Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr. 126

Visual Analytics of City Dynamics using Social Media Data

Nikolay Ivanov

Studiengang: Informatik

Prüfer/in: Prof. Dr. Thomas Ertl

Betreuer/in: M.Sc. Robert Krüger, Dipl. Phys. Qi Han, Dr. Steffen Koch

Beginn am: 7. November 2016

Beendet am: 9. Mai 2017

CR-Nummer: H.2.8, H.5.2

Abstract:

Understanding the dynamics of urban areas is essential for urban planners and geomarketing specialists. Social media platforms provide the means for such analysis, which has proved to be challenging, due to problem diversity, data volume, veracity and velocity. The scope of this thesis is to design and develop a system which overcome these obstacles and contributes towards an interactive urban analysis. To this end, this thesis presents a monitoring system, which collects data from the social media services “Foursquare” and “Twitter” in near real-time to provide a consistent stream of information for visual analysis. Furthermore, a visual analytics system has been designed and implemented, that offers methods, supported by visualizations and interaction techniques, for the derivation of knowledge to a variety of urban problems. Finally, two case studies from Tokyo, Japan are presented, which confirm the feasibility of the extracted data and the analysis capabilities of the implemented system.

Kurzfassung:

Das Verständnis der Dynamik der städtischen Gebiete ist für Stadtplaner und Geomarketing-Spezialisten unerlässlich. Social Media Plattformen bieten die Mittel für eine solche Analyse, die sich aufgrund der Problemdiversität, des Datenvolumens, der Datenwahrhaftigkeit und der Datengeschwindigkeit als herausfordernd erwiesen hat. Der Umfang dieser Arbeit besteht darin, ein System zu entwerfen und zu entwickeln, das diese Hindernisse überwindet und zu einer interaktiven Stadtanalyse beiträgt. Zu diesem Zweck präsentiert diese Arbeit ein Monitoring-System, das Daten aus den Social Media Services "Foursquare" und "Twitter" in fast-Echtzeit sammelt, um einen konsistenten Informationsfluss für visuelle Analyse zu liefern. Darüber hinaus wird ein visuelles Analysesystem entwickelt und implementiert, das Verfahren bietet, die durch Visualisierungen und Interaktionstechniken unterstützt werden, für die Ableitung von Wissen zu einer Vielzahl von städtischen Problemen. Schließlich werden zwei Fallstudien aus Tokyo, Japan vorgestellt, die die Machbarkeit der extrahierten Daten und die Analysefähigkeiten des implementierten Systems bestätigen.

Table of Contents

List of figures

Code Listings

1. Introduction

1.1 Motivation

With each year, more and more people prefer the advantages of living in the city. According to the World Health Organization [1], the majority of people lives in urban areas. As this trend continues to grow, cities become larger, more complex and dynamic. According to Juval Portugali, “the structure of cities is seen as an outcome of bottom-up spontaneous processes, on the one hand, and top-down planning and design interventions, on the other.” [2] The urban planning process requires large-scale knowledge about the people's needs to keep communities vibrant and happy. This knowledge involves the understanding of a city's everyday life from multiple perspectives. This is a crucial and challenging task to urban planners, due to the diversity of the problem. Knowing people's needs is about knowing people's habits. Where do most people go on weekends? Which outdoor park, shopping center, bar or restaurant? How do people move over the course of a workday? Do people like going shopping after work and what are their favorite places? Which train, bus stations or even airports are the most crowded during the day? Do they differ in the mornings, afternoons or evenings? Or even more specific, what events (concerts, conventions, sports) happened in the last month and how did they influence activities in its vicinity? How to make a city area more attractive to businesses? These are only a small set of questions an urban planner has to answer to discover knowledge about a city's needs. But this is often not enough. The dynamics of a city involve fast-paced changing environments. Discovering events and formation of crowds as they happen could aid analysts to plan emergency and security measures, accordingly.
Understanding this complex urban environment is as challenging as the design and implementation of the systems that support this process. The capabilities of such systems should comprise real-time analysis to discover the dynamic insights of the life and behavior of people in cities. But what data could offer such velocity? With nowadays rapid advancement of technology, mobile devices have overtaken the everyday life of people around the globe. This gives access to large volumes of location-based data and enables completely new possibilities to process and analyze this data in near real-time. Standard location-based data could be GPS or cell phone traces. Although this data is mostly accurate, complete and reliable, it cannot fully support the task of understanding city life, because it lacks the semantics of urban dynamics. Location-based Social Networks (LBSN), such as Foursquare and Twitter, do not only comprise information about position and time, but also contain semantic information about places and events. Although such data sources are often not fully reliable, as the result of noise and incompleteness, they contain information about urban activities and can characterize better human behavior. The challenge is to create a system, which can tackle the data quality, veracity and velocity obstacles to leverage the decision-making ability of the analysts and guide them through the process of urban exploration and analysis.

1.2 Research problem description

City dynamics is a complex and diverse topic. People share vast amounts of information about their activities. Such information could reveal interesting large-scale patterns of a city's everyday routines or outline unexpected behavior in (near) real-time. The extraction, management and representation of such data is a challenging task, due to its scale, incompleteness, noisiness and refresh rate. This problem requires the cooperation of experts and computers to process and analyze it efficiently, in order to derive knowledge about various urban problems. This thesis aims to solve this problem, by designing and implementing a monitoring system and a visual analytics system, which try to overcome these challenges.

1.3 Overview of thesis

The thesis is structured in the following manner:

Chapter 2 – Related work: gives an overview of related work in the fields of city dynamics and urban visual analysis.

Chapter 3 – Theoretical concepts and algorithms: describes the theory and algorithms behind the design and implementation of the monitoring and visual analytics system.

Chapter 4 – Practical concepts and implementation: describes the technical ideas and concepts behind the implementation of the monitoring and visual analytics system.

Chapter 5 – Case studies: presents two case studies and their findings, in regards to the problem description.

Chapter 6 – Conclusions and future work: concludes the thesis, by discussing the outcome of the results, possible limitations of the system and future improvements.

2. Related work

This thesis's primarily goal is concerned with the analysis and exploration of the activity and mobility patterns of people in urban areas. Different systems and methods have been

proposed to study and understand this dynamic complexity of urban life. Some researchers have focused on the subtopic of urban mobility, that is, how people or crowds move around the spatial and temporal fabric of a city. Such researches can reveal traffic congestions and aid urban planners in finding the most optimal solution to these. Other research problems are more concerned with discovering the collective behavior of people, by analyzing the semantics of the data. Also, a combination of these exist to find a reliable solution to the problem.

Different data sets have been used, from GPS and cell phone data, to the integration of multiple social media services. Achilleas Psyllidis et al. [3] argue that the integration of traditional urban data and human-generated content, e.g. from social media, is required to understand the complexity of city dynamics. They have developed a web-based application framework, called "SocialGlass" that allows for exploration, monitoring and visualization of urban data. By combining static or semi-static demographic (e.g. age, gender, population distribution) and statistical (e.g. income levels or crime rate) data, with data from social media platforms and sensors, they propose a set of tools and methods towards (near) real-time urban analysis. Furthermore, they have investigated the reliability of their system and dataset by taking into consideration the following social data dimensions:

a) Sample coverage – what is the number and demographics of people using social media? To minimize the demographics limitations, their system uses data from multiple social platforms.

b) Timeliness – different data refresh rates uncover different aspects of the city dynamics, thus making the data more reliable for each stakeholder. As already mentioned, "SocialGlass" encompasses static and dynamic data to enable more analysis possibilities.

c) Integration of multiple sources – Interoperability issues arise by the integration of different heterogeneous and autonomous data sources.

d) Veracity – Web social data is not fully reliable and contains a lot of noise, which may hinder the analysis task at hand.

e) Visualization variability – Urban dynamics involve the study of different aspects of city life, each with its respective analysis strategy, which often require a multitude of visualizations. To offer such possibilities to its users, "SocialGlass" presents a modular visualization architecture, allowing for custom on-demand visualization layouts. The usage of multiple social media platforms enables the possibility to explore and understand the spatial patterns of the data from different demographic segmentation perspectives. For example, Figure 1. shows a heat map representation of the average spatial distribution (extracted from the social media platform "Instagram") of residents and foreign tourists in Amsterdam from 6pm till 9pm during the "Amsterdam Light Festival 2015" (ALF) event.

Tatiana von Landesberger et al. [4] approached the problem in a very different manner. They used geolocated Twitter data to visualize and analyze the mobility patterns of residents in London. This task proved to be challenging, due to the emerging visual clutters by the representation of movement data over a long period of time. To overcome these challenges, they propose a visual analytics methodology with the implementation of spatial and temporal simplifications. Their approach involves spatial aggregation (grouping nodes into clusters and flows between nodes into flows between clusters) and temporal clustering (aggregating data over the temporal domain). Their algorithm starts by grouping data from multiple time steps. A DBScan-like clustering algorithm[15], based not only on spatial

proximity, but also on flow significance between places, is then performed to form the so-called “supergraph” or “superflow”, which is composed of all places, existing in the current temporal domain, together with aggregated flows between them. Figure 2. shows the spatial simplification of afternoon movements of London residents.

[a] ALF – Platform: Instagram | User type: Resident | Time Period: 18–21h

[b] ALF – Platform: Instagram | User type: Foreign Tourist | Time Period: 18–21h

Figure 1. “Activity heat maps of the ALF, illustrating the average spatial distribution of (a) Residents, and (b) Foreign Tourists, from 6pm till 9pm during the entire event period, based on their Instagram posts.”[3]

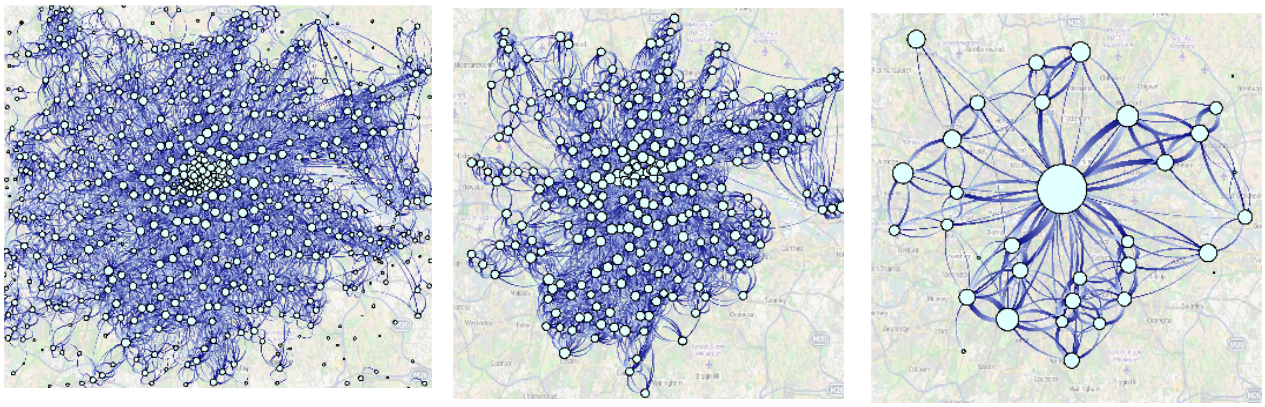

Figure. 2 Spatial simplifications of afternoon movements of London residents. [4]

Although their analysis lacks semantics about people's behavior and everyday activities, it uncovers a different dynamic aspect of city life. Their research reveals how people tend to move around the urban fabric by comparing different spatial situations over time. Their methodology and system can successfully capture urban mobility patterns, but lacks any semantical information of the reasoning behind them. Inspired by this research, this thesis tries to combine the analysis of spatial and temporal mobility and activity patterns to create a full profile of the dynamics of urban places.
Other studies have focused on the analysis of social-media based activities in the everyday life of people in urban areas. Felix Kling et al. [5] combine textual and movement data from location-based social networks, such as Foursquare and Twitter, to apply a probabilistic topic model to obtain a *"decomposition of the stream of digital traces into a set of urban topics related to various activities of the citizens in a course of the week"*[5]. The target city for urban analysis is New York. The probabilistic topic model is Latent Dirichlet Allocation(LDA)[6], used for the analysis of a set of documents. Each hour of the week is considered as an individual document, which defines the urban exploration granularity. To represent their results, they implemented several visualizations. A heat map displays the spatial distribution of the topics for different hours throughout the span of an "average" week, derived from the aggregation of observation data, while a word cloud visualizes the frequency of the words to uncover the semantics of people's activities. Figure 3. and Figure 4. show the results of the analysis for the "morning" and "evening" topics. As expected, the morning topic contains more words, related to transport facilities, when people commute to work, and typical "morning" activities, like drinking coffee. In comparison, the evening topic contains words, related to nightlife, food and entertainment places, like "theatre" and "broadway". To visualize the temporal differences of topics, they created an alluvial graph. Figure 5 shows the temporal evolution of the topics "Morning", "Lunch", "Home" and "Nightlife", derived from Central Park area on a weekend. This study successfully reveals the large-scale activity patterns of New York city, but it contains no notion of spatial mobility. Zeng et al. [7] proposed a hybrid approach to this problem. They developed a system capable of studying the mobility of people together with the semantics behind these movements. For their dataset, they used transportation data from Singapore with over 30

million trajectories and social media data, retrieved from Foursquare over the span of a week.

Figure 3. "*Heatmap and a word cloud for the "morning" topic" [5]*

Figure 4. "*Heatmap and a word cloud for the "evening" topic" [5]*

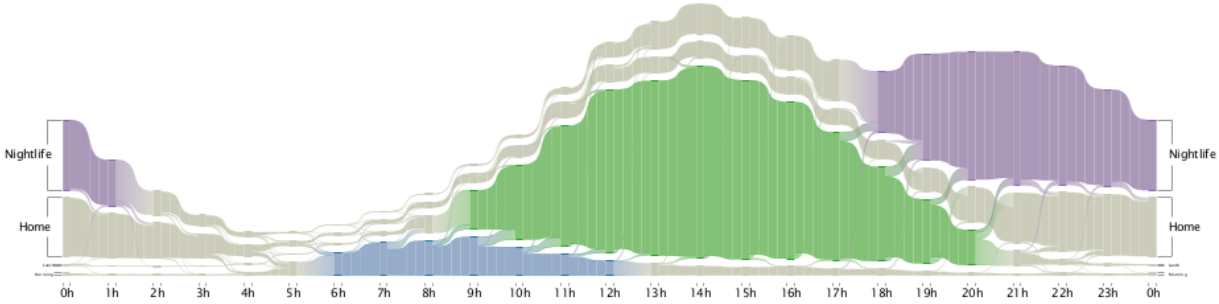

Figure 5. "*24h temporal profile of Central Park on a weekend"[5]*

They compute a relationship model between movement trips and POI check-ins, recovered from Foursquare, to study the reasoning behind these movements. To visualize this relationship, they developed a “POI-Mobility Signature”, consisting of a “POI Component”, which reveals the geographic and activity context of an area of interest and a “Mobility Component”, which aims to present the arrivals to and departures from an area of interest with a radial layout visualization. Figure 6 shows how several POI Components could be represented to compare the POI-Mobility relationship between different categorically well-defined regions. The presented area of interests are a) university, b) factory, c) shopping center, d) residence and e) airport. Although this work can successfully study the semantics behind the mobility of different areas of interest, it contains no notion of movement between them. This thesis presents a different approach, which uncovers the activity patterns of clustered areas and the mobility between them, leaving to the user of the system to derive conclusions of the reasoning behind them.

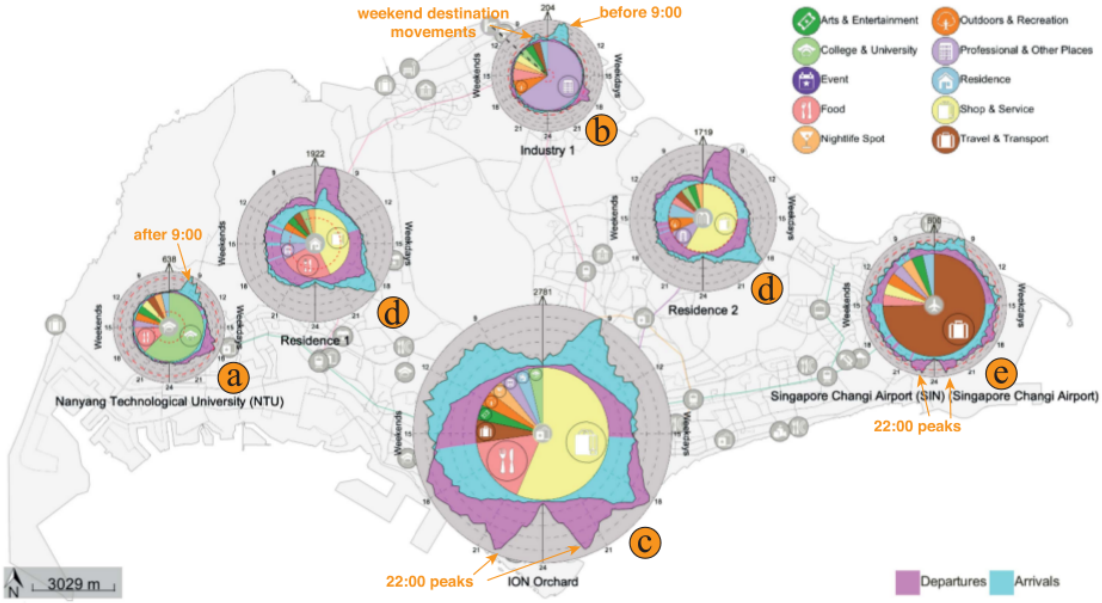

Figure 6. “*Comparing the POI-mobility relationship by exploring the POI-mobility signatures in different categories: a) university, b) factory, c) shopping center, d) residence, and e) airport”[7]*

3. Theoretical concepts and algorithms

This chapter presents the theory behind the concepts, methods and algorithms, used in this thesis.

3.1 Visual Analytics

According to Thomas and Cook [8], visual analytics is "*the science of analytical reasoning facilitated by interactive visual interfaces*". This abstract definition is further specified by Daniel Keim et al. [9]: *"Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets."*. The goal of visual analytics is to explore and analyze vast amounts of (possibly heterogeneous and/or conflicting) data to "*detect the expected and discover the unexpected*" [9] and gain deeper insights from the perspective of different stakeholders to support decision-making, planning and assessment. Reaching this goal requires the combination of multiple research fields, some of which are: data management, data mining, perception & cognition and spatio-temporal data analysis, each of which contributing to the core disciplines: information visualization and scientific visualization. (see Figure 7).
Infrastructure and evaluation are also an essential part of the visual analytics process. While the infrastructure links together all of the functions, services and system components, the evaluation assesses the effectiveness and efficiency of the methodologies and theory behind the implementation.

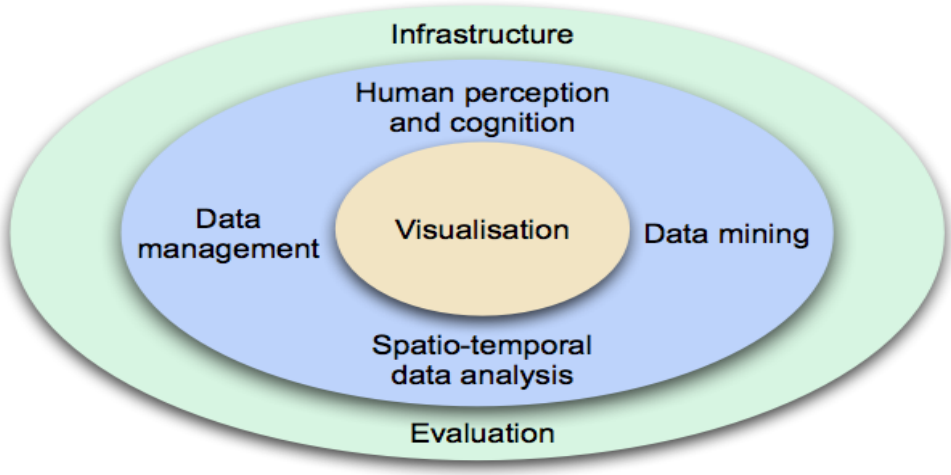

Figure 7. Visual analytics disciplines [10]. The core visualization disciplines are tightly coupled with the research fields of data management, data mining, spatio-temporal data analysis and perception and cognition. Infrastructure links them together, while evaluation assesses the used methodologies.

3.1.1 Information visualization

Information Visualization is "*the use of computer-supported, interactive, visual representations of abstract data to amplify cognition*"[11]. Challenges emerge with the visualization of different abstract data types, which could be summarized as 1-, 2-, 3-

dimensional, temporal, multi-dimensional, tree and network data, as defined by Ben Shneiderman in his famous research paper “The eyes have it: A task by data type taxonomy for information visualizations”[12]. Choosing the right visualization for the current task is of great importance for the outcome of the analysis. Sometimes several representations of the same data are needed to gain a deeper understanding. The complexity of urban dynamics requires multitude of data representations, primarily concerned with the visualization of spatio-temporal data. The visualization process requires series of steps, each of which consisting of different interaction techniques (see Figure 8), as described by Card et al. [11]. The first step involves the preparation of raw data for visualization, which includes the resolution of several data technicalities, like the integration of multiple heterogeneous data sources with data cleaning, entity resolution and data fusion strategies. This data can then be visually mapped to create visual abstractions, which could then be the subject to view transformations, which render the visualizations into multiple perspectives, each revealing a different aspect of this data.

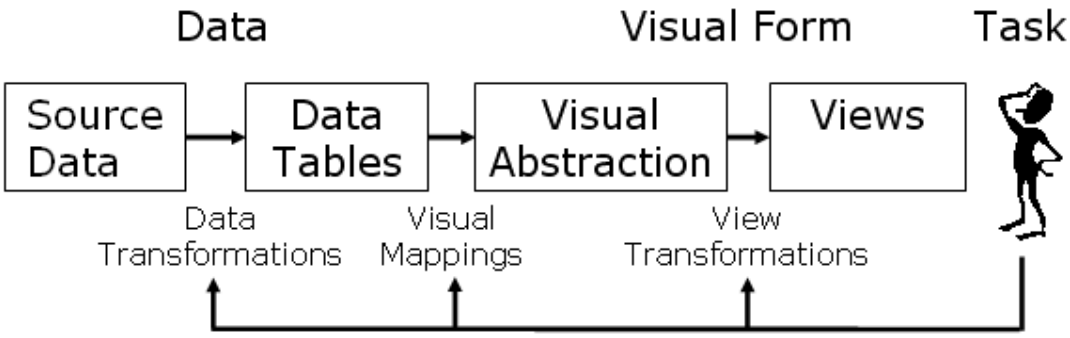

Figure 8. Information Visualization Reference Model [11]. The first step involves the transformation of source data into indexed structures, which are then mapped to visual abstractions and presented to the user through view transformations.

3.1.2 Data management

The management of data is an essential part of visual analytics, because the scale of data grows with each day and requires even more care. Often, heterogeneous data sources need to be integrated into a common global schema, which results in the need for developed data cleaning, entity resolution and data fusion strategies. A consistent database global schemata is an important part of the analysis process and the implementation of visual analytics systems. The monitoring system, implemented in this thesis, takes care of this process and provides a consistent data stream to the visual analytics system. Furthermore, it performs a data pre-aggregation step, which reduces the computational complexity and increases the response time.

3.1.3 Perception & Cognition

Understanding how humans visually perceive is essential for the creation of systems, that maximize their analysis potential. *"Visual perception is the means by which people interpret their surroundings and for that matter, images on a computer display. Cognition is the ability to understand this visual information, making inferences largely based on prior learning"* [10] Its aim is to enhance the ability of humans to discover new insights and use this knowledge to solve the task at hand. This ability to give meaning to experience is called "sense-making" and is an important part of visual analytics, because it encompasses the use of visual analytics systems for searching, organizing and deriving conclusion and new knowledge from given data. Sense-making is a broad interdisciplinary research field, combining several research fields, like sociology and cognitive science. The sense-making process [13], as shown in Figure 9, is known to be organized in two major loops, the foraging loop and the sense making loop.
The information foraging loop involves searching, filtering, reading and extraction of information. Challenges are the abstraction and formalization of found data, finding and identifying trends and outliers and forming hypotheses from the derived information. [13]
The sense-making loop involves the analysis and cognitive stages users traverse to give meaning to a large scale data in a certain situation. It involves the steps to schematize information, building of cases and its presentation. Schematization of information could encompass in itself the creation of simple schemas and models, but also complex and computer-based. A suitable visualization could be a timeline, which organizes and coordinates events, forming an evidence list and conclusions out of different data cases. Using these evidence, one could build a case to confirm or disconfirm hypotheses. The last step is the creation of suitable presentation or publication to support and explain the derived conclusions to a specific audience. [13] Visual analytics systems extend this knowledge to define their own workflow, which utilizes the cognitive capabilities of the human mind.

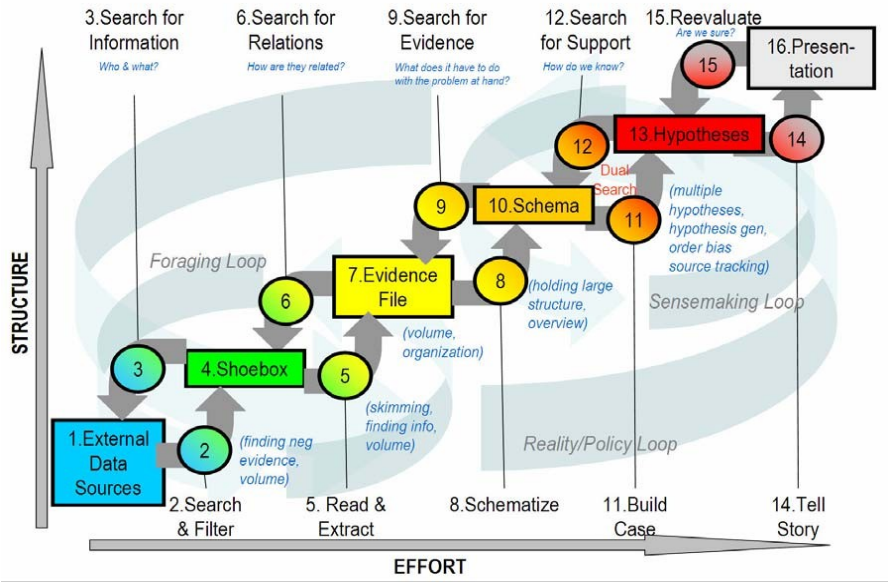

Figure 9. *"Notional model of sensemaking loop for intelligence analysis"*. [13]

3.1.4 Data mining

Data mining is the process of finding patterns in large data sets. These patterns are usually impossible to discover without computational assistance. According to Fayyad et al. [14], data mining refers to a single step in the process of knowledge discovery in databases (KDD). It is the "*application of specific algorithms for extracting patterns from data*" [14]. Data mining is a broad term, involving methods from multiple disciplines: machine learning, database systems, artificial intelligence, statistics and more. Fayyad et al. group the goals of data mining into two distinctive characterizations: "prediction" and "description". "Prediction" refers to the prediction of unknown or future values of interest, while the subgoals of "description" focus on finding patterns in large-scale data [14]. Data mining presents several methods to achieve these goals [14]:

Clustering Analysis – The grouping of data objects on the base of common attributes or dependencies. This results in a finite set of data descriptors, which reveal different data aspects.

Association – Retrieves dependencies between variables in large-scale data.

Regression – The mapping of a data object to a real-valued prediction variable.

Classification – The mapping (classification) of data objects to a set of predefined classes.

Outlier detection – The discovery of anomalies, that is, data objects which deviate from the expected routine or data pattern.

Summarization – Involves the finding of compact representation of the problem data at hand.

This thesis presents a clustering analysis approach to find patterns describing the spatial mobility of people (or check-ins) and identifying areas of different densities and semantical meaning. Furthermore, an outlier detection method is developed to find objects that do not conform to a pre-calculated routine value, e.g. the presence of people (or number of check-ins) at different places and times. Although not included in this thesis, predictive analysis could also be performed to gain further knowledge about the dynamics of urban life.

3.1.4.1 Clustering analysis

Many clustering algorithms have been proposed over the years. Choosing the right algorithm almost always requires domain knowledge about the analysis task, the data and the run-time/space complexity requirements. Multiple clustering classifications exist. Clustering algorithms can be categorized into hierarchical, centroid-based, density-based and more. A density-based algorithm has been implemented in this thesis to group

spatially-close areas of different densities, in order to reveal the mobility and semantical patterns of cities.

Density-based clustering

One of the most popular density-based clustering algorithms is "Density Based Spatial Clustering of Applications with Noise", proposed by Martin Ester et al. [15]. The following definitions formalize the intuition behind the pseudo code of the algorithm:

Let D denote the set of data points of some k-dimensional space S in the database. The idea is to find similar data points within a given radius, denoted by "Eps" or "ε", according to some distance function, denoted as dist(p, q), where $p \in D \wedge q \in D$. Cluster is formed if a data point contains at least a minimum number of points(denoted by "MinPts"), including the current one. [15]

Definition 1. (Eps-neighborhood of a point): *"The Eps-neighborhood of a point p, denoted by* $N_{Eps}(p)$ *, is defined by* $N_{Eps}(p) = \{q \in D | dist(p,q) \leqslant Eps\}$ *". [15]*

Definition 2. (Directly density-reachable): *"A point p is directly density-reachable from a point q wrt. Eps, MinPts if*
1) $p \in N_{Eps}(q)$ *and*
2) $|N_{Eps}(q)| \geq MinPts$ *".[15]*

Definition 3. (Density-reachable): "*A point p is density-reachable from a point q wrt. Eps and MinPts if there is a chain of points* P_1 *..* P_n *,* $P_1 = q$*,* $P_n = p$ *such that* P_{i+1} *is directly density-reachable from* P_i *.*" [15]

Definition 4. (Density-connected): *"A point p is density-connected to a point q wrt. Eps and MinPts if there is a point o such that both, p and q are density-reachable from o wrt. Eps and MinPts."[15]*

Definition 5. (Core object): "*A core object refers to such point that its neighborhood of a given radius (Eps) has to contain at least a minimum number (MinPts) of other points*". [15]

Definition 6. (Cluster): *"Let D be a database of points. A cluster C wrt. Eps and MinPts is a non-empty subset of D satisfying the following conditions:*
1) $\forall p,q$ *: if* $p \in C$ *and q is density-reachable from p wrt. Eps and MinPts, then* $q \in C$ *. (Maximality)*
2) $\forall p,q \in C$ *: p is density-connected to q wrt. EPS and MinPts. (Connectivity)". [15]*

Definition 7. (Noise): *"Let* C_1 *.....* C_k *be the clusters of the database D wrt. parameters* Eps_i *and* $MinPts_i$ *, i = 1 ..... k. Then we define the noise as the set of points in the database D not belonging to any cluster* C_i *, i.e. noise =* $\{p \in D | \forall i : p \notin C_i\}$ *". [15]*

These definitions are visually exemplified in Figure 10. The idea is to find clusters with MinPts = 3 and an arbitrary radius length “Eps”, visualized in the figure as circles defining the density regions of each node. The example contains 13 spatially distributed nodes, which form two clusters with the current algorithm parameters. The red and green nodes define the core points of each cluster. Although not core points, the yellow nodes are defined as “density-reachable” and therefore are contained in the respective clusters. The blue nodes are noise. Some additional remarks follow below:

1) Nodes 7, 8, 13 are defined as 'noise' points, because they are not contained in either of the two clusters.
2) Nodes 1, 2, 3, 5 are the core points of the red cluster, because each of their Eps-neighborhood contains at least MinPts points. Same goes for the nodes 9, 10, 11, 12 of the green cluster.
3) Although nodes 4 and 6 are not core objects, because their respective “ε”-neighborhood contains only 2 points, which is less than the defined MinPts value, they are “directly density-reachable” respectively from the core nodes 3 and 5 and “density-reachable” from the rest of the core points.
4) The green cluster contains only core points.

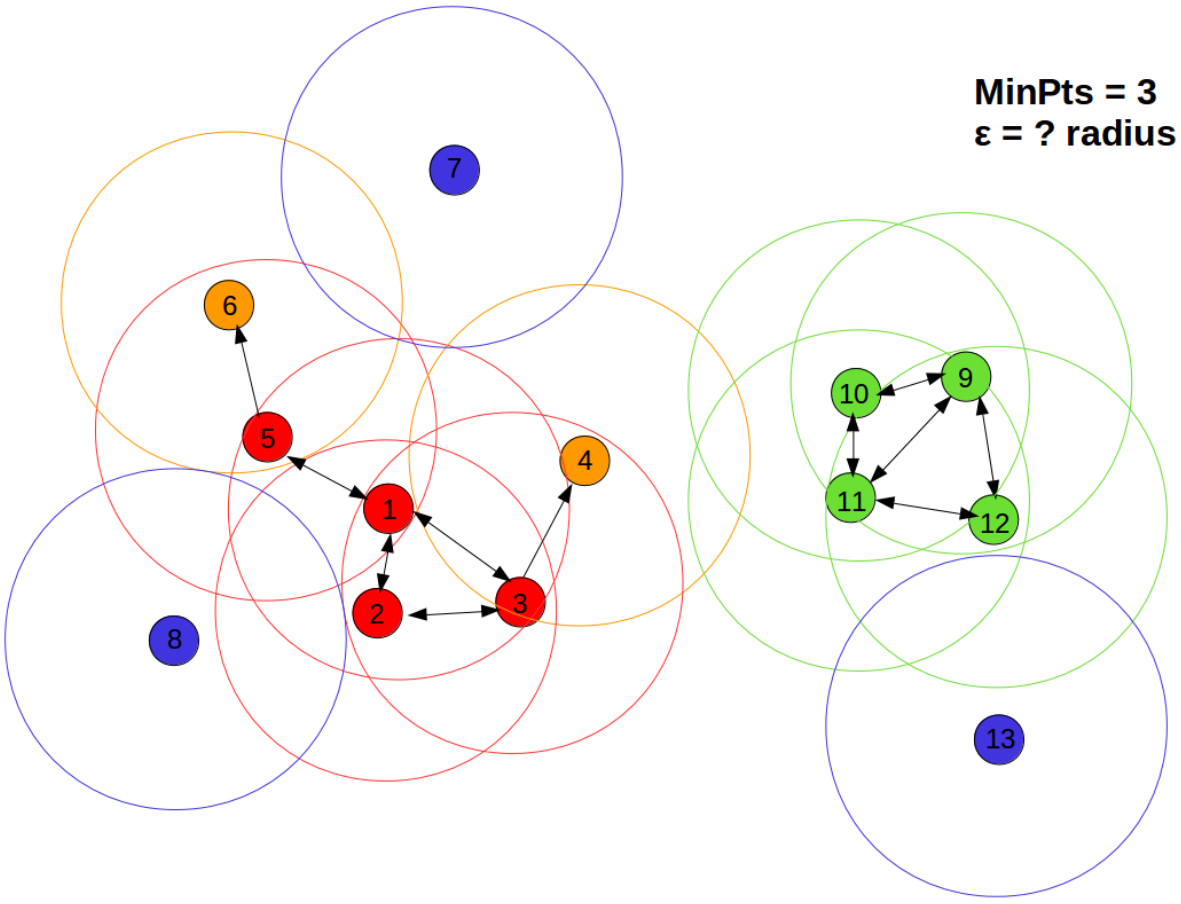

Figure 10. DBScan example, containing low-density clusters.

Intuitively, increasing the MinPts value creates denser clusters and adjusting the Eps value creates clusters of different sizes (based on the defined distance function). The combination of these parameters is important to reveal unexpected patterns for the current data and task at hand and finding the right balance between them is essential.
Figure 11. depicts the same example as the one described above, but with different MinPts value. The radius of the Eps-neighborhood is the same. As one can initially see, the red cluster is, as expected, denser, while the green cluster has the same number of points, although not all of them are core objects. The difference between the two examples is described in details below.

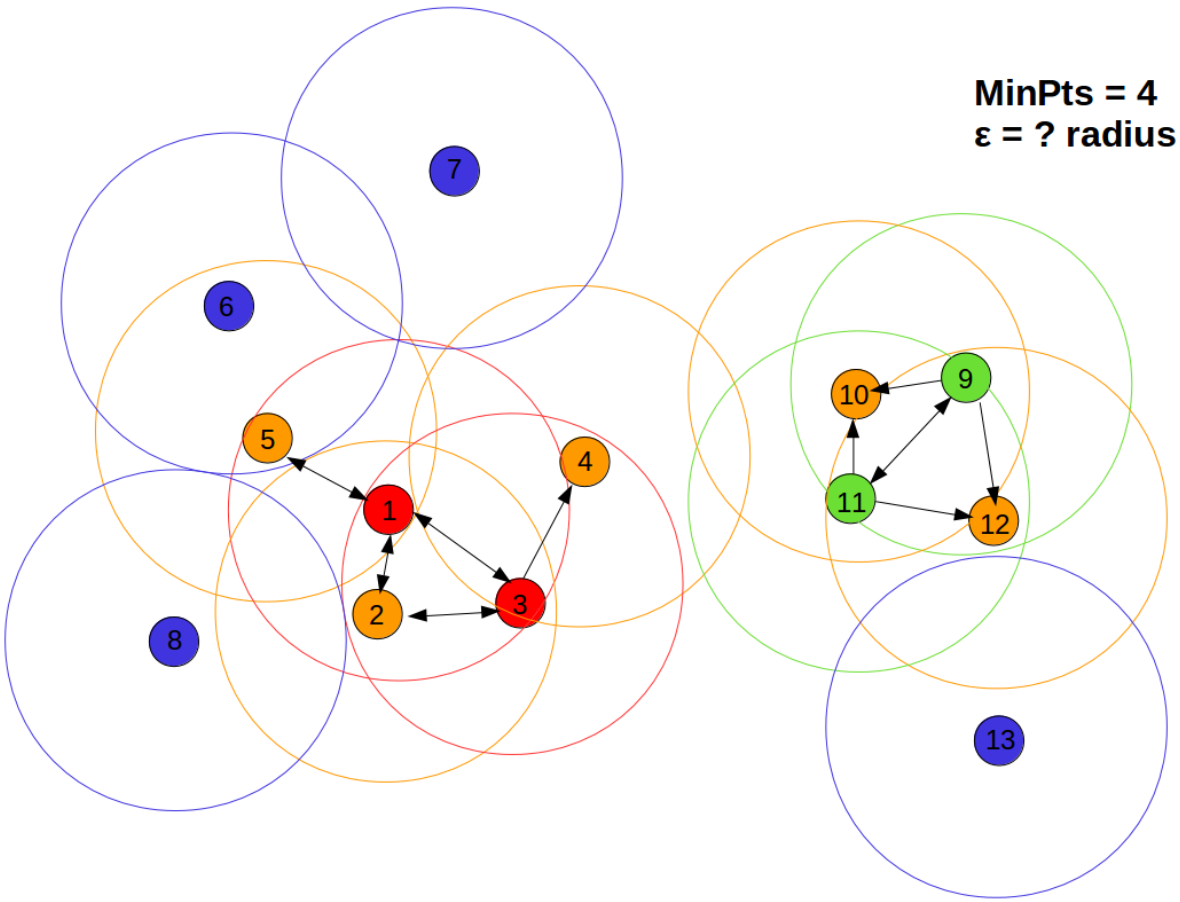

Figure 11. DBScan example, containing higher density clusters.

1) The red cluster has only two core points (1 and 3), due to the increased MinPts value.
2) Node 6 is no longer contained in the red cluster, due to node 5 no longer being a core object and therefore its Eps-neighborhood not considered. This leaves node 6 as part of the "noise" set.
3) The green cluster has only two core points (9 and 11), compared to all points in the previous case. This shows the increased density of the region, due to the fact that both nodes (10 and 12) do not anymore consider points around their neighborhood.

The pseudo code of the algorithm is presented in Listing 1.

Algorithm DBScan(D, MinPts, Eps):

```
// D is the database and contains the set of data points
//MinPts is the variable for the minimum amount of points around an Eps-neighborhood
//Eps is the radius of the density region for each data point (dependent on the chosen distance function)
//"Retrieve_Neighbors" returns the set of all points in the Eps-neighborhood of a point.
//The function is explained in details in the next chapter "Quadtree"

LET Cluster_Label = 0

FOREACH data point AS p FROM D:
        IF p not in a cluster THEN:
                X = Retrieve_Neighbors(p, Eps)
                IF |X| < MinPts THEN:
                    MARK p AS "noise"
                ELSE:
                    Cluster_Label = Cluster_Label + 1
                    LET Cluster_Stack = STACK

                    FOREACH data point AS xP FROM X:
                        MARK xP AS Cluster_Label
                        PUSH xP TO Cluster_Stack
                    END FOREACH

                    WHILE Cluster_Stack NOT EMPTY:
                        LET currentPoint = POP FROM Cluster_Stack
                        LET Y = Retrieve_Neighbors(currentPoint, Eps)

                        IF |Y| >= MinPts THEN:
                            FOREACH data point AS yP FROM Y:
                                IF yP NOT MARKED AS "noise" AND NOT IN A cluster:
                                    MARK yP AS Cluster_Label
                                    PUSH yP TO Cluster_Stack
                                END IF
                            END FOREACH
                        END IF
                    END WHILE
                END IF
        END IF
END FOREACH
```

Listing 1. DBScan algorithm for density-based clustering. [16]

3.1.4.2 Outlier detection

Outlier detection methods are used to discover anomalies in large data sets. This thesis uses a statistical method to discover outliers in the temporal and spatial domain of each data object, which narrows down the problem to the finding of anomalies in linear data. Consider the following definitions:

Assume the existence of a database D and a set of linear values s_1 ..... s_n , $s_i \in D$.

Definition 1. *The arithmetic mean of a data set, denoted as μ, is defined by* $\frac{\sum_{i=1}^{n} s_i}{n}$

Definition 2.*The standard deviation of a data set, denoted as σ, is defined by* $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(s_i-\mu)^2}$

Normal (Gaussian) distribution

Normal distribution in statistics is a continuous probability distribution and is used to represent randomly sampled values. They are useful with randomly sampled data, mostly because of the central limit theorem (CLT), which states that " *the distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution*" [17]. The probability density of the normal distribution or also known as the normalized Gaussian function is:

$$f_{\mu,\sigma^2}(x) = \frac{1}{\sqrt{2*\Pi*\sigma^2}} * e^{\frac{-(x-\mu)^2}{2*\sigma^2}}$$

where μ *is the mean,* σ *is the standard deviation and* σ^2 *is the variance.*

Varying the standard deviation or variance of the function, changes the width of the Gaussian distribution. Figure 12 illustrates several continuous probability density functions of the normal distribution. The blue, red and green functions have the same mean value, but their variance values vary. The green function's variance and mean value are different. This functions (or their 2D equivalents) can be used as smoothening or sharpening convolution kernels in many visualization fields. (e.g. Gaussian blur in image processing).

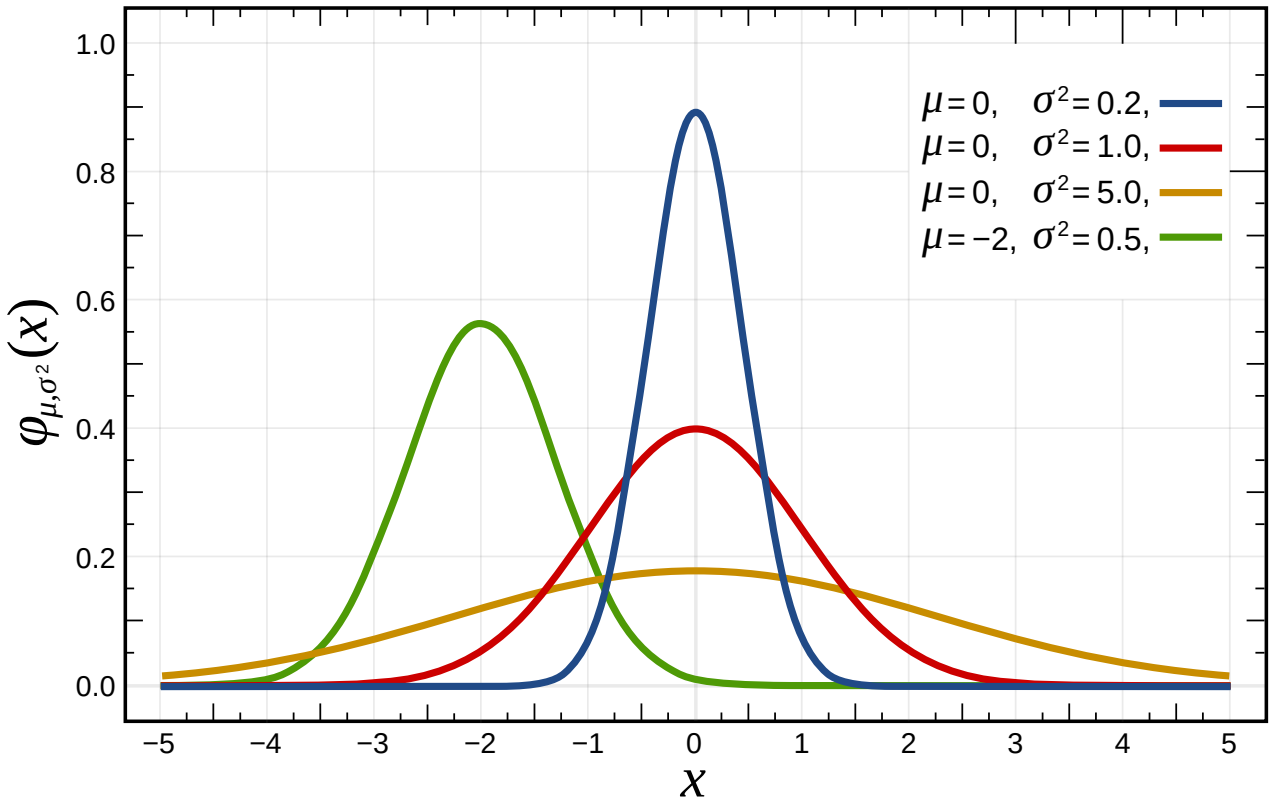

Figure 12. Probability density functions of the normal distribution with varying variance and mean.

[Graph source https://upload.wikimedia.org/wikipedia/commons/7/74/Normal_Distribution_PDF.svg]

Statistical method

Multitude types of data exist, e.g. multi-dimensional, temporal, hierarchical or network data. This variety contributes to the complexity of the task to find anomalies in large data sets. The described clustering algorithm (DBScan) in chapter 3.1.4.1 can successfully detect noise data points in multivariate data. Other popular techniques for the detection of outliers involve different clustering methods (centroid-based clustering, e.g. k-means), support vector machines, neural networks and more. Choosing the right algorithm is dependent on the scale, type and dimensionality of the data. This thesis aims to detect the check-in routines of users from the social media platform “Foursquare” and highlight outlying behavior, such as significant increases and decreases of check-ins per venue, cluster of venues or category. Each object of interest can be interpreted as the set of individual check-in values over the entire temporal domain, which narrows down the problem to the discovery of outliers in linear data. One of the most popular methods of finding anomalies in linear data is using statistics. This thesis involves the usage of a statistical method, described below, to discover anomalies in the check-in behavior of people.

Assuming the data values are normally distributed, Figure 13. depicts the probabilities of values to be contained in 1, 2 or 3 standard deviations. More formally:

$$Pr(\mu-\sigma \leqslant S \leqslant \mu+\sigma) \approx 0.6827$$
$$Pr(\mu-2\sigma \leqslant S \leqslant \mu+2\sigma) \approx 0.9545$$
$$Pr(\mu-3\sigma \leqslant S \leqslant \mu+3\sigma) \approx 0.9973$$

where S is the control value.

The standard deviation defines how spread the values in a data set are and lets the user define the standardization for the routine or normal behavior. The problem of finding outliers with this method narrows down to a user-defined multiplier, which controls the value deviation level. This allows the user to define what an anomaly or unexpected behavior is for a particular analysis task.

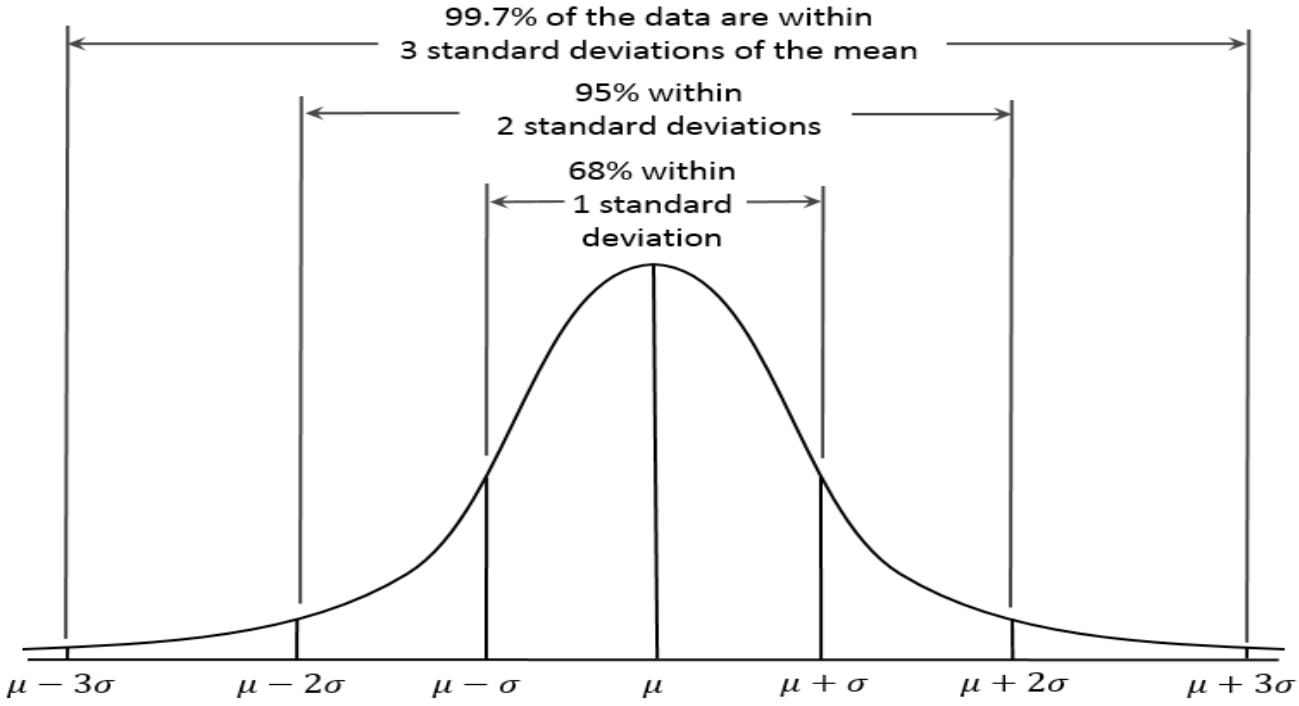

Figure 13. Data probability percentages using a standard normal distribution (with variance σ^2 = 1)
[Graph source: https://upload.wikimedia.org/wikipedia/commons/a/a9/Empirical_Rule.PNG]

3.1.5 Spatio-temporal data analysis

According to Keim et al., "s*patial data, is data with references in the real world, such as geographic patterns in spatial and/or temporal data requires special techniques measurements, GPS position data, and data from remote sensing applications*" [10]. Finding patterns in such data sets is challenging, due to "*scale and uncertainty*"[10]. Most of the time, such data has high velocity, which contributes to its temporal resolution and defines its large scale.

Uncertainty, on the other hand, due to the fact that spatio-temporal data (especially coming from LBSN) is often integrated (from multiple heterogeneous sources), interpolated or aggregated along its spatial or temporal dimension.
Understanding such data is the main goal towards the derivation of knowledge about the dynamics of urban areas. The analysis of spatial data is often concerned with the implementation of algorithms to identify clusters, their visual mappings, the integration of data structures (e.g. a Quadtree) to increase the efficiency of the involved algorithms and the definition of distance or similarity functions, which often require domain knowledge about the problem. Analysis along the data's temporal dimensions is primarily concerned with finding trends and correlations between individual temporal units and their aggregation [10].

3.2 Computational geometry

Computational geometry involves the study of algorithms and methods, in regards to geometry problems. This chapter presents such problems, contained in this thesis, and the solutions to these.

3.2.1 Quadtree

Quadtrees are tree data structures where each node of the tree has exactly four children nodes and recursively subdivides a two-dimensional space into four subregions. Quadtrees have different applications, e.g. spatial indexing, collision detection, image compression or even physical simulations. Depending on the task, different types of Quadtrees exist.

This thesis uses a Quadtree for several tasks:

1) Collision detection between rectangles.
2) Retrieving neighbors around a given perimeter of a data point.
3) Finding the visual centroid of a polygon.

3.2.1.1. Rectangle Collision Detection

Problem: Given a set of rectangles and a target rectangle, find if the target rectangle collides with at least one rectangle from the set.

Naive approach:
The naive approach involves, as expected, iterating through the whole set and checking each rectangle for collision. This results in O(n) run-time complexity, where n is the size of the set.

Using a Quadtree:
The algorithm involves adding a rectangle to the Quadtree and searching for collision for a given target rectangle. The tree could be used for fast and efficient searches, once all objects have been added once to the tree.

1) ADD function

Function ADD (currentNode, rectangle):

```
// currentNode is the pointer to the current node of the tree
// rectangle is the candidate object for adding to the tree
// The function GetBounds recovers the bounds (x, y, width and height) of an object
// The x and y coordinates are assumed to be in the top left corner of the object

LET nodeBounds = GetBounds(currentNode)                    i)
LET rectBounds = GetBounds(rectangle)

LET Q1 = Top right quadrant of current node                ii)
LET Q2 = Top left quadrant of current node
LET Q3 = Bottom left quadrant of current node
LET Q4 = Bottom right quadrant of current node

LET Quadrants = Array(Q1, Q2, Q3, Q4)

FOREACH Quadrant AS Q FROM Quadrants:                      iii)
        IF rectBounds.x >= Q.x AND rectBounds.width <= Q.width
     AND rectBounds.y >= Q.y AND rectBounds.height <= Q.height:
                        //Recurse until the rectangle is not fully contained in any of the subnodes
                        ADD(Q, rectangle)
                        RETURN
        END IF
END FOREACH

// Push the rectangle to the leaf node of the current node
PUSH rectangle TO currentNode.leafNode                     iv)
```

Listing 2. ADD function, describing the process of adding a rectangle object to a Quadtree.

The idea of the algorithm is to add the target rectangle to a leaf node in any of the root's subnodes. The requirement is that the target rectangle is not fully contained in any of the current node's subnodes or quadrants. Figure 14 illustrates this. For example, consider the rectangles A and D to be added to the Quadtree. Rectangle A cannot be fully contained in any of the root's sub-nodes, therefore it is added to the root's leaf node. Rectangle D is contained in the third quadrant of: the root, its first and second sub-node levels and is added to the displayed leaf node, using recursion. Figure 9 depicts the tree representation

of this example and shows how the tree expands, as more rectangles are added.
Some remarks on the above presented algorithm:
i) The algorithm starts by retrieving the bounds of the current node of the tree (The start node is always the root).
ii) Each of the 4 quadrants is calculated from the dimensions of the current node and added to an array for iteration.

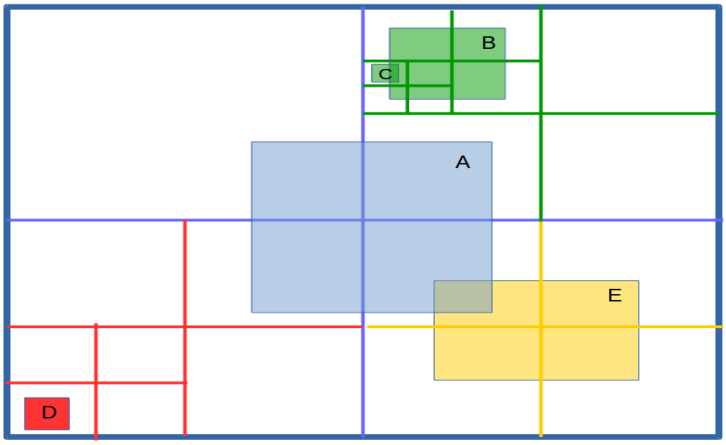

Figure 14. Spatial representation of a Quadtree. Shows the process of adding and searching for collision between the represented rectangles.

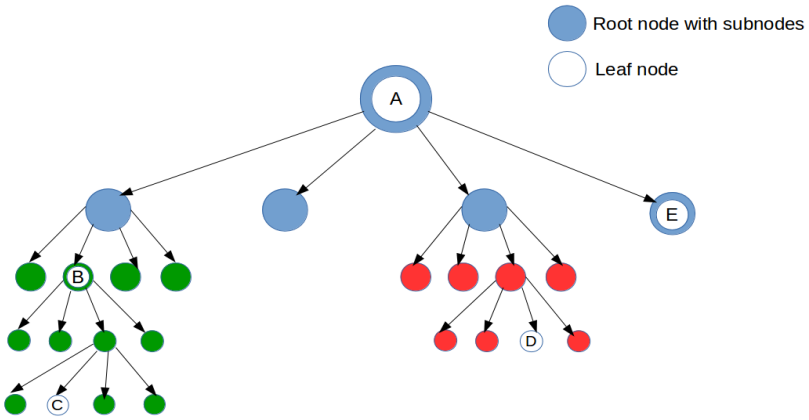

Figure 15. Tree representation of a Quadtree. Shows tree expenditure as new objects are added to the tree.

iii) The quadrants are being iterated and the rectangle's bounds is being checked to each one of them for containment. If the rectangle is contained in any sub-nodes, the algorithm recurses with the containing quadrant as the current node, until it reaches a deeper tree level, where the rectangle is not fully contained in any of the current node's quadrants.
iv) Reaching this part of the function means that no quadrant of the current node could fully contain the rectangle. Therefore it is added to the leaf node of the current node.

2) SEARCH function

The idea behind the search function is to find if and with which rectangles (if needed) a target rectangle collides. The search starts from the root node by traversing through the sub-nodes, which fully contain the target rectangle. The target rectangle is checked for collision at each tree depth's leaf node, containing an array of earlier included rectangles. The following check is performed, to discover if two rectangles collide:

Assume r_1 with
w_1 = width of r_1
h_1 = height of r_1
x_1 = Center X of r_1
y_1 = Center Y of r_1

is the target rectangle and respectively r_2 is the rectangle to check if the two collide.
1) Compute the “Minkowski sum” of the two rectangles, which produces a third rectangle with the following parameters:

$w_m = \frac{(w1+w2)}{2}$, defining the width.
$h_m = \frac{(h1+h2)}{2}$, defining the height.
$d_x = x_1 - x_2$, defining the center x coordinate.
$d_y = y_1 - y_2$, defining the center y coordinate.

Then, the two rectangles collide if $|d_x| \leqslant w_m \wedge |d_y| \leqslant h_m$.

The pseudo code snippet of the function is described in Listing 3.

Consider Figure 16. to depict the algorithm's search process. Since rectangle A is not fully contained in any of the quadrants, all existing rectangles have to be checked for collision. This results in the worse-case scenario with run-time complexity of O(n), where n is the number of nodes in the tree. Where this data structure and algorithm exceeds is when dealing with large number of smaller objects. For example, consider the rectangles B and C. As it can be easily seen, these two collide with each other. The algorithm can find this out only by traversing through the green part of the tree, while narrowing down the spatial search area at each iteration and excluding rectangles outside of it.

Function SEARCH (currentNode, rectangle):

```
// currentNode is the pointer to the current node of the tree
// rectangle is the target rectangle for the collision check
// Assume rectangle is an object containing its bounds(x, y, width and height)
// Assume currentNode is a node object, containing its subnodes in an array called "Quadrants"
// The function checkCollision(rect1, rect2) checks for collision between the two objects with the
// method described above

// An array containing objects colliding with the target rectangle
LET  collisionObjects = ARRAY

// 1. Check for collisions with all leaves in current node

FOREACH leaf objects AS checkObject FROM currentNode.leafNode:
        IF checkCollision(rectangle, checkObject) == TRUE:
                        PUSH checkObject TO collisionObjects
        END IF
END FOREACH

//2. If object can be contained fully in one of the subnodes, recurse
FOREACH Quadrant AS Q FROM currentNode.Quadrants:
        IF rectangle.x >= Q.x AND rectangle.width <= Q.width
      AND rectangle.y >= Q.y AND rectangle.height <= Q.height:
                        //Recurse while extending the array containing the found colliding objects
                        EXTEND collisionObjects WITH RESULT FROM SEARCH(Q, rectangle)
                        RETURN
        END IF
END FOREACH

//3. If object is not contained fully in any of the subnodes, check all of the children of the current node
//    recursively for collision
FOREACH Quadrant AS Q FROM currentNode.Quadrants:
                EXTEND collisionObjects WITH RESULT FROM SEARCH(Q, rectangle)
        END IF
END FOREACH

RETURN collisionObjects
```

Listing 3. SEARCH function, describing the algorithm to detect collisions between between rectangles using a Quadtree.

3.2.1.2. Retrieve neighbors around a perimeter of a data point

Solving this task fast and efficient is of high importance for the density-based clustering,

described previously. The data points are represented as squares, instead of circles, for the sake of simplicity and code reusability.

Problem: Given a data point, find all neighboring data points around a given perimeter.

Naive approach:
Finding neighbors around a perimeter is a challenging task in terms of run-time complexity. The naive approach involves iterating through all data points and calculating the distance (dependent on the defined distance function) to find out if two points are neighbors. The run-time complexity is O(n), where n is the number of all data points. If an algorithm requires to find the neighbors of each point in the data set, the run-time complexity grows to quadratic, O(n^2), which is unacceptable when working with large data sets and user interactions.

Using a Quadtree:
The algorithm involves adding all of the data nodes into the Quadtree once. The ADD function, described above, can be reused for this task. The second task involves retrieving the set of neighbors around the perimeter of a given target data point.
The algorithm for retrieving neighbors around a point narrows down to finding all collisions between the target rectangle and the set of rectangles. The target rectangle is the initial rectangle expanded with the given parameter size. The green rectangle in Figure 16 illustrates the target rectangle and the green dashed rectangle around it, the perimeter size. The red data points are the found neighbors and the blue rectangles depict the ones, that are not defined as neighbors by the algorithm.
The run-time improvements can be easily seen. The target rectangle does not need to check all rectangles for collision, but only the ones that are contained in the same sub-node or its children. As the area and the data set grow larger, it gets even more time-consuming to retrieve neighbors using the naive approach. This data structure and algorithm perform best for smaller defined perimeters, as this can minimize the number of node traversals and checks needed.

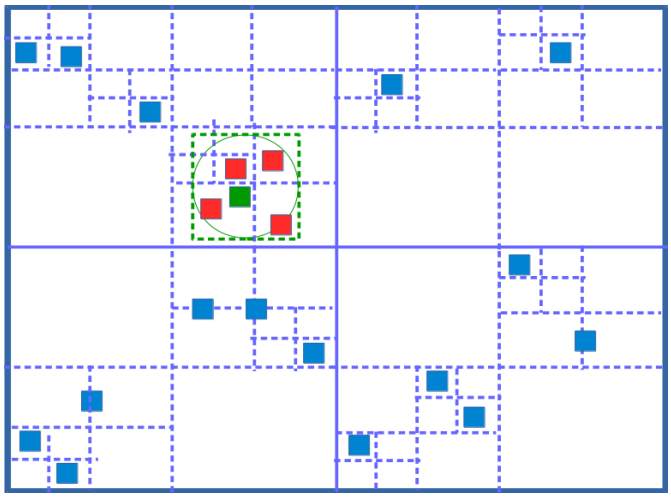

Figure 16. Finding neighbors around the perimeter(dashed green rectangle) of a given point (green rectangle). The red points are the recovered neighbors from the algorithm.

3.2.1.3. Finding the visual centroid of a polygon

This thesis uses concave hull of points as nodes to a graph. When dealing with concavity, finding the centroid is not always the best solution, due to it being outside the concave polygon. This leads to edges of the graph leading to nothing. Vladimir Agafonkin, the creator of “Leaflet”, and the engineers at “Mapbox”[18] came up with an efficient algorithm for finding the visual centroid of a polygon. To achieve this efficiency, they used a Quadtree. Consider Figure 17, which contains a concave polygon. The centroid of the polygon, defined as the average of all points, should lie somewhere outside the polygon, which is not an ideal solution. A better solution would be to find the point or a Quadtree cell that is guaranteed to be within a global optimum, in respect to the distance from the polygon. Their algorithm for finding this point, named the visual centroid, is as follows:

“1) Start with a few large cells covering the polygon-center
2) Recursively subdivide them into four smaller cells, probing cell centers as candidates and discarding cells that can't possibly contain a better solution better than the one we already have“. [18]

Figure 17. shows a rough idea of the process behind finding the visual centroid of the displayed (in blue) polygon. Each node of the Quadtree is divided into four quadrants and the distance from the polygon to the node's center is calculated. Then, cells are discarded if the algorithm found another cell with better solution. The algorithm proceeds until it reaches the global optimum.
This thesis uses the solution provided from “Mapbox”[18] to produce the visual centroids of clusters, visualized by a concave hull of points.

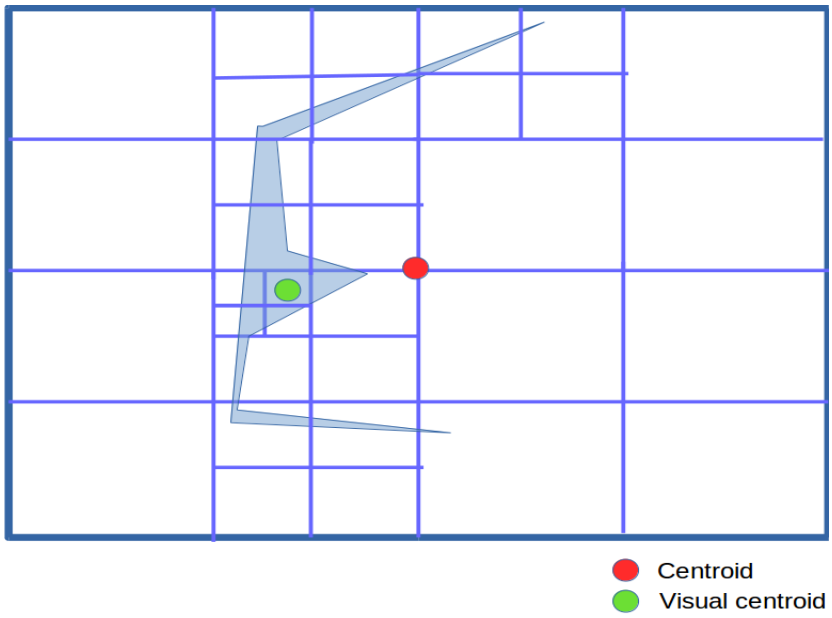

Figure 17. Rough idea behind the process of finding the visual centroid of a concave polygon.

3.2.2 Bezier curves

Bezier curves, discovered by the french engineer Pierre Bezier, are parametric curves used frequently in computer graphics. Bezier curves can be both described in terms of polynomial functions or the combination of multiple linear interpolations [19]. The polynomial representation is defined by using Bernstein polynomials:

$$Bezier(n,t)=\sum_{i=0}^{n}\binom{n}{i}*(1-t)^{n-i}*t^{i}*w_i$$

for an n^{th} order curve where $0 \leqslant t \leqslant 1$ and w_i is the weight of each point. [19]

A more intuitive way to present the idea behind Bezier curves is by using linear interpolations. Figure 18, 19, 20 and 21 depict the process behind calculating a curve starting from the point S and ending at the point E. The points C1 and C2 are control points, needed for controlling the curvature when calculating a cubic curve. The process can be seen as simultaneous movement along the lines A and C1, C1 and C2 and C2 and E. As the points move along the lines, the linear interpolation and the linear interpolation of the linear interpolation between them is calculated for each given t, $0 \leqslant t \leqslant 1$. The parameter t can be seen as percentage left before finishing the full curve between the points. Figure 18, 19 and 20 show steps of the process, respectively for t=0.25, t = 0.5 and t = 0.75. At each step, a tangent point to the target curve is retrieved (T1, T2 and T3). Figure 21. shows how the tangent points align after a finite amount of steps between t = 0 and t = 1.

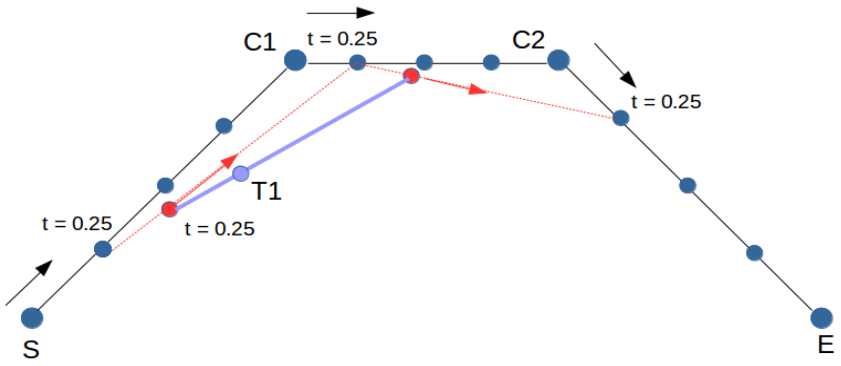

Figure 18. An intermediary step for t=0.25 of the process of calculating a Bezier curve using linear interpolations. C1 and C2 control the curvature. T1 is a calculated tangent point to the curve.

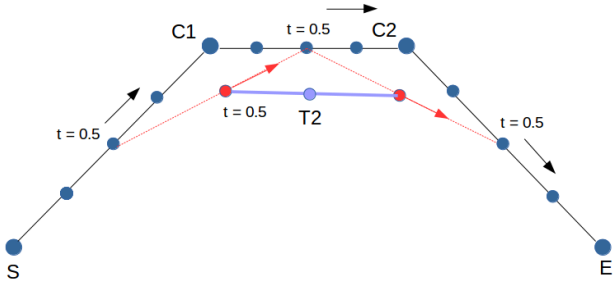

Figure 19. An intermediary step for t=0.5
C1 and C2 control the curvature.
T2 is a calculated tangent point to the curve.

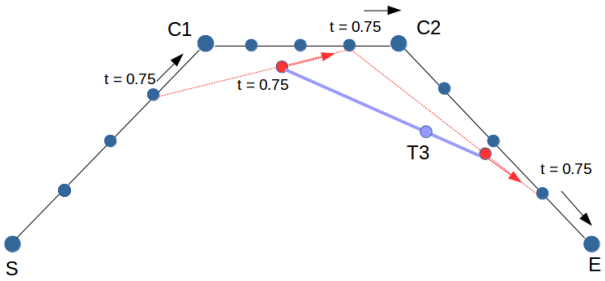

Figure 20. An intermediary step for t=0.75
C1 and C2 control the curvature.
T3 is a calculated tangent point to the curve.

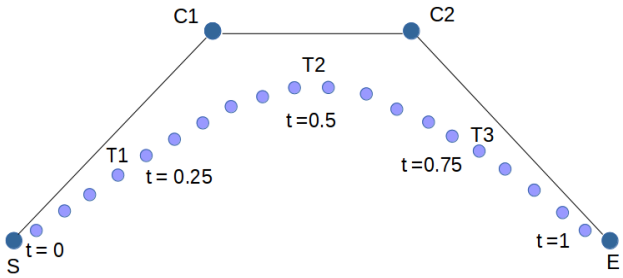

Figure 21. Tangent point alignments after a finite amount of steps between t = 0 and t = 1.

3.2.3 Concave hull of a set of points

This chapter describes an algorithm, proposed by Emil Rosén et al.[20], to recover the concave hull of a set of points. A couple definitions are needed, before presenting the algorithm:

***Definition 1.* Convex hull:**
"A convex hull of a set of points is the uniquely defined shape that minimizes the area that contain all the points, without having any angle that exceed 180 degrees between two neighboring edges." [20]

Definition 2. Concave hull:
"A concave hull of a set of points could be defined as the shape which minimizes the area of the containing shape, but allowing any angle between the edges."[20]

Figure 22. shows several formed hulls from the same set of points. The red hull is a convex hull, because it does not contain an angle between two neighboring edges that is bigger than 180 degrees. The blue hulls represent concave hulls with different levels of concavity. As the level of concavity rises, the concave hull gets more and more distorted, so choosing this parameter wisely is important for the current task at hand. [20]

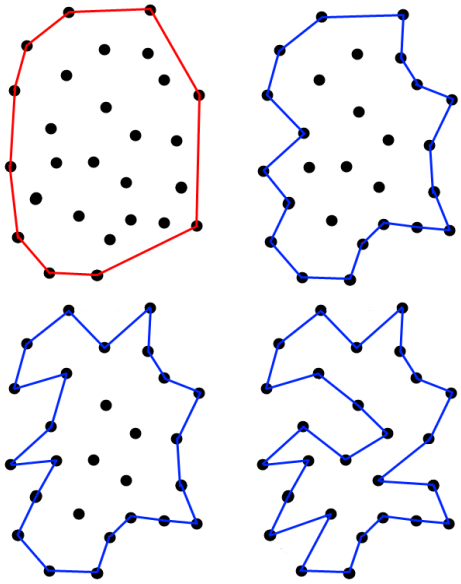

Figure 22. A convex hull (red) and concave hulls (blue) with different levels of concavity. [20]

Emil Rosén et al.[20] propose a fast and efficient algorithm for constructing the concave hull of a set of points. First, they calculate first the convex hull of the set, which is then "iteratively opened up to create the concave hull"[20]. The first phase of the algorithm is the construction of the convex hull.

Convex hull construction:

Emil Rosén et al.[20] used a "Divine and conquer" algorithm to construct the convex hull. A summary of the algorithm is as follows:

"1)*To find the hull, the points are first sorted in, for example, lexicographic order on the x*

coordinate using quicksort.
2) *All points are divided into sets of one or two neighboring (according to the sorted axis) points.*
3) The points are then merged together creating an edge between them. The resulting subset of edges are merged together to create subsets of convex hulls, where the points on the boundary know about their clockwise and counterclockwise neighboring boundary points.
4) This goes on recursively until all subsets have been merged together into one convex hull."[20]

Concave phase:
The idea is to remove the longest edge from the convex hull at each iteration. *"All points are then searched to select the best candidate to add to the boundary between the endpoints of the selected edge. The search criteria for the points are that the largest angle from the new point to the end points of the old edge should be as small as possible"*. [20]
The pseudo code of the algorithm described by Emil Rosén et al. [20] follows below:

```
Data: list A with edges for the convex hull
Result: list B with edges for a concave hull

Sort list A after the length of the edges;

while list A is not empty
        Select the longest edge e from list A;
        Remove edge e from list A;
        Calculate local maximum distance d for edges;

        if length of edge is larger than distance d
                Find the point p with the smallest maximum angle a;

                if angle a is small enough and point p is not on the boundary
                        Create edges e2 and e3 between point p and endpoints of edge e;
                        if edge e2 and e3 don't intersect any other edge
                                Add edge e2 and e3 to list A;
                                Set point p to be on the boundary;

        if edge e2 and e3 was not added to list A
                Add edge e to list B;
```

Listing 4. Algorithm for the calculation of the concave hull of a set of points, after the derivation of their convex hull. [20]

4. Practical concepts and implementation

This chapter describes in details the technical ideas and concepts behind the implementation of the monitoring and visual analytics system, presented in this thesis.

4.1 Monitoring System

Monitoring systems or more specific social media monitoring systems deal with the heterogeneity of multiple communication channels or APIs and their limitations in terms of API limits and Quality of Service, different data models, schemas and confronting or problematic data. Other tasks include the preparation of data for (in this case) the analysis process. This can involve different pre-aggregation and data transformation strategies. This thesis presents a monitoring system, which collects data in (near)real-time from multiple social media sources, transforms it and stores it in a database with fast access.

4.1.1 Concepts and architecture

This chapter discusses the practical concepts and architecture behind the implementation of the monitoring system. The first chapter argues about the need of a flexible database management system for the problem at hand. The next few chapters describe the architecture of the system, its components and the design decisions made to overcome several limitations.

4.1.1.1 Non-relational DBMS

Non-relational (NoSQL) database management systems (DBMS) are concerned with the storage, management and retrieval of non-relational data. Tabular relations used in relational DBMS are not always the best answer to a problem, even though they are easier to maintain. There are many diverse ways to store data in a non-relational way. To fulfill the fast-growing needs of organizations for the storage and maintenance of different types of data, many non-relational database management systems have been developed over the years. Each one with its benefits and disadvantages in terms of problem specifics. Some of the most popular ways to store data in a non-relational database are:

Key-value store: Uses an associative array (also called map) to store data.
Document databases: Again, uses an associative array, but instead of simple values, assigned to each key, the values used are complex data structures, called documents. Each document can contain multiple key-value relationships with different hierarchies.
Wide-column stores: This type of data storage can be easily compared with relational data storage, but with the significant difference, that each column in a table can contain multiple encapsulated key-value pairs.
Graph databases: As its title suggests, this type of NoSQL DBMS stores data in a graph-related way. This type of storage is suited for a specific task-related problem, which

requires large-scale data to be represented as a network or a graph.
So, why are non-relational DBMS better and more efficient for some data types and tasks than relational DBMS? Consider some of the advantages and disadvantages of both:

Advantages:

Scalability: It is easier to transparently scale data "horizontally" across multiple machines, when using document stores for example, because the data is stored into a single document, instead of being spread out across multiple relations.
Speed: Storing data in a non-relational, instead of a relational, database results in a trade-off between speed and storage. For example, document stores collect multiple relations of the data into a single document with multiple hierarchies, which does not require slow and heavy operations, like JOINs, to recover a complete entity object. On the other hand, storing data in a non-relational way results in many duplicated data fields, which requires more storage.
Design simplicity: The schema-less storage of data simplifies the design and allows for the easy storage of all kinds of data.

Studying the dynamics of urban life in near real-time requires the storage and management of great amounts of data. This data often comes in different data models and schemata. To account for this, the database management system should be flexible enough to give fast access to this data and fulfill the analysis needs.
To summarize these advantages, consider the following simplified scenario of the problem, described in this thesis:
Social media platform "Foursquare" exposes their data through a RESTful API. The data consists of users, who check-in at venues at different times of the day. Each venue is assigned at least one category, which uncovers its semantics. Each (sub)category can contain multiple subcategories. Figure 23. represents a possible relational solution to this problem.

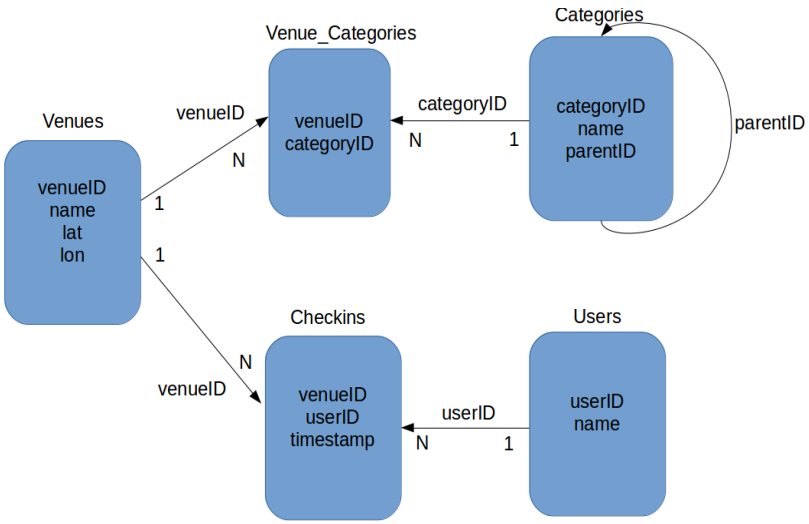

Figure 23. Relational schema of a simplified scenario.

The solution consists of 3 relations, stored in 5 tables. “Venue_Categories” table connects categories to venues in a many-to-many relationship. “Checkins” table connects venues to users the same way. The parent-child relationship between the categories is represented with the “parentID” attribute.
Consider the simple query, which recovers the check-in count for the category “Travel & Transport”. The output has to be the name of the category together with its check-in count. An example of the SQL query is shown in Listing 5.

```
SELECT c.name, COUNT(ch.userID)
FROM Categories c, Venue_Categories vc, Checkins c
WHERE c.categoryID = vc.categoryID
AND vc.venueID = ch.venueID
AND c.name = 'Travel & Transport'
```

Listing 5. A simple SQL query, which recovers the check-in count for the category “Travel & Transport”.

It requires two JOINs to recover the check-ins for a single category. Things get problematic when tables consist of millions of records, with JOIN operations between them, which is a necessary, but also a very time-consuming task. As a system becomes more complex, allowing for a multitude of operations to be executed, the complexity of database queries grows as much. Consider another query example. Assume a system needs to retrieve the check-in count of each user for the same category (“Travel & Transport”). The output has to contain the category name, user name and the check-in count. The more advances query is shown in Listing 6.

```
SELECT c.name, u.name, COUNT(ch.userID)
FROM Categories c, Venue_Categories vc, Checkins c, Users u
WHERE c.categoryID = vc.categoryID
AND vc.venueID = ch.venueID
AND u.userID = ch.userID
AND c.name = 'Travel & Transport'
GROUP BY u.userID
```

Listing 6. An advanced SQL query, which recovers the check-in of each user for the category “Travel & Transport”

This query requires even more JOIN operations to retrieve the needed information. Although index structures and query optimizers help tremendously, as the scale of the data grows, so does the number of relations and the size of each table.
Non-relational databases allow the possibility to structure the stored data in any way possible using hierarchical relations. This gives the user flexibility to choose how to store his data according to the current problem at hand.
Figure 24. shows a non-relational solution to the previous problem. This time, a collection named "CategoryCheckins” has been defined with a sample document inside. The document

is schema-less and contains all the attributes and relations needed to answer easily the above defined queries, by just collecting data from a single document, without having to perform heavy JOIN operations.

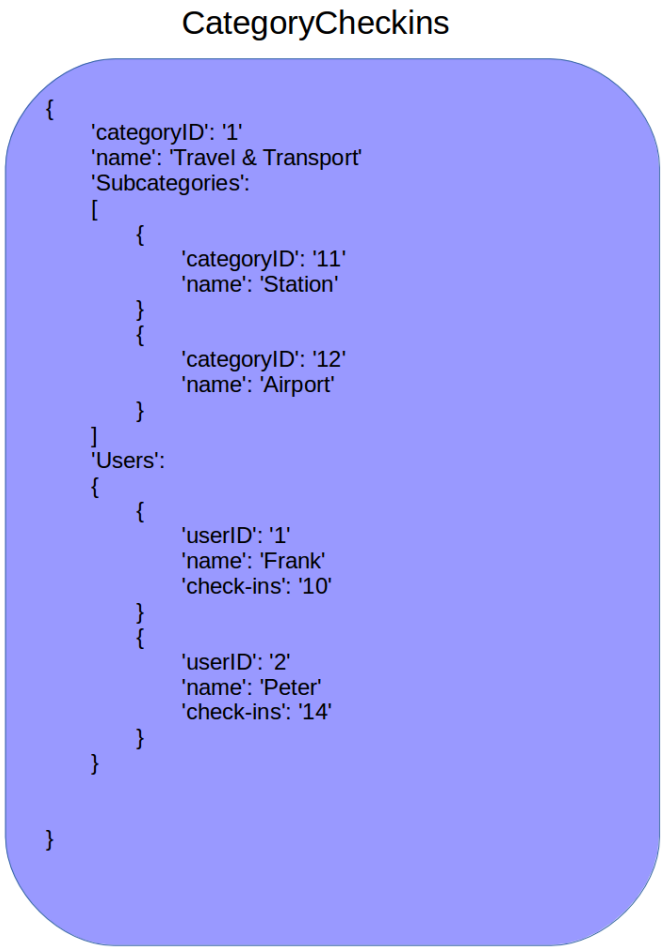

Figure 24. Non-relational example of a simplified scenario.

This example illustrates some of the advantages of non-relational DBMS, but what are their drawbacks, in comparison to relational DBMS? Which one is better suited for the storage of social media data for not only (near) real-time, but also historic urban analysis?
An essential advantage of relational database is the storage of data in relations, which allows to delete or update rows with less effort, compared to e.g. document-oriented databases. Also, data type constraints, used in relational databases, help to preserve the integrity of the data, which is crucial for many organizations.

In summary, there are multiple weighing factor for the decision of using a non-relational, or more specific, document-oriented database management system. The process of collecting information from social media services in real-time requires the fast insertion and retrieval of data, without the need to update or delete it even once. In addition to this, the data coming from such services is often type-less. Another important factor for choosing a document-oriented non-relational database is its flexibility. It provides the means for easier information integration, fast data retrieval and schema-less design. This simplifies the process of storing and retrieving large-scale data, which is important for a fast and interactive analysis. Another big advantage is the possibility to easily scale this data horizontally, as it grows larger, which is inevitable with data from location-based social platforms.
This thesis uses “MongoDB” to store and manage the data, integrated from multiple social media sources. “MongoDB”, a document-oriented database, provides the means to store such data in any way imaginable. The next few chapters dive deeper into the data model and schema of this thesis's task and outline the importance of having a flexible database management system.

4.1.1.2 Architecture

This chapter reveals the architecture behind the monitoring system and explains roughly the whole monitoring process.
Figure 25. depicts a high-level architecture of the implemented monitoring system. Two social media services are used as sources to be collected regularly: Foursquare and Twitter. To account for this, the monitoring system consists of two sets of scripts. “Foursquare Monitoring” is concerned with the extraction of data from the Foursquare API. The Foursquare API exposes information about its venues and the number of people currently checked-in at each one. Although this data is useful, it does not contain any information about urban mobility, due to a limit on exposed user information, thus the need for Twitter data integration. Some Foursquare users can link their account to the Twitter platform, which automatically sends a tweet at each check-in. This tweet can be collected in real-time from the Twitter Streaming API. The “Twitter Monitoring” script has the task of managing this process. It initiates the connection and passes collected data to the “Foursquare API” script, which communicates with the MongoDB database to store it through a database wrapper. The connections are established through series of security checks, respectively managed by the “Foursquare API” and “Twitter OAuth” scripts, through the use of user-defined configuration files.

4.1.1.3 Pre-aggregation

As more and more information is being collected, the process of searching and retrieving data from this large data pool becomes time-consuming. In addition to this, the data requires further transformation steps before being loaded into the analytics system. The pre-aggregation step has been designed to tackle this problem. As seen in Figure 25, the pre-aggregation script is part of the monitoring system, but is working totally independent.

It collects data from several collections at each time unit (hour, day or month), aggregates them accordingly and stores it as a new document into their respective collections (hourly, daily, monthly). When the user requests history data for analysis, the chosen time frame (e.g. a day) has to be collected and aggregated.

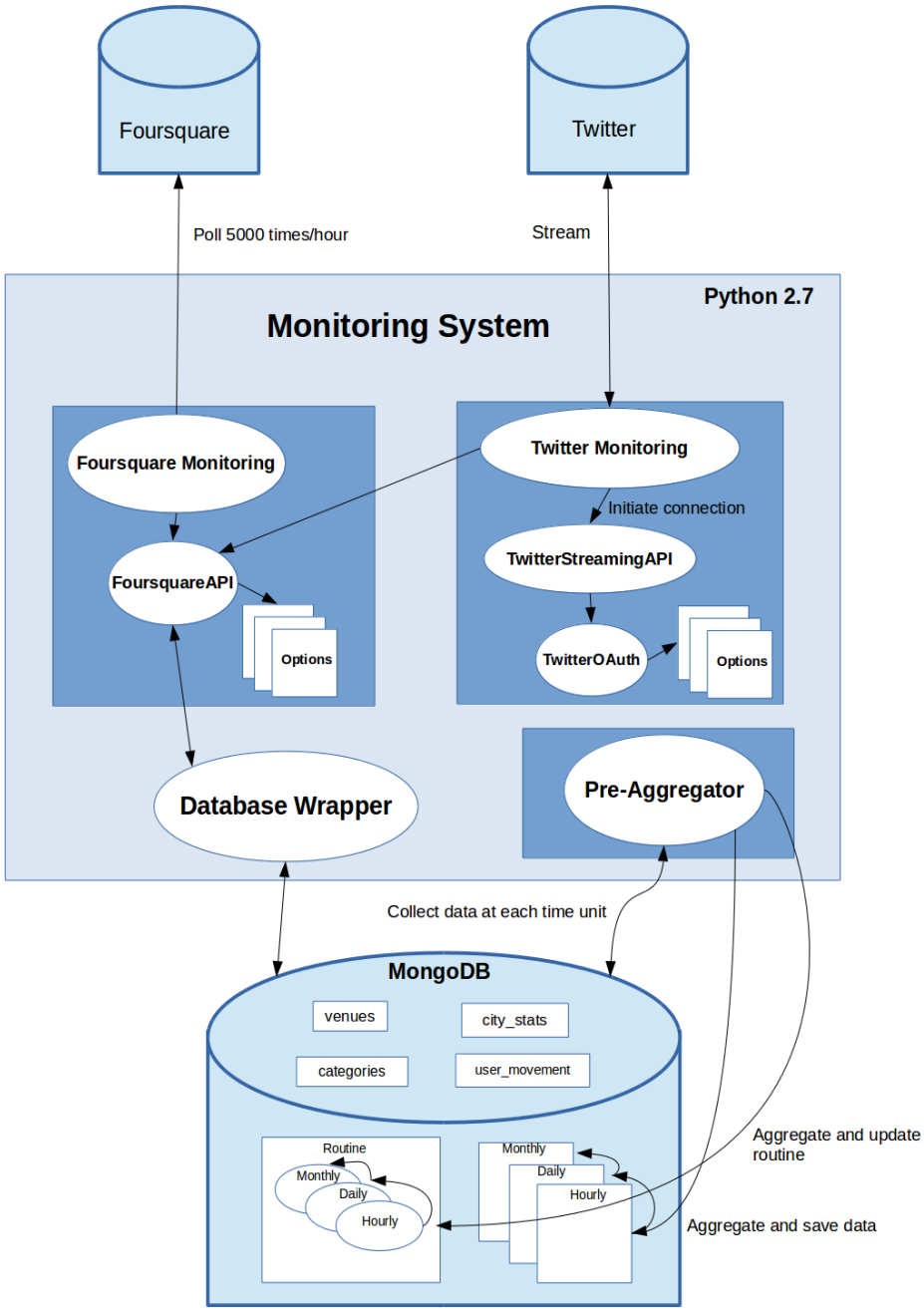

Figure 25. High-level architecture of the monitoring system, implemented in this thesis.

The process of aggregating this data includes the following:

1) Collect data for each time unit (dependent on the type of the aggregation(hourly, daily or monthly)) for the chosen time window.
2) Aggregate data between the time units. The aggregation strategy is described in chapter 4.1.2.3.

The temporal resolution of the data varies significantly, depending on the API limitations and the chosen spatial resolution. Assume the temporal resolution of the data equals 6 minutes, meaning its refresh rate is 10 times per hour and 240 times per day. If a user of the system decides to explore the data of a chosen day, the system has to recover each one of these time units and calculate the aggregations between them, which involves the calculation of deltas between check-in counts for the Foursquare data and the sum of movements between venues or categories for the movement data. This process becomes time-consuming as the selected time window becomes larger.
The pre-aggregation step does this beforehand, automatically and transparently to the user. This speeds up the process of loading history data, because it takes a lot less aggregation steps to complete the task. In regards to the previous example, the system would need only to recover the data from the daily, already aggregated, collection and display it to the user for analysis.
In addition to this, this step involves the pre-calculation of routines per point of interest or category, explained in details in chapter 4.1.2.4.

4.1.2 Implementation

This chapter discusses the strategies and mechanisms behind the implementation of the various monitoring system's components.

4.1.2.1 Foursquare API

Foursquare is a social media platform, which provides search-and-discovery service to its users, using their current location and based on search criteria (e.g. category). Furthermore, it offers a personalized recommendations list for each user to visit and check-in. To stimulate their users to check-in at places, they offer different incentives, like coins and category stickers. Studying the patterns of this check-ins can reveal the city's pulse and lifestyle.

Limitations and decisions

Foursquare exposes information about their venues through a RESTful API and a Streaming API. The Streaming API would provide the ideal means to collect check-ins in real-time, as they come, but there are a few limitations. To collect check-ins from the Foursquare Streaming API, the collecting system needs to:

a) be authenticated as a manager of a certain venue or
b) collect data only for the users, added as “friends”.
As this is not the case, this API cannot provide the needed service for this system. Why not use the Twitter Streaming API for the job? Although, real-time data could be collected from Twitter, it contains only a small subset of user check-ins, which may greatly deviate from their real distribution and in turn hinder the analysis task. Foursquare contains full real-time information about the check-in distribution of its venues. But how to collect this data?
By polling the RESTful API. It exposes the platform's data through different interfaces. The “Venues Search” interface (or also called endpoint) returns a list of venues, matching the search criteria. But there is another problem. This endpoint can be requested at most 5000 times per hour for userless requests and 500 times per hour for authenticated users, in order to scale down the traffic to their servers. This limitation raises more questions. What temporal and spatial resolution of the extracted data can be achieved? Can both be improved? The temporal resolution of the data refers to its refresh rate. The spatial resolution refers to the number of points of interest to be refreshed at each update iteration. The “Venues Search” endpoint returns at most 50 points of interest per request. To overcome this limitation, the endpoint can be called with a category parameter, returning the top 50 venues (by number of check-ins) for this category. This gives the possibility to request this endpoint for each of the categories, their subcategories and so on, going at least 4 levels deep into the hierarchy. There are approximately 850 subcategories, which increases the spatial resolution of the data tremendously, but for what cost? As already mentioned, the endpoint can be polled at most 5000 times per hour, meaning the temporal resolution of the data is at least 10 to 15 minutes, depending on several factors, one of which is the API availability. And vice versa, if the system is set up to poll data only from the 10 main categories, it could achieve a temporal resolution of approximately 12 seconds. The trade-off is clear, the bigger the temporal resolution, the lower the spatial resolution and vice versa. There is no answer on what is best, because it depends on the specifics of the analysis task, which requires domain knowledge of the problem. That is why, this decision is left to the user of the system, dynamically adjustable from a beforehand defined configurations file.

Configurations

The configuration file, depicted in Figure 25. in the Foursquare section, is described in Listing 7. It is split into three sections. The “FoursquareAuth” section is concerned with the authentication to the Foursquare API. Before being able to poll data, the system has to successfully authenticate through a registration and security process. This process requires a client id, client secret for user-less access and “OAuth” token for authenticated access. Although the “Venues Search” API can be accessed user-less, it is worth noting, that not all API endpoints support it, thus the need for an “OAuth” token. The possibility to add more than one token is related with the movement data coming from the Twitter API, explained in the next chapter (4.1.2.2) . The next section, “Foursquare”, sets the search parameters passed to the “Venues Search” endpoint. The “venue_limit” parameter defines the number of venues that will be returned for a single request. As already discussed, the maximum is 50. The “radius” parameter defines the size of the venue search perimeter. The “cat_hierarchy” parameter defines the category hierarchy level, on which, as already

discussed, the temporal and spatial resolution depend.

```
[FoursquareAuth]
client_secret=[CLIENT_SECRET]
client_id=[CLIENT_ID]
oauth_tokens=[OAUTH_TOKEN1], [OAUTH_TOKEN2], ……
[Foursquare]
venue_limit=[VENUE_SEARCH_LIMIT, MAX 50]
radius=[RADIUS_IN_METERS]
cat_hierarchy=[CATEGORY_HIERARCHY_LEVEL, MAX 4]
[City]
country:[COUNTRY]
city:[CITY]
timezone:[TIMEZONE, e.g. Asia/Tokyo]
```

Listing 7. Foursquare configuration file.

The currently developed prototype of the system does not support multiple cities for analysis and comparison. To define the target city for analysis, the user has to set the “city” variable in the “City” section. The “timezone” parameter is needed as well for the correct time-stamp calculation. Changing these parameters should give the user the flexibility to accustom the system to the analysis problem at hand.

Data model

The hierarchical data model of the data, received from the Foursquare API, allows for easier integration into the MongoDB database, which in turn is effortless loaded into to the visual analytics system. Although the structural data model of the data is well-defined, the system needs also to be able to semantically recognize real-world entities and how they are organized, in order to process them efficiently. A lightweight schema is developed for this task. The objects, explained in Listing 8. and Listing 9., are stored in JSON format by the monitoring system and used by the visual analytics system.
Consider Listing 8., The included sub-category for each venue allows the system to dive deeper into its semantics, e.g. a venue could have a “Nightlife” category, but also be a “Bar”. The “count” field contains the number of people, currently checked-in at this venue. A Foursquare user is checked-out of a venue, after having checked-in earlier on two occasions:

1) If a user checked-in at a different venue.
2) If three hours have passed and the user hasn't checked-in at a different venue.

The category object, described in Listing 9., illustrates the flexibility of the non-relational

database management system. The “hereNow” section contains the number of Foursquare users currently checked-in at this category. This check-in count essentially represents the sum of all check-in counts for a single time unit from venues, assigned to the same category. Even though this can be considered as redundancy (in normalized relational data schemata), it can speed up tremendously the interactive analysis process.

Venue object

```
{
        “id”: “object”,                                  // An internal ID of the venue
        “name”: {“type” : “string”},                     // The name of the venue
        “location”:
        {
                “lat”: {“type” : “number”},              // The latitude coordinate
                “lon”: {“type” : “number”}               // The longitude coordinate
        }
        “categories”:
        [
                {
                        “id”: “object”,                  //An internal id of the main category
                        “name”: {“type” : “string”},     //Name of main category for this venue
                 },
                 {
                        “id”: “object”,                  //An internal id of the sub category
                        “name”: {“type” : “string”},     //Name of sub category for this venue
                }
        ]
        “hereNow”:
        {
                “count”: {“type”:”number”}               // Contains the number of people
                                                         // currently checked-in at this venue
        }

}
```

Listing 8. Lightweight schema of a venue object, stored in the database.

Category object

```
{
        “id”: “object”,                                  // An internal ID of the venue
        “name”: {“type” : “string”},                     // The name of the venue

        “categories”: {“type”: “array”},                 // Contains an array of
                                                         //all subcategories to this category

        “hereNow”:
        {
                “count”: {“type”:”number”}               // Contains the number of people
                                                         // currently checked-in at this category
        }

}
```

Listing 9. Lightweight schema of a category object, stored in the database.

4.1.2.2 Twitter API

The Twitter API exposes information about tweets from the homonymous social media platform. This monitoring system collects tweets only related to Foursquare. Data, extracted from the Foursquare API, does not give a sense of movement, since the exposed user information is limited. To account for this, this thesis integrates tweets, coming from the Twitter Streaming API, which are linked to a specific Foursquare check-in. The Streaming API enables the possibility to collect movement data from Twitter in real-time. The monitoring process starts by connecting to the Twitter Stream, which pushes tweets from a user-defined location and a bounding box perimeter. The tweets are then filtered by keywords to extract only those, describing check-ins of Foursquare users, which contain an URL with an identification number. The Foursquare API provides an endpoint to resolve check-ins, collected from the Twitter API.

4.1.2.2.1 Limitations and decisions

The main limitation of the Twitter Streaming API is … the Foursquare API. The endpoint to resolve check-ins collected from Twitter provided only authenticated access, allowing for only 500 requests per hour. To overcome this problem, the Foursquare configurations file, as already described, contains multiple "OAuth" tokens, which are dynamically swapped when their limit is exhausted.

4.1.2.2.2 Data model

Check-ins, recovered Foursquare, contain information about the user and the current checked-in venue. A user map is created to extract the movement of the currently checked-in user. This map contains the previously checked-in venue as the value and the user as the key. If a user switches venues, his so-called "movement" is recorded. Only movements between two different venues are considered if there are consecutive check-ins of a user in the span from 5 am to 5 am the next day. Listing 9. describes a lightweight schema of the user movements.

User Movement object

```
{
        "prevVenue": "Venue object",        // The previously checked-in venue
        "nextVenue": "Venue object",        // The current checked-in venue
        "user":
        {
                "id": "object"              // Internal id of the user who has switched venues.
                                            // No name to remain anonymous.
        }

}
```

Listing 10. Lightweight schema of a movement object, stored in the database.

4.1.2.3 Pre-aggregation

The pre-aggregation step, as already discussed, simplifies and speeds up the process of loading large amounts of data interactively from the database. It involves the aggregation of check-in data from Foursquare together with the movement data from Twitter.
There are three different pre-aggregation strategies: hourly, daily and monthly.

Hourly:

Let H be the current to be aggregated hour for a target venue or category. Consider the consecutive time steps $t_n \dots t_m$, containing check-in data from Foursquare where $n \in H, m \in H$. Let $c_n \dots c_m$ be the check-in count for each respective time step. Then the aggregation of hour H for the target venue or category is defined as the sum of the deltas between every two consecutive check-in counts:

$$\sum_{i=n}^{m} c_i - c_{i+1}$$

Aggregating the movement data is even simpler. Let $e_n \dots e_m$ be movements from a source venue to a target venue for the above defined time steps. Then, the movement aggregation is defined as the sum between all movements:

$$\sum_{i=n}^{m} e_i$$

Daily and monthly:

The aggregation process of the movement data is the same as in the above defined hourly aggregation.

The aggregation of the check-in data is slightly different.
Let $H_0 .. H_{23}$ be the already calculated hourly aggregations of the target day for a single venue or category. Then its daily aggregation is defined as:

$$\sum_{i=0}^{23} H_i$$

The same process is used for the monthly aggregation, but with daily aggregations, instead of hourly.

4.1.2.4 Routine calculation

The routine is used for the automatic detection of outliers per point of interest or category. The theory behind this process is defined in chapter 3.1.4.2 “Outlier detection”. This thesis investigates routines for each aggregation type: hourly, daily and monthly. Hourly routines are defined as the tuple (hour, weekday), daily as (weekday) and monthly as (month of the year).
As described in chapter 3.1.4.2, to perform the outlier detection, the system needs to keep track of the mean and standard deviation values for each of the above defined routine tuples, containing data about Foursquare venues and categories. Although there are multiple ways to define routines, e.g. by defining the daily routine as (weekday, month of the year) or the hourly as (hour, weekday, month of the year), as tuples get more and more specific, more and more data samples are needed for the convergence of the mean to a reliable value.
The routine calculation process is implemented in the pre-aggregation step, by reusing the already calculated values for each aggregation type.

4.2 Visual Analytics System

The amount of data, coming from location-based social networks, is huge. Studying this data would be impossible without the assistance of computers. The task of a visual analytics system is to aid the sense-making process, described in chapter 3.1.3, with automatic data analysis and interactive visualizations.
City dynamics is a broad topic, encompassing multiple perspectives of urban life. For example, what is the spatial distribution of people (or check-ins) at different times of the day, week or even month? What is the semantical reasoning behind this distribution? How do people move around the urban fabric? What is the semantical reasoning behind this movement? What are the typical everyday routines of people and is there an interesting outlying behavior? Answering these questions requires the cooperation of experts with domain knowledge and a system, supporting the analysis process. This thesis tries to answer these questions by proposing a visual analytics system, tightly cooperating with the monitoring system to reveal spatial and temporal patterns from real-time and historical data. The monitored city is the capital of Japan, Tokyo, due to its population and size, making it one of the largest metropolitan areas in the world.

4.2.1 Concepts and architecture

This chapter discusses the practical concepts and architecture behind the implementation of the visual analytics system.

4.2.1.1 The visual analytics process

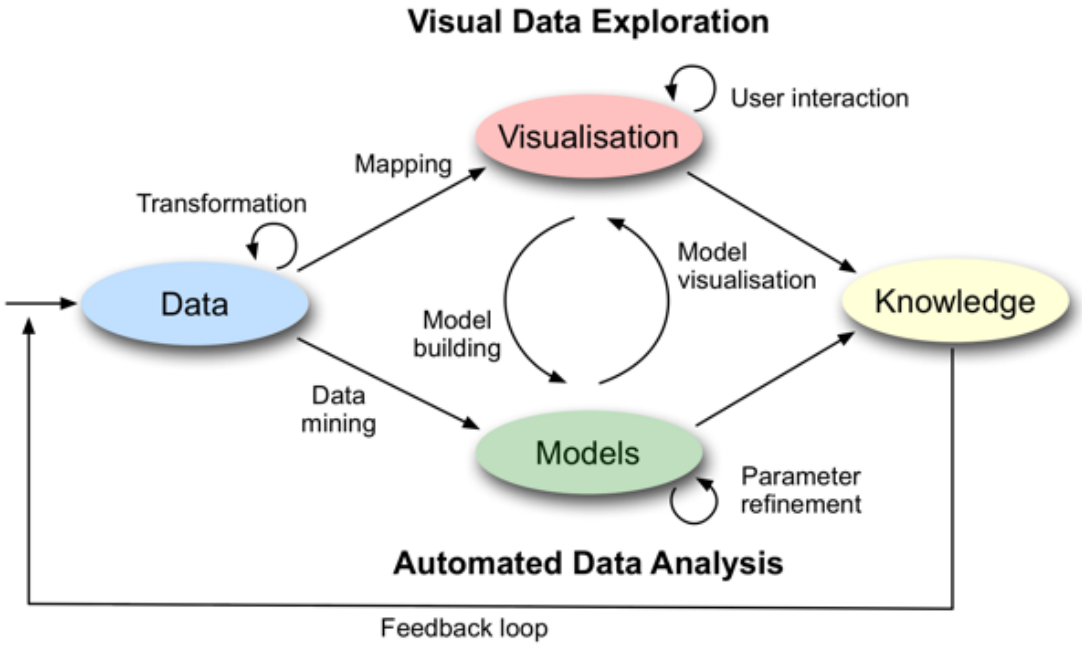

Figure 26. Visual Analytics Process. [9]

The sense-making process cooperates tightly with visual analytics to explore and analyze large data sets to confirm the expected and discover the unexpected. Figure 26. shows the generic knowledge discovery process using visual analytics.
The input data is subject to different transformations (e.g. data integration, cleaning or aggregation) to match the needs of the system. The user of the system can choose between applying data mining methods (see chapter 3.1.4) for automated data analysis or visual mappings for exploratory analysis. Automated analysis forms models from the input data, which are then mapped to visualizations for further evaluation and refinement. The user can interact with the visualizations and the data models (by refining the parameters of the data mining methods) to gain insights and knowledge of the original data and prove or disprove hypotheses. This process can repeat indefinitely, until the user has discovered the knowledge needed to solve the analysis task.

4.2.1.2 Overview and Workflow

The workflow of this thesis's visual analytics system follows closely the visual analytics process, described in the previous chapter. Before presenting its specifics, consider an overview of the system's capabilities (see Figure 27). The visual system consists of 7 main sections:

a) Stream- and alluvialgraph: The stream graph visualizes the temporal domain of the

data, split into multiple categories:

- Travel & Transport
- Shop & Service
- Residence
- Professional & Other Places (Other Places contains tourist sites, government buildings, shrines and more)
- Outdoors & recreation
- Nightlife spot
- Food
- Event
- College & University
- Arts & Entertainment

These are the main categories, provided from the social media platform “Foursquare”. The alluvial graph is another representation of the extracted temporal data, which reveals a different important city aspect: Mobility between categorical entities.

Figure 27. An overview of the visual analytics system's prototype, presented in this thesis.

b) Map – An interactive map, provided by “Leaflet” and “Open Street Maps”. The map can be zoomed in/out and panned for exploration. The spatial visualization of data objects and models is done via the usage of different methodologies and visualization layers. They comprise a Mobility Graph, which groups points of interest together as nodes and visualizes aggregated movements between them as edges to this graph. and a heat map algorithm and visualization, used to represent the distribution of check-ins.

c) Map and tool options – This section contains a venue description panel and a set of interactive parameters for the data mining methods. The description panel contains a sorted by check-in count list of venues contained in a chosen node (or cluster) of the mobility graph or heat map representation. The parameters of the used algorithms are dynamically adjustable, which dynamically updates the impacted models and visualizations on each user interaction.

d) Mode options – Switches between two different working modes of the system. Real-time and history exploration.

e) History slider options – Contains several options, allowing the exploration of the time-series data, visualized in a). The options comprise:

- Loading data, via its aggregation, from a user-defined time window, displayed in a).
- Expanding the currently chosen time window, by traversing deeper into the temporal hierarchy: Monthly->Daily->Hourly->Hourly section->and so on … until only two elements are left.
- Traversing in the opposing direction of the temporal hierarchy.
- History of (slider) interactions with the capabilities to go back to a certain point of time.
- An option to switch between the stream- and alluvial graph at any given time.

f) Interactive categorical legend – Clicking on each of the categories, filters out all currently loaded data and models, not assigned to the chosen categories, and updates all enabled models and visualizations (Stream- and alluvial graph, mobility graph and heat map). Multiple categories can be stacked at the same time, so that the user can analyze only relations between them.

After a rough presentation of the system's capabilities, its workflow can be further refined (see Figure 28). The data mining models of this thesis's prototype are the following:

- Density models : Models, created by the density-based clustering algorithm DBScan. They can be refined by adjusting the parameters Eps (or radius) and MinPts (see chapter 3.1.4.1)
- Gaussian group models: Models, created by a 2D- Gaussian distribution around each point of interest and assigning them to groups, based on that value. They can be refined by adjusting the size of the 2D kernel and sigma for the Gaussian function. (see chapter 4.2.2.1)
- Statistical models: Models, created by a statistical method to detect outliers (see chapter

3.1.4.2). They can be refined by adjusting the multiplier of the standard deviation value.

The system's visualization capabilities comprise:

- Heat map: Uses the Gaussian group models to assign a linearly interpolated color scheme to each group.
- Mobility Graph: Uses the density models to visualize them as nodes to a graph.
- Outlier visualization: Uses the statistical models to highlight the found outliers. The current visualization is limited to a red border line around the object, although glyphs or extensions would suit better.
- Stream- and Alluvialgraph: Visualizes the temporal dimension of the data, while also using outlier visualizations to highlight anomaly behavior.

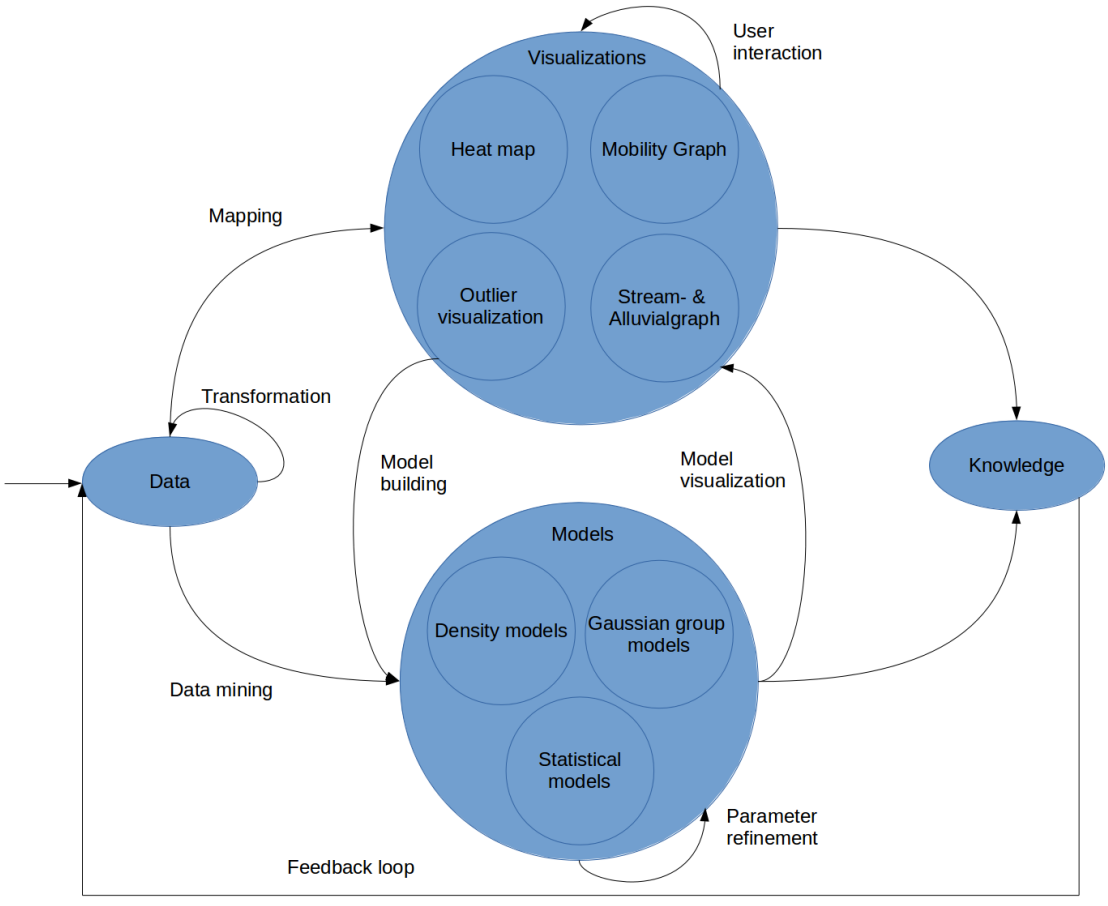

Figure 28. Refined Visual Analytics process of the system's prototype.

4.2.1.3 Architecture

A high-level architecture of the system's prototype is shown in Figure 29. At the bottom is the back-end part, which communicates with the MongoDB database through a database wrapper by polling data regularly to collect the most recent updates for real-time analysis or to request history data to be aggregated and returned to the front-end for model building and visualization. The whole process is controlled by a single controller, located in the front-end. It communicates with each method controller and regulates the incoming data. The purpose of each method controller is to build models and map them to appropriate views. The user, shown on top, can interact with the system through parameter refinements and visual interactions to discover knowledge, as depicted in Figure 26. The event-based interactions are then appropriately managed by the controllers to control the process of model building and visualization.

4.2.2 Controllers and views

This chapter presents and discusses the implemented methods and their respective visualizations.

4.2.2.1 Heat map

The heat map is used as a graphical representation of the spatial distribution of the points of interest (venues). Each venue is assigned a value and a group according to this value. Each group is presented with a color, linearly interpolated to a sequential color scheme, pre-attentively displaying the differences between them. The more “hotter” the region, the bigger the check-ins count. There are multiple algorithms for calculating the heat map of a location-based data set. The idea is to first project their coordinates (latitude and longitude) on the map. A Mercator projection could be used for smaller regions (like cities). Then use the grid of the canvas as distance between the venues. This distance function allows for the formation of different “heat” patterns on different zoom levels of the map.

Algorithms

One of the most popular algorithms is the so-called “splatting” algorithm, which uses a normalized Gaussian kernel (see chapter 3.1.4.2) over all projected points of interest. An example of the algorithm is shown on Figure 30. Initiate all pixels or grid points with a value, which equals 0. For a given kernel size and standard deviation, “splat” the kernel value around the given point onto its corresponding pixel, by adding the sum to its grid value. In the end, each pixel will posses a certain value. To assign colors to values, linearly interpolate the minimum and maximum values over all canvas pixels to a given color scheme. This creates the feeling for a “continuous heat image” of the point's distribution.

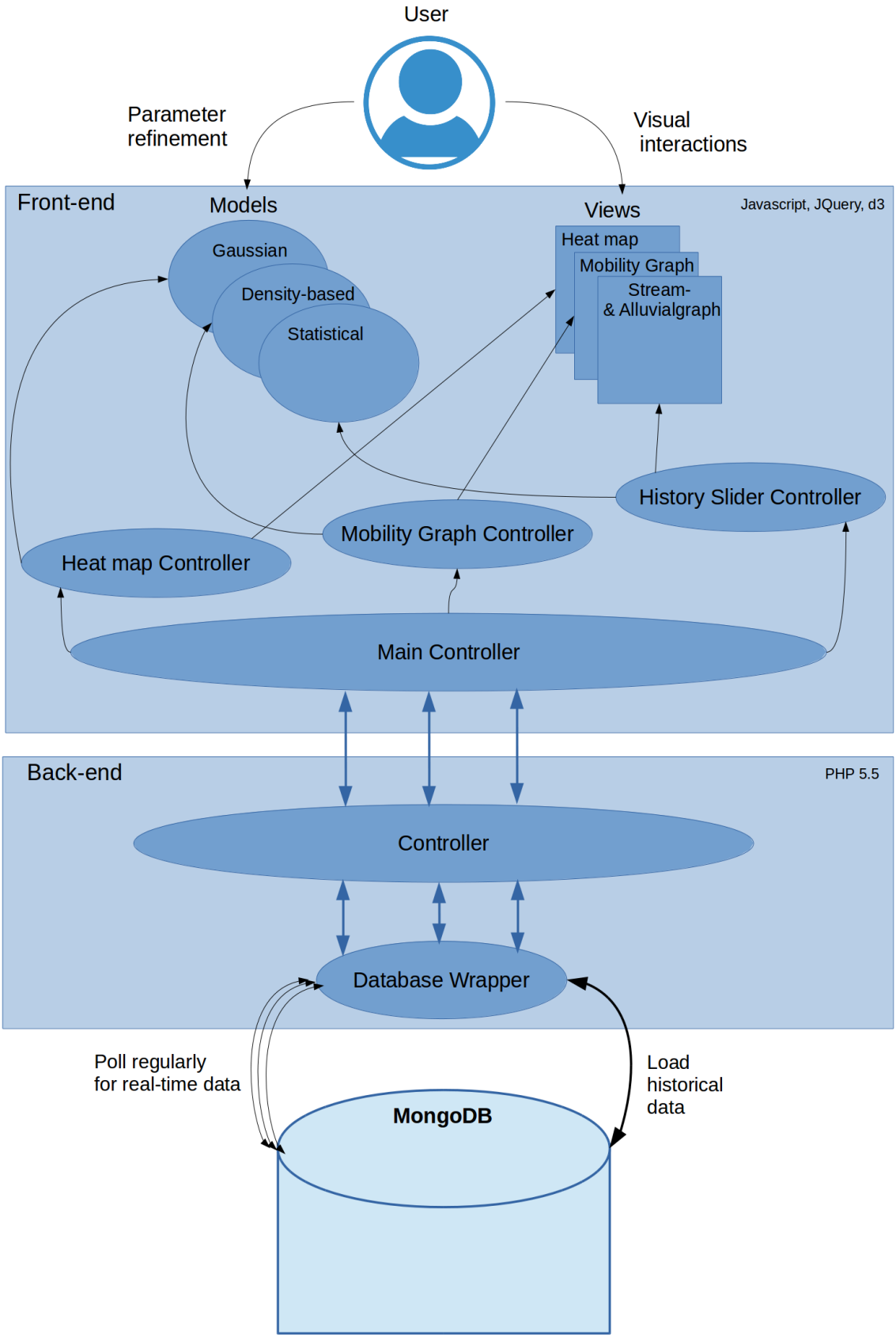

Figure 29. High-level architecture of the Visual Analytics system.

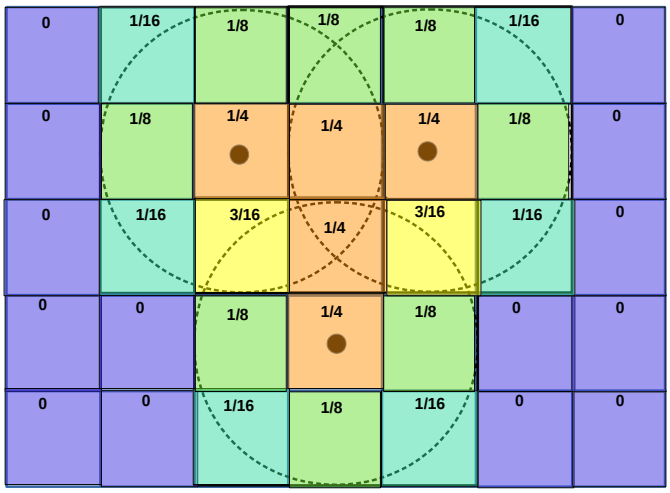

Figure 30. “Splatting algorithm” for the creation of heat maps.

The simplicity of this algorithm makes it very attractive and popular. But there is a problem with this algorithm for the current data set and analysis task. The created image doesn't give sufficient information about the check-in distribution of each point of interest or venue, because the values around the perimeter of a point (kernel size) are assigned to a different color. To account for this, this thesis presents a different algorithm for the “heat” value calculation. The main idea is to assign a certain value and color to a venue and its belonging perimeter. The heat map objects could be then represented as color-mapped finite elements over the spatial domain. The algorithm works as follows:

Assumptions:

1) A normalized 2D Gaussian kernel with user-defined value for sigma (standard deviation) and kernel size.

2) An array of venue objects (see 4.1.2.1), containing check-in counts.

3) Each venue object is projected on the map's canvas using its coordinates (latitude and longitude)

Algorithm:

Figure 31 illustrates the algorithm. Point A is the target venue to be calculated. The algorithm starts by assigning the assumed kernel on top of the target point. It then starts iterating through the kernel, while testing each of the kernel's grids for venue containment. If a venue is contained in a grid point inside the kernel, the current kernel's Gaussian value is multiplied with the check-in count of the found venue and added to the final “heat” value of the target venue. During the kernel iteration phase, if the current iterated grid is part of the kernel's border, a scan process begins. Figure 32 depicts this process. The red area is the kernel, the blue areas are the pixels to be scanned from each of the border cases and the green areas represent pixels scanned from border corner cases. The arrows show the scan directions. The idea is to find whether the target venue collides with venues outside its

kernel. If there is a collision, the colliding venue's check-in count is again multiplied by the Gaussian value of the border grid, from where the scanning process had begun. This value is then added to the target venue's final “heat” value. The final value is then assigned a linearly interpolated color. Although this algorithm is not optimal, due to points further away from the center, but contributing more to the result (see points B and D), it still produces fine results for the current data set.
The presented algorithm assigns each venue to a certain Gaussian group, represented on the map with sequential colors. This lets the user explore the check-in distribution of each venue or clusters of venues. The algorithm is calculated for each chosen zoom level of the interactive map. The results can be seen in Figure 33. Picture a) illustrates a heat map example, constructed from all venues with a check-in count of 1, without recalculating it at each zoom level to increase the resolution of found clusters. Picture b) shows a zoomed-out heat map visualization, computed with the above explained algorithm with a kernel size of 5 and $\sigma > 2$. Picture c) uses the same parameters to display a zoomed-in example of the algorithm. It can be observed that at higher zoom levels, each individual venue receives a certain value and a respective to this value color, depending on its check-in count. The overlapping of venues adds up to each of the point's values and creates groups (clusters), spotted in picture a) (see Figure 33).

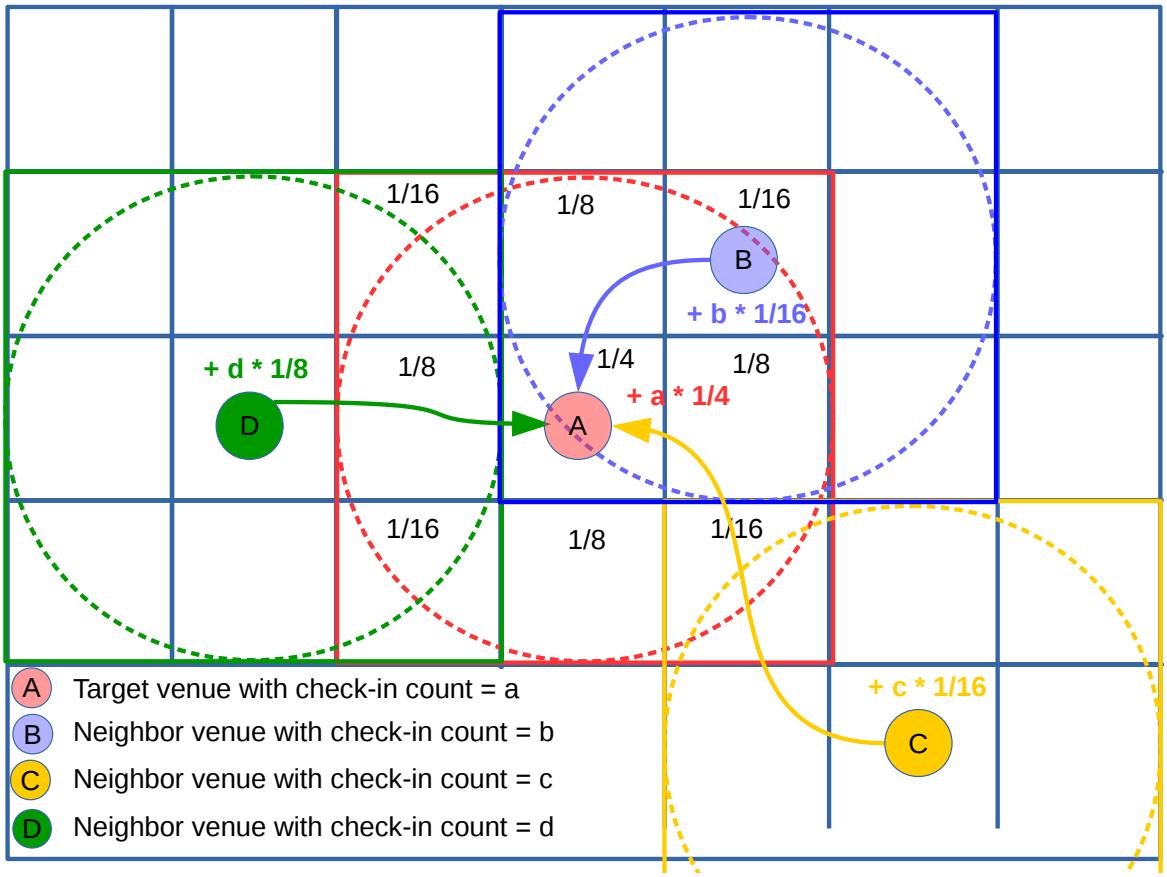

Figure 31. “Heat” value calculation for a target venue.

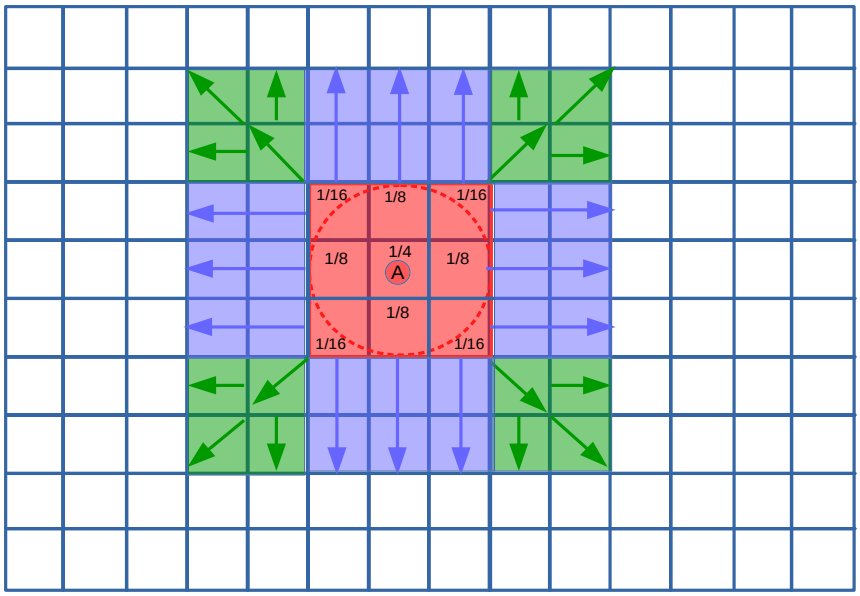

Figure 32. Scanning process of an area around the kernel to find neighboring points.

4.2.2.2 Mobility Graph

The spatial distribution of check-ins reveals the busy areas of a city in real-time or throughout different temporal aggregations. Although this information is useful, it uncovers only a single aspect of the urban life's complexity. To study the dynamics of a city, an analyst would need multiple and different set of tools. The mobility graph tells a distinctive story. It shows the movement of people between venues and clusters of venues. Dense regions of venues often reveal patterns about the popular areas of the city. These clusters of people (or venue check-ins) contain semantic information about the reasoning behind them, which is the key towards studying the behavior and habits of crowds throughout the spatial and temporal fabric of the city. The visualization of movement data over a long period of time could result in visual clutters, which could hinder the cognitive understanding of the data. The implemented mobility graph tries to solve this problem and contributes to the analysis of spatial patterns and mobility patterns. Assume D is the current set of points of interest (or venues). The mobility graph is defined as a directed graph G = (V, E) with $V \in D$ as the set of nodes and E as the set of edges, connecting these nodes. The system traverses the following steps to create and visualize the set of nodes V.

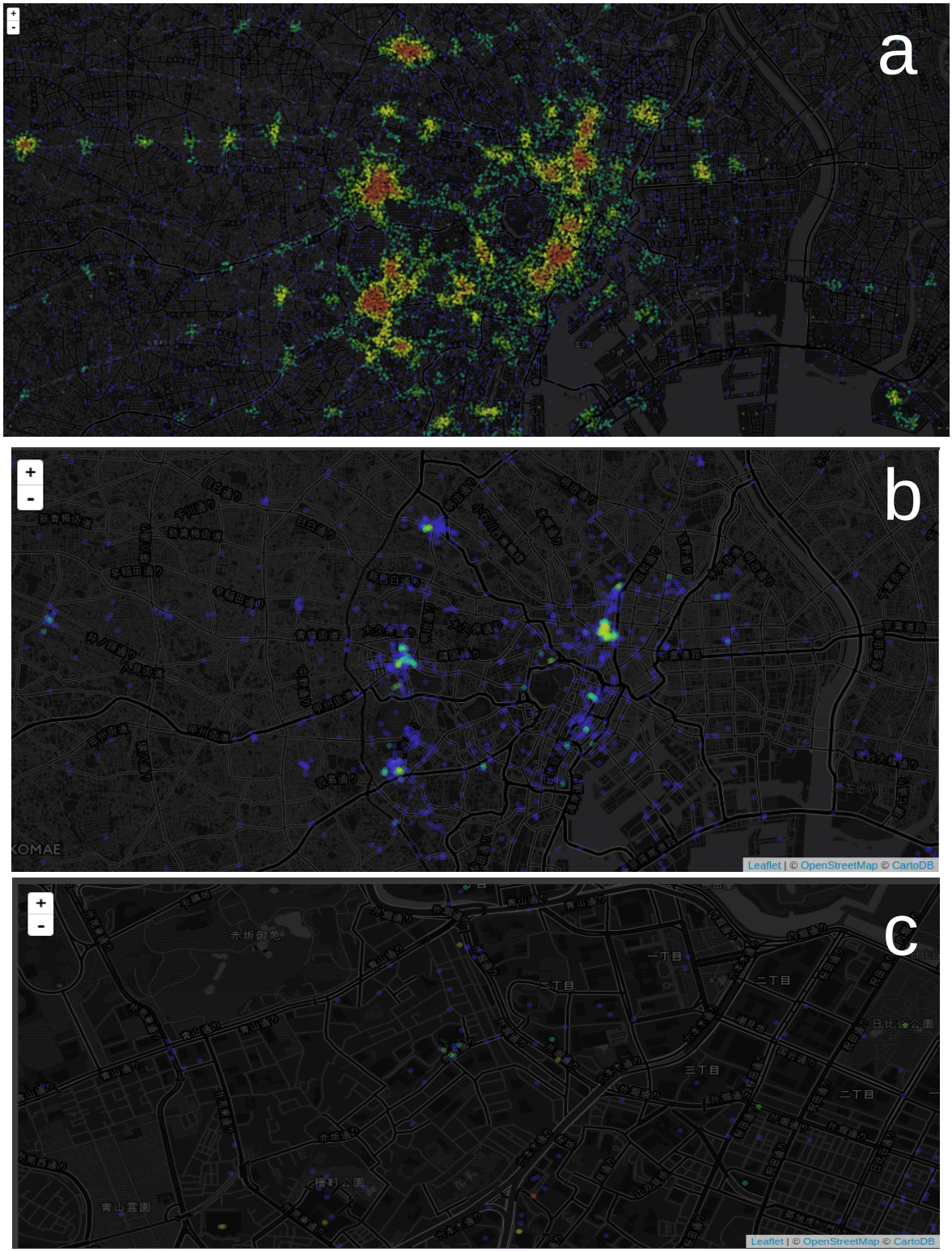

Figure 33. Two heat map examples. a) Formation of “heat” clusters for all venues without recalculating at each zoom level. b) Formation of “heat” clusters with check-in size data. b) Bottom picture shows the heat representation of individual venues.

1) Use a density-based clustering algorithm (see chapter 3.1.4.1) with user-defined parameters to group together venues close to each other, based on a grid-based distance function. Such function can group together projected venues into bigger clusters on zoomed-out map exploration and smaller clusters on higher zoom levels with the same user-defined parameters. Varying these parameters, uncovers areas of different densities. Density-based clustering allows to learn clusters by introducing the notion of noise. Unlike centroid-based clustering algorithms, which group together all points into a defined number of clusters, density clusters can highlight areas of interest with varying densities, while remove data anomalies. The process of finding density-based clusters with DBScan requires the retrieval of neighbors around the grid-defined perimeter of each point. The efficiency of this task is essential for the interactive process. A Quadtree has been implemented to speed up this process (see chapter 3.2.1.2).

2) The next task would be to visualize the clusters. For this purpose:
    a) First, calculate the concave hull of the set of clustered points (3.2.3)
    b) Then, visualize this hull with a randomly chosen diverse color.
    c) Define the visual centroid (3.2.1.3) of the concave hull as its center.
3) Create and visualize a tag cloud (see chapter 4.2.2.3) for this cluster, using its visual centroid from 2.c).
Now that the nodes of the graph are visualized, they need to be connected with edges, representing movement between them. The next few steps describe this process:

1) Calculate the aggregation of movements between all clusters (or venues inside those clusters).
2) Interpolate this aggregation size to a certain edge width.
3) Visualize the set of edges between clusters as animated arrows with the above defined width, using the visual centroids of the clusters as start and end points.
4) Overall, curves are perceived as better than straight lines, due to their smoothness. The edges are visualized as curves using Bezier curves (see chapter 3.2.2)

Figure 34. shows the end result of these steps. Although, the animation cannot be shown, it is an essential part of the pre-attentive edge visualization. The user-defined parameters (see chapter 3.1.4.1) can change the density of the clusters. This example contains more low-density clusters. Picture a) displays a zoomed-out overview of the city. It contains one big cluster of venues in the center and other smaller clusters around its vicinity. This allows the user to explore the movement of people between the city center and its rural areas. Zooming in further on the blue cluster (see picture b and c), reveals new patterns. That way, a user can focus on a single area and explore the movement and activities inside it, while the system creates new clusters at each zoom level. Figure 35. shows how refining the clustering parameters affects their size and density. The parameters in picture a) are fairly chosen for both the neighborhood radius and the minimum number of points needed. As the size of the cluster area gets lower and the minimum number of points higher, so does the density of the clusters. Pictures b and c illustrate the areas of very high densities. Choosing the right value for both of the parameters requires domain knowledge about the problem.

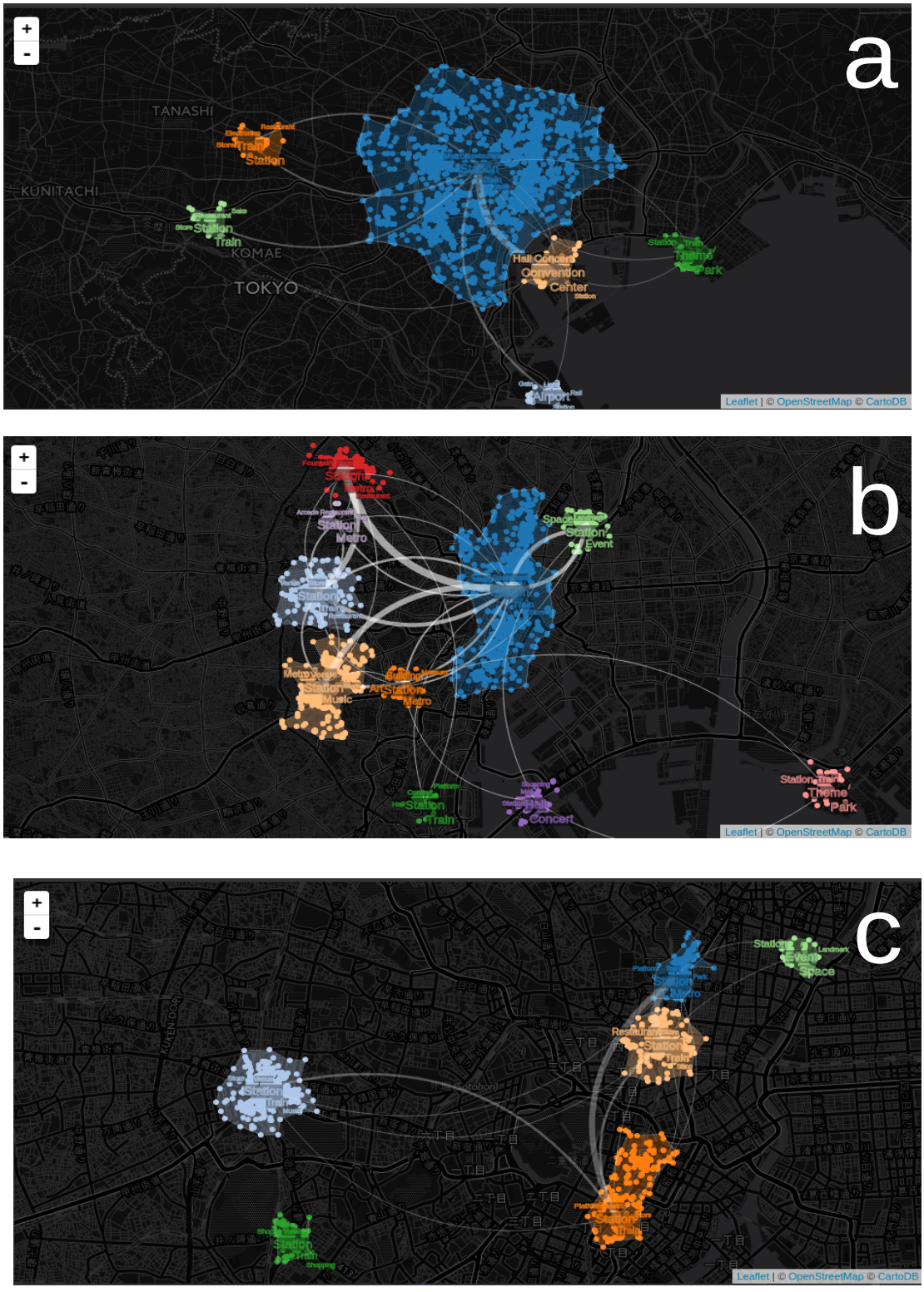

Figure 34. Mobility graph clusters of different zoom levels. a) Overview of the city. b) Zooming in on the blue cluster. c) Zooming in further on the blue cluster.

Figure 35. Mobility Graph clusters of different densities. a) low → c) high

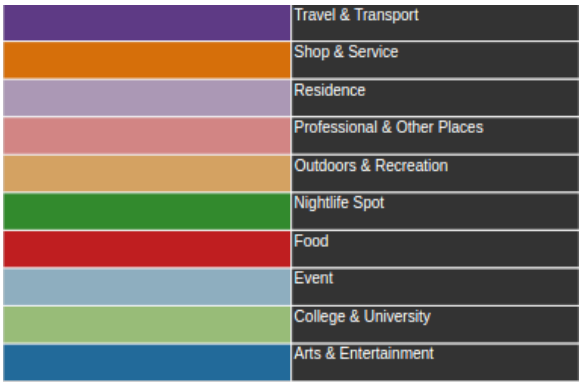

Figure 36. Category legend for convenience

The system allows the user to interactively change them through the usage of horizontal scroll bars, which display the changes on the visualizations instantly at each scroll unit. This way, the system can “animate” how clusters evolve over different parameter values, which gives the user an easier decision on the cluster density level for his task.

Another important task is the analysis of each individual cluster and the comparisons between them. The prototype of this system offers these capabilities to its user. Figure 37 shows an example. Picture a) contains the details of three different clusters. The center(blue) cluster is the “downtown” cluster, the right cluster is Tokyo's “Disney World” and the left cluster is a cluster containing many restaurant (“Food”) venues and a university campus. Clicking on each of the clusters, reveals a radial stream graph of this cluster's category activities over the chosen aggregated temporal domain. This example contains an hourly aggregation for a randomly chosen weekday. The details of the individual clusters can be moved across the map for convenience. First, consider the color mapping of the categories in Figure 36. Picture a) shows the “Disney World” cluster as very active in the “Arts & Entertainment” category (marked as light blue in the legend). Its peak hours are the morning hours when most of the people arrive to visit the area. In comparison the “downtown” cluster contains check-ins from multiple categories, as it can be expected. Two check-in peak hours are visible for the “Travel & Transport” category (marked as purple), in the mornings and early night time, when the rush hours are happening and people travel to work and home after work. The third cluster is most active in the “Food” category (marked as red) in the evening, when people have dinner and mornings until noon in the “College & University” category when students visit the university campus. Picture b), on the other hand, shows the number of incoming and outgoing people (edges) to and from the respective clusters over the aggregated domain. The inbound traffic (marked with purple) and the outbound traffic (marked with light blue) can reveal the “busyness” of a cluster in respect to movement traffic. For example, the “downtown” cluster clearly shows an increased size of incoming traffic in the mornings, when people go to work and outgoing traffic in the evening, when people leave the downtown area to return home.

One last important aspect of each cluster is their semantics. The main categories are sometimes not enough to tell the whole story about cluster's activities. For example, “Arts &

Entertainment” is quite generic for “Disney World” theme park. Deriving the semantical specifics of each cluster would help determining a better solution to the analysis problem. The next chapter reveals the design and implementation of “Tag clouds” for this purpose.

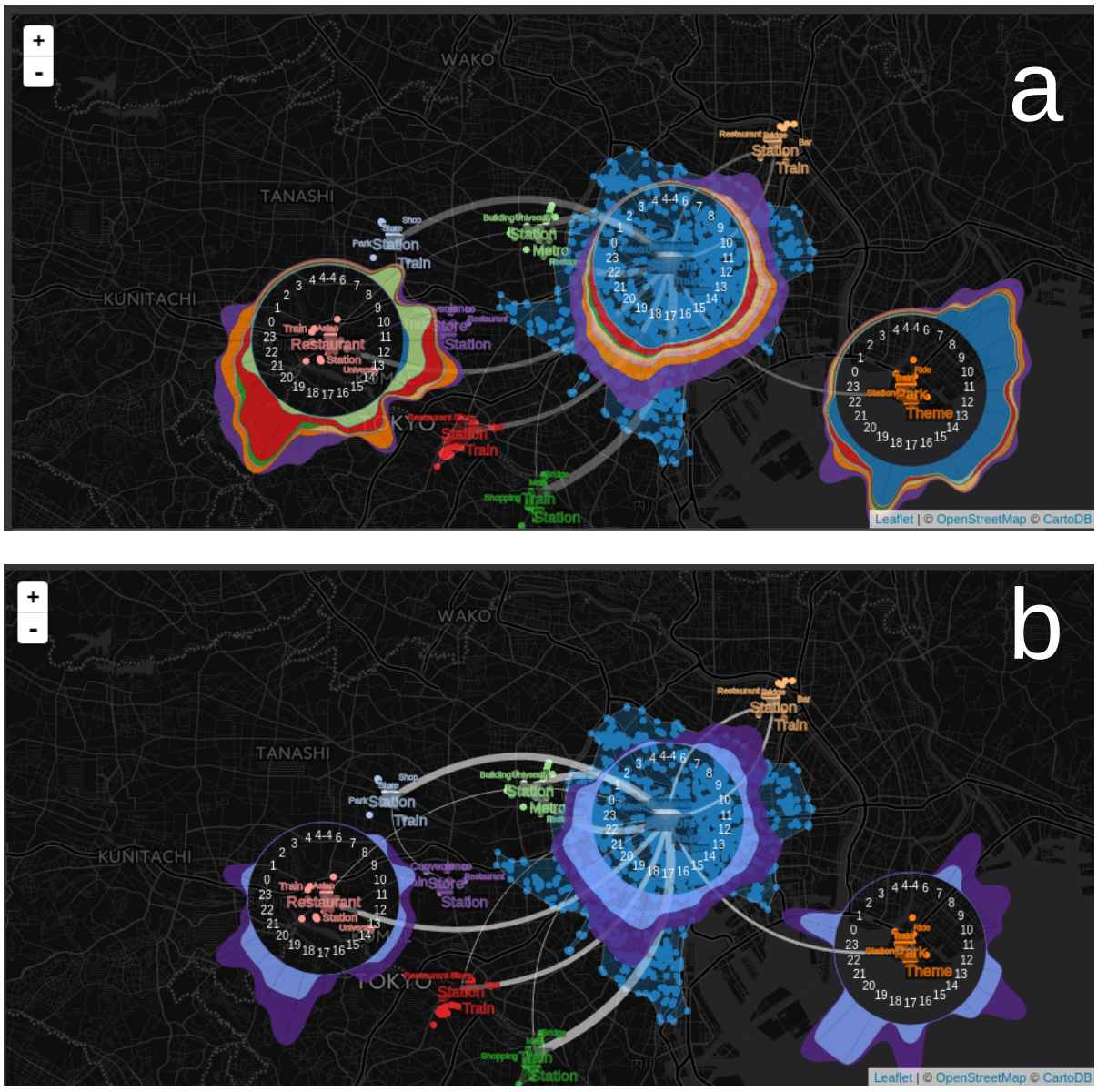

Figure 37. Radial visualization of the temporal evolution of each cluster. a) Categorical b) Movement (Incoming traffic as purple, Outgoing traffic as light blue)

4.2.2.3 Tag Clouds

Tag clouds are visual representations of textual data. This thesis uses tag clouds to uncover

the semantics of single venues or cluster of venues. The main idea is to use the subcategories of each venue, provided by the Foursquare API, to uncover its check-in activities. For example, a venue could belong to category “Food” with different subcategories in the morning and evening. Morning's activity of the venue could be a “Cafe” and evening's could be a “Restaurant”. Another example would be a venue, belonging to “Professional & Other Places”. The venue could be a government building, office, convention center, tourist attraction or even a shrine. The diversity of the subcategories gives the user a deeper meaning of the data set, which simplifies the analysis task.
The algorithm to create a tag cloud is inspired by Jonathan Feinberg's Wordle[21]. The following steps are involved:

Assume the algorithm needs to create a tag cloud for a cluster of venues.

Processing phase:

1) First, start off by creating a map for each cluster, containing words as keys and check-in count as values.
2) Then, split each subcategory of each venue from the cluster into a set of words. For example, the subcategory “Asian restaurant” would split into two words: “Asian” and “Restaurant”. This ensures that if e.g. a cluster contains many restaurants, but not all of them are asian, the word “Restaurant” would be high valued.
3) Add the venue's check-in count to the created word map with its respective subcategory's words as keys.
4) Repeat steps 1) and 2) until all venues from the cluster are visited and their respective subcategories are aggregated in the cluster map.
5) Transform the map into a sorted by value array (descending) and pick the top 5 words.
6) Map the aggregated word values to font sizes (to be used as text within their respective bounding boxes) using interpolations.

After the processing phase has finished, the text's bounding boxes need to be arranged around the cluster's center:

Placement phase:

Consider Figure 38, showing the box's placement process:

1) Start off, by placing the first text bounding box (which has the biggest size according to the previously sorted array) on top of the visual centroid (see chapter 3.2.1.3 on how it is calculated).
2) Add the box to a Quadtree data structure, as shown in chapter 3.2.1.1. All text boxes from all clusters must be added to the same Quadtree structure, in order to avoid overlaps of texts from two closely-located clusters.
3) Start moving in a spiral, by testing each spot for placement compatibility for the next bounding box from the array. A spot is compatible, if the current bounding box to be placed (target rectangle) does not collide with any other text box (already added to the Quadtree

structure) from all clusters combined. The collision detection is explained in chapter 3.2.1.1. This way, overlaps between any two word's on the canvas will be avoided.

4) Repeat steps 2) and 3) until all word's bounding boxes are placed and visualized on the canvas.

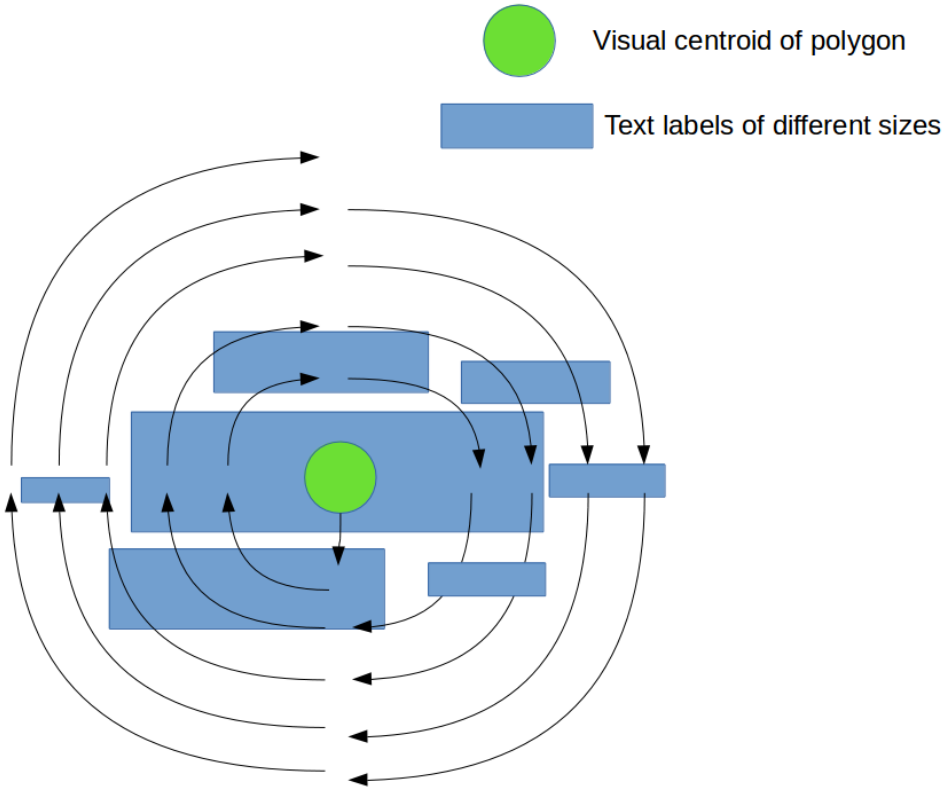

Figure 38. Spiral movement and positioning of text around a point.

Figure 39. shows four cluster examples with their respective tag clouds. Picture a) is Tokyo's "Disney World" cluster. As expected, it contains the words "Park" and "Theme" with the highest check-in counts. The other words are "Ride", "Train" and "Station", meaning the cluster also contains a nearby train station, probably very active when most of the people come and go from the theme park. Picture b) illustrates a well-defined university cluster, containing the words "University", "College", "Building", "Academic" and "Cafeteria". The third picture shows a cluster with an on-going concert at the time. It contains the words "Music", "Venue", "Concert", "Hall" and "Movie", meaning it probably contains a popular

movie theater as well, due to it making it into the cluster's top 5 subcategories. Picture d) is the “Tokyo station” cluster, containing these expected words: “Station”, “Train”, “Platform”, “Metro”, “Bus”. These examples show how the semantics of a cluster can be further inspected to achieve a better understanding of the reasoning behind the check-ins activities.

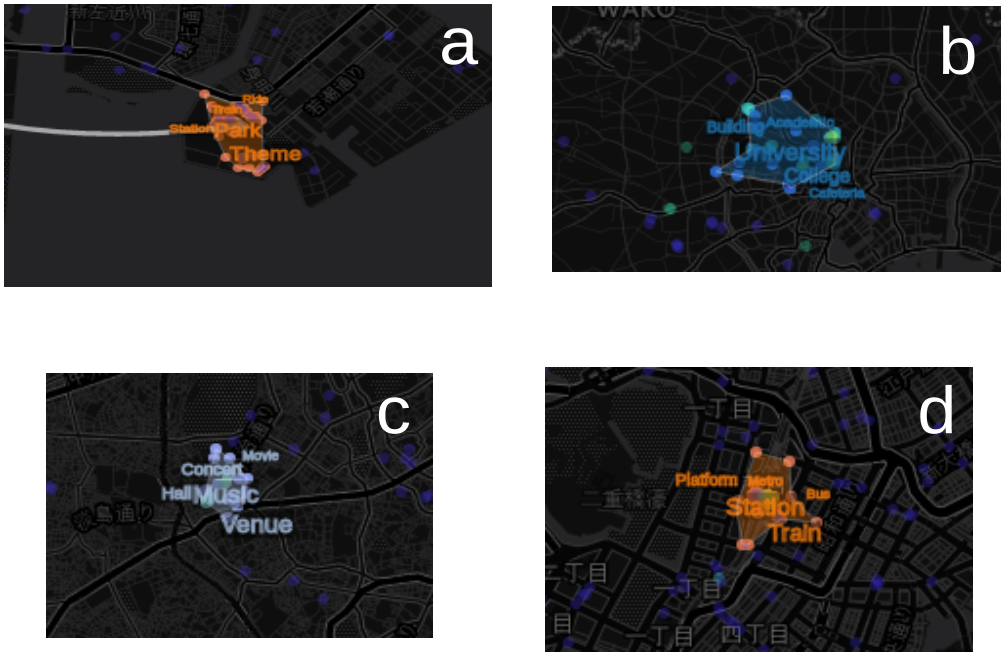

Figure 39. Tag cloud visualizations of different semantically well-defined clusters. a) Theme Park. b) University. c) Music/Concert Hall. d) Train/Metro Station.

4.2.2.4 Stream- & Alluvialgraph

The Stream- and Alluvialgraph are both graphs, used for time-series data visualization, but each one serving a different purpose. They consist of nodes for each category and time unit and edges connecting those nodes. The edges are created using Bezier curves (see chapter 3.2.2) to create smooth transitions. The edges of the Streamgraph connect only nodes from the same category between sequential time units. This creates the feel of a “river” flowing through the time fabric and visualizes how the evolve over time.
Consider Figure 40. Picture a) depicts the daily temporal check-in distribution of categories between the end of February 2017 and mid-April 2017. The weekly pulse of the city of Tokyo can be easily recognized, with weekends having more check-ins than weekdays. The capabilities of the graph comprise loading aggregated data from the database and

expanding or zooming-in on a selected region of the graph. The selection is done through a slidable time window with a user-defined width. The user can interact with the window by dragging it from either side, while it expands or shrinks. Zooming-in or out of (if possible) a selected graph area switches the time unit aggregation type (e.g. from days to hours). This assures the scalability of the graph , while also allowing the user to explore a specific type and area of the temporal domain. Picture b) shows a zoomed-in example of the selected region from a). It contains the hourly distribution of the selected week, which reveals how each category changes over the course of a day. Zooming in further (see picture c), allows the user to freely explore the selected distribution, by covering only small parts of the temporal domain.

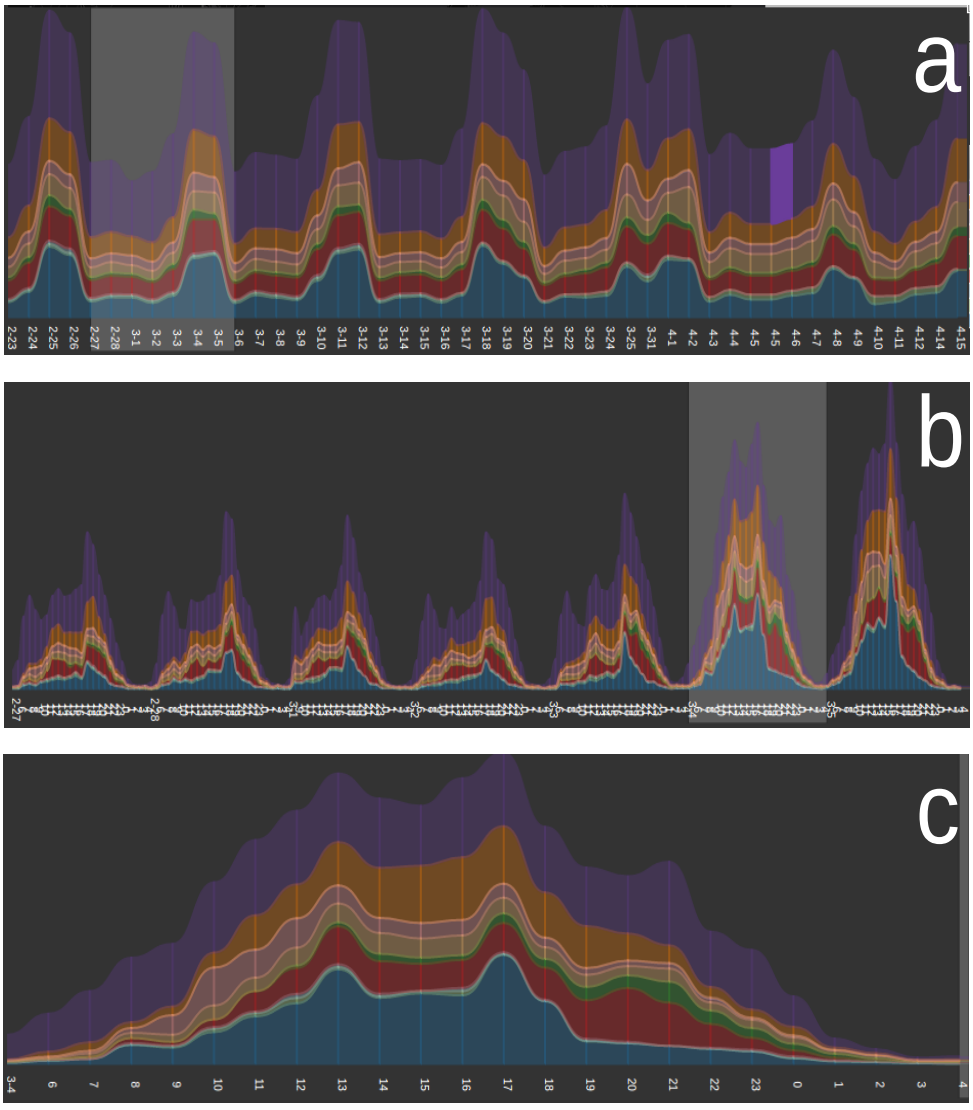

Figure 40. Streamgraph of different temporal aggregations. a) Daily data over the course of several days. b) Hourly data over the course of a week c) Hourly data over the course of a day.

The Alluvialgraph is concerned with another aspect, the mobility of users between categories. The displayed data is the extracted movement data from Twitter (see chapter 4.1.2.2). An edge from the Alluvialgraph represents movement between a source and a target category. The width of the edge represents the size of the movement aggregation. It is worth noting, that the size of the edge can change between time units before reaching its target node. This is due to the fact, that there is a possibility of larger size of people moving between venues during the transition phase between the previous and the current time unit. To account for this, the visualization of the edge is the result of interpolation between the source and the target node. Figure 41., picture a) shows the Alluvialgraph of day's hourly distribution. There are different strategies for the placement of the category nodes, some of them involving the “relaxation” of the connecting edges. The proposed solution in this thesis keeps each node at its respectful position throughout all time units. This is due to the fact that the user gets accustomed to the position of each category by the regularly use of the Streamgraph and by looking at the category legend, shown in Figure 36. Changing the positions of the nodes at each time unit would result in chaos, which would disturb the pre-attentive perception of the visualization. Picture b) of Figure 41. shows how the Alluvialgraph can be better explored. Zooming-in on an area (the same way as in the Streamgraph) allows the user to easily spot the movement patterns of the data. The graph is interactive and allows the user to hover over a node or an edge to highlight it, while also displaying the aggregated check-in count. Clicking on a node, highlights the selected node together with its outgoing and incoming edges. This allows the user to select multiple nodes, compare them or see how they form over time. The category legend, shown in Figure 42, also serves another purpose. After clicking on a chosen category, the system filters out all venues and categories, not belonging to it. The views/visualizations are updated respectively to reflect these changes. Figure 43 shows an example of these chosen categories: “Travel & Transport” , “Professional & Other Places” and “Nightlife Spot”. After the selection is made, all models and visualizations should contain only data from these 3 categories. This allows the user to better compare them and investigate relationships amongst them. Figure 44. illustrates an example of how the Stream- and Alluvialgraph look like after the filtering process. The time domain is an hourly aggregation of a typical workday (Monday). Picture a) shows how the check-in counts for the category “Professional & Other Places” (pink) rise in the morning and fall at night. On the other hand, the night life category (green) has almost no check-ins in the morning, but its numbers are high during the night, when people leave work. Although this information is useful, it does not reveal any information about the assumed transitions between these categories. The Alluvialgraph (picture b) uncovers these characteristics. For example, it can be seen that between 9 am and 13 pm, a lot of people transition from category “Travel & Transport” to category “Professional & Other Places”, which is when most of the people commute to work. As expected, there is no movement between “Nightlife Spot” and any of the other 2 categories in this time frame. In comparison, a lot of people transition from category “Travel & Transport” to category “Nightlife Spot” between 20 pm and 22 pm and vice versa between 23 pm and 0 am, when people return home with public transport. Filtering unneeded categories, allows the user of the system to focus on the analysis problem, by removing redundant data. The combination of categories for analysis requires domain knowledge about the current task at hand. The above described scenario

demonstrates the usage of the graphs, by confirming an expected workday hypothesis.

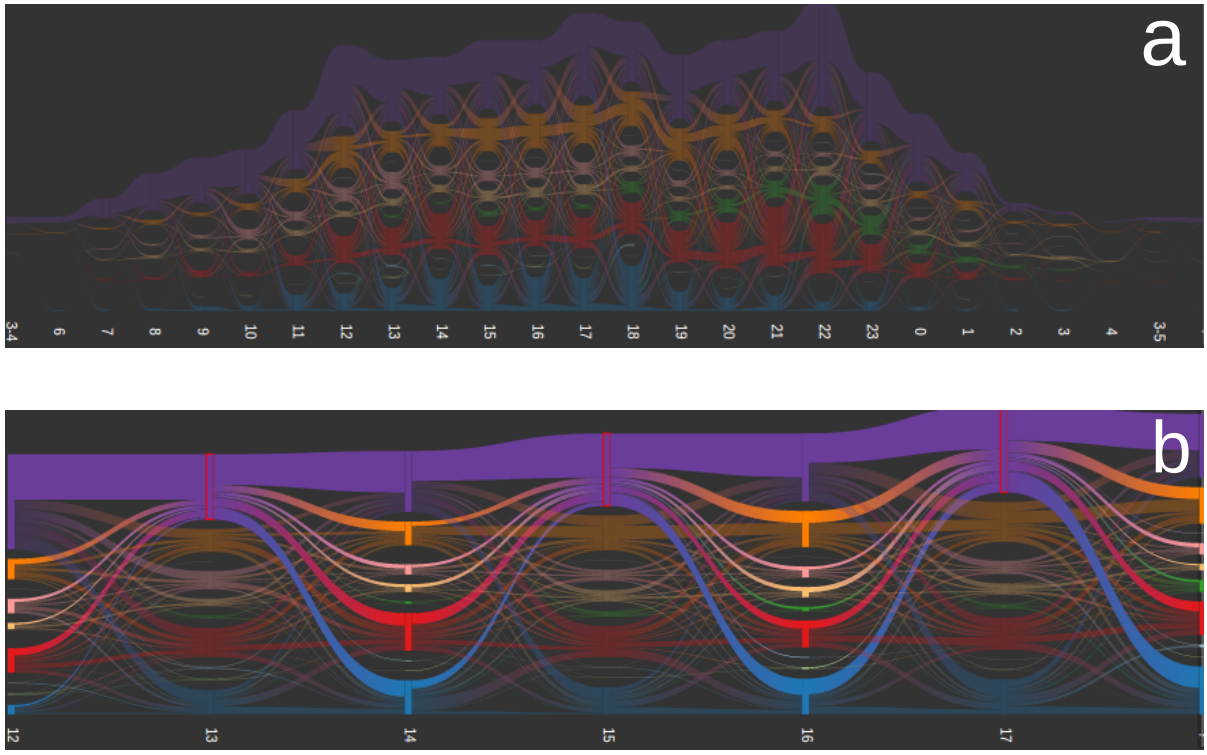

Figure 41. Alluvial graph example. a) Shows the transitions of categories in the time span of a day b) Shows how a single category can be highlighted and explored

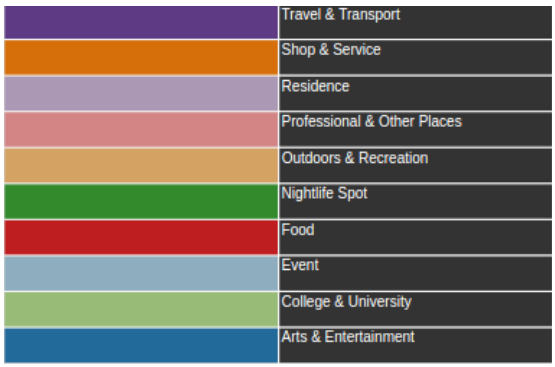

Figure 42. Category legend.

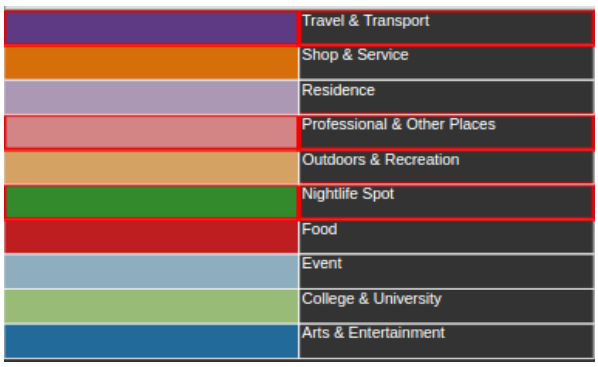

Figure 43. Filtered categories.

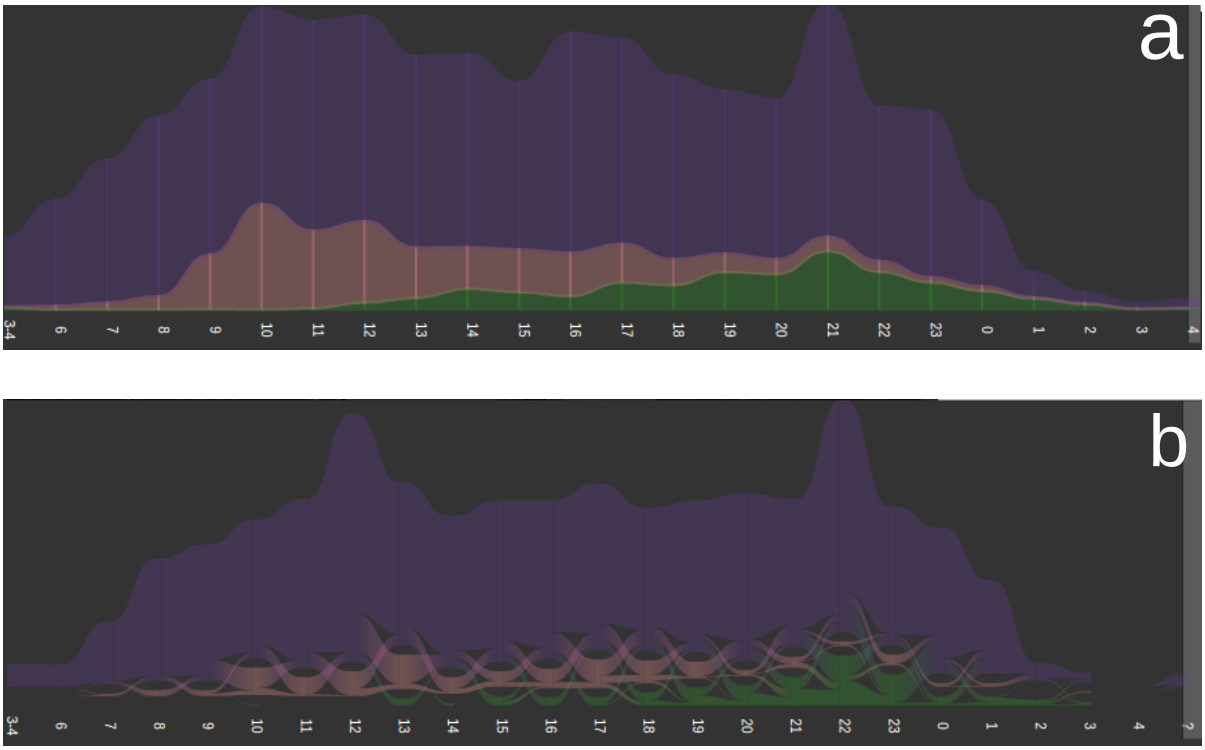

Figure 44. Graphs containing only the categories “Travel & Transport”, “Professional & Other Places” and “Nightlife Spot”. a) Streamgraph, visualizing their distribution over time (for a normal workday). b) Alluvialgraph, visualizing the relations between the selected categories.

4.2.3 Routines & outliers

Urban data, extracted from location-based social networks services, is vast and most of the time noisy. Understanding the regular occurring routines in such data is an important but also a challenging task. This thesis already proposed several methods and visualizations, which aim to assist the analyst with this problem. Another important task is finding anomalies in such data. Knowing when and why they exist, could aid urban planners improve citizen's daily life or plan emergency scenarios. Data, coming from Foursquare, contains semantical information about location-based places. Mining this information could help to achieve a better understanding of the spatial and temporal patterns and deviating behavior. This thesis proposes a statistical solution (see chapter 3.1.4.2) to the detection of outliers from the city's usual routine. The main idea is to find if there is a check-in count of a venue or category in the current data set, which deviates from the usual routine by a user-defined standard deviation multiplier. The routine is calculated at each aggregation in the monitoring system and is defined as the mean of all samples per venue and category for the tuples, described in chapter 4.1.2.4. All values, contributing to this result are stored, in

order to calculate the standard deviation of the samples. This deviation and mean are used in the following process to discover data anomalies:

1) Allow the user of the system to define what he considers an anomaly in the current data set, by choosing an appropriate standard deviation multiplier (refer to Figure 13. in 3.1.4.2).
2) For each venue or category:

Assume S is the control value of the current object to be examined, X is the user-defined multiplier, σ is the standard deviation and μ is the mean. Then, the object is marked as an outlier, if $S > \mu + X * \sigma \vee S < \mu - X * \sigma$, that is it deviates too much from the normal routine.

3) Visualize the found outliers. Currently the visualization consist of marking the anomaly object with a red border, which is not an ideal visualization. Another way to visualize this is to use glyphs or extend or shrink somehow the edges of the streamgraph. However, this approach still captures the attention of the user towards this object. Hovering on top of it, gives more information about the statistics behind it.

The found outliers are reflected in the Stream- (for the categories) and Mobilitygraph (for the venues). Figure 45 shows the temporal distribution of a non-typical workday (Monday), due to national holidays in Japan. Pictures a), b) and c) display how the anomalies are marked interactively, by changing the standard deviation multiplier parameter. The top picture contains anomalies having 3 times higher or lower values than the standard deviation. As it can be seen, this example contains only a few outlying values. Decreasing the multiplier by 1 each time, reveals more anomalies (picture b and c). Finding outliers in the temporal domain is useful, but not enough for explaining the reasoning behind them. For this task, the system must also consider the spatial distribution of the data. Consider Figure 46., containing the same data set. Assume, the user's attention has been grabbed by the deviations in the category “Professional & Other Places”.
For convenience, the data set has been constrained to venues only from this category. The Mobility Graph reveals its spatial patterns. There are several clusters, containing event spaces and convention centers. After further inspecting the clusters, the user can reveal the causes for the spike in “Professional & Other Places” check-ins between 9 am and 10 am. The right cluster contains an unusual amount of check-ins for this Monday hour. There are over 200 people checked-in with a routine value of only ∼33. The cluster's venue descriptions (top right of the system) displays a sorted list (by check-in count) of venues, contained in the cluster, and aims to aid the user to identify and investigate specific points of interests. Looking at the displayed venues for this cluster, it is easy to assume, that the responsible venue for the unusual check-in behavior between the 9 am and 10 am, is the “Makuhari Messe”. The streamgraph in Figure 46. shows another small spike of check-ins between 13 pm and 14 pm. Inspecting the “downtown” cluster, containing event space venues, reveals that he is responsible for the unusual amount of people checked-in at this hour. To find out the specifics of this cluster, the user could zoom-in further on the chosen cluster, adjust the clustering parameters or use the heat map visualization to discover the “guilty” venues.

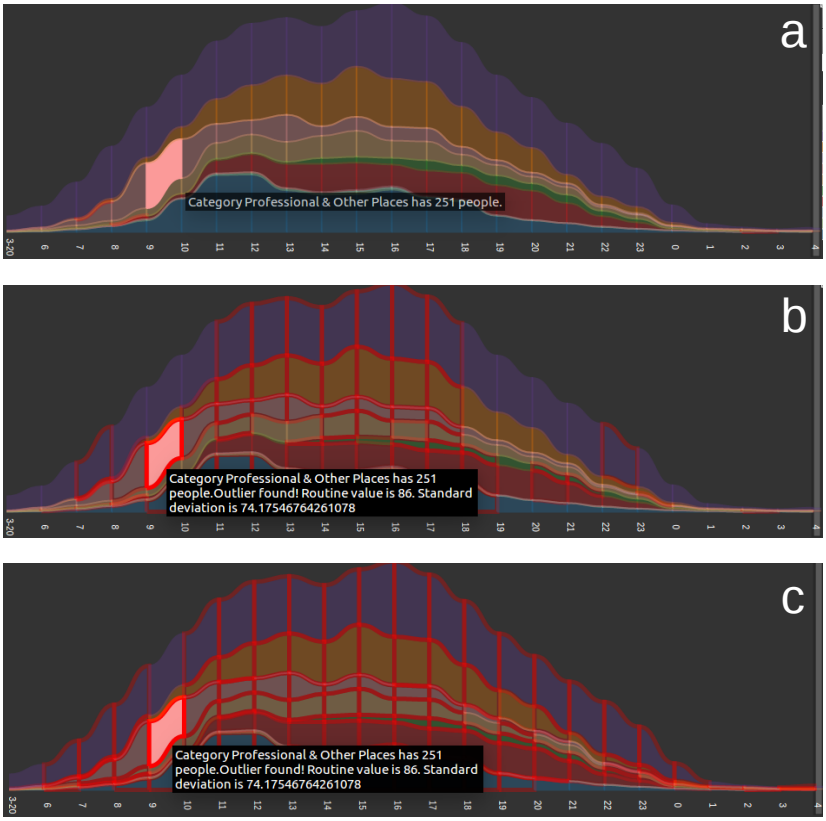

Figure 45. Found outliers in the Streamgraph for different standard deviation multipliers (denoted as X). a) X = 3 b) X = 2 c) X = 1

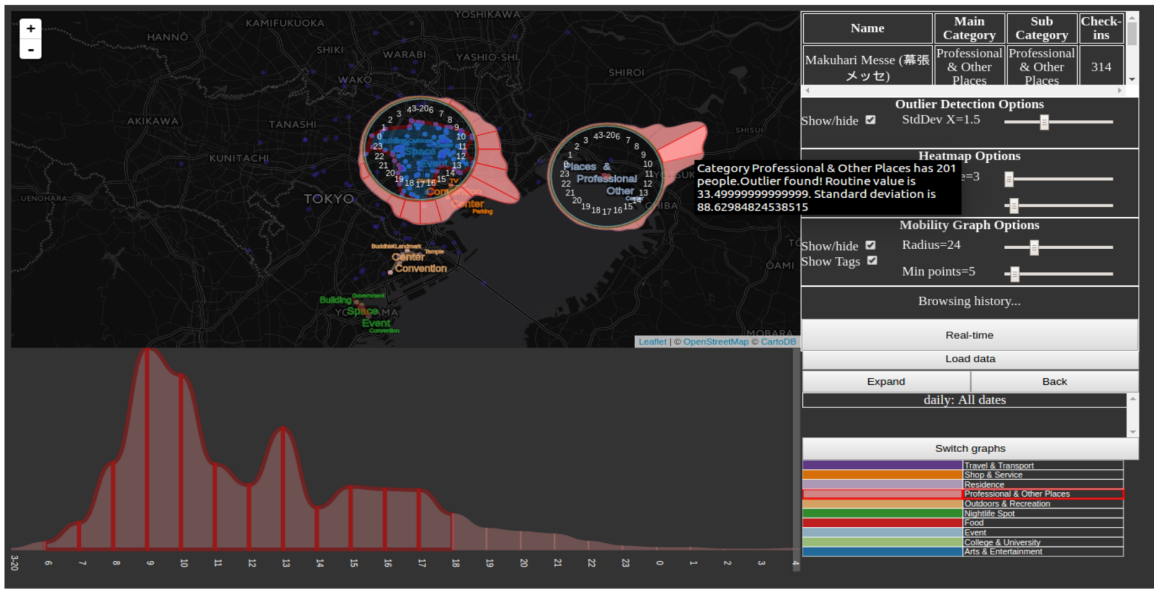

Figure 46. Spatial outliers of clusters. The right cluster (or more specific the “Makuhari Messe” venue) is responsible for the deviations in the check-in behavior of the category (“Professional & Other Places”) between 9 am and 10 am.

5. Case studies

This chapter presents two case studies, which test the feasibility of the data and the capabilities of the system.

5.1 Typical routines

This case study analyses the typical check-in behavior of Foursquare users in Tokyo city, Japan. As one of the largest cities in the world, it poses a great challenge towards its understanding. Knowing about the routine activity behavior of workdays or weekends could aid urban planners to improve the everyday life of the citizens. The implemented monitoring system in this thesis collects data from the end of February 2017 in a 30 km perimeter around the city center. This chosen perimeter misses on opportunities to analyze "Residence" category venues, due to their greater distribution in the rural areas of the city, but it allows to achieve a much better temporal resolution of the data. Before proceeding with the analysis, the user of the system needs to define what he considers as "routine". Figure 47. depicts the process of finding the most "routine"-like workday and weekend. Picture a) shows the data anomalies of the daily distributions using a 2x standard deviation multiplier (refer to 3.1.4.2, Figure 13 for the probability percentages). The week after (13.03) seems like the best candidate for routine behavior. Zooming in on its workdays hourly distribution (see Picture b) reveals Thursday (16.03) as the least deviating workday in all categories. The choice for a typical weekend is Sunday (19.03), as seen in Picture c). The next day (20.03) happens to be a public holiday in Japan, thus the anomalies in the data. This day could be useful for case comparisons between the typical weekend and a holiday. Consider the chosen typical workday (see Figure 49). Refer to the category legend (Figure 48) for details on the color mappings. The distribution follows the typical expected workday behavior. There is a high number of check-ins in the morning (rush hour) for the "Travel & Transport" category, when people commute to work. The "professional" category starts climbing from 8 am to 11 am, as expected, when people arrive to work. There is a large amount of check-ins in the "Food" category from 11 am until 14 pm, when people are on lunch break. There are no large deviations for each of the categories from 14 pm to 17 pm. The second rush hour begins at 17 pm when people start leaving their workplaces. After work, some people go shopping, to restaurants or bars, movie theaters or concerts and so on. This usual behavior can be confirmed by the peaks in these categories, respectively "Shop & Service", "Food", "Nightlife", "Arts & Entertainment" for the hours between 17 pm and 20 pm. After 20 pm, many shops close and so do theaters and so on, which is probably the reason for the drop in their check-in numbers. Figure 50. illustrates the temporal distribution for each individual category for the chosen workday. Although some of these category distributions are not fully reliable, namely "Residence" and "Event" due to lack of enough data or no big events happening at the time, "Residence" still matches the expectations of people checking in at home after work.

More specifics are needed to explore the spatial routines of people in the span of a typical workday. Before the definition of some typical problem scenarios, consider Figure 51, representing an overview of the spatial distribution.

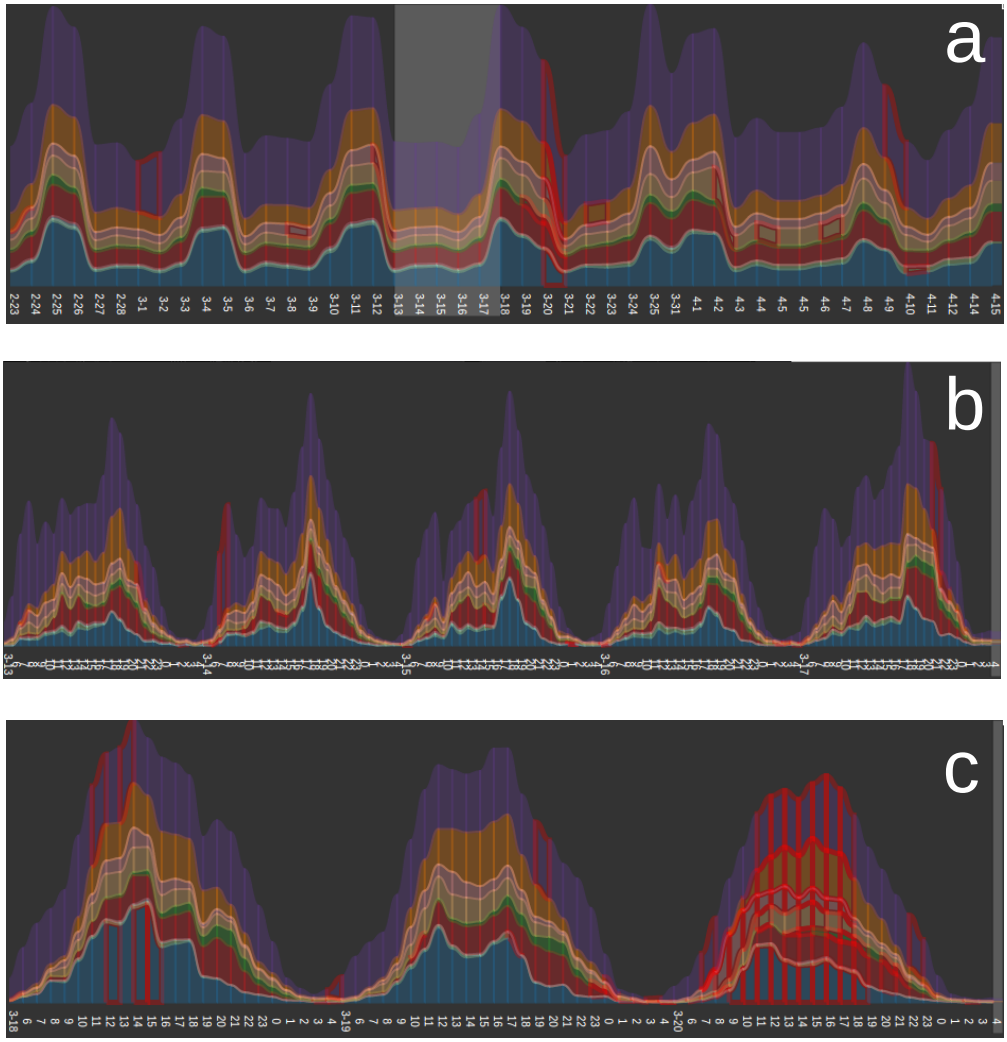

Figure 47. Process of finding the most “routine”-like week behavior using a standard deviation multiplier of 2x a) Most “routine” week. b) Most “routine” workday seems to be Thursday (16.03) c) Most “routine” weekend seems to be Sunday (19.03). The third displayed day (20.03) is a national holiday in Japan on a Monday, thus the number of data anomalies.

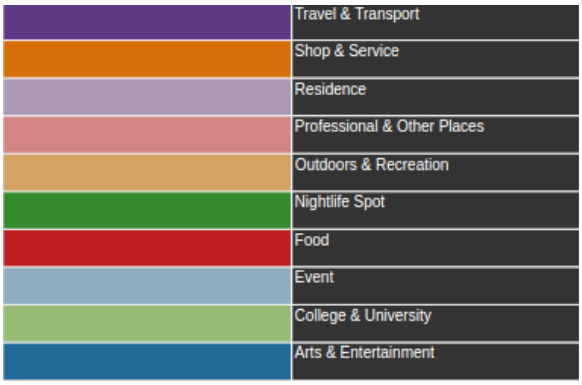

Figure 48. Category legend for convenience.

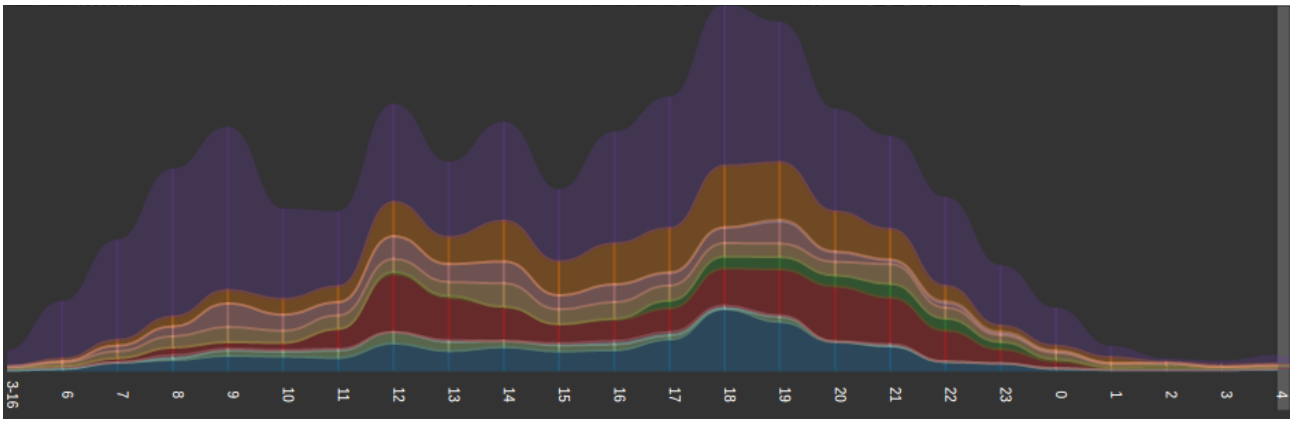

Figure 49. Temporal distribution per categories for a typical workday. (Thursday)

Exploring the formed clusters, reveals some interesting insights on the spatial data. The clusters follow the usual routine (for the same 2x standard deviation multiplier) for each of the categories, but one, the “Food” category. The outlier detection algorithm finds anomalies in the distribution of “Food” venues throughout the whole day. This, however, does not automatically mean that the temporal distribution of the category “Food” should deviate from the typical routine. The routine value is calculated per venue or category, which creates two possibilities:

1) Different “Food” venues are visited each Thursday, thus the routine value for each is low. (Which could be expected, since people frequently change restaurants.)
2) There are big deviations of check-ins for some frequently visited “Food” venues or clusters.

Exploring further the spatial distribution of only “Food” venues, shows the “guilty” downtown clusters (see Figure 52). The user could further analyze them by zooming in. Determining the reasoning could require domain knowledge or more exploration (e.g. loading “food” data from different Thursdays). For the sake of simplicity, this case study won't consider any “food”-related problems.

Studying the spatial patterns of a workday requires specific problem cases. Consider these two:

1) What are the most congested areas, in terms of public transport, throughout the span of a typical workday?

2) What are the most popular shopping places after work in a typical workday scenario?

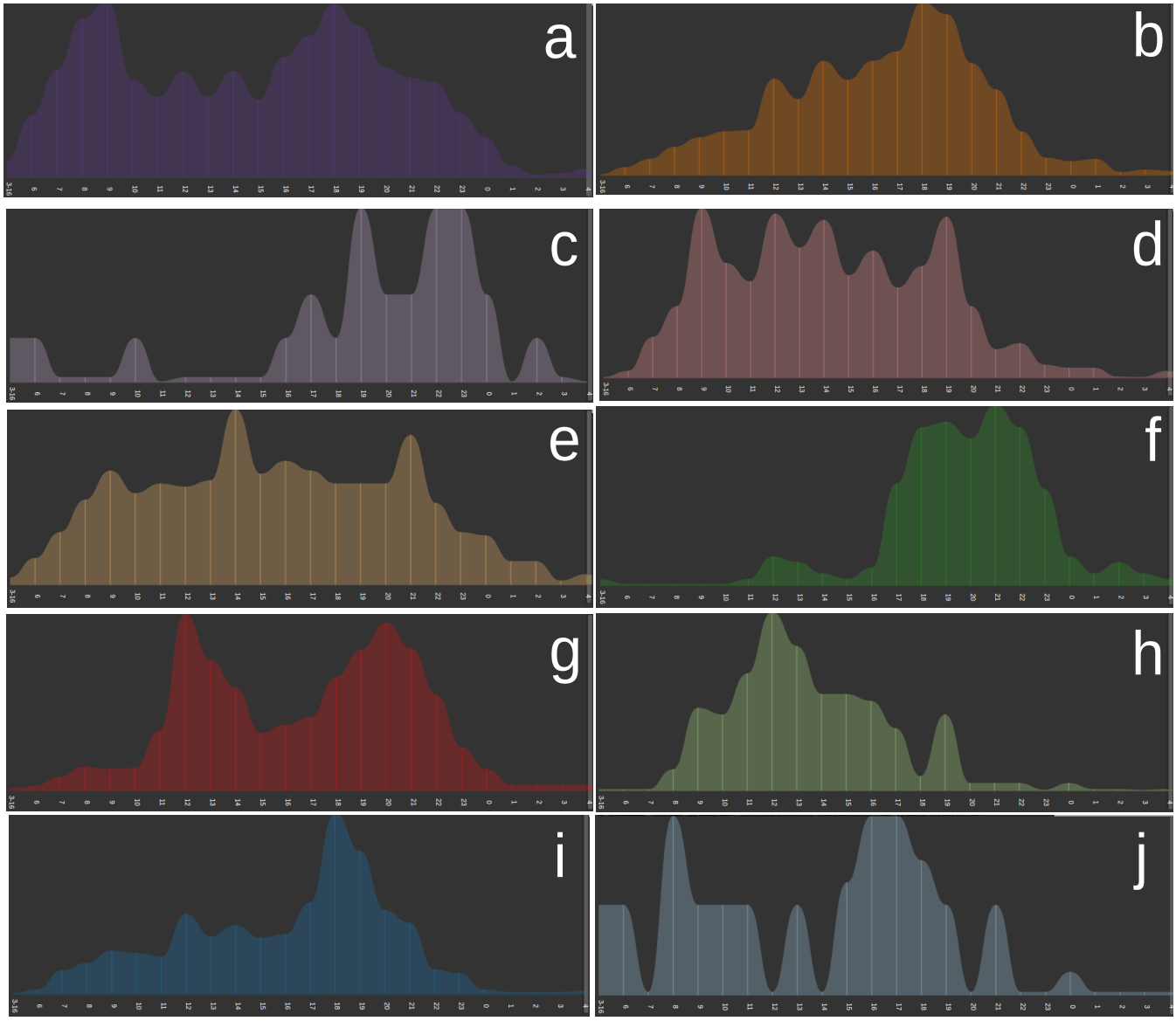

Figure 50. Temporal distribution of each category of the typical workday.
a) Travel & Transport, b) Shop & Service, c) Residence, d) Professional & Other Places, e) Outdoors & Recreation, f) Nightlife, g) Food, h) College & University, i) Arts & Entertainment, j) Event

Figure 51. Spatial distribution of a typical workday (Thursday). Inspecting each cluster and further zooming in on them shows no large-scale spatial anomalies for all categories but the “Food” category.

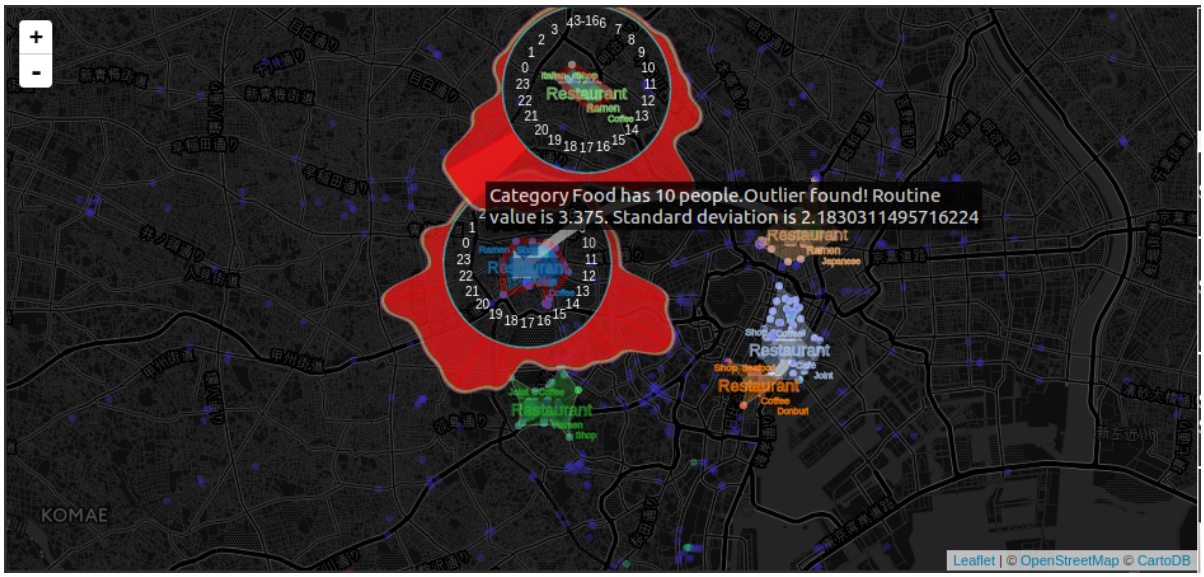

Figure 52. Several “Food” clusters with routine deviation for the typical workday. Although, the overall “Food” activity distribution of the clusters could be the same, the venues could be different.

1) What are the most congested areas, in terms of public transport, throughout the span of a typical workday?

A possible workflow is displayed in Figure 53:

1) Filter out all data, not contained in the category “Travel & Transport”.
2) Explore the spatial distribution with a heat map. (as shown in picture a)
3) Adjust the clustering parameters to match the expectations from 2).
The bottom “heat” cluster, which is the airport, is not considered in this case. (as shown in picture b).
4) Explore each cluster individually and compare their temporal check-in distributions (as shown in picture c).
5) Compare the traffic data for each cluster. (as shown in picture d).

2) What are the most popular shopping places after work in a typical workday scenario?

A possible workflow is displayed in Figure 54:

1) Filter out all data, not contained in the category “Shop & Service” and load “after work” data, from 17 pm to 22 pm (not including). (as shown in picture a)
2) Explore the spatial distribution with a heat map (as shown in picture b)
3) Adjust the clustering parameters to match the expectations from 2).(as shown in c)
4) Explore the semantics of the most popular clusters (see picture c). For example, the light blue cluster contains electronics stores and hobby shops. The light green clusters consists of electronics, department and book stores.

Just for the sake of the example, problems can get even more specific. Consider the problem of discovering the most popular activities (out of “Shop & Service”, “Food”, “Arts & Entertainment”) before people go to a bar or any other “Nightlife” category on a Friday night. Consider Figure 55, which tries to answer this problem using an alluvial diagram. It clearly shows that most people prefer going to “food” venues, like restaurants, before visiting a bar.
Consider the defined most “routine”-like weekend day (Sunday, 19.03) on Figure 56. Most of the categories are “normally” distributed. Their peaks are in the middle of the day, when the people are most active. The “Nightlife” category is active after 17 pm, as expected. The category “Professional & Other Places” contains tourist places, conventions and shrines, which contribute to its overall check-in counts on weekends. Refer to Figure 57. for the temporal distribution of each category. The category “College & University” contains a small outlier, which can be ignored. It is worth noting that the universities are in their non-lecture period. The summer semester in Tokyo starts on the 5th of April.

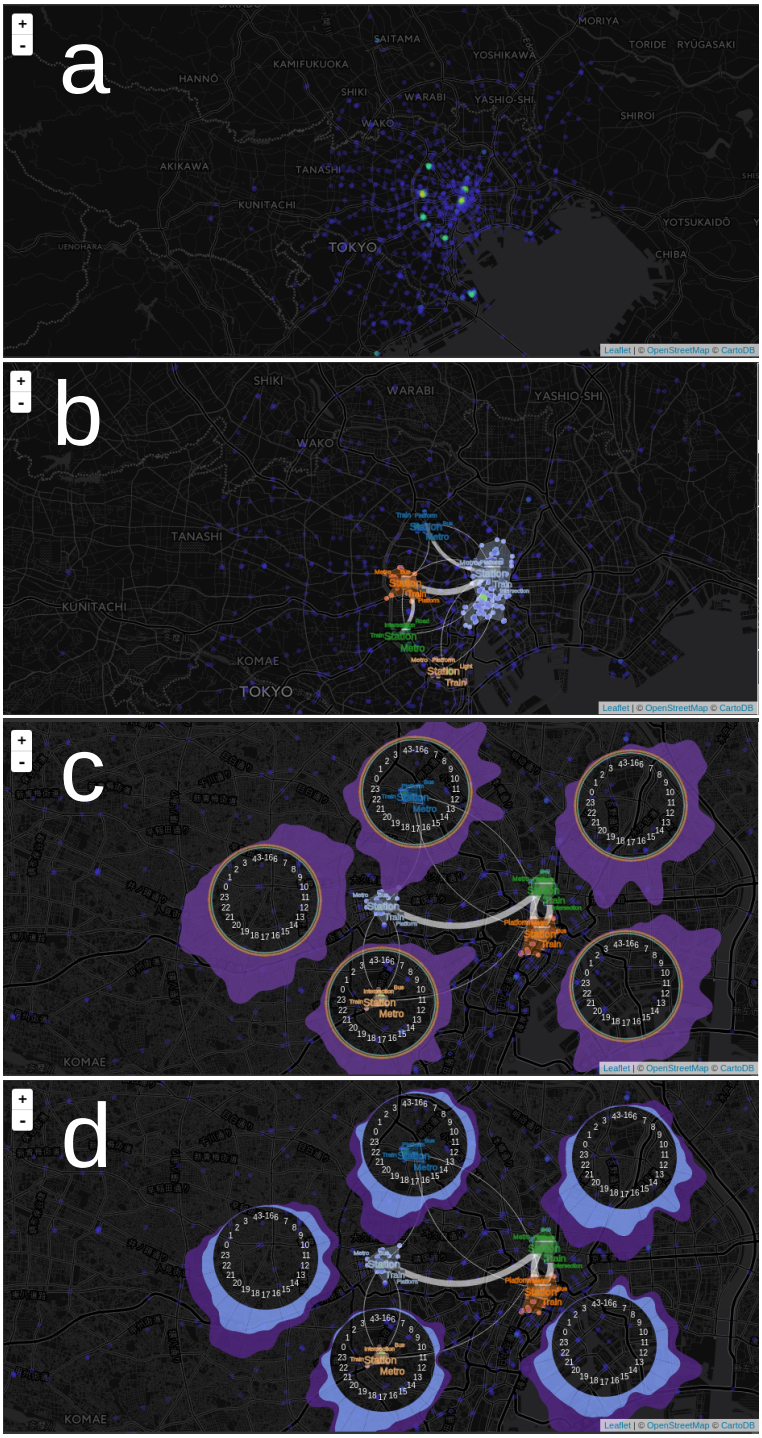

Figure 53. Possible workflow for the discovery of congested public transportation areas throughout the time frame of a typical workday. a) Overview distribution b) Zoomed-out clusters. c) Zoomed-in clusters with temporal activity distribution. d) Zoomed-in clusters with traffic distribution (Light blue refers to outbound traffic, purple refers to inbound traffic).

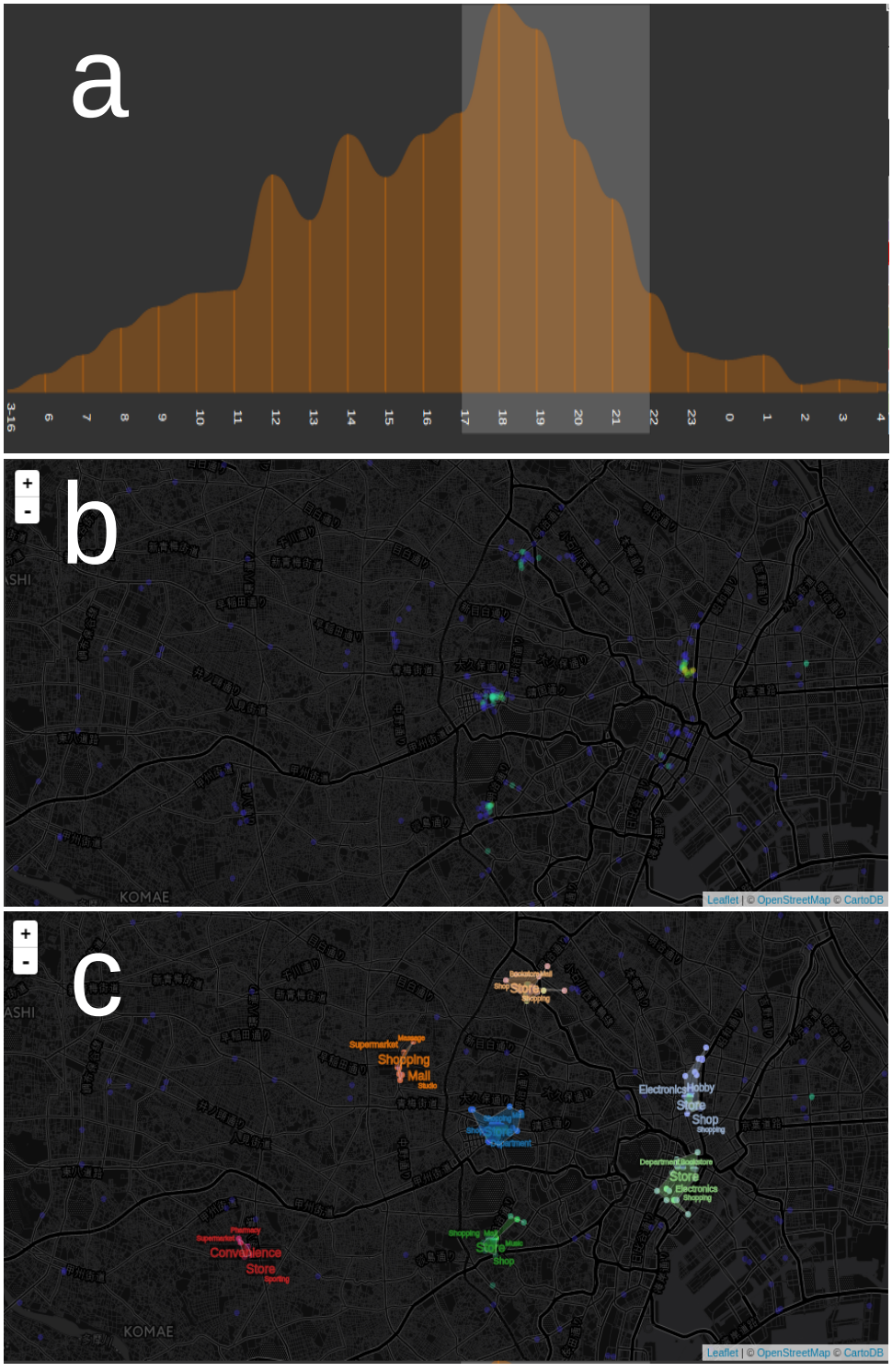

Figure 54. Possible workflow for the discovery of popular shopping areas after work. a) Filtered temporal distribution b) Spatial distribution c) Clusters, containing “shopping” venues for the time frame 5pm - 22pm.

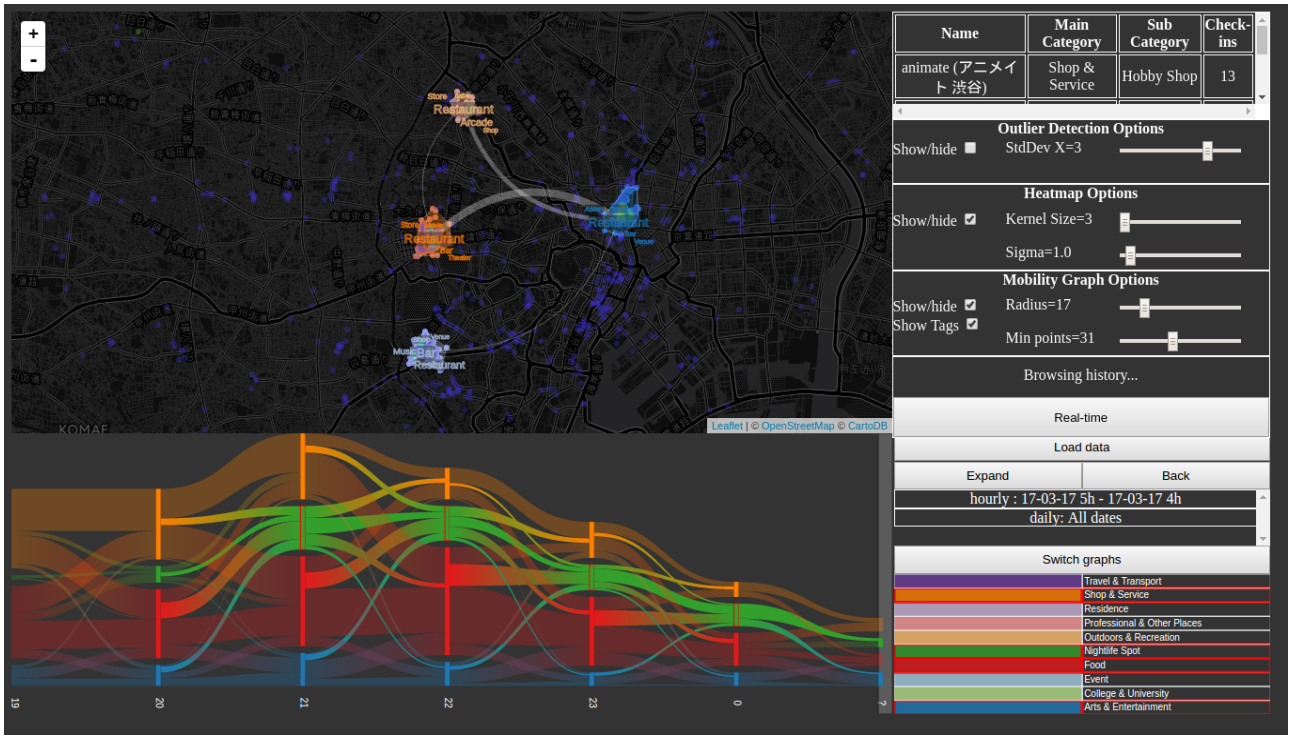

Figure 55. Alluvialgraph and spatial representation of the categories “Shop & Service”, “Nightlife Spot”, “Food” and “Arts & Entertainment” on a Friday night (19 pm – 0 am on 17.03.2017).

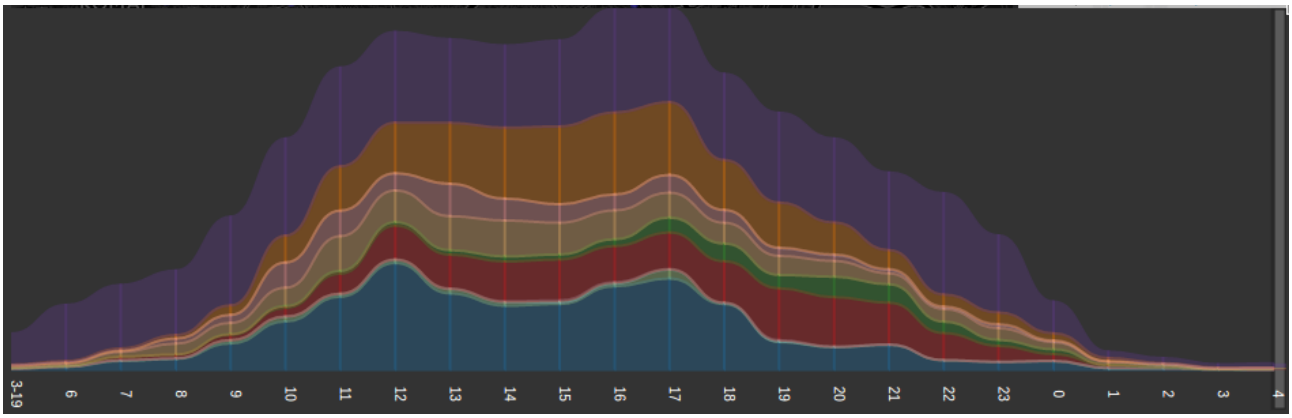

Figure 56. Temporal distribution of a typical weekend.

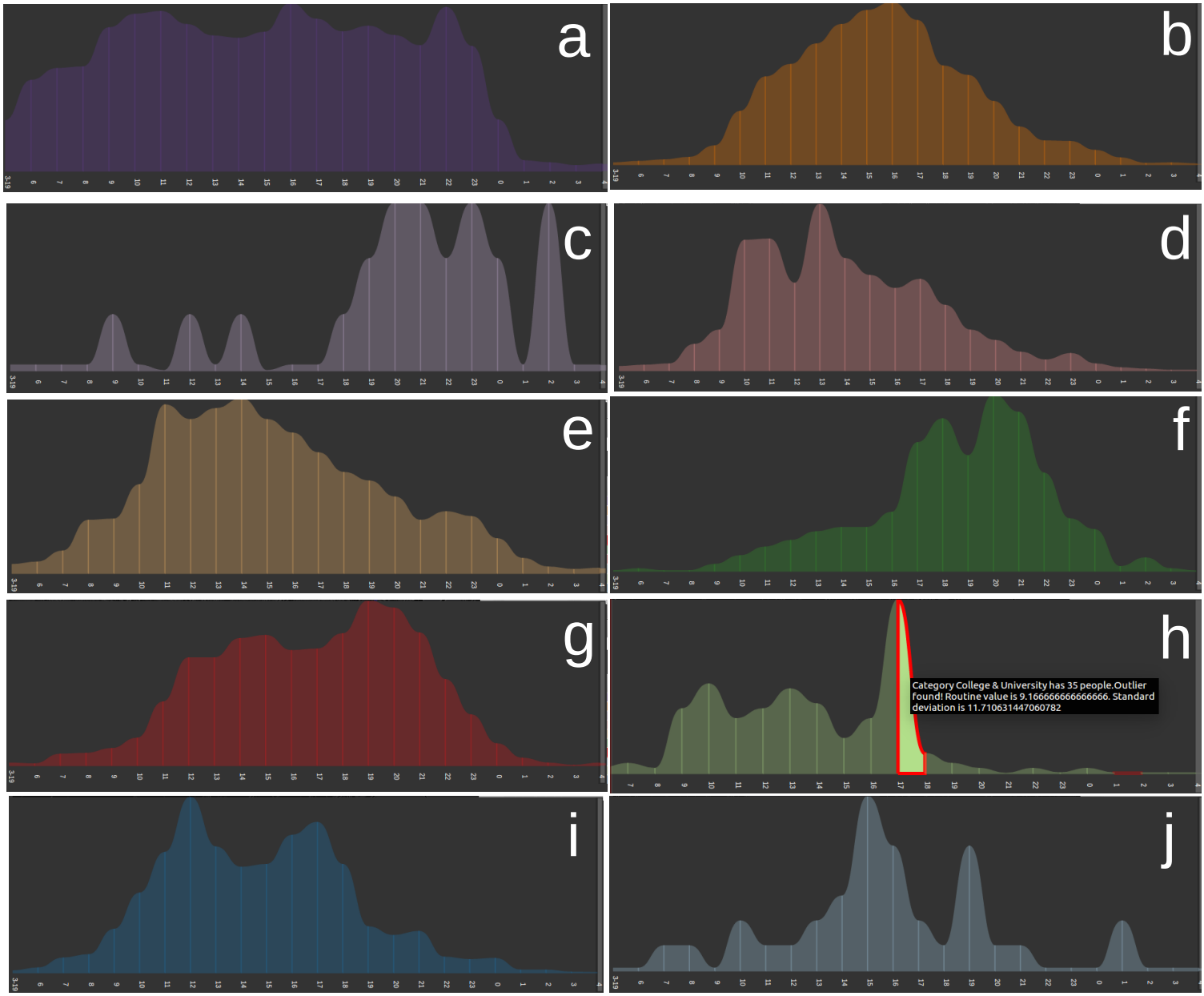

Figure 57. Temporal distribution of each category of the typical weekend. a) Travel & Transport, b) Shop & Service, c) Residence, d) Professional & Other Places, e) Outdoors & Recreation, f) Nightlife, g) Food, h) College & University, i) Arts & Entertainment, j) Event

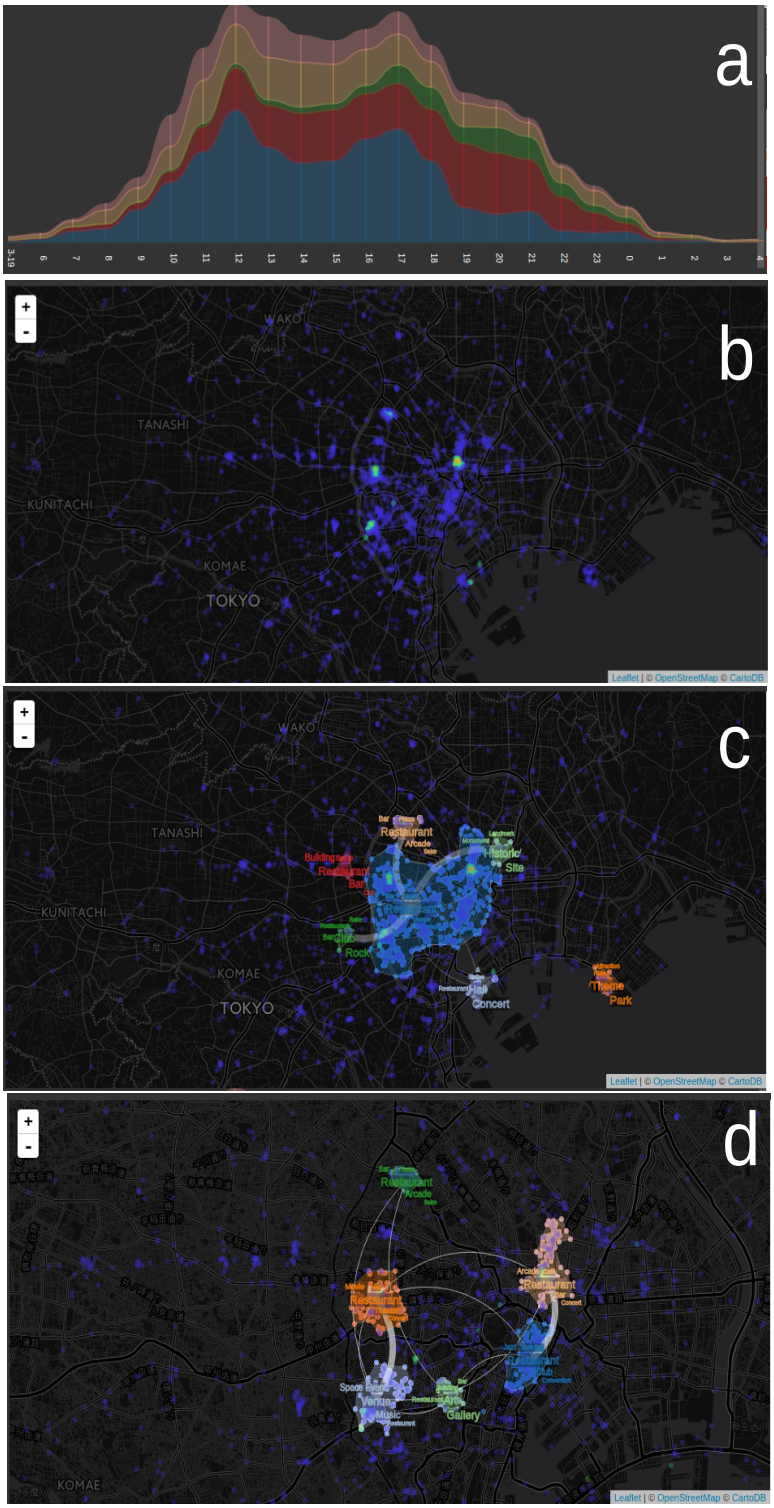

Figure 58. Possible workflow for finding the “rest” places on a typical weekend (Sunday).
a) Temporal distribution, containing the categories “Professional & Other Places”, “Outdoors & Recreation”, “Nightlife”, “Food” and “Arts & Entertainment”. b) Finding the most active areas of the city using a heat map. c) Exploring the “downtown” and rural clusters first d) Zooming-in further on the “downtown” cluster reveals more specific “resting” place clusters.

A typical problem question for a “weekend” spatial analysis could be:

What are the favorite “rest” places for people on a typical weekend?

Consider Figure 58. for a possible solution to this problem:
1) Filter out all data, not contained in the categories “Professional & Other Places”, “Outdoors & Recreation”, “Nightlife”, “Food”, “Arts & Entertainment”. Consider the temporal distribution. (as shown in picture a).
2) Find the most popular areas using a heat map representation (as shown in picture b)
3) Refine the algorithm parameters accordingly to find clusters of different densities . (as shown in picture c).
4) Zoom-in to further explore a chosen cluster (as shown in picture d).
5) Furthermore, the user can filter out more of the categories, to focus on specific activities.

This solution reveals some of the most popular “resting” places, which are restaurants, arcade gaming clubs, art galleries, music events, theme park, bars and rock clubs, theaters and so on...

5.2 Events and public transportation

Events, like concerts and conventions, gather masses of people in one place. Many people use the public transportation to arrive at such places. This movement flows can be studied to plan and possibly prevent future traffic congestions. Another big concern with masses of people is their security. Knowing how people travel to such events at certain places (stadiums or convention centers) could help in the planning of emergency and security measures. This chapter presents two events (discovered from the extracted data) and their mobility analysis, in terms of public transportation.

“Nippon Bundokan” Concert Hall (The “Idolm@ster” Music Festival):

Finding such events in large scale social media data is as challenging, as its analysis. Consider the following steps a user has to traverse using the prototype of the system (see Figure 59):

Assume the user of the system needs to find events that are in the arts and entertainment category. (sport games, concerts, etc.)
1) Define a standard deviation multiplier (1.3x in this case) to find anomalies in the data.
2) Filter temporal data by the category “Event” and explore interesting outliers (as shown in picture a).
4) Filter temporal data by the category “Arts & Entertainment” and explore interesting outliers (as seen in picture b).
5) Exploring further the hourly distribution, shows the same peak hour for both (as shown in picture c)

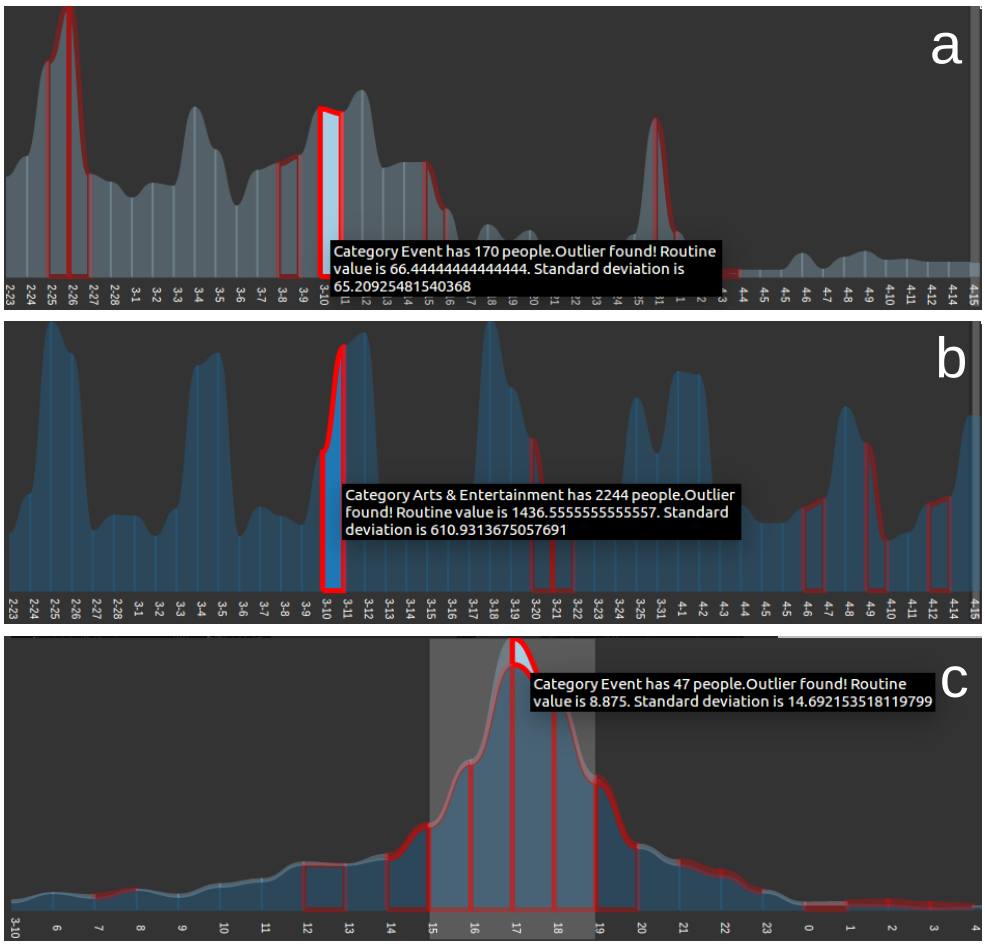

Figure 59. The process of finding entertainment events.

The next step would be to find the event spatially. For this task, consider the heat map representation (Figure 60) of the spatial distribution of data from these 3 categories: “Travel & Transport”, “Event”, “Arts & Entertainment”. The represented time frame is from 15 pm to 18 pm (see Figure 59. c), when people start gathering for the event. The heat map clearly shows a small “heat” cluster in the center of the city, with much larger check-in sizes than the rest. Exploring further this small cluster of venues reveals more details about the event and place. The analysis process can begin after the event has been discovered in the temporal and spatial domain. Consider Figure 61. Picture a) visualizes (with an alluvial graph) the flows of people between the categories “Travel & Transport” (purple), “Arts & Entertainment” (blue) and “Event” (light blue) for the temporal frame of the event (10.03.2017). The graph clearly shows a large flow of people transitioning from the category “Travel & Transport” to “Arts & Entertainment” between 17 pm and 18 pm, which is probably when the event begins and confirms the hypothesis that lots of people are traveling with public transportation towards the event. The next step would be a spatial analysis of their mobility patterns. Consider Picture b). It shows a possible density parameter refinement, allowing the algorithm to cluster together small public

transportation regions (like metro or train stations). Hovering over the event area, highlights only the edges of the graph coming out of or going into the selected node. Animations improve further the visualization perception (although it cannot be shown). All flows, concerned with the event area, are incoming, as it can be seen on the radial graph, representing the temporal distribution of traffic for the event cluster (see picture b). The biggest inflow of people to the event comes from venues, contained in the blue cluster. His neighborhood cluster (light orange) contributes with a fair amount as well. Distant clusters are also responsible for the large number of inbound traffic to the event (green and purple). Picture c) reveals the semantical meaning behind these clusters. It can be seen that the largest number of people traveling to the event is coming from public transportation clusters, like train and metro stations.

Figure 60. Finding the event using the heat map representation.

A small amount of people arrived at the event from other concert halls, music venues or the nearby ongoing baseball match. Such big events can affect positively (from a marketing perspective) and negatively (from the formation of large crowds) nearby neighboring areas. For example, consider picture d), showing the outbound traffic from a nearby (to the event) train station. The most people travel to the event area, but there are some exceptions. A good amount of people flow to the neighboring clusters as well. The dark green (left) cluster has an event on its own happening (baseball match), which accounts for its inbound traffic. The light green (bottom) cluster shows a lot of people transitioning back and forth between the both areas, which is an indication for an ongoing traffic in the area. Further exploration and some domain knowledge about the clusters can reveal deeper insights on the influence of the event on the nearby regions.

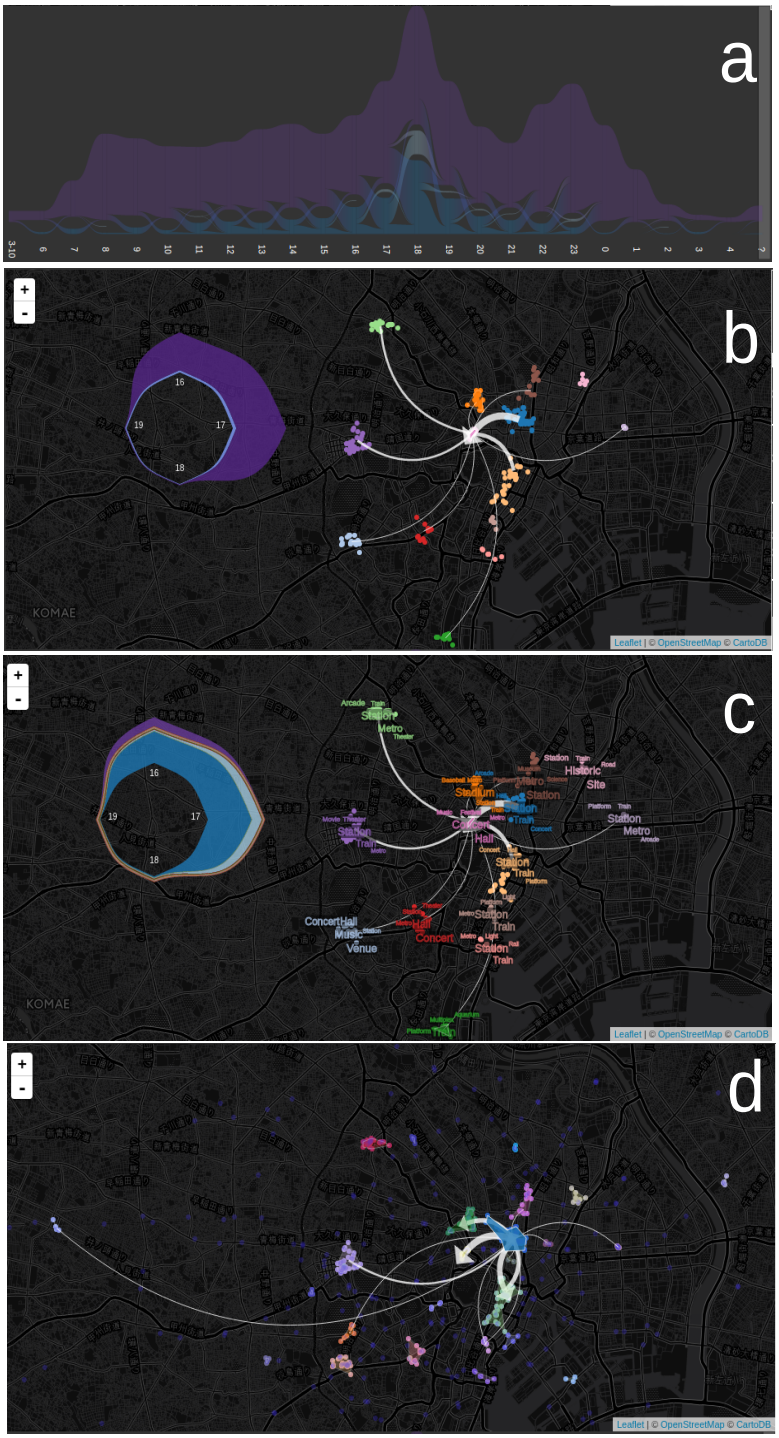

Figure 61. Possible analysis process of the movement flows to a concert event between 15 pm and 19pm. a) Shows temporal transitions for the defined time frame. Large amount of people flow from the category “Travel & Transport” (purple) to “Arts & Entertainment” (blue). b) Mobility between venue clusters. The radial graph shows the distribution of incoming flows towards the region of the event. c) Same as b), but displays the semantics behind the clusters. Most of them are train stations (there are also other concert clusters and a baseball match cluster). d) Shows how the event affects nearby clusters of venues.

“Tokyo Big Sight” Convention Center:

As the previous example, the first step would be the discovery of such events in the temporal domain. The process is exactly the same, but with the exception that the user is assumed to search for events, regarding conventions, from the category “Professional & Other Places”. Finding the event spatially is a simple task using the heat map representation. Figure 62. shows venue data for the time frame 8 am to 13 pm on the 25th of March. The small “heat” cluster is easy to spot. This group of venues should contain a much larger number of people checked-in, due to the lack of other “heat” clusters. Inspecting further the venues, reveals their name (refer to the title) and check-in count (~500 check-ins). Such large-scale event involves the movement of large amounts of people. The assumption is that many of these people use the public transportation service to arrive at the event location. To prove this hypothesis, consider picture a) on Figure 63. It shows a large amounts of flows from the category “Travel & Transport” to the category “Professional & Other Places” in the same time frame of the event, which proves the hypothesis. The next challenge is to study these flows to discover the spatial movement of people using public transportation services. Picture b) on Figure 63. shows these movement flows, while picture c) reveals the semantics of these clusters. As expected, all of them are train, metro or bus stations. With further knowledge of the city's transportation structure, the user of the system can study in details each of the clusters and plan better traffic, security and emergency measures for big conventions happening in the same area. Another analysis possibility is studying the flows of people from venues of different categories, like restaurants and shops, to the event area, in order to increase business revenues.

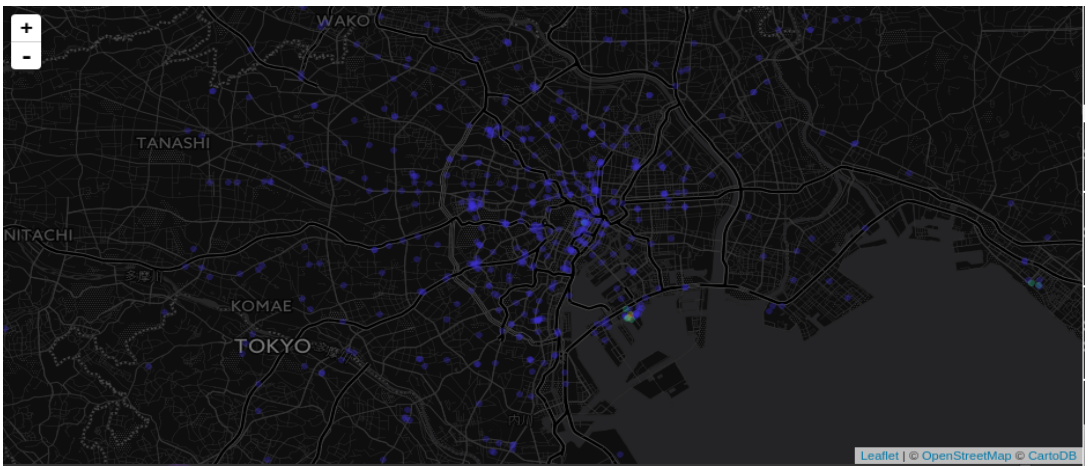

Figure 62. Heat map representation of the convention event on 25th of March from 8 am to 13pm.

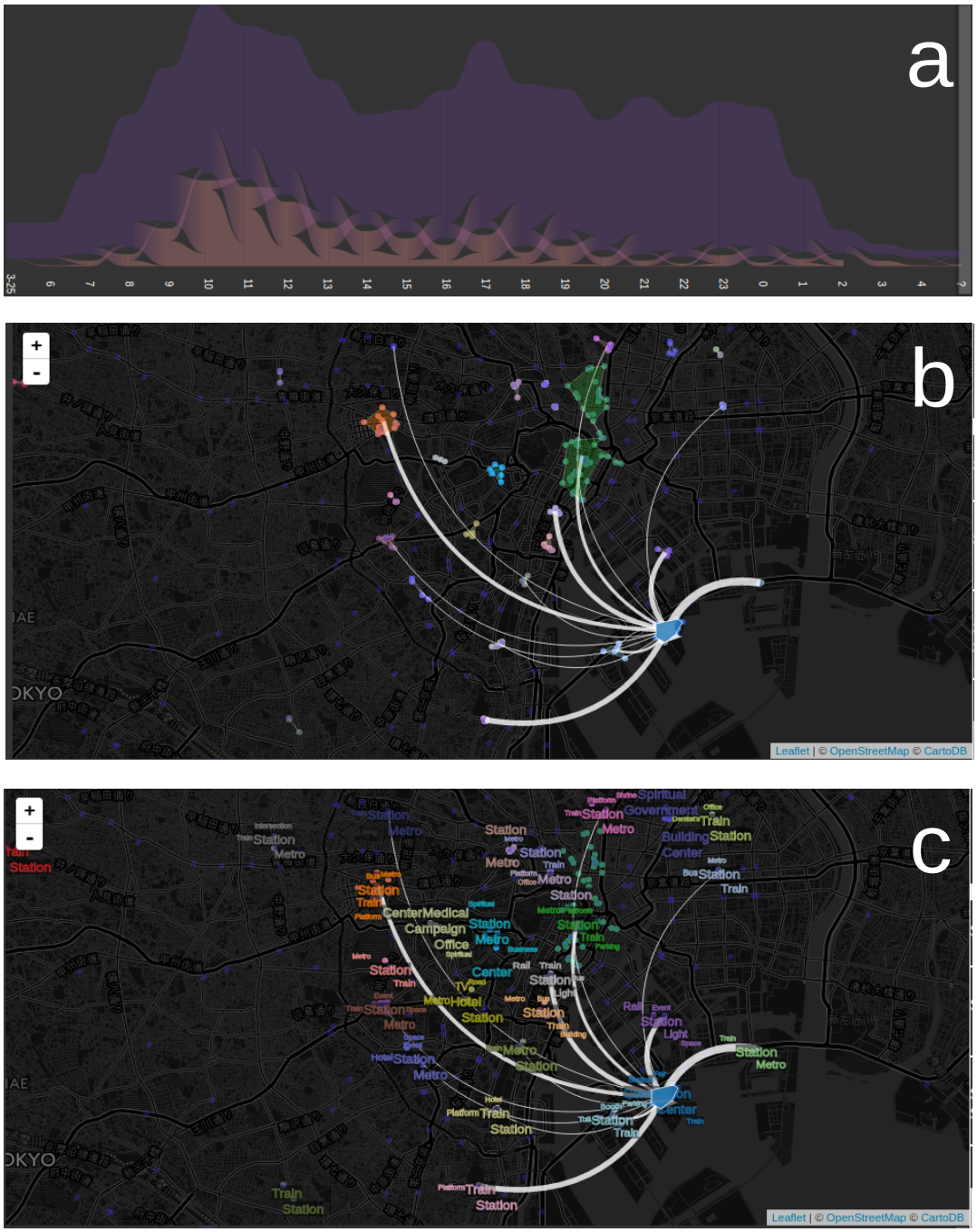

Figure 63. Possible analysis process of the movement flows to a convention event between 8 am and 13 pm. a) Shows temporal transitions for the defined time frame. Large amount of people flow from the category “Travel & Transport” (purple) to “Professional & Other Places” (pink). b) The distribution of flows between venues clusters. c) Same as b), but displays the semantics behind the clusters.

6. Conclusion

Cities are complex and dynamic environments. Social media platforms provide the means towards studying these dynamics. The scope of this work includes the implementation of a monitoring system, which collects data in (near) real-time from such social media services, namely Foursquare and Twitter. This is a challenging task, due to API restrictions and limitations, and requires a trade-off between the temporal and the spatial resolution of the data. Furthermore, a visual analytics system has been developed to guide urban planners or marketing specialists towards the goal of understanding the dynamics of urban areas. The capabilities of the system comprise (near) real-time inspections and analysis of historical data. The spatial distribution of POIs is represented by a heat map, which reveals the “hottest” regions, in terms of check-ins from the social media platform Foursquare. Mobility of people is an important part of the dynamics of urban areas. Data, extracted from Twitter, enables the possibility to study people's movements. To this end, a mobility graph has been implemented, which uses a density-based algorithm to cluster together POIs into a set of nodes to the graph. The respective edges involve the usage of data from Twitter to represent aggregated movements between these clusters. Each node contains a tag cloud representation of its top five activities, extracted from the Foursquare sub-categories of the contained POI. These tags reveal the semantics of the clusters, which contributes to a better analysis. The temporal domain of the data is represented and managed by a stream- and alluvialgraph. The streamgraph allows for the analysis of the data's temporal distribution per Foursquare category, while the alluvialgraph visualizes transitions between them. Lastly, a statistical method has been implemented for the semi-automatic detection of anomalies in the data. Such methods could help analysts to find unexpected behavior or big events and plan accordingly. Two case studies have been presented to test the feasibility of the extracted data and the capabilities of the system in real life scenarios. The two scenarios involve the analysis of typical urban routines in Tokyo, Japan and the movement of masses of people towards large-scale events using public transportation services.

6.1 Limitations

The current state of the system allows for the analysis of a single user-defined city. This is an obvious limitation, due to the impossibility to compare multiple urban areas and their routines. Furthermore, the extracted data contains no information about its demographic features. Knowing, whether the majority of population are young or old, residents or tourists and more can reveal different patterns and allow for new analysis possibilities. Another limitation of the system involves the mobility graph and its clusters. Sometimes, the proposed clusters from the system are not ideal for the analysis focus. Allowing the users to define areas of interest in multiple ways would be beneficial for the analysis process.

6.2 Future work

The system could be expanded further to allow the possibility for real-time simultaneous analysis of multiple cities. This would require the restructuring of the monitoring system to be able to store large amounts of data. Furthermore, the visual analytics system would need to process this data more efficiently, e.g. by introducing spatial pre-aggregation strategies. Another possibility would be the integration of more social media platforms with different analysis strategies (e.g. analysis of text data from tweets) and static government data, to contribute to the complete analysis of urban dynamics.

References

[1] World Health Organization, Urban Population Growth, 2017, URL http://www.who.int/gho/urban_health/situation_trends/urban_population_growth_text/en/, Last Accessed 01.05.2017.

[2] Portugali, J, What makes cities complex?, 2013, URL http://www.spatialcomplexity.info/files/2013/10/Portugali.pdf, Last Accessed 01.05.2017.

[3] Achilleas Psyllidis, Alessandro Bozzon, Stefano Bocconi and Christiaan Titos Bolivar. Harnessing Heterogeneous Social Data to Explore, Monitor, and Visualize Urban Dynamics, 2015.

[4] T. von Landesberger, F. Brodkorb, P. Roskosch, G. Andrienko, N. Andrienko, and A. Kerren. MobilityGraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. IEEE Transactions on Visualization and Computer Graphics, 22(1):11–20, 2016.

[5] Kling, F., & Pozdnoukhov, A. . When a city tells a story: Urban topic analysis. In Proceedings of ACM SIGSPATIAL 2012 (pp. 482–485). Redondo Beach, CA, 2012.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, 2003.

[7] Wei Zeng, Chi-Wing Fu, Stefan Arisona, Kwan-Liu Ma Visualizing the Relationship Between Human Mobility and Points of Interest. In IEEE Transactions on Intelligent Transportation Systems PP(99):1-14, 2017.

[8] Thomas, J.J., Cook, K.A.: Illuminating the Path. IEEE Computer Society Press, Los Alamitos, 2005.

[9] Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melancon, G. (2008): Visual Analytics: Definition, Process, and Challenges. In Kerren, A., Stasko, J.T., Fekete, J.-D., and North, C. (Eds.) Information Visualization – Human-Centered Issues and Perspectives. Volume 4950 of LNCS State-ofthe-Art Survey, Berlin: Springer, pp.154-175, 2008.

[10] Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis and Florian Mansmann, Mastering the Information Age Solving Problems with Visual Analytics, 2010.

[11] Card, S., Mackinlay, J., Shneiderman, B. Readings in Information Visualization: Using Vision to Think, Morgan Kaufmann, 1999.

[12] Shneiderman, B. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. IEEE Visual Languages, 1996.

[13] Pirolli, Peter, and Stuart Card. "The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis."Proceedings of international conference on intelligence analysis. Vol. 5. 2005.

[14] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In Pro. ACM KDD, 1994.

[15] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial database with noise. In I&‘1 Conference on Knowledge Discovery in Databases and Data Mining (KDD-96), Portland, Oregon, August 1996.

[16] D. Birant and A. Kut. St-dbscan: An algorithm for clustering spatial-temporal data. Data Knowl. Eng., 60(1):208–221, 2007.

[17] The Central Limit Theorem, URL, http://www.math.uah.edu/stat/sample/CLT.html, Last Accessed 01.05.2017.

[18] Mapbox, A new algorithm for finding a visual center of a polygon, 2016, URL, https://www.mapbox.com/blog/polygon-center/, Last Accessed 01.05.2017

[19] Kamermans, Mike, A Primer on Bézier Curves, 2011-2017, URL, https://pomax.github.io/bezierinfo/, Last Accessed 01.05.2017.

[20] Emil Rosén, Emil Jansson, Michelle Brundin. Implementation of a fast and efficient concave hull algorithm.Project in Computational Science, Uppsala University, Department of Information Technology, 2014, URL, http://www.it.uu.se/edu/course/homepage/projektTDB/ht13/project10/Project-10-report.pdf, Last Accessed 01.05.2017

[21] Jonathan Feinberg, Wordle, URL, http://static.mrfeinberg.com/bv_ch03.pdf, Last Accessed 01.05.2017.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

_____

Ort, Datum, Unterschrift