Duale Hochschule
Baden-Württemberg
CAS
Center for Advanced Studies

Evaluation of data-driven methods to locate convenient pickup locations for ride hailing

Studienarbeit

(Student Research Project)

at the Center for Advanced Studies Heilbronn
of the Cooperative State University Baden-Württemberg

Author	Ernesto Elsäßer
Matriculation Number	2016242
Cooperative Partner	moovel Group GmbH, Stuttgart
Reviewer	Frau Prof. Katja Wengler
Submission Date	February 9, 2019

Declaration of originality

I confirm that this assignment is my own work and that I have not sought or used inadmissible help of third parties to produce this work and that I have clearly referenced all sources used in the work.

This work has not yet been submitted to another examination institution neither in the same nor in a similar way and has not yet been published.

Stuttgart	09.02.2019	[illegible]
Place	Date	Signature

Abstract

A central business process of the emerging ride-hailing industry is the on- and off-boarding of customers. Although most of the services available today are powered by software, finding suitable spots to pick up their customers is usually left to the drivers on the street. In metropolitan city centers, where traffic is dense and curb space is scarce, this practice often leads to unwanted delays and traffic violations.

The purpose of this research project is to explore data sources and processing methods that could be utitlized to support and improve this business process. More specifically, this project aims to find, compare and evaluate ways to deduce convenient pickup locations from publicly available data. Such data might be acquired from governmental Open Data portals, community-driven data platforms or commercial data services. The desired outcome is a proven, scalable method applicable for productive use.

Through broad exploration, four usable categories of data and twelve separate methods are identified. After comparison and evaluation, it is found that even the two most promising methods - using a curb data API and historical trip records - do not satisfy the formulated requirements. In turn, none of the proposed methods can be recommended for productive use. This negative outcome can be generally attributed to two factors. The first is a lack of data directly describing the curb and its allowed usages. The second is a lack of powerful tools to simplify the processing of digital material such as photos, videos or 3D scans of the curb. Both factors can potentially be compensated by technological advancement. Therefore it is concluded that better approaches will become available with time.

Contents

List of Abbreviations

API	Application Programming Interface
CCTV	Closed Circuit Television (video surveillance)
CRAN	Comprehensive R Archive Network
CSV	Comma-separated Values (a common file format)
DBSCAN	Density-based spatial clustering of applications with noise
DEM / DSM / DTM	Digital Elevation / Surface / Terrain Model
DOT	Department of Transportation
GIS	Geographic Information System
GTFS	General Transit Feed Specification
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
NACTO	National Association of City Transportation Officials
NASA	National Aeronautics and Space Administration
OECD	Organization for Economic Co-operation and Development
OSM	OpenStreetMap
TNC	Transportation Network Company
YOLO	You Only Look Once (Object Detection Framework)

Chapter 1

Introduction

In recent years, city dwellers have been witnessing a steady increase in the amount of transportation options they can chose from. Aside of conventional transport modes like privately owned cars, public transit and taxis, various forms of carsharing, bikesharing, carpooling and ridesharing solutions have emerged. Along with this diversification, the transportation industry is undergoing a wide wave of digitalization. Telephone centrals and fixed schedules are being replaced by demand-driven software solutions and intelligent prediction algorithms. In addition, smartphone apps are making it possible to buy tickets, reserve vehicles or request rides on the go, at any time. Among all these developments, the branch that arguably had the biggest impact on the daily commuting habits of people around the world is the so-called ride-hailing services (definition follows). In less than ten years, these services have managed to build globally renowned brands and did establish themselves as a valid alternative to taxis and even public transit [Schaller, 2018] [Fitzsimmons and Hu, 2017].

Unsurprisingly, the new business model brings along new challenges. The areas in which ride-hailing services are most popular and thus most successful are dense, urban centers with lots of people and traffic [Schaller, 2018]. There, the new fleets of service vehicles occupy already limited curb space, violate parking rules, interfere with traffic and cause congestions - leading to growing tensions with authorities [Rodriguez, 2017] [Fitzsimmons and Hu, 2017]. One way to reduce these negative impacts on traffic, and at the same time improve service quality, would be to ensure that drivers actually find suitable spots to pickup customers. This will be the focus of this research project.

The following sections provide a definition of the term ‘ride-hailing service’ and present the approaches used today to choose pickup locations. The shortcomings of these approaches then lead to the problem formulation and goal of this project.

1.1 Ride-hailing Services

Ride-hailing services - also called on-demand or real-time ridesharing services - are a relatively new class of mobility services that use modern technology to serve their customers. The International Transport Forum of the OECD[1] provides the following attribution and classification [ITF/OECD, 2018]:

"Ride services – both ride-hailing and on-demand micro-transit – represent a minority share of overall trips in all cities where they operate but are growing rapidly. Since their first appearance in 2009, these services have spread rapidly across the globe in an increasingly complex playing field of large regional and global actors. These services match supply and demand for rides in several very different contexts and with an increasingly diverse set of services ranging from upper-tier premium door-to-door services, to shared use of vehicles along semi-fixed routes. They depend on professional licensed drivers in some contexts (including taxi drivers) or are open to all drivers meeting a minimum set of qualifications in others."

Popular examples for ride-hailing services include Uber (United States), Lyft (United States), Didi Chuxing (China), Ola (India) and Grab (Southeast Asia). Companies that offer ride-hailing services are also called transportation network companies (TNCs).

1.2 Choosing Pickup Locations

Most ride-hailing services share the same core business process: A customer requests a ride, and if the request can be served, a vehicle is dispatched to pick up the customer and drive him or her to the chosen destination. One decision that has to be made in this process, is to which exact coordinate the vehicle is guided to pick up the customer. Described below are two strategies applied today for making this decision.

Common Approach

Most TNCs do send their drivers directly to the GPS location of the customers mobile phone, letting them figure out themselves where to stop and possibly wait for the customers. This is being done on the premise that there will be free curb space somewhere close to the current location of the customer. Problems arise when there is no free curb space available. In dense urban areas like city centers, due to the multitude of different parties competing for curb access, this does happen quite often. For reference, the In-

[1] Organisation for Economic Co-operation and Development `https://www.oecd.org`

ternational Transport Forum lists ten major uses that curbs are serving besides vehicle parking [ITF/OECD, 2018]:

- Pedestrian access to/from sidewalk
- Emergency vehicle access
- Access to public transport
- Goods delivery and pick-up
- Interface for cycling infrastructure
- Pick-up and drop-off of passengers from taxis, ride services, private vehicles
- Repair and maintenance work access (infrastructure, utilities)
- Waste management and surface water runoff
- Commercial activities (kiosks, restaurants, food trucks, cafés, etc.)
- Leisure, green and planted space (trees and parklets)

With the rise of ride-hailing services and online shopping, the competition for curb access is growing even further [ITF/OECD, 2018].

The negative consequences of this over-demand becomes visible in traffic violation statistics. In 2017, on a typical weekday in San Francisco, TNCs did perform more than 170,000 pickups, accounting for approximately 20 percent of traffic [Castiglione et al., 2017] – but for over 60 percent of traffic violations [Rodriguez, 2017]. The situation in other major North American[1] cities is similar. What these numbers seem to indicate is that the lack of available curb space, paired with the desire of customers to be picked up just where they are, is pushing drivers towards choosing risky - even illegal - pickup locations.

Virtual Stops

An option to circumvent the problem described above is to restrict pickups to a set of pre-selected locations, which are known to be legitimate and usually free. Because these spots basically serve the same purpose as bus stops - yet lacking any visible indication in the real world - these spots may be called *virtual stops*. The strength of this concept

[1] From [ITF/OECD, 2018]: "At present, evidence on ride service impacts has been essentially limited to North American markets where they have first been deployed and where they have been most popular."

is that it is simple to understand and implement. Coming up with a well-suited set of virtual stops for a given service area however, proved to be non-trivial[1].

A straight-forward approach to obtain an initial set of virtual stops is to inventorize real bus stops. In many cities, the locations of public transport stops are publicly available, often in the standardized GTFS (General Transit Feed Specification) format [Google, 2019] [2]. If the density of bus stops is not sufficient, or if the ride-hailing service is simply not granted permission to use them, alternative approaches are required though.

Another possibility is to place virtual stop locations based on two-dimensional maps of an area. The required map geometry data can be obtained from map databases like OpenStreetMap (see section 3.2). Unfortunately - as will be demonstrated later - these databases only provide limited insight into the curb situation on a street. Yet another possibility is to pick virtual stops in a fully manual process. This approach however does not scale well. The effort and budget required to repeat a manual process for each new area a service is expanding to will presumably largely offset the benefits of using a virtual stop based approach in the first place.

1.3 Motivation

In summary, neither of the approaches considered above does produce good pickup locations for ride-hailing. Instead, providing good pickup locations seems to require richer data and more sophisticated algorithms. Exploring such data sources and algorithms, and fusing them into software systems that autonomously locate good pickup locations, will be the goal of this research project.

[1] Based on practical experience from developing and operating a ride-hailing service at moovel.

[2] An extensive list can be found at `https://transitfeeds.com/feeds`

Chapter 2

Method

This chapter specifies the process that will be used to find and evaluate methods to locate good pickup locations.

In order to assess the quality of a given pickup location, it will be necessary to define criteria for ‘good’ locations. Consequently, in order to avoid ambiguity and allow precise diction, the first step will be to introduce and define the property ‘convenient’ as a classification for curbside locations.

The next step will be an exploration of available data and possible processing techniques. The goal of this step is not yet to find the best solutions, but to obtain a broad range of options that each can in one way or the other provide information about the curb and its usage. The result of this step should be a list of applicable methods. In this context, ‘method’ will always refers to a combination of a processing technique and a data source that provides the data to be processed (see figure 2.1).

The next step will be a comparison of the found methods, focusing on utility, complexity and universality. Through a comprehensible rating system that assigns points in each category and calculates overall scores for each method, the list will be brought into order. Based on that ranking, the most promising methods will be selected for further evaluation.

Next, the selected methods will be evaluated individually. This includes a prototypical implementation of each method that will be verified with real data. After the concept is

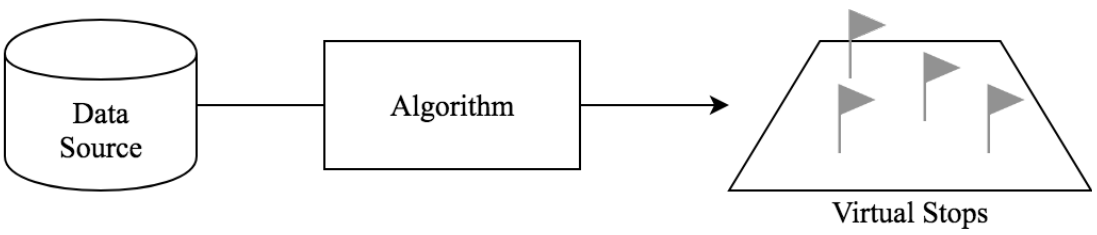

Figure 2.1: Schema of a resulting method

proved, the assumptions made in the previous step will be reviewed. In addition, each methods viability for real-world application will be assessed based on the results from the prototypical implementation.

In the last step, the results of the detailed evaluation will be discussed and compared against initial expectations. If a method proved useful and possesses all demanded qualities, it will be recommended for productive use. Otherwise, an analysis of the limiting factors will be made.

In summary, the process involves the following five steps:

1. Definition of Criteria for “Convenient” Pickup Locations
2. Collection of Possible Methods
3. Selection of Promising Methods
4. Evaluation of Selected Methods
5. Recommendation for Implementation

Chapter 3

Results

This chapter presents the results of the five-step process described above. Every section describes the implementation and outcome of the respective process step.

3.1 Definition of Criteria for "Convenient" Pickup Locations

To decide whether a spot at the curb is suitable for passenger loading, it is necessary to define a set of criteria to evaluate it against. If a pickup location fulfills these criteria, it will be called "convenient".

For the purpose of customer pickup, there are two aspects to consider in a locations: The spatial situation and the legal situation. Clearly, for a vehicle to stop and wait somewhere, an appropriate amount of free curb side space is required. But this alone is not sufficient. It also needs to be ensured that passenger pickup or stopping is prohibited at the considered location. Consequently, to allow a smooth and legal[1] pickup, suggested curb side locations should

1. physically allow on- and off-boarding
2. not be blocked by other curb side activities
3. be in a zone where stopping (for passenger pickup) is allowed
4. be in a zone where parking is forbidden

[1] For an argumentation on why it is important to respect parking rules, see section 1.2.

Declared parking zones are considered inconvenient because it must be assumed that all available space in these zones is occupied by parking cars. However, if there was data available that allowed to predict the usage of parking zones, or even obtain real-time usage data, this risk could be minimized.

In the following, the property "convenient" will be used synonymously for these four properties, and the quality of pickup locations will be measured by these criteria.

3.2 Collection of Possible Methods

The goal of this step is to explore data sources and processing techniques that can provide information about the convenience of curb locations. To come up with a broad set of options, a systematic approach is used. The approach consists of three phases:

- Review of literature, press articles and online content from transportation providers, industry experts, public institutions, etc. to learn about available data sources
- Categorization of data sources that can provide insight into the curb situation
- Enumeration of processing techniques that can be applied to the collected data sources

The following sections present the results of each phase.

Data Sources

Listed below are all data sources that were identified to potentially contain useful information. The data sources are grouped into categories according to the type of data U.S. contain. The four data types that are considered useful are rules data, coordinate data, shape data and visual data.

A. Rules Data

This category contains data sources that directly encode traffic rules. Usually, the rules are linked to geographic entities (i.e. street segments) to make them retrievable by address or coordinate. Data within this category can answer questions like 'Is it allowed to park on XYZ Street between 6th and 7th Avenue at 6pm on a saturday?' or 'On which curb in Downtown Los Angeles is it allowed to pick up passengers?'. It is easy to see that this kind of data can be directly used to verify three of the four requirements for convenience.

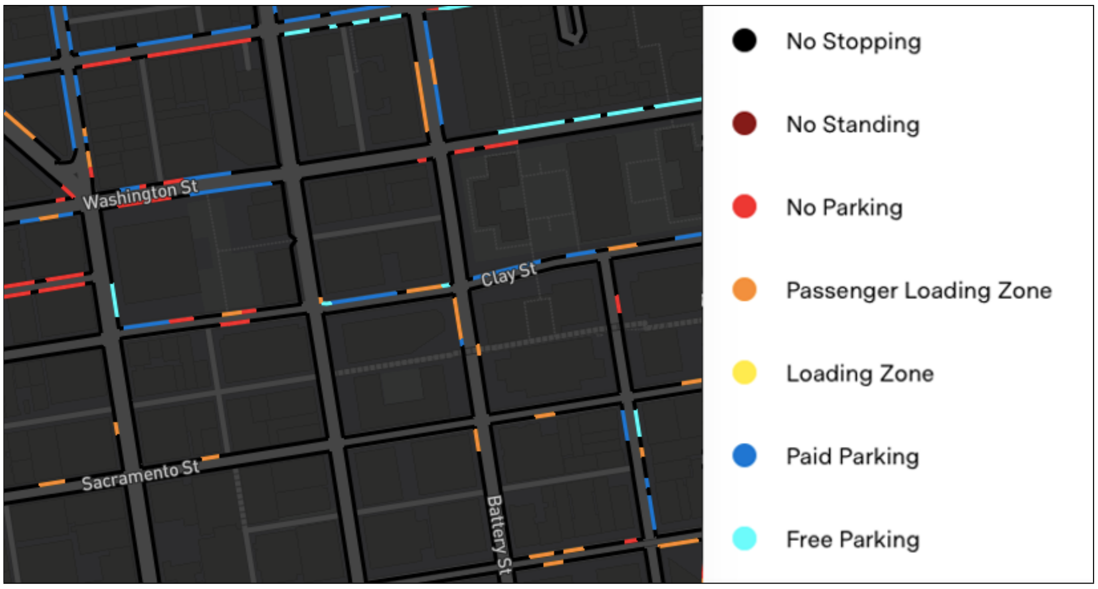

Figure 3.1: Rules data example: curb segments in San Francisco [Source: Coord]

However, such data is rarely available in digital form. [ITF/OECD, 2018] explains this situation as follows:

"Inventories of curb space allocation are often kept in multiple departments responsible for different uses (parking, traffic, fire and emergency services, freight delivery and logistics, bicycle and car-share, street vendors and parklets, utility maintenance, etc.). These inventories may be digital but are often paper-based and are infrequently updated. Inter-department use is not always easy and their use by the public and other stakeholders, not possible, or only very difficultly so."

Accordingly, there is no global data base for curb usage rules, and if data is available locally, it tends to be incomplete or fragmented. However, there are initiatives aiming to build consistent and comprehensive data bases.

Authoritative Rules Data

Many government bodies operate Open Data platforms that grant the public access to authoritative data sets [Kitchin, 2014]. All of these portals are potential sources for authoritative rules data. However, because of the large number of Open Data portals ([OpenDataSoft, 2019] lists over 2500), it is impossible to scan them all individually within the time frame of this project. Therefore the scope of the exploration was narrowed down to a manageable subset. The chosen approach was to select the region that generally offers the best access to the largest amount of Open Data. This was found to be the United States of America (hereafter U.S.). According to [OpenDataSoft, 2019], U.S. possesses the highest density of Open Data portals in the world (almost 1000). The U.S. government further operates a service called *Data.Gov*, which is a search engine for authoritative data sets published on local or regional Open Data platforms in the U.S. [U.S. General Services Administration, 2019]. It grants access to an catalog of over 300000 indexed and categorized data sets, thus speeding up the search process immensely. A review of this catalog, together with selected municipal Open Data portals, produced following findings:

Overall it stands out that almost all relevant data sets originate from a small group of highly developed cities - namely Seattle, San Francisco, Los Angeles, New York and Chicago. Among them, the only city that provides comprehensive rules data for all its curbs is Seattle [City of Seattle GIS Program, 2018][1]. Apart from that, the only other notable instance of curb rule data is an inventory of all commercial loading zones in Washington DC [U.S. DOT, 2017]. More commonly found were data sets of parking zones and parking meters [San Francisco, 2018] [Chicago, 2019] [Los Angeles DOT, 2019a]. However, these data sets generally do not designate stopping or loading zones. Another source of potentially useful information are data sets featuring traffic signs, especially parking signs. Out of the Data.gov catalog, 17 data sets are tagged as 'signs'[2], among them a data set of street signs in Seattle, a data set of parking regulation locations and signs in New York, as well as a collection of pictures of parking signs in San Francisco. For smaller cities, virtually no data was found.

During the review process it became clear that the availability of authoritative curb usage rule data is fully dependent on local digitalization and Open Data efforts of individual city departments. Further it was found that there is no standard format for any of the described data types (see also [Baskin, 2019]). Because of that, data quality may vary from data set to data set. Note also that the findings mentioned above represent the total amount of usable data for the whole U.S. - a region that was chosen because of its rich Open Data landscape. In other regions of the world, the supply of data might be even more limited.

Third-Party Rules Data

In the 2018 report of the International Transport Forum, the authors compiled a list of private companies and initiatives that work on digitalizing curb space. The list only features two entries [ITF/OECD, 2018]:

SharedStreets (`https://www.sharedstreets.io`)

[1] Interactive map: `http://web6.seattle.gov/SDOT/seattleparkingmap/`

[2] Number may vary over time, compare `https://catalog.data.gov/dataset?tags=signs`

SharedStreets is a project by NACTO (National Association of City Transportation Officials) and the Open Transport Partnership. Its goal is to develop a data standard and a platform to share transportation and mobility data between public and private sectors. As of this writing, the project has only published a draft standard, and does not yet offer a platform to publish or access data.

Coord (`https://coord.co`)

The Coord platform is developed by Sidewalk Labs, a company owned by Alphabet Inc. The platform collects information regarding new mobility services as well as curb infrastructure and use:

"Coord builds its inventory of curb uses by parsing a massive and continually updated dataset of digital photographs (collected by Coord's parent company) of parking signs and other curb-adjacent signs and objects. Integrating this real-time map of street and curb regulations into back-end code can help ensure that mobility services are fully and automatically compliant with these rules. At present, Coord provides API access to curb-related data regarding freight loading and passenger pick-up zones, street parking rules, bus stops, location of taxi stations and more. It currently has developed curb inventories for New York City, Seattle, San Francisco and Los Angeles. Future plans include integrating bicycle share referencing and automated access to tolling and parking prices (with an eye to also providing a back-end to support automatic transaction clearing for toll and parking payments)." [ITF/OECD, 2018]

B. Coordinate Data

This category contains data sets of annotated geo-coordinates.

Vehicle Position Data

One way of finding convenient pickup locations is to look at historical data. If a spot on the curb has been used for passenger pickups before, it might as well be a good candidate for future pickups. If a spot is used frequently, it might be more convenient than others. If a spot is regularly used for parking however, it might be located in a parking zone, or at least on a curb segment which is used for parking. In this case, it will probably not be a good location for passenger pickup.

Potential sources for such vehicle position data are software-driven companies in the mobility and transport sector, such as taxi, car-sharing or delivery services. By tracking and collecting the locations of pickups, drop offs, delivery stops or parking spots, these companies can understand and optimize how their business processes. This however,

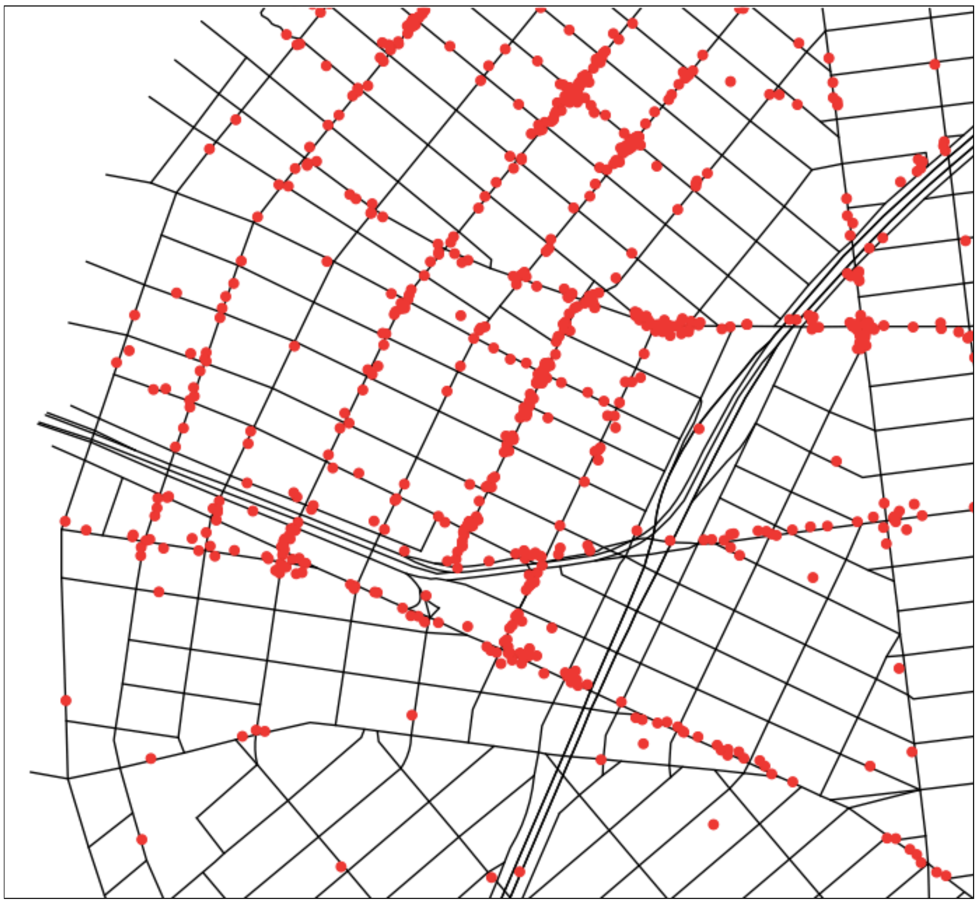

Figure 3.2: Coordinate data example: taxi pickups locations in Brooklyn, NY

makes this kind of data valuable and sometimes even a business secret, which must not get into the hands of the competition. For these reasons, the chances that vehicle position data from private companies is publicly available are low.

One notable public example of coordinate data is a data set collected and made available by the New York City Taxi and Limousine Commission on taxi and ride service trips [NYC Taxi and Limousine Commission, 2019]. This data set contains pickup and drop-off locations for all taxi rides operated by this commission[1].

One often referenced data set featuring trip records from a private company is a public data set of Uber trips in New York in 2014 and 2015, obtained by a political news website through a Freedom of Information Law request against Uber [FiveThirtyEight, 2019].

C. Shape Data

This category contains data that describes the geometry of the real world. Information that can be extracted from such data sets are for example the courses of roads, the locations of intersections or the shapes of buildings. If the geometry data is detailed enough, it might even be possible to derive the shape of curb lines and zones. Shape data can be divided into two subcategories. Modelled data which is manually created to model the shape of the real world, and measured data which is captured directly from the real world through surveying. It is to be noted that there is no hard border between both categories, as modelled data must always be based on measurements and raw measured data might get post-processed.

Modelled Shape Data

Modelled data can be obtained from map databases. These databases usually only describe infrequently changing entities like streets and buildings. The most popular provider for map data is OpenStreetMap (OSM). OpenStreetMap is an open, crowd-sourced map database that covers most of the populated surface of the earth and is maintained by private and corporate contributors. It offer two-dimensional as well as three-dimensional models [OpenStreetMap Foundation, 2019].

Measured Shape Data

What modelled shape data cannot provide is the actual curb usage situation of a street segment. This information can only be obtained from three-dimensional snapshots of that

[1] At least until June 2016, later data only specifies the origin and destination zone.

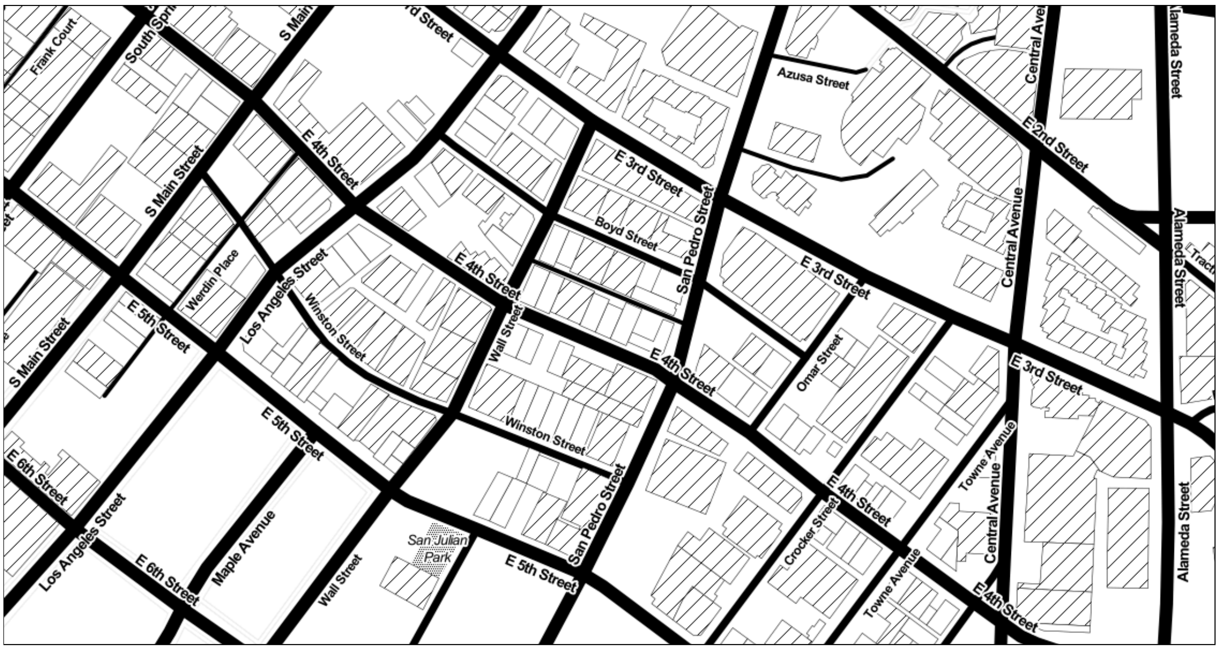

Figure 3.3: Shape data example: streets and buildings in downtown Los Angeles [Source: Stamen/OSM]

segment. Such snapshots are created and used in surveying, where U.S. are referred to as Digital Elevation Models (DEM), Digital Surface Models (DSM) or Digital Terrain Model (DTM). Today these models are usually captured by planes or satellites equipped with measuring devices, and can achieve resolutions of up to one meter [Wilson, 2012]. Digital elevation models can be obtained from Open Data platforms, from satellite operators such as the NASA, or from community platforms like OpenTopography.

D. Visual Data

This category contains visual depictions of the real world.

Street-Level Visual Data

Any image or video depicting a street can be analyzed for curb space availability and usage. Sources for such material are dash cams, CCTV cameras, Google Street View, Mapillary[1] or any other collection of published pictures taken on street-level.

[1] Mapillary is a platform for sharing crowdsourced geotagged photos.

Top-down Visual Data

Aside from street-level imagery, high-resolution aerial or satellite images also contain information about the layout and usage of curbs. Such images can be obtained from commercial sources like Google Earth and DigitalGlobe, or from community sources like ArcGIS or the U.S. Geological Survey Earth Explorer [Clancy, 2014].

Processing techniques

Listed below are all identified processing techniques, sorted by the category of data U.S. operate on.

A. Rules Data

In the best case, rule data does not require specific processing techniques, as it can be used as is to determine what zone a given spot is located in or to find the closest curb segment that allows passenger pickup. All that is needed is a database that does support respective queries. If only parking rule data is available, this information can be used to mark certain zones or location as not convenient.

More sophisticated techniques are required if only traffic sign data is available. If annotated sufficiently, this data can be used to derive locations and even shapes of non-parking zones. The challenge of this scenario is to determine which traffic sign and rule actually is in effect at a certain curb coordinate. For example, in the U.S., a sign is in effect from the location of the sign post to the location of the next sign post with a different sign, or to the end of the curb. In Europe, parking and stopping signs are often used in combination with markings on the street that delimit their actual area of effect. Further, especially in the U.S., the parking rules expressed by a single sign can be quite complex and signs are often annotated with special conditions or exemptions (compare [Seattle DOT, 2019] [Los Angeles DOT, 2019b]). Because of this complexity and heterogeneity, all of the reviewed data sets stored the traffic rules themselves in freeform text columns. Parsing such data requires natural language processing techniques and knowledge of the local regulations and systems.

The concrete methods found to process rules data are:

A1: Find curb segments where passenger loading is allowed, but parking not

A2: Derive curb usage rules from traffic sign data sets, then perform A1

A3: Extract parking zones from public parking spot database

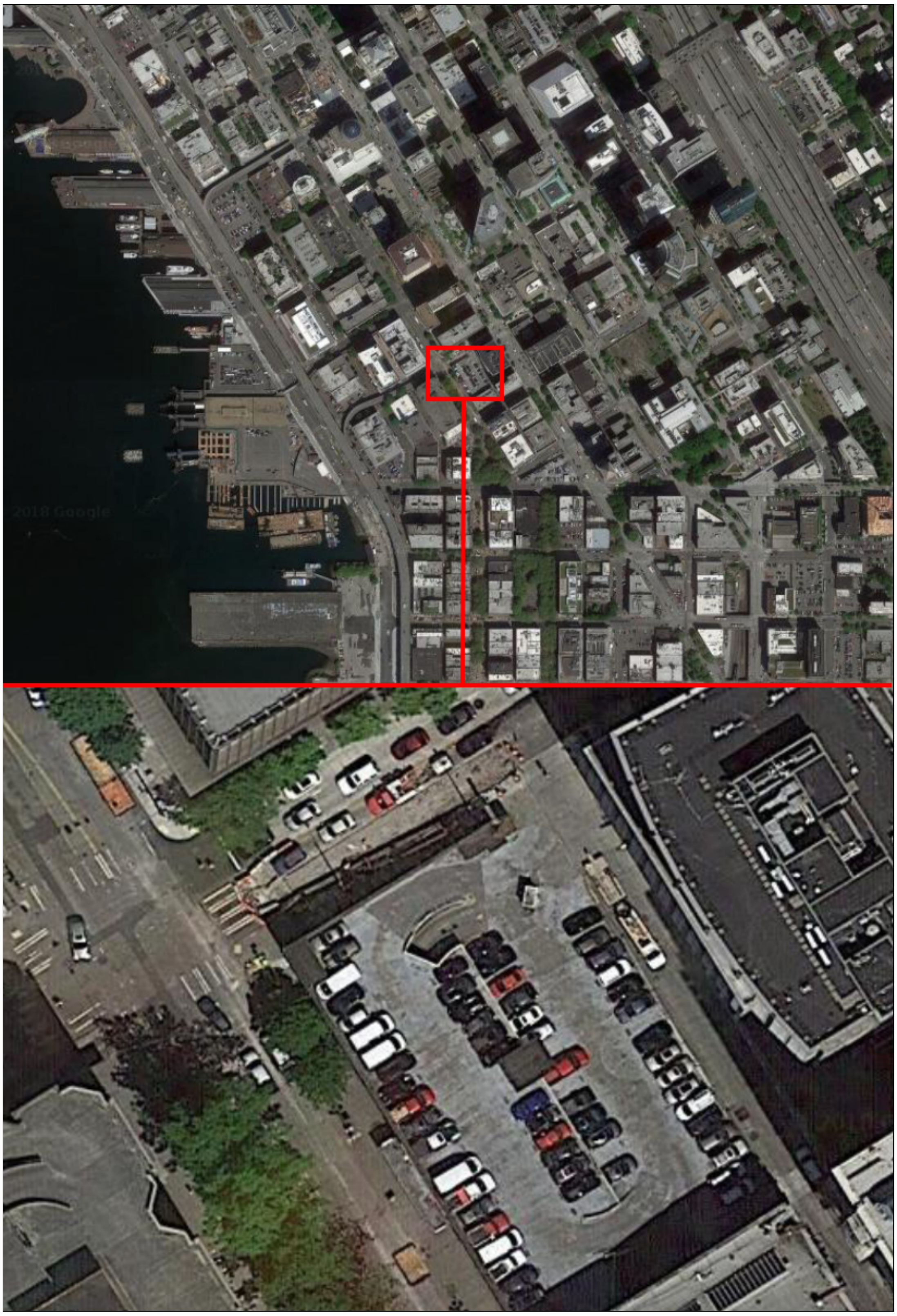

Figure 3.4: Visual data example: satellite picture of downtown Seattle [Source: Google]

B. Coordinate Data

Every entry in a vehicle position data set represents a historic location of a single vehicle. To obtain information about the convenience of specific spots or zones, the data needs to be filtered for relevant events and then condensed by using some sort of clustering technique.

The concrete methods found to process coordinate data are:

B1: Use cluster analysis of historical pickups from other mobility services to extract pickup zones

B2: Use cluster analysis of parcel delivery vehicle stop data to extract drop-off / stopping zones

B3: Use cluster analysis of parking positions of free-floating carsharing vehicles to extract parking zones

C. Shape Data

Maps describe the geometry of roads, intersections and sidewalks. Through geometrical transformation and interpolations, it is possible to derive the location and shape of curb zones from this data.

Measured shape data represents a snapshot of the real world at a certain point in time. If the resolution is high enough, such a snapshot can be used to identify blocked and non-blocked curb space. To make assumptions about the usage and availability of curb space over time - or at a certain point in the future - it is necessary to analyze multiple snapshots and use prediction models to find regularities.

The concrete methods found to process shape data are:

C1: Extract curb zones from map geometry

C2: Use machine learning to find parking patterns in 3D models of a street

D. Visual Data

Techniques used to extract information from visual data are studied in the field of Computer Vision. The discipline that explores methods to detect objects in images is called Object Detection [Prasad, 2012]. Techniques from this discipline can be applied on both static and moving images.

Street-level footage can be scanned for all the information that a real driver would use to find a pickup location: traffic signs, street markings, parking cars and free curb spots.

In aerial images, traffic signs are usually not visible. However, the curb usage situation can also be analyzed from a top-down perspective. [ElMikaty and Stathaki, 2017] for instance describes an accurate detection model for cars in high-resolution aerial images.

The concrete methods found to process visual data are:

D1: Detect curb usage related traffic signs in street-level images

D2: Find free curb spots on street-level footage

D3: Identify parking zones on top-down images

D4: Collect information about parking habits from top-down images

3.3 Selection of Promising Methods

The goal of this step it to put the methods described above (A1 through D4) into relation with one another. This is done by assessing the potential and limitations of each method, and then assigning scores based on selected properties. Through the scores it becomes possible to select the methods which appear most promising for further evaluation.

In order to compare and rate the individual approaches, it is necessary to establish a set of comparison criteria. These criteria should reflect the requirements imposed by the scenario this research project is working towards: the development of a productive virtual stops generator for a scalable ride-hailing service. For a solution to be selected for production, it should be economical, meaning that it should not require exceeding amounts of money and resources for implementation or operation. It should also reliably produce virtual stops, and the produced stops should be convenient to use for drivers and customers. Finally, the solution should be scalable, meaning that once implemented, it should be reusable - with as little adjustments as possible - for every new operating area that the service expands to. These requirements can be reduced to the following three comparison criteria:

- Utility
- Complexity
- Universality

To avoid repetition, the common properties of the four different data categories will be examined first, followed by a discussion of the individual methods. It has to be noted that at this point, there is no empirical data available about any of the methods. Hence, the following assumptions are based purely on reasoning.

In terms of processing effort, methods that use rules data, especially curb usage rules, promise the least overhead. The utilized data already contains direct information about the convenience of a given curb segment. Therefore no additional processing steps to derive this information are required. However, authoritative rules data is rare, and lacks a common format. Therefore, exclusively relying on authoritative rules data will most certainly require manual data collection and preprocessing efforts for each new operating area. There is also a risk of not finding any usable at all. This limitation can be circumvented by the use of non-authoritative data provided by third parties. In practice, the only source that was found to provide such data is Coord. At the time of writing, their Curb Search API (partly) covered four American cities: Los Angeles, New York City, San Francisco and Seattle [Coord, 2019a]. This makes non-authoritative rules data just as rare as authoritative data.

The other data categories require more elaborate processing techniques. Among them, approaches based on coordinate data are most predictable, because U.S. employ well-understood algorithms, namely cluster analysis. Efficient implementations of such algorithms are freely available for many languages and platforms[1]. It can therefore be assumed that the implementation effort for methods using coordinate data will be reasonably small. However, most data sets in this category are collected and owned by private companies, which usually will not share their data. Hence, the total number of usable data sets in the category is estimated to be small. The number of sources that can provide such data for multiple cities around the world must be expected to be even smaller, as there are only a few companies that operate fleets on a global scale.

For shape data, the availability is less problematic. Google Maps and OpenStreetMap both virtually cover the complete surface of the Earth, and their data is updated regularly. However, the effort required to extract anything beyond basic curb geometry from shape data is extremely high. The information that can be extracted from modelled data does provide no notion of convenience, and is therefore only usable in conjunction with other data. Gaining insight into the usage of curbs requires an analysis of measured data. The amount of extractable information within such data varies with age and resolution of the data. If the data is too old, the curb situation might have changed significantly in the meantime. If the resolution is too low to discern individual cars, a model cannot

[1] Compare for example `https://en.wikipedia.org/wiki/Hierarchical_clustering#Software`

provide any insight at all about curb usage patterns. The only widespread measurement technique that achieves sufficient resolutions (below 1m) is LIDAR [Carter et al., 2012], and finding recent high-resolution scans for a specific area is thought to be very unlikely [Fisher, 2018].

The processing effort for measured shape data is expected to be extremely high. Due to the way U.S. are created and used[1], DEMs usually come in big data files covering large areas. Locating and extracting the geometry of specific street sections from such files requires extensive pre-processing. Together with the machine learning models required to make sense of the geometry, and the automation required to apply these techniques on multiple data sources in order to find usage patterns, the resulting solution would become very complex and costly. Further, that solution would have to be tailored to an individual data set and would therefore lack universality. It is also not clear if such approaches can deliver usable results at all. Therefore, shape data based methods do not appear very promising in general.

For visual data, the processing effort is high as well. Extracting any useful information from images or videos requires Computer Vision techniques, such as Object Detection to recognize cars and traffic signs, or 3D Reconstruction to recreate depth from two-dimensional imagery. While there are frameworks that provide implementations and learning models for these techniques, the effort to prepare the data and to select, configure and train a model for the specific tasks is very high. A review of scientific work focusing on similar problems also suggests that the required models would become very complex and the expectable results not very reliable [Wang and Cheng-Yue, 2016] [Lee, 2009].

The availability of street-level visual data in general is good, however obtaining video footage of a specific street section will most probably involve manual steps. An advantage of using street-level imagery is that it is cheap and simple to create. Apart from that, static images covering complete networks of streets can be obtained from Google Street View or Mapillary. For video data, a comparable source that allows the retrieval of footage of a specific location could not be found. Therefore, the data has to be either captured manually (e.g. via dash cams), collected from local, fixed camera systems (e.g. traffic cameras, public CCTV) or scraped from video hosting platforms (like YouTube).

For top-down images, the situation is different again. Aerial and satellite images are widely available for every city in the world. As for shape data, the limiting factor here is resolution. Among the reviewed material depicting cities or city district, none of the publicly available options offered sufficient resolutions to identify objects in the size of cars.

[1] DEMs are traditionally used to survey the geologic, geographic or demographic features of a region.

The utility of visual data based methods is completely dependent on the accuracy of the applied technique. In the best case, the locations of traffic signs derived from street-level footage might be exact enough to recreate the shape of passenger loading zones. The probability of achieving such good results is expected to be minimal though. If it is possible to locate parkings zones, the results are not directly usable, but can still contribute to more convenient pickup locations by avoiding potentially blocked curb segments. If the resulting curb zones are only based on recurrence patterns in a small amount of snapshots, derived suggestions must be regarded more as a hint than a guarantee. In the worst case, when only few static images of a distinct area are available, or the resolution of the data is simply too low, or the material is too old, derived suggestions will represent mere guesses and can deliver no additional use for real-world operation.

Based on these assumptions about the data situation in general and the individual techniques specifically, each methods has been assigned points for utility, complexity and universality. Table 3.1 shows all scores, as well as the resulting ranks. For each of the three criteria, ten points is the highest achievable score, while zero points is the lowest. Consequently, a score of ten in *Complexity* would indicate that a method is extremely simple, while a score of ten in *Utility* would be rewarded to a method that generates virtual stops which are guaranteed to be free and legitimate at the time of pickup. In the *Score* column, the three individual scores are summed up using the factors specified in the second row. Complexity is weighted heavier because it has direct impact on service quality and customer experience. Universality, although being important for scalability, is weighted lighter because a solution that will only work well in limited regions is still valuable. Also, universality can in many cases been compensated by manual steps, which is not optimal, but might still be more economical than developing a very complex universal solution. The methods **A1** and **B1** are marked with an asterisk because U.S. achieved the highest scores. Both will be evaluated in detail in the next section.

For reference, all methods are listed again below:

A1: Find curb segments where passenger loading is allowed, but parking not

A2: Derive curb usage rules from traffic sign database, then perform A1

A3: Extract parking zones from public parking spot database

B1: Use cluster analysis of historical pickups from other mobility services to extract pickup zones

B2: Use cluster analysis of parcel delivery vehicle stop data to extract drop-off / stopping zones

	Utility	Complexity	Universality	Score	Rank
Factor	5	4	3		
A1*	9	9	1	84	1
A2	7	6	3	68	6
A3	5	8	5	72	4.5
B1*	7	8	5	82	2
B2	7	8	4	79	3
B3	5	7	3	62	10
C1	2	8	10	72	4.5
C2	4	0	2	26	12
D1	7	2	8	67	7
D2	4	3	6	50	11
D3	5	3	9	64	8
D4	4	4	9	63	9

Table 3.1: Method Comparison

B3: Use cluster analysis of parking positions of free-floating carsharing vehicles to extract parking zones

C1: Extract curb zones from map geometry

C2: Use machine learning to find parking patterns in 3D models of a street

D1: Detect curb usage related traffic signs in street-level images

D2: Find free curb spots on street-level footage

D3: Identify parking zones on top-down images

D4: Collect information about parking habits from top-down images

3.4 Evaluation of Selected Methods

The goal of this step is an in-depth evaluation of the methods that performed best in the previous comparison. The basis for this evaluation is a prototypical implementation operating on real data. Each section first presents the design decisions and the resulting software architecture used to implement the respective method, followed by a critical review of the achieved results and the effort involved. Each section closes with a re-assessment of the methods utility, complexity and universality, based on the insights gained from implementing and testing the method.

Method A1

The approach for this method is to find curb segments where stopping and passenger loading is allowed, but parking is not. The method requires a data source that allows locating curb segments around a given coordinate, as well as retrieving curb usage rules for these segments. The Curb Search API provided by Coord, introduced in section 3.2, is the only known data source that can provide such information for more than one area. It also uses a standardized format for all areas. Therefore this data source promises the best scalability and least complexity among the considered options, and was thus chosen for implementation.

Figure 2.1 described a method as a component that takes a batch of data and derives a set of virtual stops from it for further use. However, if the algorithm used to produce virtual stops is efficient enough, it is also possible to derive virtual stops just in time, i.e. at the moment when a customer requests a ride. This approach, subsequently referred to as *dynamic stop generation*, has two advantages. First, in contrast to pre-calculated stops which have to be adjusted whenever curb rules change, dynamically generated stops are always derived from the latest data. Also, when generating stops dynamically, the algorithm can pick the coordinate within the selected curb which is closest to the customers position. Of course, this optimization only makes a significant difference for longer curb segments.

Because this method uses rules data, deriving virtual stops is a relatively simple process:

1. Find the curb closest to the user location
2. Chose a coordinate within that curb as virtual stop

If the geometric shapes of all eligible curbs are available, and the database queries to find the curbs closest to a target location are sufficiently fast, dynamic stop generation can be applied. A test with curb data from Los Angeles showed that these requirements can be met. Therefore the prototypical implementation was designed accordingly.

The proposed software architecture consists of three components. The first component is a script that retrieves the shapes of all eligible curbs within a chosen area from the Curb Search API. The downloaded data is being stored in a relational database with support for geo-spatial queries, which is the second component. The third component is a web service that dynamically creates virtual stops for a given coordinate based on the data in the database. The architecture is visualized in figure 3.5.

At the center of the architecture sits the database. It holds only one table, the table of curb shapes. The schema of this table needs to allow the fast retrieval of all curbs within a certain distance of a chosen coordinate. On order to do so, the used database

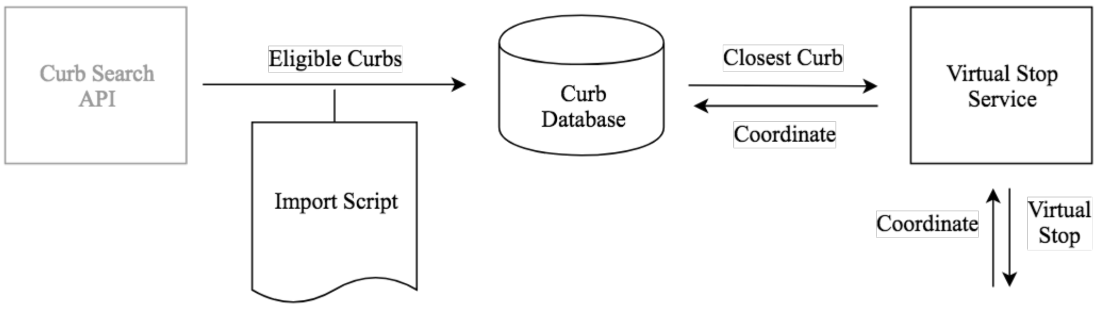

Figure 3.5: Overall architecture for method A1

management system has to support indices over geo-spatial shape columns. The database management system that was selected for implementation is PostgreSQL, because it provides good performance and scalability and can be extended to support geo-spatial queries and indexing. The resulting schema can be found in appendix A.

The import script needs to fetch all eligible curbs within a chosen area and then store them in the aforementioned database table. The source of the curbs data it the Curbs Search API. This API offers an endpoint that allows the retrieval of rules by bounding box. In addition to the API key and the corner coordinates of the bounding box, the endpoint accepts three parameters that allow a pre-filtering of the returned data: *primary use*, *permitted use* and *vehicle type*. To set these parameters appropriately, it is important to understand the data model that the API uses. The following paragraph, taken from Coords documentation of the Curb Search API, describes how the response data is structured [Coord, 2019b]:

“Every segment of a curb has, at any given time, a primary use and permitted uses. The primary use is what the regulator has defined as the desired use of the curb at that time. The permitted uses comprise everything that is allowed, including the primary use if any. We distinguish these so that we can tell apart areas signed, say, ‘PASSENGER LOADING ZONE’ from those signed ‘NO STANDING’ (which may also allow passenger pick-up and drop-off).A segment of curb can also have a vehicle type. In this case, only vehicles of this type are allowed to perform the permitted uses on this segment, though other vehicles may have a smaller list of permitted uses.”

The API documentation then specifies a list of permitted uses and vehicle types that are available. Of the four permitted uses (*park*, *load goods*, *load passengers* and *none*) only the option *load passengers* is of interest. Since it also includes the ridesharing use case, the correct vehicle type to choose is *taxi*.

The shell command used to perform the HTTP request, as well as a sample JSON response are provided in appendix A. For reasons of simplicity and portability, the script was implemented as bash script. It is executed manually to fill or update the database. According to information provided by Coord, it is sufficient to update the retrieved data only a few times a year, because the data rarely changes[1]. Because of this, no periodic scheduling was implemented.

The web service provides a single endpoint that accepts a location coordinate and returns a virtual stops if possible. On receiving a request, the service queries the database for the curb segment closest to the given coordinate. If such a segment is found, a coordinate within that segment is returned as virtual stop. The main configuration parameter of this service is the distance limit between the provided coordinate and curb segments. When choosing a value for this limit, two aspects have to be considered. If the chosen distance is too long, the customer might have to walk a lot until he or she reaches the pickup location. Especially in areas where TNCs like Uber and Lyft are operating, the willingness of customers to walk more than a block in order to get a ride is said to be very low[2]. However, if the chosen distance is too short, the query might not return any curb segment at all, because no eligible curbs are close enough. This factor is especially critical in dense metropolitan areas, where the already limited curb space has to be distributed amongst various different needs (compare section 1.2). Based of these considerations, the distance limit for the prototypical implementation with curb data from Los Angeles was set to 100 meters. The web service was implemented as a REST service in TypeScript, using Node.js as JavaScript engine and express.js as web framework. For reasons of brevity, the implementation itself is not provided.

Following insights were gained from implementing and testing the described architecture:

As expected for a rules data based approach, the implementation effort was comparatively small. After an initial exploration of the Curb Search API, the proposed architecture could be implemented within one week. The import script was realized in under 100 lines of code and the TypeScript code that queries the database and returns virtual stops spans about 200 lines of code[3]. The time to generate a virtual stop for a coordinate is in the range of milliseconds and downloading and importing all eligible curbs for covered areas of Los Angeles is possible in less than a minute.

In section 3.3, it was argued that the usability of methods based on rules data is very high, because U.S. produce certain results. However, when applied to real data, the method did perform poorly. If the web service returned a virtual stop, that stop did fulfill all convenience criteria. In downtown Los Angeles however, with the recommended distance limit of 100 meters, it proved to be almost impossible to find eligible curbs at

[1] According to verbal information provided by a Coord employee, the change rate of curb usage rules is approx. 3% per year, based on surveys U.S. conducted with city planning departments.

[2] This statement has been made by people within the Los Angeles Department of Transportation, a collaboration partner of moovel. It is not based on scientific evidence.

[3] Approximated number, as this code is embedded in a larger web service implementation.

all[1]. Considering these results, the estimated usability for productive use needs to be corrected downwards significantly.

Another major drawback that confirmed itself is the limited scalability of rules based approaches. At the time of writing, the Curb Search API covered only four North-American cities [Coord, 2019a]. It is to be noted that Coord does encourage partners to contribute data themselves by manually surveying curb segments using a provided smartphone app. However, for the purpose of building a scalable ride hailing service, this option does not provide any advantage compared to manually creating the virtual stops in the first place. Neither is it desirable to depend on the data acquisition efforts of a third, privately operated company in order to scale to new operating areas.

It can be concluded that this method is simple to implement, but also provides very limited usability and scalability, at least at the time of writing.

Method B1

This method performs a cluster analysis on historical pickup locations from other mobility services to extract pickup zones. The precondition for this method is the availability of sufficiently large historical data sets for the respective area. The taxi trip data set provided by the New York Taxi and Limousine Commission (mentioned in section 3.2) contains historical records for millions of taxi trips in the Manhattan district in New York (among other data) [NYC Taxi and Limousine Commission, 2019]. It therefore allows a prototypical implementation of location clustering for the Manhattan area.

The architectural pattern for this method is a data pipeline that transforms a list of coordinates representing historical pickup locations (provided in tabular form) into a list of coordinates representing recommended virtual stops (see figure 3.6). The primary design decision for this pipeline is the choice of clustering algorithm and its configuration. The algorithm should locate relatively small and dense clusters in a two-dimensional cloud of data points. A dense cluster suggests that the corresponding curb spot is frequently used for pickups. By allowing only small clusters, accidental merging of multiple curb spots can be avoided. One algorithm that identifies such clusters is called DBSCAN, which stands for *density-based spatial clustering of applications with noise* [Ester et al., 1996]. Given a set of two-dimensional data points, DBSCAN finds and groups points that are closely packed together, i.e. have many nearby neighbors. DBSCAN takes two parameters: the maximum distance[2] between points in a cluster and the minimum number of points

[1] Of the first 1000 requests forwarded from productive systems, the prototypical implementation could serve mere three.

[2] Theoretically, most clustering algorithms can work with any measure of distance. In practice, imple-

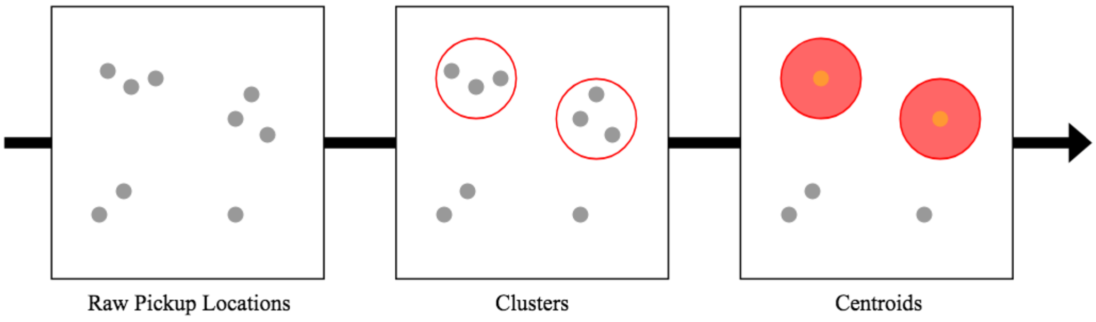

Figure 3.6: Data pipeline for method B1

required to form a cluster.

Reference values for these parameters can be estimated based on curb geometry and characteristics of the available data. The maximum distance between cluster points should be in the range of the length of an average parking spot within the operating area. If two coordinates are further apart than the length of 2-3 cars, U.S. probably do not represent the same spot on the curb. If the building density in the observed area is very high, the distance limit should be reduced further, because curb zoning in such areas tends to be very granular. The limit should also be adapted to the measuring accuracy of the coordinates (if known). If the provided location coordinates are inaccurate, the limit should be increased to account for measurement errors.

The minimum number of points required for a cluster (and therefore a virtual stop) should be chosen relative to the density of pickups in the data set. This density can be expressed as the quotient of the total number of trips and the size of the covered area. The minimum number of points should be a multiple of this density. The chosen multiplication factor can be understood as a confidence level that the clusters have to match. While the feasibility of physical on- and off-boarding (criteria 1) is already confirmed by a single pickup, frequent usages indicates that the spot is in a stopping zone and usually unblocked (criteria 2 and 4). The more often a stop is used in a fixed amount of time, to more likely it is convenient. (As demonstrated in section 1.2, frequent usage should not be used as an indicator for legitimacy of pickup stops (criteria 3)).

The technology used to implement the described design needs to be able to efficiently process and visualize big data sets. A common choice for such tasks is R, a software environment for statistical computing and graphics. The Comprehensive R Archive Network (CRAN) offers a vast collection of user-created packages, among them an efficient implementation of DBSCAN called *dbscan*:

mentations usually assume that the sample data is in Euclidean space and therefore use the Euclidean distance. This is also the case for the implementation that will be used here.

"This implementation of DBSCAN implements the original algorithm as described by Ester et al (1996). DBSCAN estimates the density around each data point by counting the number of points in a user-specified eps-neighborhood and applies a used-specified minPts thresholds to identify core, border and noise points. In a second step, core points are joined into a cluster if U.S. are density-reachable (i.e., there is a chain of core points where one falls inside the eps-neighborhood of the next). Finally, border points are assigned to clusters. The algorithm only needs parameters eps and minPts." [Hahsler, 2019]

This package was chosen to perform the central clustering step in the prototypical implementation. The algorithm was parameterized based on the used data. The New York City taxi trip data is provided in CSV format, one file for each taxi service and month [NYC Taxi and Limousine Commission, 2019]. For June 2016, the file[1] with trips in the Manhattan area (59 km^2) contains approx. 11 million rows, resulting in a little less than 0.2 pickups per square meter. Taking into account the high density of pickup locations in the data set, and the fact that downtown Manhattan has one of the highest building densities in the world, the chosen parameters were minPts=25 and eps=3m.

The DBSCAN algorithm produces a partitioning of the input coordinates. Each coordinate is either assigned to a cluster, or remains unclustered (these coordinates are called 'noise'). This partitioning can be visualized on a map using colors. Figure 3.7 shows the plot of a 3 km^2 section of Upper Manhattan, with orange dots indicating clustered coordinates and small grey dots indicating noise. It can be seen that the method successfully identified several clusters. The centroids of these clusters can be used as virtual stops. The plot was generated using Rs default plotting library and water body outlines from the U.S. Census Bureau for reference [Walker, 2019].

These results proof that applying cluster analysis to historic vehicle position data can deliver useful results. The implemented only works for the specific format of the data sets provided by the New York Taxi abd Limousine Commission though. It cannot be directly applied to differently structured data sets. In the best case, a preprocessing step that adapts the data to the implementation can solve this. In the worst case, the implementation itself needs to adjusted. Nonetheless does the prototypical implementation allow a better estimation of the overall complexity and usability of the approach.

The implementation effort for for the data pipeline was extremely small. The pipeline was developed over the course of two days and processing of the complete sample data set with over 11 million trips took less than 30 minutes on average consumer hardware. The R script that calculates the clusters and centroids is less than 50 lines of code (see appendix B). The resulting list of 4710 coordinates was directly usable as an initial set

[1] `https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2016-06.csv`

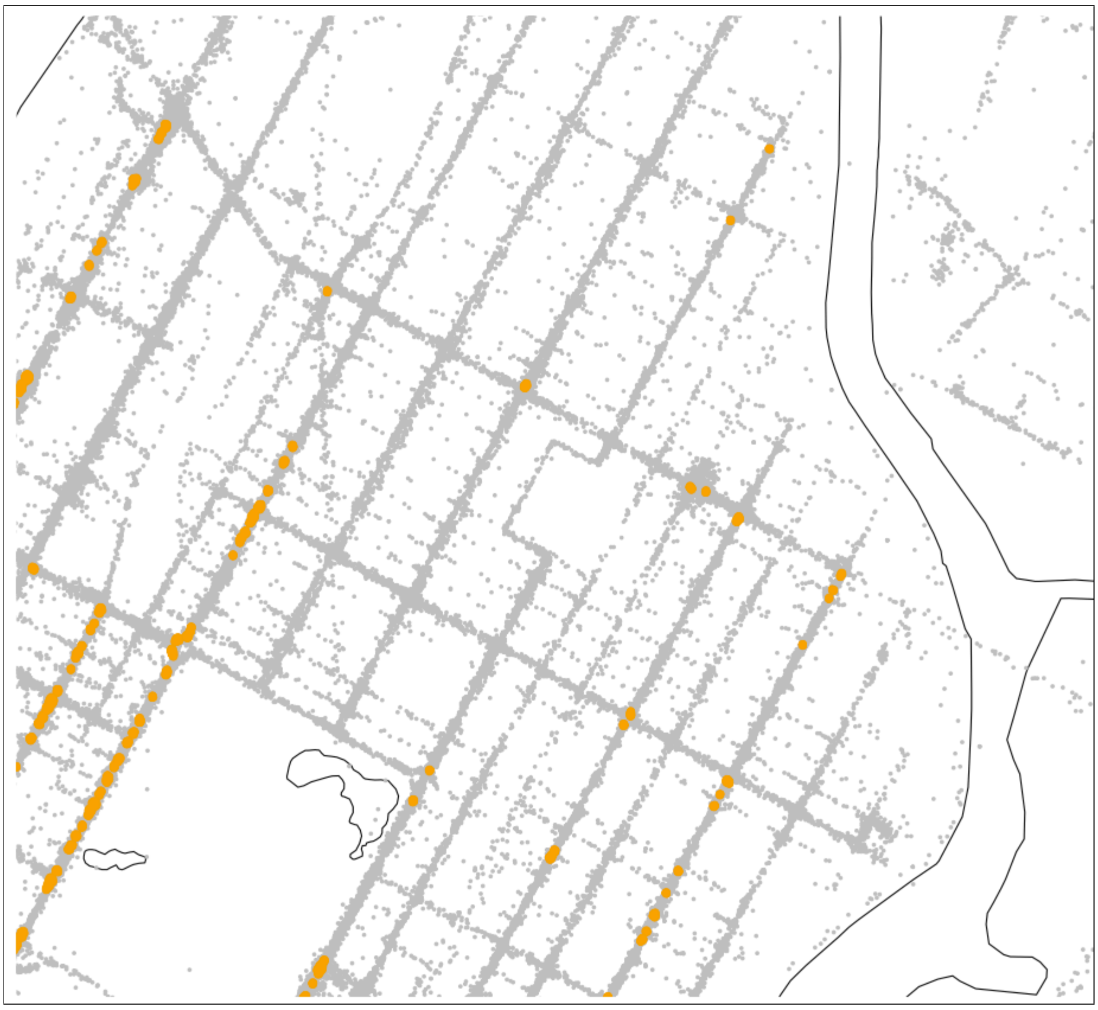
Figure 3.7: Clustering of pickup locations in Upper Manhattan, New York

of virtual stops. With the chosen minimum of 25 individual pickups per cluster, the confidence that the generated virtual stops fulfill the convenience criteria is high.

It can be concluded that this method provides good usability with very low complexity, yet cannot be applied universally but requires manual adjustment for every new data set.

3.5 Recommendation for Implementation

The intended process for this research project was to first find several different methods to generate virtual stops, then to select the most promising among them and finally to thoroughly evaluate those, in order to find the ones most suitable for productive use. Inconveniently, neither of the two methods selected for in-depth evaluation proved to be useful and universal enough to qualify for recommendation. Method A1 is limited by the coverage of the utilized API and the scarcity of eligible curb space. Method B1 - while delivering good results - fully depends on the availability of very specific data sources for the exact regions the ride hailing service is supposed to operate in. In turn, neither of these methods will work well for the scenario of an expanding ride-hailing service. Therefore, this research project does not recommend any of the evaluated methods for standalone implementation. It rather shows that, with the currently available data, there exists no generally satisfactory approach to automatically generate virtual stops. Yet, when manual preparation is an option, the achieved results can serve as a starting point for choosing the approach that will work best for a specific area and situation.

Chapter 4

Conclusion

The purpose of this research project was to identify and evaluate data-driven methods to locate convenient pickup locations. The desired outcome was a method proven to deliver usable results for various operating areas. To approach the objective systematically, a five-step process was outlined and implemented. The strategy was to start with a broad set of options and to condense these options until the most practicable method is found.

The result of this process however, was not a recommendation for implementation, but rather the insight that none of the twelve considered methods could satisfy all demanded requirements. Interestingly, for each of the three criteria - utility, complexity and universality - one or more methods could achieve high ratings. However, just like in the magic triangle known from project management, it seems to be impossible to fulfill all three requirements. None of the methods reached overall scores higher than 85, from 120 possible points. This result can partially be explained by looking at the available categories of data. The most useful category, rules data, is also the most rare. The categories in which data is widely available, on the other hand, provides the least insights into the situation that is to be analyzed (namely modelled shape data and static visual data). Both of these observations imply measures that could be taken to improve this situation and allow better results in the future.

The first measure is to improve the availability of curb usage rules. This would require a systematic inventorization and digitalization of curb space and curb rules by government offices, private companies or collaborative communities. If in the future, at some point, all major cities possessed comprehensive curb zone databases, ride hailing services and various other road users could operate more efficiently. The same could be achieved if global mapping communities like OpenStreetMap would start to annotate already existing data with curb zone information, or if businesses would find a way to monetize the collection and maintenance of such data. For all three scenarios, the foundations are being layed already. However, they all require time as well as an increased awareness of the benefits that a digitalized curb could provide for society.

The second measure is to develop more efficient means to comprehend digital representations of the real world. Algorithmically extracting information from such data requires an ‘understanding’ of what the sensor - be it a camera, a radar or a LIDAR - was looking at when capturing the data. Exactly this ‘understanding’ is the declared mission of Computer Vision, yet the tools available today are mostly limited to a subset of the information that is available in the data (e.g. faces, motion, geometry, or a QR code). In contrast to that, imagine a framework that takes a short dash cam video and automatically produces an accurate three-dimensional model of the scene. Within that scene, it automatically detects and classifies all recognizable objects using both the color information from the original video and shape data from the generated three-dimensional model. If such technology was freely available, the complexity of methods like D2 (finding free curb spots in street-level footage) and C2 (finding parking patterns in measured shape data) would be significantly reduced. Similar to the first measure, these developments require time, as well as novel scientific methods. But research on such methods has already started, and when the technology becomes available, it will enable various new applications, not just in the mobility sector but for every aspect of digital life.

Bibliography

[Baskin, 2019] Baskin, J. (2019). Why there's no GTFS for curbs (yet). Retrieved from https://medium.com/coord/why-theres-no-gtfs-for-curbs-yet-fbb12d8e2209.

[Carter et al., 2012] Carter, J., Schmid, K., Waters, K., Betzhold, L., Hadley, B., Mataosky, R., and Halleran, J. (2012). Lidar 101: An introduction to lidar technology, data, and applications. *National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, Charleston, South Carolina*, 30.

[Castiglione et al., 2017] Castiglione, J., Chang, T., Cooper, D., Hobson, J., Logan, W., Young, E., et al. (2017). TNCs Today: A Profile of San Francisco Transportation Network Company Activity. *San Francisco County Transportation Authority (June 2017).*

[Chicago, 2019] Chicago (2019). Parking Permit Zones. Available from https://data.cityofchicago.org/Transportation/Parking-Permit-Zones/u9xt-hiju.

[City of Seattle GIS Program, 2018] City of Seattle GIS Program (2018). Blockface. Available from http://data-seattlecitygis.opendata.arcgis.com/datasets/blockface.

[Clancy, 2014] Clancy, M. (2014). High Resolution Aerial Imagery. Retrieved from http://www.elecdata.com/blog/high-resolution-aerial-imagery/.

[Coord, 2019a] Coord (2019a). Coord Location Coverage. Retrieved January 11, 2019, from https://coord.co/locations.

[Coord, 2019b] Coord (2019b). Curb Search API. Retrieved January 11, 2019, from https://coord.co/docs/searchcurbs.

[ElMikaty and Stathaki, 2017] ElMikaty, M. and Stathaki, T. (2017). Detection of cars in high-resolution aerial images of complex urban environments. *IEEE Transactions on Geoscience and Remote Sensing*, 55(10).

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

[Fisher, 2018] Fisher, D. (2018). Lidar: A Data Resource Worth Sharing. Retrieved from `https://datasmart.ash.harvard.edu/news/article/lidar-data-resource-worth-sharing`.

[Fitzsimmons and Hu, 2017] Fitzsimmons, E. G. and Hu, W. (2017). The downside of ride-hailing: More new york city gridlock. Retrieved from `https://www.nytimes.com/2017/03/06/nyregion/uber-ride-hailing-new-york-transportation.html`.

[FiveThirtyEight, 2019] FiveThirtyEight (2019). Uber TLC FOIL Response. Available from `https://github.com/fivethirtyeight/uber-tlc-foil-response`.

[Google, 2019] Google (2019). General Transit Feed Specification Reference. Retrieved from `https://developers.google.com/transit/gtfs/reference/`.

[Hahsler, 2019] Hahsler, M. (2019). Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms. Retrieved from `https://www.rdocumentation.org/packages/dbscan/versions/1.1-3`.

[ITF/OECD, 2018] ITF/OECD (2018). The Shared-Use City: Managing the Curb. Retrieved from `https://www.itf-oecd.org/sites/default/files/docs/shared-use-city-managing-curb_3.pdf`.

[Kitchin, 2014] Kitchin, R. (2014). *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage.

[Lee, 2009] Lee, T. (2009). Robust 3d street-view reconstruction using sky motion estimation. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1840–1847.

[Los Angeles DOT, 2019a] Los Angeles DOT (2019a). Parking Meter Inventory. Available from `https://data.lacity.org/A-Livable-and-Sustainable-City/Parking-Meter-Inventory/s49e-q6j2`.

[Los Angeles DOT, 2019b] Los Angeles DOT (2019b). Parking signs. Available from `https://ladot.lacity.org/what-we-do/parking/can-i-park-there/parking-signs`.

[NYC Taxi and Limousine Commission, 2019] NYC Taxi and Limousine Commission (2019). TLC Trip Record Data. Available from `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`.

[OpenDataSoft, 2019] OpenDataSoft (2019). Open Data Inception. Available from `https://data.opendatasoft.com/explore/dataset/open-data-sources%40public/`.

[OpenStreetMap Foundation, 2019] OpenStreetMap Foundation (2019). Planet OSM. Available from `https://planet.openstreetmap.org`.

[Prasad, 2012] Prasad, D. K. (2012). Survey of the problem of object detection in real images. *International Journal of Image Processing (IJIP)*, 6(6):441.

[Rodriguez, 2017] Rodriguez, J. F. (2017). SFPD: Uber, Lyft account for two-thirds of congestion-related traffic violations downtown. Retrieved from `http://www.sfexaminer.com/sfpd-uber-lyft-account-two-thirds-congestion-related-traffic-violations-downtown/`.

[San Francisco, 2018] San Francisco (2018). Parking regulations. Available from `https://catalog.data.gov/dataset/parking-regulations-f7c91`.

[Schaller, 2018] Schaller, B. (2018). The new automobility: Lyft, uber and the future of american cities. Retrieved from `http://www.schallerconsult.com/rideservices/automobility.pdf`.

[Seattle DOT, 2019] Seattle DOT (2019). Parking signs. Available from `https://www.seattle.gov/transportation/projects-and-programs/programs/parking-program/parking-regulations/parking-signs`.

[U.S. DOT, 2017] U.S. DOT (2017). Commercial Loading Zone Management Program: Washington, D.C. Retrieved from `https://ops.fhwa.dot.gov/publications/fhwahop17022/index.htm`.

[U.S. General Services Administration, 2019] U.S. General Services Administration (2019). Data.Gov Data Catalog. Available from `https://catalog.data.gov/`.

[Walker, 2019] Walker, K. (2019). Load Census TIGER/Line Shapefiles. Retrieved from `https://www.rdocumentation.org/packages/tigris/versions/0.7`.

[Wang and Cheng-Yue, 2016] Wang, C. Y. and Cheng-Yue, R. (2016). Traffic Sign Detection using You Only Look Once Framework. Retrieved from `http://cs231n.stanford.edu/reports/2016/pdfs/263_Report.pdf`.

[Wilson, 2012] Wilson, J. P. (2012). Digital terrain modeling. *Geomorphology*, 137(1):107–121.

Appendix A

Listings for Method A1

SQL statement to create the database table that stores curb rules (using a PostGIS column type to store the curb geometry):

```
CREATE TABLE curb_rules (
  id text NOT NULL, -- created based on curb_id and selected metadata
  shape geography(LINESTRING), -- PostGIS polygon
  side_of_street text, -- N | NE | E | SE | S | SW | W | NW
  curb_id text,
  rules json DEFAULT '[]'::json,
  timezone text DEFAULT 'UTC'
);

ALTER TABLE ONLY curb_rules ADD CONSTRAINT curb_rules_pkey PRIMARY KEY (id)
    ;
CREATE INDEX curb_rules_shape_index ON curb_rules USING gist (shape);
```

SQL statement to retrieve curbs within range of a coordinate (using PostGIS transforms):

```
SELECT
  id,
  side_of_street,
  curb_id,
  ST_AsGeoJSON(shape) as shape,
  ST_Distance(shape, ST_MakePoint($[longitude],$[latitude])::geography) as
     distance,
  rules,
  timezone
FROM curb_rules
WHERE ST_DWithin(shape, ST_MakePoint($[longitude],$[latitude])::geography,
   $[maxDistance])
```

```
ORDER BY distance ASC;
```

Bash command to request all rules in the operating area (downtown Los Angeles) from the Curb Search API:

```
curl -G "https://api.coord.co/v1/search/curbs/bybounds/all_rules?
   vehicle_type=taxi&permitted_use=load_passengers&min_latitude
   =34.02492092332343&min_longitude=-118.27777862548828&max_latitude
   =34.06368117691715&max_longitude=-118.21701049804689&access_key=$API_KEY
   "
```

Sample JSON response from the the Curb Search API documentation [Coord, 2019b]:

```
{
  "features": [
    // ... some features omitted for brevity.
    {
      // Note that geometry and metadata are structured the same in "
          all_rules" and "time_rules"
      // requests.
      "geometry": {
        "coordinates": [
          [-73.9773964310734, 40.751917110446385],
          [-73.97715562215994, 40.75181739119557],
          [-73.97689946982429, 40.75171053502362],
          [ -73.97624207098005, 40.75143628919809]
        ],
        "type": "LineString"
      },
      "properties": {
        "metadata": {
          "curb_id": "bnljOjEwNDMzNQ",
          "distance_end_meters": 129.7443128858502,
          "distance_start_meters": 18.653505116785006,
          "end_street_name": "LEXINGTON AVENUE",
          "side_of_street": "SW",
          "start_street_name": "EAST 42 STREET",
          "street_name": "EAST 42 STREET",
          "time_zone": "America/New_York"
        },

        // Every feature can have multiple rules that apply at different
            times. There is always
        // exactly one rule that applies at any one time.
        "rules": [
// Rule 1: passenger loading only from 7am-10am and 4pm-7pm Mon-
   Sat.
{
  // "other_vehicles_permitted" describes what vehicles not
     matching "vehicle_type" can
  // do when this rule applies. It is empty when "vehicle_type"
     is "all", as it is for
  // this rule.
  "other_vehicles_permitted": [],
  "permitted": ["load_passengers"],
  "price": [{"price_per_hour": {"amount": 0, "currency": "USD
     "}}],
  "primary": "none",
  "reasons": ["sign"],
  "times": [
    {
      "days": [1, 2, 3, 4, 5, 6],
      "time_of_day_end": "10:00",
      "time_of_day_start": "07:00"
    },
    {
      "days": [1, 2, 3, 4, 5, 6],
      "time_of_day_end": "19:00",
      "time_of_day_start": "16:00"
    }
  ],
  "vehicle_type": "all"
},

// Rule 2: 3-hour pay parking for commercial vehicles from 10am-4
   pm Mon-Sat.
{
  // Any vehicle can stay here at most three hours.
  "max_duration_h": 3,

  // This rule has "vehicle_type" of "commercial", so "
     other_vehicles_permitted"
  // describes what non-commercial vehicles can do while "
     permitted" describes what
  // commercial vehicles can do.
  "other_vehicles_permitted": ["load_goods", "load_passengers"],
  "permitted": ["park", "load_goods", "load_passengers"],
  "price": [{"price_per_hour": {"amount": 350, "currency": "USD
     "}}],

  // The primary use is parking.
```

```
    "primary": "park",

    "reasons": ["sign", "meter"],
    "times": [
      {
        "days": [1, 2, 3, 4, 5, 6],
        "time_of_day_end": "16:00",
        "time_of_day_start": "10:00"
      }
    ],

    // The primary vehicle type is commercial.
    "vehicle_type": "commercial"
  },

  // Rule 3: free parking all day Sunday, before 7am, after 7pm.
  {
    "other_vehicles_permitted": [],
    "permitted": ["park", "load_goods", "load_passengers"],
    "price": [{"price_per_hour": {"amount": 0, "currency": "USD
        "}}],
    "primary": "park",

    // reasons is null because free parking is the default
        regulation in the absence of
    // any other information.
    "reasons": null,

    "times": [
      {
        "days": [0],
        "time_of_day_end": "24:00",
        "time_of_day_start": "00:00"
      },
      {
        "days": [1, 2, 3, 4, 5, 6],
        "time_of_day_end": "07:00",
        "time_of_day_start": "00:00"
      },
      {
        "days": [1, 2, 3, 4, 5, 6],
        "time_of_day_end": "24:00",
        "time_of_day_start": "19:00"
      }
    ],
    "vehicle_type": "all"
  }
```

```
        ],

        // No temporary rules in effect for the coming week for this curb
           segment.
        "temporary_rules": null
      },
      "type": "Feature"
    }
  ],
  "type": "FeatureCollection"
}
```

Appendix B

Listings for Method B1

R script to calculate clusters and centroids for the NYC Taxi and Limousine Commission Trip Record Data:

```
library("dbscan")

#-- load pickup coordinates from CSV

trips <- read.csv(file="yellow_tripdata_2016-06.csv",
                  header=TRUE, sep=",")

# extract coordinates into a matrix
columns <- c("pickup_longitude", "pickup_latitude")
coords <- trips[columns]
m <- data.matrix(coords)

#-- perform DBSCAN

# maximum distance between clustered points, in meters
distance = 3

# meters per longitudinal degree, at 45 degrees latitude,
# according to https://en.wikipedia.org/wiki/Decimal_degrees
metersPerDegree = 78710

eps = distance / metersPerDegree
minPts = 25
fr <- frNN(m, eps)
db <- dbscan(fr, minPts)

#-- calculate centroids
```

```
centroids <- matrix(0, 0, 2)
numClusters <- max(db$cluster)
for (n in 1:numClusters) {
  cluster <- m[db$cluster == n, ]
  centroids <- rbind(centroids, colMeans(cluster))
}
```