	0.2651	0.3835
Mthd. 1	200000	400/20	118	52	0.2634	0.3689

Table 4.2: Table representing the different F-value and optimal parameters of the two DBscan executions

For a more in-depth study of method two we tried multiple executions with different sets of parameters. We stored a performance summary in this table 4.3. First note that we have duplicated executions (see results for ids 4-5 and 6-7). This is to study the variability within executions. As we can see the F-values for both pairs are almost identical, these are great news, we can say that method to is avoiding variability issues. As we predicted through the Validation module the highest F-values derive from our optimal parameters (120 and 40).

4.1.3 Experiments over the Visual Clustering Module

This is considered as the final step needed to identify interest points in the location under our study. Using the results provided by the Spatial Cluster Search module (the final list of GPS coordinates), we perform a new petition to Flickr asking for 100 images located around each detected location of interest. It is worth noting that, for this module, we lack methods to generate Ground Truth data-sets without too much human manual effort. Hence, a quantitative assessment of the algorithms similar to that one provided for the previous module is not feasible.

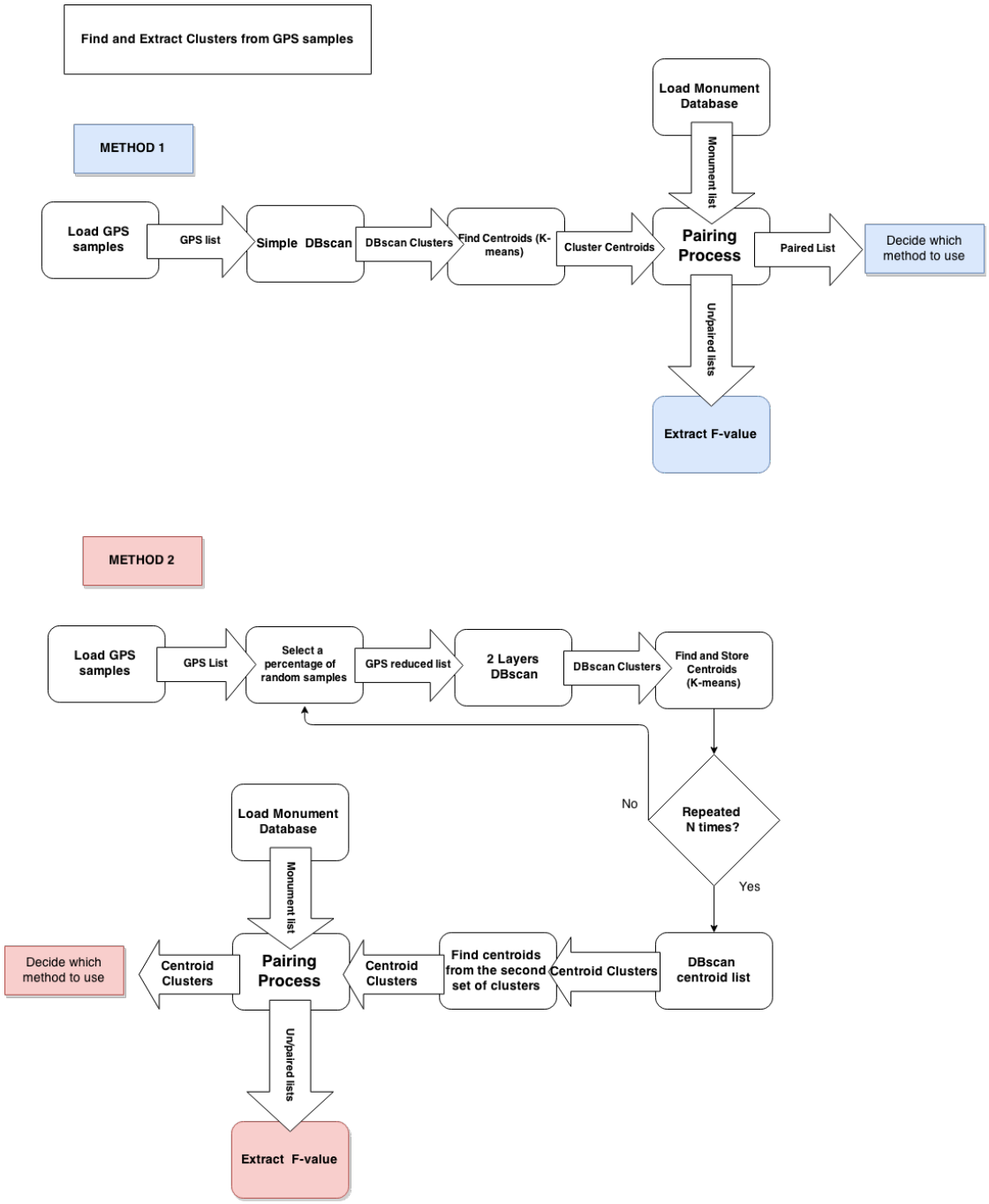

Figure 4.11: Diagram showing the process of an Exhaustive Analysis (Phase 2)

ID	Samples	N Datasets	Min_Pts(P1/P2)	Max_Dis(D1/D2)	TH=100	TH=200	Exe.Time(s)
1	30000	20	120/120	35/35	0.2459	0.3442	1716
2	20000	20	100/100	40/40	0.2378	0.3077	491
3	10000	20	70/70	40/40	0.1440	0.1440	147
4	30000	20	120/130	40/40	0.2570	0.3775	1236
5	30000 (2)	20	120/130	40/40	0.2500	0.3710	1273
6	30000	20	100/120	40/30	0.2400	0.3378	1222
7	30000 (2)	20	100/120	40/30	0.2511	0.3478	1301
8	150000	20	110/110	30/30	0. 2374	0.3525	14530
9	150000	20	500/300	30/50	0.2143	0.3003	93245

Table 4.3: Data store from multiple Method 2 executions

Instead, in order to select the optimal value of the parameters involved in this processing module, we have followed a qualitative evaluation of the results by looking at the resulting images.

Visual Clustering Design

In this section, we describe the experiments we have made to decide the optimal value for the N parameter: the number of visual representations (images) associated to each detected place of interest. By using k-means over the GIST descriptor of the downloaded images, we divide the whole image set into N different cumulus, where N is the number of key pictures we want to obtain. So K-means will separate N-centroids and we will store them in a N by M (length of GIST descriptor) table. The final step is to identify the image that gets the closest value to each of the N different K-means descriptors, as this set of images will become the visual representation of the landmark.

Determining the Number of Pictures extracted by Visual Clustering

In this section, we describe the experiments we have made to decide the optimal value for the N parameter: the number of visual representations (images) associated to each detected place of interest. Another decision to be made. Do we represent all clusters with the same number of elements? Some places like museums of markets might have many different expositions or shops to visit. Problem is that this differentiation should be made by labels. For now we look to use the same number disregarding which kind of location it is. Other possibilities will be contemplated during the Future Work section.

Note that, from the possibility of choosing a wide range of pictures to represent a landmark, we have only chosen a number of 3. Originally, the experiment was first based on selecting a single picture for each discovered landmark. We soon realized that this was not the way to proceed, so we chose to experiment using a different number of pictures.

When using a single representation, we were missing out on some of the multiple sights that the location was offering. Additionally, there was a high probability of selecting the wrong picture. This was considered as mayor risk. We took the most practical approach. We represented each location by using multiple images instead of one. We judged the results by our own subjective criteria. The figures 4.12 4.13 4.14 4.15 4.16 showed some examples that helped us in our decision. We estimated, by looking at many experiments, that we needed at least 3 pictures to represent the different sights that a location could offer. Also, we detected that by using 4

or more representations we got duplicated sights over the same place. In addition, as this is a selective process, we wanted to keep the number of possible representations to a minimum. Therefore, we decided was an ideal number for our purposes.

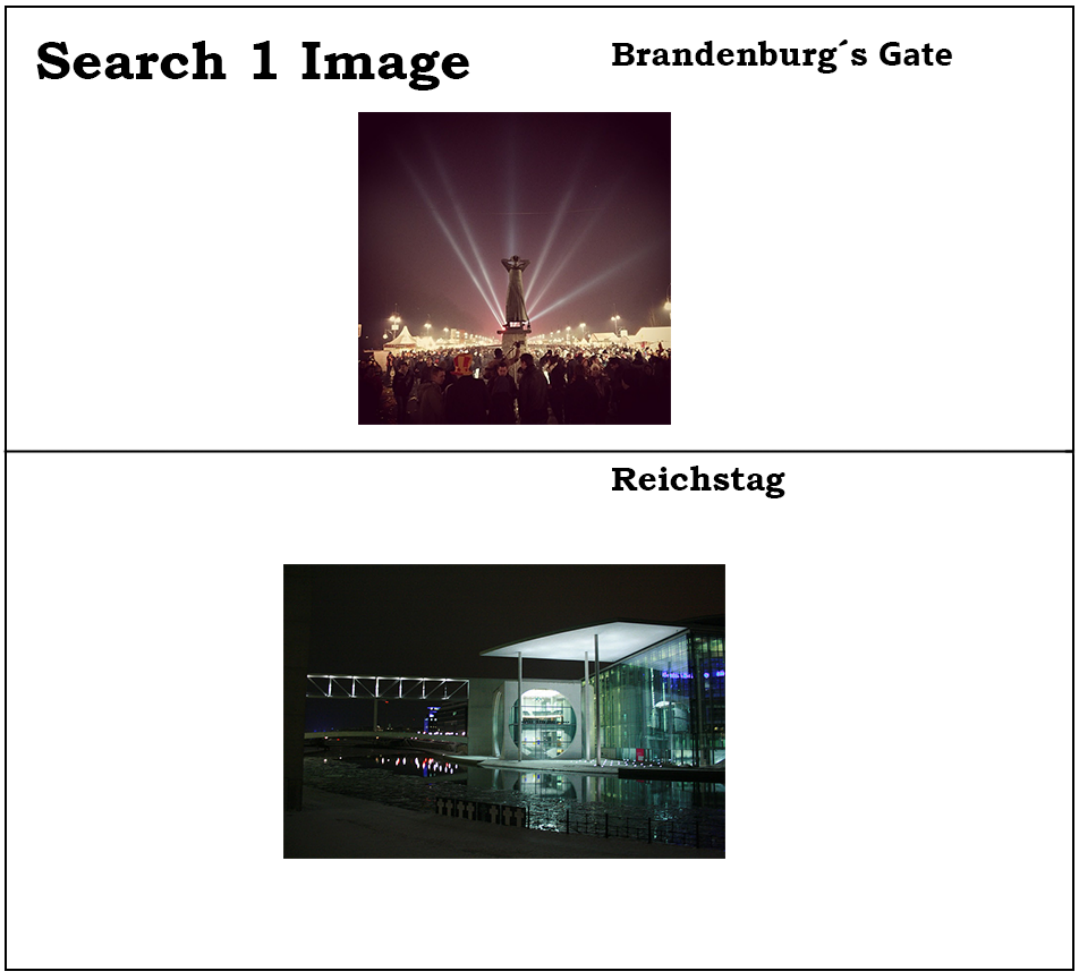

Figure 4.12: Set of 1 selected pictures to represent different cases

Taking into account, that we have to consider that the imported pictures come a public source. The variety of sights that we are getting for each location depends directly on the uploads made by the Flickrs users. We are also getting wrongly tagged uploads. These pictures are usually inaccurate o totally unrelated, they are inconveniences that confuse our K-means algorithm. In most cases, these images do not influence the outcome and can be ignored since the bast majority of imported images are relevant and K-means provides a statistical study of the most frequent cases.

There are some particular cases which can be represented by more than 3 different sights. This could be the case of museums, parks or some large structures. The figure 4.17 shows a museum as an example of a place with multiple representative sights. For these cases, we have contemplated

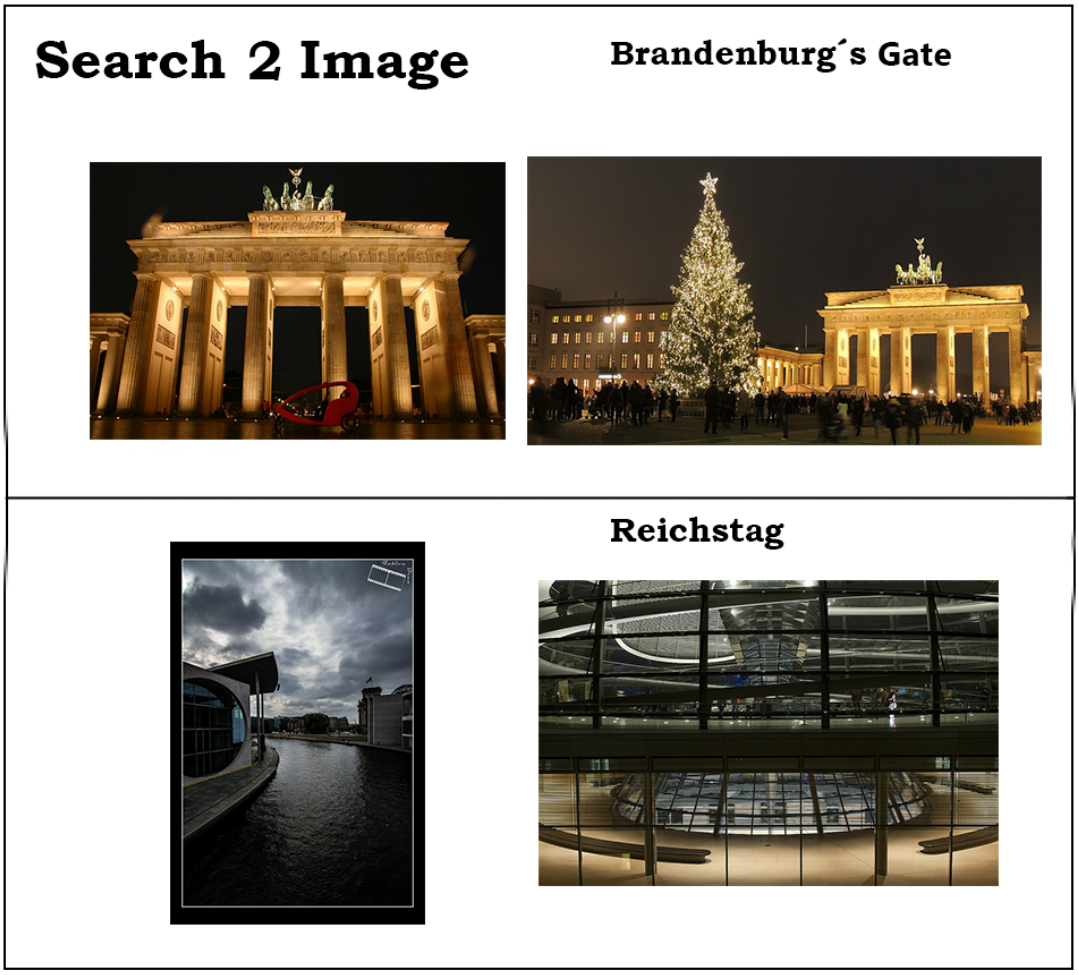

Figure 4.13: Set of 2 selected pictures to represent different cases

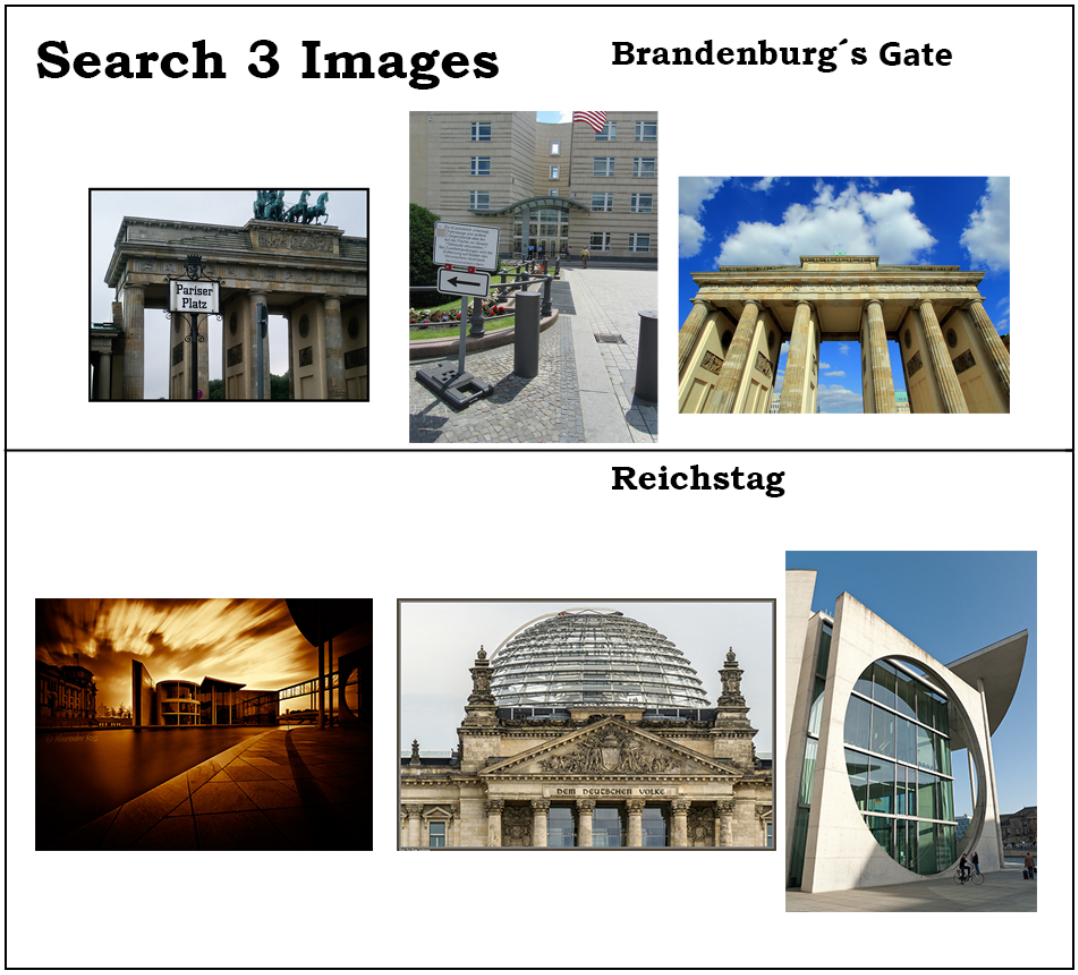

Figure 4.14: Set of 3 selected pictures to represent different cases

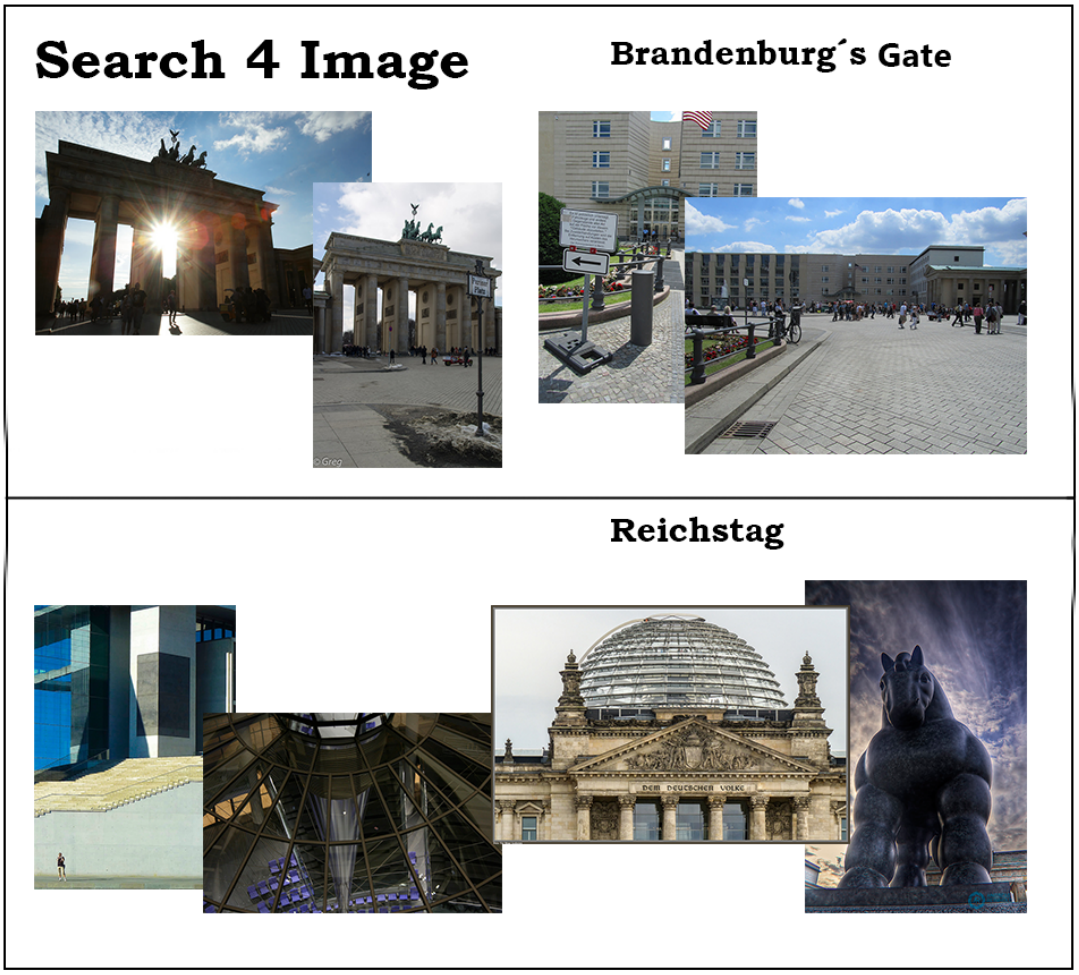

Figure 4.15: Set of 4 selected pictures to represent 2 different cases

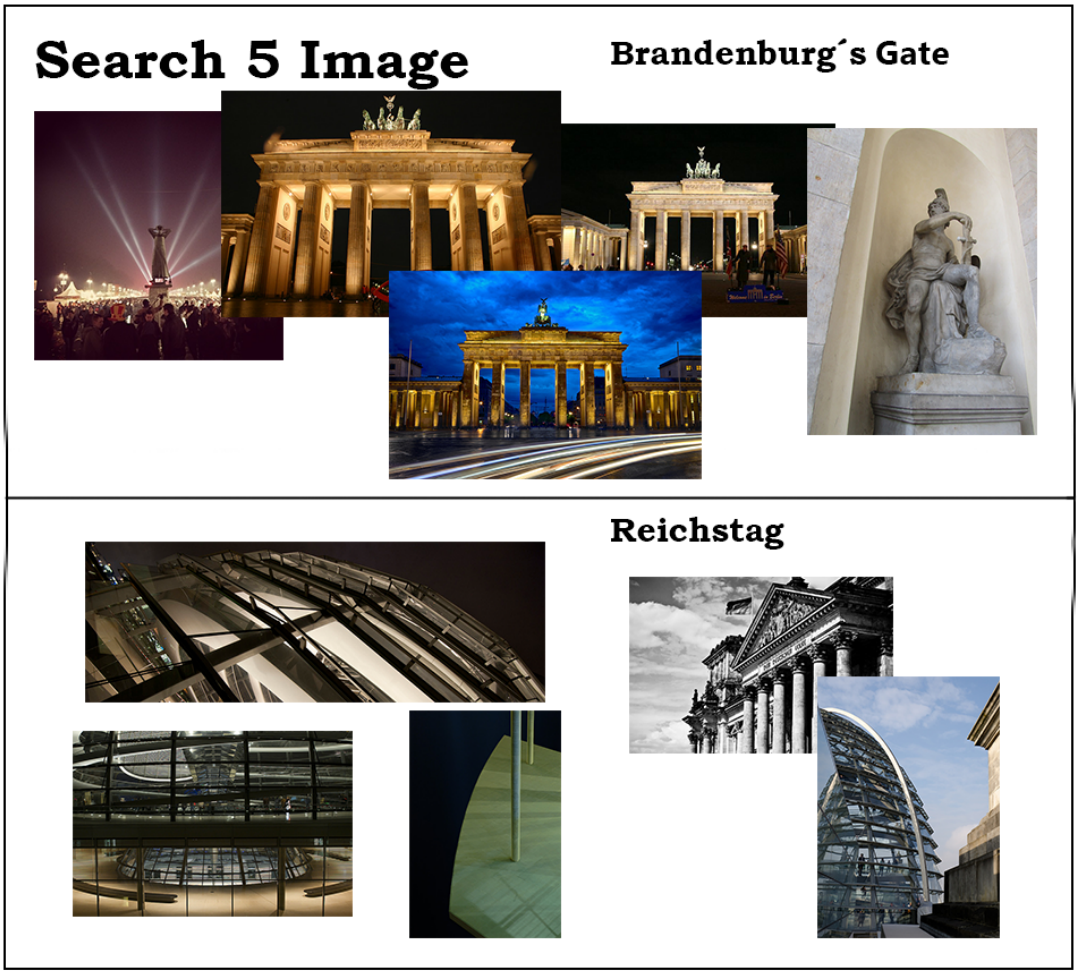

Figure 4.16: Set of 5 selected pictures to represent 2 different cases

the possibility of extending the number of representations to 5 or even 6. However, by using the K-means approach we will find many duplication cases. To achieve a better performance and avoid using multiple instances of the same element, we must use the tags associated to each picture. This process is beyond the scope of the project. We will further discuss this subject during the Future Work chapter.

Figure 4.17: Set of selected pictures representing the Jewish Museum in Berlin

4.2 Error and Particular Cases Analysis

In this section, we have identified the main errors of our approach. During the next paragraphs, we will thoroughly describe each error, explain its causes and consequences, as well as propose potential solutions.

4.2.1 Large and distant monuments

Since we have only studied Berlin this section is just a working theory. Berlin does not really provide an interest point which could fulfill this statement. The point is that some large monuments, such as the Eiffel Tower or the world-known Statue of Liberty, which are photographed at both close and long distances. This originates different clusters that represent the same monument.

Taking on the examples again, the Eiffel Tower's, shown in figure4.18, pictures are taken from its base and structure and also from the 'Elysian Fields' with almost the same frequency. So we have a cluster at the monument itself and some others at more than a hundred meters from it. We are disregarding the fact that given the monument's popularity we could find pictures of it that were taken from almost anywhere in Paris.

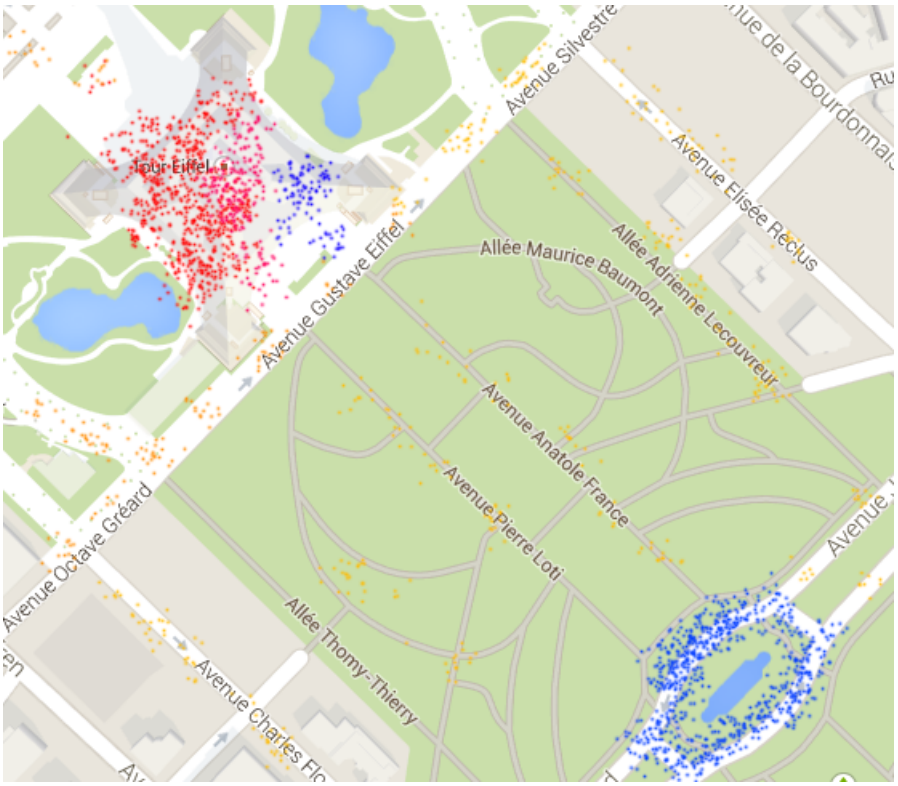

Figure 4.18: Eiffel's Tower Picture Mapping Estimation

In the case of the Liberty's Statue , also shown in Figure ,4.19, we can find pictures close to the statue and other taken from Manhattan's coast or some boat in the proximity. For this case we have to consider the water surface as a non-Inaccessible surface, meaning that Ferry boats probably circle the area in a regular pattern so tourist can take pictures. Following our predictions we will probably find a trace of the Ferry's route marked by the pictures tourist have taken during the short trip.

Also, given the size of this constructions we can expect that our algorithm will detect multiple clusters around this places, all pointing to the same monument. This confuses our project, our

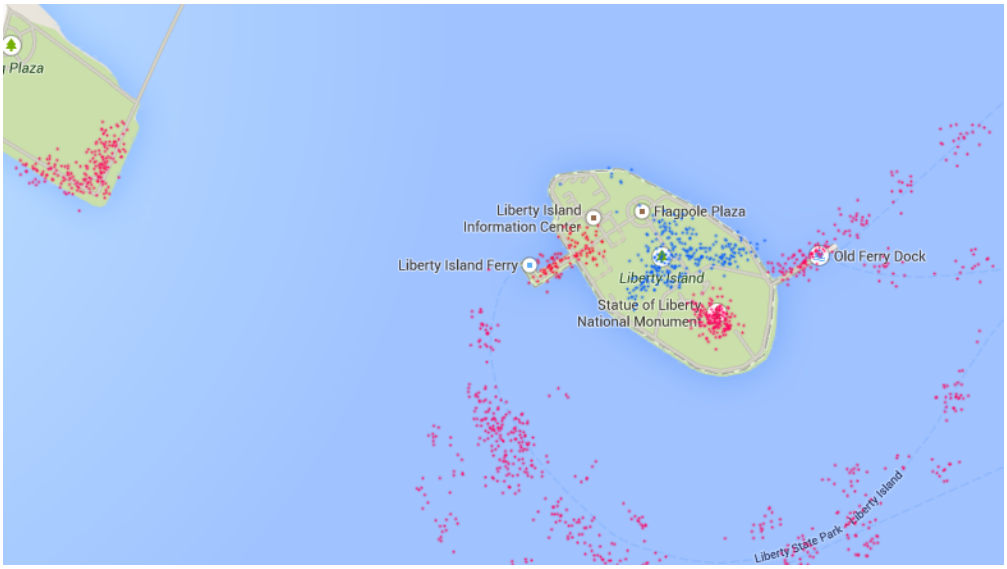

Figure 4.19: Liberty's Statue Picture Mapping Estimation

script thinks that a monument is pointed by a single cluster, and so it makes pairs of 1 to 1. The problem is not that a monument is surrounded by clusters that want to be paired, the real problem is that, given the large distances in which the pictures are taken, one of these clusters may be occupying the surface of another monument. So in the worst possible case scenario a true positive detected monument could be filled with pictures of another. This could be very confusing for the Image Recognition step and is one of the reason why an UI was designed. Manual intervention could easily mitigate this effects by adding a small non-automatic step (User Interface).

4.2.2 A single Point Clusters.

Firstly, in order to avoid, to avoid confusion due to this section's title, 'a single point cluster' refers to clusters which have more than Min_points as their number of points but they all share the same exact location.

During our experiment we have confirmed that this is not an error have generate by our execution took, but rather a curious output we receive from Flickr's queries.

Since we consider statistically impossible the fact that every single Flickr user took his picture at the exact same location, we have formulated two different theories.

- It could be a self tagging process, meaning that if the points is in radius to a reference or previously set point. So a set of points inside the radius of a key point will be considered the same point as the reference point.
- Another theory is that the pictures where taken with a device which used mobile data roaming to be able to have internet connection. This means that coordinates are set by

triangulation of repeaters antennas. Now if a device takes a very reduced time to respond to one of the three repeaters, meaning that is very close to it, it could be hard to determine the exact position since there are a huge difference between the times to reach each repeater. Probably the protocol implements that if a device's delay is equals or less than a minimum delay, then the device's position could be set to the repeaters coordinates.

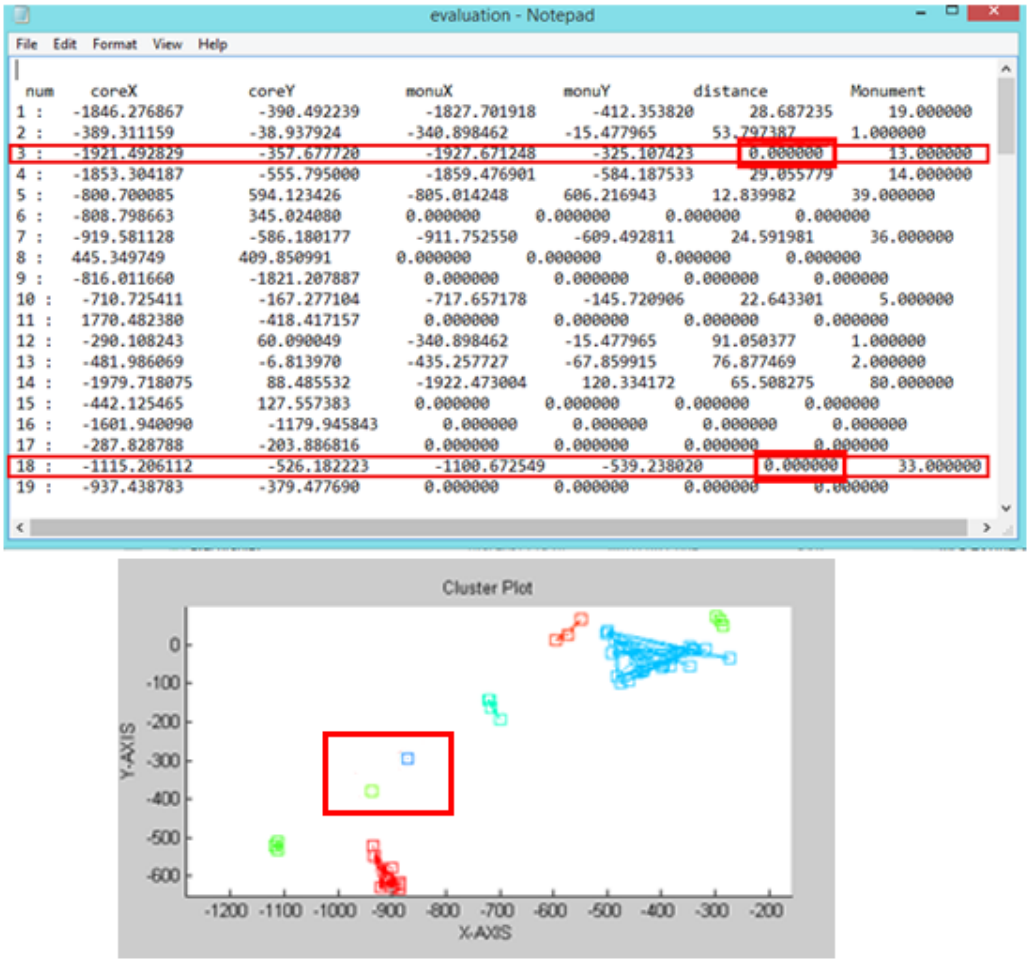

Figure 4.20: Single-point cluster examples, the marked clusters occupy a single point in space. Below there is a debugging log. The red marked '0' values depict that there is no distance between the conforming points and the reference point in the cluster

After consulting Flickr's API and its developers forum we are still unable to determine which of the two is a valid theory. We have found no complete evidence about this phenomena but it is probably related to the second theory. Also the fact does not affect our results in the present, the single-point clusters are no different from any standard clusters we have encountered apart from their peculiar concentration.

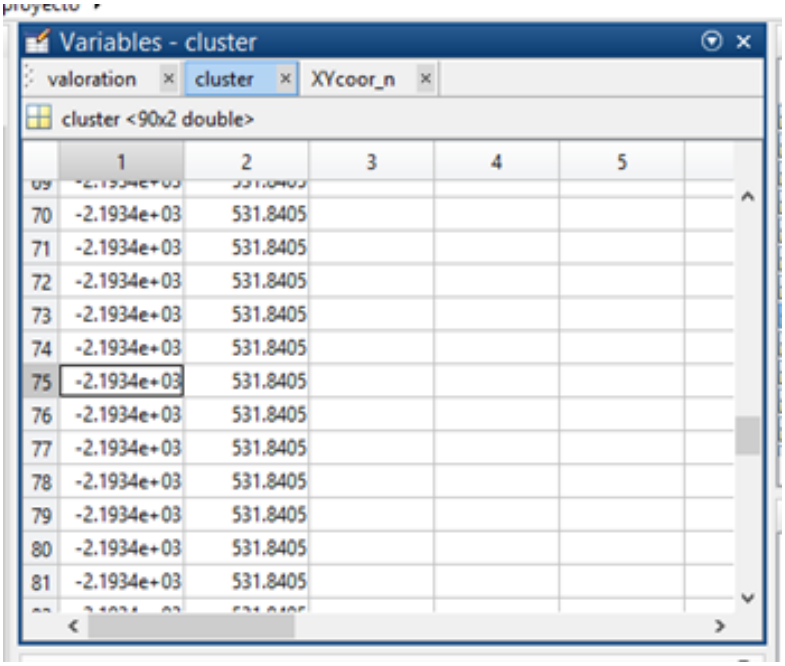

Figure 4.21: Single-point cluster log, displaying that all points in a cluster were identical

During our experiments we extracts multiple logs. In the figure 4.20 a log can be seen, it indicates the cluster-monument pair and in the column highlighted the distance between both points it is indicated. As you can see, some of the pairs have this register equals to 0, this probes the fact. Further proof can be found by looking at the additional figure 4.21

4.2.3 Bridges and Rivers

We do not see this section as a real problem for the project, it is more like a curious fact. During our numerous executions we could see that some of our final cluster centroids were exactly located in the middle of bridges and rivers. This was a regular fact and there was no clear variation from execution to execution. There is a easy and reasonable explanation for this, since we use the mean to determine the position of every cluster centroid, chances are that probably half the pictures taken of that location were taken from one side of the river and the other half from the other. This is expected considering water as an inaccessible surface where pictures can not be taken by normal means.

Figure 4.22 reflects a some examples of clusters located over rivers and briges.

4.2.4 Variability Between Executions

This is a fact that really needs to be taken into serious consideration. For any of the two search strategies there is a random component. This is caused because we decided that it will take so

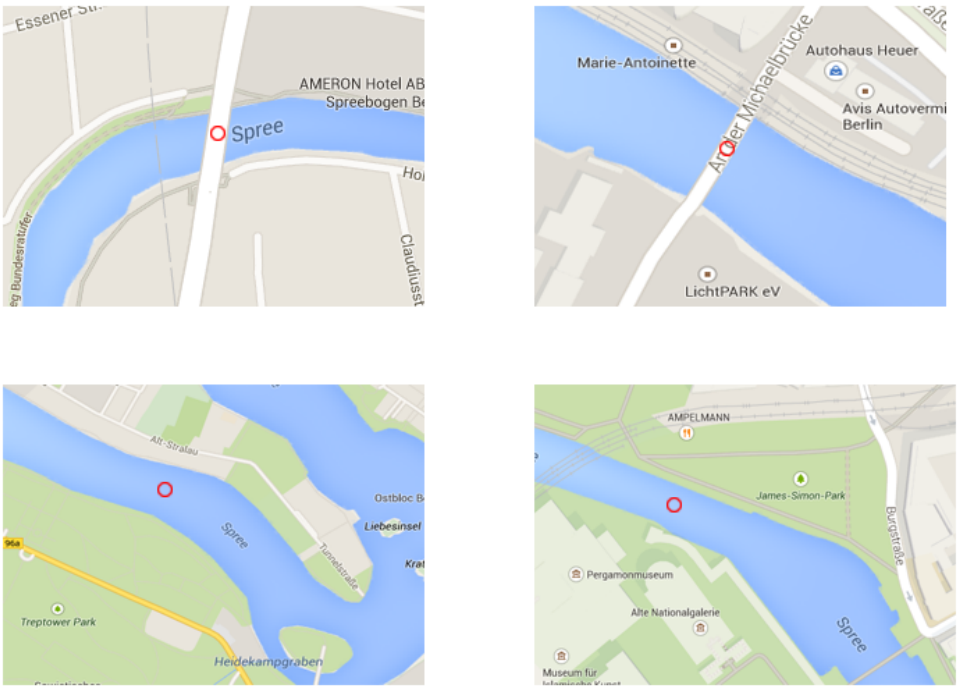

Figure 4.22: Examples Clusters located on Rivers and Bridges

long to do a DBscan with all the sample from the Flickr's query. Instead we took 30000 of those elected randomly. This is why we can see a slight variation in the number and the position of the clusters centroids.

This small variation should not be and issue most of the time, but in some cases it can be decisive when pairing clusters and interest points. Also when using a reduced DBscan (it is considered reduced when we do not use all the samples at our disposal), we will probably be unable to reveal all the clusters that otherwise we could have found by using all the samples. This is why we recommend using the statistical study seen in 2.4.3. The main reason is that during some executions the points conforming such cluster can be absent or more dispersed, as a consequence the algorithm will sometime not be able to find those. So we can not say that the value of the most optimal DBscan may not be accurate. For example, lets say that in a execution certain DBscan parameters leads to a F-value of 0,4. And after a second execution with the same DBscan parameters we get a different value of 0,38. This difference can mean that the most optimal pair of DBscan parameters may change between executions. However, there is an estimated fixed ranges which will not be surpassed for the bast majority of execution. We have estimated that this ranges goes more less 20 points and 10 meters from the corresponding values of 120 and 40. By observing the following figure 4.23, we can see an example of how this variations take effect between executions. In the figure Red elements represent the ones

that are unpaired while Blue ones represent the opposite. Also 'X' represent monuments and found clusters are represented by 'O'. You can see that some pairs are under a rectangular mark. This are cases that have a high chance to change from paired to unpaired, or vice versa, during different executions. We captured a case of variation on the green rectangle.

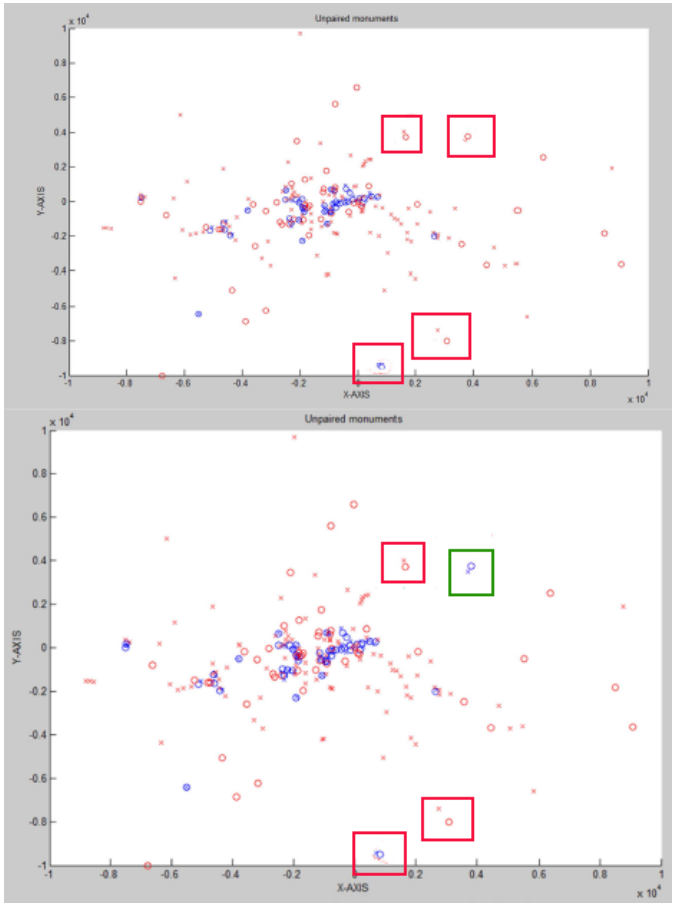

Figure 4.23: Examples of Pairs that are on risk of changing the state during different executions

Take this as a side note, even that there are many pairs under variation risk most of them remain constant for the 90% of the cases. This factor almost completely disappears after applying our second method of our Cluster Search module.

4.2.5 Cluster and Monument not pairing with the closest in the list

Take into account that we are not referring to those cluster-monument pairs that are simply separated by a larger distance than the threshold allows. This section is devoted to explain why after our execution we find a cluster really close to a monument but they would not pair together.

The reason behind this behavior is that the order in which we pair our clusters following an order. An order based on which clusters have more points or appear more times during a given

number of executions. This causes that big clusters search for their pair first and disregard the others's interest. They will take the closest monuments for their own and let the next one try to choose. As an example in figure 4.24 (remember the legend form the previous section), we find two two points that would have normally being paired. But as we can see, the monuments are paired with clusters that are farther apart from them. This is a demonstration of what we have explained before.

The possible problem is that in some cases the larger cluster could be representing another location, and the smaller unpaired cluster could be right. This is one of the reasons why we use the User Interface for small calibrations in our estimations.

Figure 4.24: Examples of clusters that are not paired with the closest monument

Chapter 5

Conclusions and Future Work

During this section we will do a self-evaluation both objective and subjective of the result got in every step..

5.1 Conclusions

Spatial clustering algorithms are the key to the designed module. This way, we are able to locate those places that have some special interest to the public. Our success can easily be noticed by analyzing the optimal cases. Our Berlins Ground Truth database is finally composed by 112 different locations while our application was able to find a total 137 possible locations of interest. During the pairing process, 54 of the Ground truth elements where left unpaired (48%). This means that a 52% of the monuments were correctly identified, leading to a 52% of recall factor. That leaves us with 79 unpaired clusters. We can conclude that the application is automatically identifying many Ground Truth locations. Although its precision might not suit the most accurate tasks, the application is recognizing the most photographed landmarks in Berlins region.

Secondly we need to measure quantitatively the performance of the spatial clustering process. Here is an objective analysis of our numeric results. By observing previous results , we have achieved a maximum value around 0,3. This might not seem high enough but considering that our database is automatically generated and includes any picture on Flickr that was tagged as we requested. We had imported multiple pictures wrongly tagged by Flickr's users. Also, some of our Ground Truth elements could have been considered as non-relevant. If it was not for either of this facts, we would have a higher F-value. Lets now study our F-value:

Secondly, we need to measure quantitatively the performance of the spatial clustering process. As an objective analysis of our numeric results, we must highlight that, by observing previous results, we have achieved a maximum F-value around 0,3. This might not seem high enough, but we must also take into account the fact that our database is automatically generated and that it also includes any tagged picture on Flickr, even if they were wrongly tagged. Added to that, some of our Ground Truth elements could have also been considered as non-relevant. If these

errors could have been removed, we would have got a higher F-value. Now, on to the study of our F-value:

The F-value is dependent of the precision and the recall. During our execution we extracted the value of both parameters.

- **Precision:** during our experiments, the precision in the output Exhaustive Search (Method 2 Cluster Search) gets the highest precision value. Of a total of 137 cluster we have successfully identified 58 of those as points of interest in Berlin. This results in an 42% precision factor.
- **Recall:** in our executions this value was around 52% taken that we identified 58 locations of the 112 of our Ground Truth Monuments list.

We have designed a module that determines a visual representation for the spatial clustering output. As it has already been said, this process cannot be analysed from an objective point of view. This section must be analyzed by using subjective criteria. Remember that this process selected three pictures to represent each cluster found during the spatial cluster search. We have to consider that every picture that we got from the Flickr query will be influencing the selection of those three pictures. Consequently, our results can be mixed for each one of the clusters. One may be represented by those three images while others may not even have one that makes a clear picture of the location. Still, for more than 90% of the clusters, we were capable of identifying, at least one of them, as the actual monument that occupied the location.

Mathlab was used as a primary developing tool for the creation of this application. We created a functional script that can be run and tested over any GPS location. The final script, formed by over 2000 lines of code, does not include external code functions (such as K-means or DBscan). We have implemented 12 functions that were used in the different modules. We have additionally created some debugging scripts that helped during the experimentation process by using graphs and text logs.These last two added extra information about clusters. The results can verify the success of this application.

The application can guarantee a 50% precision in the analysis of any large cities in the world. Additionally, if they are taken as touristic locations, we can also safely say that our application will detect the monuments that are considered as the most relevant around that area.

As for smaller towns and remote areas our clustering algorithms would only work if enough pictures were taken in the proximity. Same happens for remote locations (a mountain, a field, ...), if multiple users found an interesting sight we will detect it. Also if we are using multiple tags when searching for GPS coordinates, we might narrow our search too much. As a consequence our algorithms will not find enough points to make proper estimations.

In some occasions, our application failed to identify multiple clusters when we have a large amount of pictures. The reason is that a large quantity of GPS coordinates may end up composing a large 'carpet', identified by DBscan as a single point of interest. Therefore, both lack and excess of samples, can have a negative impact on our cluster search.

Our performance is completely dependent of what Flickr database has to offer in the analyzes region. If users took pictures there we will be able provide a new customized database. On the

other hand, if there are not enough pictures, our application will not detect every relevant landmark in the region.

5.2 Future Work

In this section we will discuss future plans for the application that were outside the scope of the project.

Furure Experiments

An important future line of work is to validate our results by doing as many experiments as possible. Since we have worked on the city Berlin, the first step will be to analyze similar cities. London, New York or Tokyo are also great examples. After these studies, we should take our experiments one step further. Obtain data from smaller towns (e.g. Heidelberg, Salamanca...) and find out which optimal parameters should DBscan use for these cases. Continuing this line, another interesting study would be to look into world-famous isolated landmarks such as the Grand Canyon or Mount Rushmore, and see how our application behaves.

In the end, and as a learning procedure, we should get random remote locations and particular populations that are not in a regular situation. We thought that it was interesting to find out if mapping locations such as Pionyang (North Korea) or Bagdad (Irak) was possible. We do not expect to be able create a custom map of these cities but it will serve as a way of testing the limits of our application.

After taking notes from a large variety of experiments, we could finally design a more intelligent search algorithm. A process able to differentiate the analyzed subject (a city, a small town,...) and that adapts accordingly, therefore, optimizing the application. By studying multiples locations we stored data, such as number of found samples, density around the center... in order to carry on making future estimations. If the process determines that we are analyzing a remote location, we will use a specific set of DBscan parameters. If instead, we face a small town, the application will set up with another set. This could be a mayor improvement and help in the study of any location.

Optimization: Design a DBscan Variant

As we have explained in previous sections, DBscan does not provide a perfect solution as our clustering algorithm. It could be argued that some of the most point populated areas in the city may not be differentiating a interest point from another. Yet we picked DBscan as the most convenient for our experiments and it has been the algorithm we used for running the whole project. Another possibility was designed, a idea of a new clustering algorithm.

As previously explained, DBscan does not provide a perfect solution as our clustering algorithm. It could be argued that some of the most point populated areas in the city may not be differentiating an interest point from another. Yet we picked DBscan as the most convenient for our experiments. Therefore, it has become the spine of this project. Given the importance of the task given, we studied a new alternative that could replace DBscan, increasing performance, as our spatial clustering algorithm.

There is no real need for a completely new clustering algorithm. Instead we can just make some simple modifications to an already existing one. So DBscan is used as the base of the algorithm, but one of the modifications that we need to introduce, must make it capable to differentiate clusters with more precision. The idea behind this modification is to give each point a value inside the cluster, so the farther away from the center of the cluster, the smaller the value for that cluster will be. This predicament could be fulfilled in multiple ways. One could be that the value decreases with each step from a point to another. Another could be that the distance to the centroid is directly proportional to the value given to that point.

We can expect two different problems from this new approach.

- With this new algorithm we still require to select a centroid. The idea is to run it as a DBscan during a first step. We will detect the clusters as we did in the application. After that we proceed to find the point that will act as a centroid for the cluster, we can use an already existing function FindCenterCoor (designed during this project and based in the mean value). From that point we must start evaluating all the points within a maximum distance. In the end, we will have N number of matrixes, where N is the number of clusters created and the evaluation will go over all the samples we go from the first query (200000 samples).
- A second problem is that multiple points will try to claim the points they have in range as their own. The easy way to solve this conflict is by assuming that the point will belong to the cluster which has given it the largest value after the evaluation. So after evaluating every point, we go point by point allocating each of them in the corresponding cluster. After solving those conflicts, the final step will be to discard those clusters which do not have a minimum number of points assigned to them. As in our original application, the order is key, and we must find a hierarchy to face this problem.

As it is expected, this procedure will take a lot of time and memory compared to the simple DBscan process. Due to this fact, the execution of this script will be recommended to be run by a particularly powerful computer. One of the reason why this algorithm was never further implemented deals with the need to run many experiments within a maximum scope of time. To sum up, this script, in theory, would be able to get better results than the default DBscan, but, at the same time, executing the function will also probe very time-consuming.

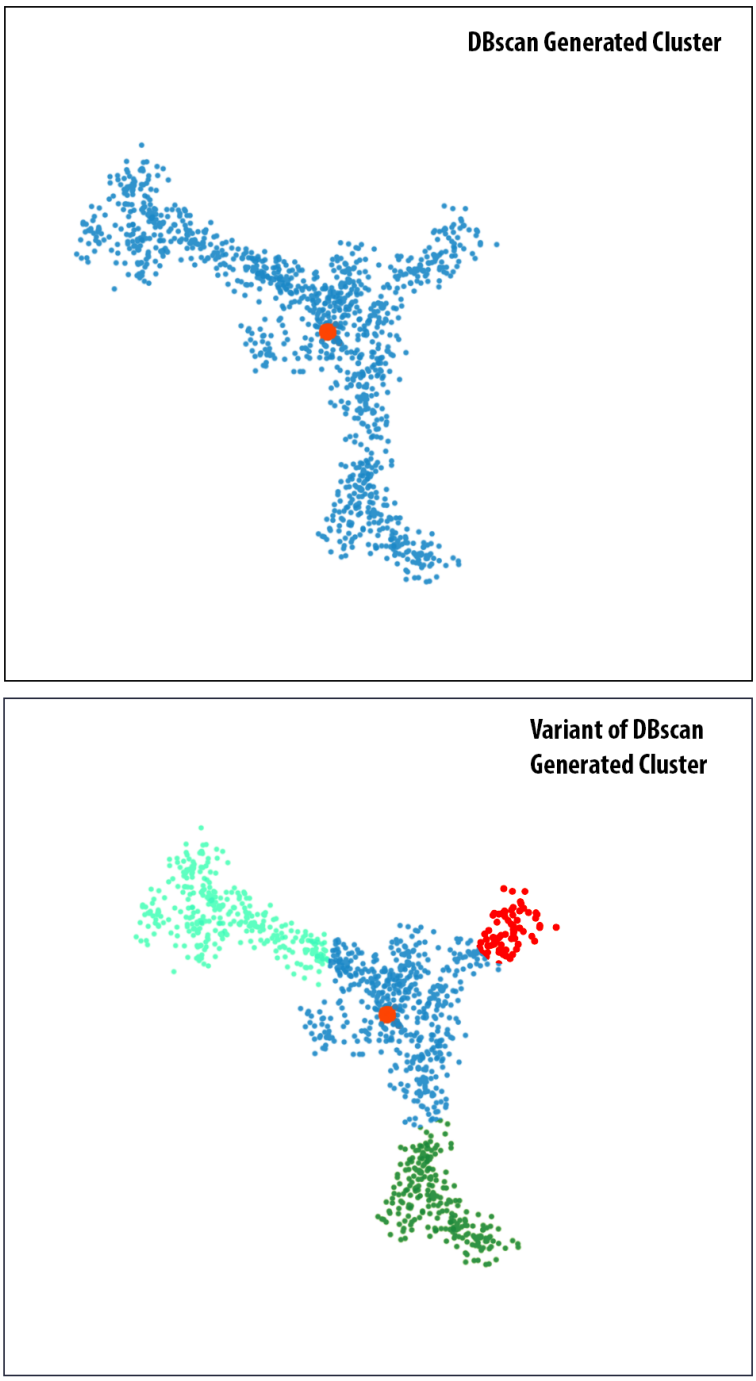

Figure 5.1: DBscan variant example 1

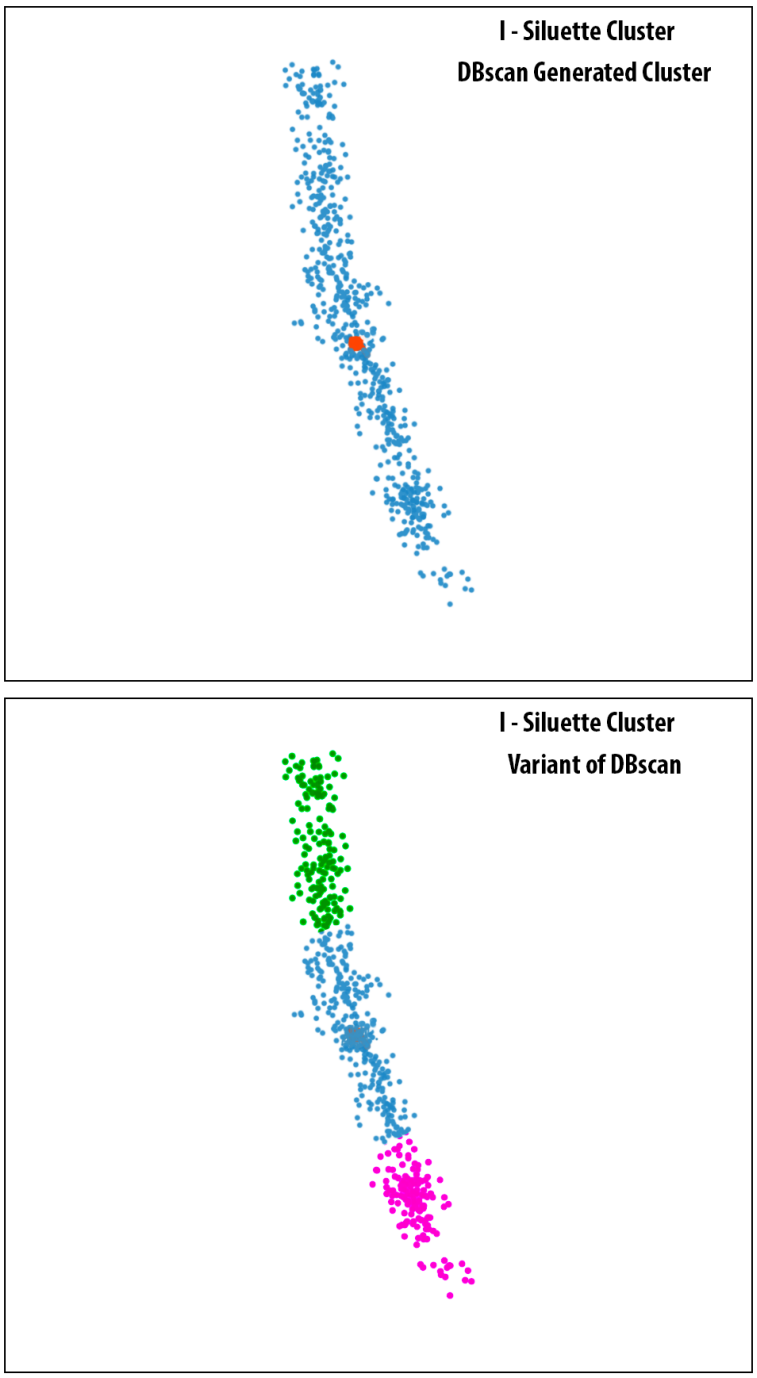

Figure 5.2: DBscan variant example 2

Appendices

Appendix A

Appendix A: Glossary

This appendix serves as an introduction to some of the concepts that we used during the explanation of this Project.

- **Outliers.** Speaking of clusters, outliers relates to those points which are not part of any found cluster.
- **Ground Truth Coordinates.** Can be GPS or Euclidean coordinates to an origin point. This coordinates are the verified inputs that informs of the exact location of a point of interest. They are absolute and shall not be modified during the experiment.
- **Precision.** Applied to this project, it refers to the fraction of clusters that have successfully found the Ground Truth coordinate compared to the total number of Ground Truth points available.
- **Recall.** Also known as sensitivity, it is a parameter that refers to the percentage of monuments that we were able to retrieve from the whole Ground Truth Database.
- **Density Reachable.** In our density based cluster, the main reason why we recognize the clusters is that within each cluster we have a typical density of points which is considerably higher than outside of the cluster. Furthermore, the density within the areas of outliers is lower than the density in any of the clusters. Being density reachable means that when we make a step away from a point, we will remain in the same cluster y N points are closer or equal as a distance Max_dist.
- **Neighborhood.** A neighborhood is considered as the set of points which are directly density reachable from any of the points in the same neighborhood.
- **Directly Density Reachable.** A point p is density reachable from a point q with a certain Eps and MinPts if there is a chain of points p1, , pn, p1 = q, pn = p such that Pi+1 is directly density reachable from Pi.
- **Density Connected.** A point p is density-connected to a point q with a given distance and Minimum number of Points (Min_Points) if there is a point such that both, p and q are density-reachable for those given the distance and MinPts.

- **Inaccessible surface.** This term refers to a portion of terrain where normal users would be unable to take pictures, under normal circumstances. As an example, a river or a lake would be an inaccessible surface. A restricted area would be considered as one as well.

Appendix B

Appendix B: Budget

In this appendix we estimate the overall cost of the development of this project during the 8 months of development. This is considered as a one-person project and it needs a small budget to be funded. We have classified the expenses in categories:

- **Material Cost.** Here are included the computer and software used to develop the project.
- **Workers fee.** There are two persons involved in this project. A supervisor and a worker. Through 8 months, they have work 230 days and being payed 30 and 35 euros together. Their fee ascends to 14950 euros.

Cost	Price (€)
Working space	3.500
Computer	1200
Matlab License	200
Employer´s Fee	14950
TOTAL	**19850**

Figure B.1: Table representing the diferent cost needed to fund this project

Bibliography

[1] Spain internet usage, population and telecommunications report. http://www.internetworldstats.com/eu/es.htm.

[2] Google maps official site. https://www.google.es/maps/.

[3] Google static maps developer site. https://developers.google.com/maps/documentation/staticmaps/.

[4] Google places api. https://developers.google.com/places/?hl=en/.

[5] Berlins wikipedia list. http://en.wikipedia.org/wiki/List_of_sights_in_Berlin.

[6] Apple as a company. https://www.apple.com/.

[7] Google as a company. www.google.com/about/.

[8] Adobe photoshop official website. http://www.adobe.com/en/products/photoshop.html.

[9] Flickr main webpage. https://www.flickr.com/.

[10] Panoramio main webpage. http://www.panoramio.com/.

[11] Flickr api. https://www.flickr.com/services/api//.

[12] Source of the image for dbscan. http://blog.csdn.net/chjjunking/article/details/6751026.

[13] Example using gist descriptor. http://cs.brown.edu/courses/csci1290/2011/results/final/gen/.

[14] Finding groups in data: An introduction to cluster analysis. http://onlinelibrary.wiley.com/book/10.1002/9780470316801/.

[15] Ejcluster and suncluster configuration guide. http://es.scribd.com/doc/50271303/sun-cluster-3-configuration-guide-412383#scribd.

[16] Picasa main webpage. http://picasa.google.com//.

[17] Flickcurl api library. http://librdf.org/flickcurl/.

[18] Image displaying different cluster silhouettes. http://tech.knime.org/whats-new-in-knime-211.

[19] Iván González-Díaz; Tomás Martínez-Cortés; Ascensión Gallardo-Antolín; Fernando Díaz de María. Temporal segmentation and keyframe selection methods for user-generated video search-based annotation in expert systems with applications. *Paper 42(1) 488-502 (2015)*, 1996.

[20] Blei David M.; Ng Andrew Y.; Jordan Michael I; Lafferty John. Latent dirichlet allocation. *Journal of Machine Learning Research 3: 9931022. doi:10.1162/jmlr.2003.3.4-5.993.*, 2003.

[21] Kaufman L. and Rousseeuw P.J. Wikipedias k-medoids documentation. *http://en.wikipedia.org/wiki/K-medoids*, 1987.

[22] Ester Martin; Kriegel Hans-Peter; Sander Jrg; Xu Xiaowei (1996). Simoudis Evangelos; Han Jiawei; Fayyad Usama M. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226231. ISBN 1-57735-004-9. CiteSeerX: 10.1.1.71.1980.*, 1996.

[23] William S. Massey. A basic course in algebraic topology, graduate texts in mathematics. *Graduate texts in mathematics 127 (3rd ed.)*, 1991.

[24] A. Oliva and A. Torralba. Modeling the shape of thescene: a holistic representation of the spatial envelope. *http://people.csail.mit.edu/torralba/code/spatialenvelope/*, 2001.

[25] Mihael Ankerst; Markus M. Breunig; Hans-Peter Kriegel; Jrg Sander. Optics: Ordering points to identify the clustering structure. *ACM SIGMOD international conference on Management of data.*, 1999.

[26] Mihael Ankerst; Markus M. Breunig; Hans-Peter Kriegel; Jrg Sander. Optics: Ordering points to identify the clustering structure. *Proc. ACM SIGMOD99 Int. Conf. on Management of Data, Philadelphia PA*, 1999.

[27] M. Douze; H. Jgou; H. Sandhawalia and L.Amsaleg. Evaluation of gist descriptors for web-scale image search. *https://lear.inrialpes.fr/pubs/2009/DJSAS09/gist_evaluation.pdf*, 2014.

[28] Vishwakarma Singh; Sharath Venkatesha; Ambuj K. Singh. Geo-clustering of images with missing geotags. *In Proceedings of the 2010 IEEE International Conference on Granular Computing (GRC '10). IEEE Computer Society, Washington, DC, USA, 420-425. DOI=10.1109/GrC.2010.76 http://dx.doi.org/10.1109/GrC.2010.76*, 2010.

[29] Liangliang Cao; Jiebo Luo; Gallagher A.; Xin Jin; Jiawei Han; Huang T.S. "aworldwide tourism recommendation system based on geotaggedweb photos. *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference*, 2010.

[30] Soumen Chakrabarti; Martin Ester; Usama Fayyad; Johannes Gehrke; Jiawei Han; Shinichi Morishita; Gregory Piatetsky-Shapiro; Wei Wang. Data mining curriculum: A proposal. *Intensive Working Group of ACM SIGKDD*, 2006.

[31] Soumen Chakrabarti; Martin Ester; Usama Fayyad; Johannes Gehrke; Jiawei Han; Shinichi Morishita; Gregory Piatetsky-Shapiro; Wei Wang. Encyclopdia britannica: Definition of data mining. *Intensive Working Group of ACM SIGKDD*, 2006.

[32] Tapas Kanungo; Christine D. Piatko; Ruth Silverman; Nathan S. Netanyahu; Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 2*, 2002.

[33] Wang Y. and Cao L. Tdiscovering latent clusters from geotagged beach images. *in Shipeng Li; Abdulmotaleb El-Saddik; Meng Wang; Tao Mei; Nicu Sebe; Shuicheng Yan; Richang Hong and Cathal Gurrin. ed.'MMM (2)'. Springer. pp. 133-142.*, 2013.

[34] F.; Tat-Seng Chua; Neven H. Yan-Tao Zheng; Ming Zhao; Yang Song; Adam H.; Buddemeier U.; Bissacco A.;Brucher. Mean shift, mode seeking, and clustering. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on , vol., no., pp.1085,1092, 20-25 June 2009*, 2009.

[35] Cheng Yizong. Tour the world: Building a web-scale landmark recognition engine. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.