UNIVERSITY OF CALIFORNIA

Los Angeles

Self-Localization of Autonomous Vehicles Using Landmark Object Detection

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical and Computer Engineering

by

Juan Carlo Rebanal

2020

ABSTRACT OF THE THESIS

Self-Localization of Autonomous Vehicles
Using Landmark Object Detection

by

Juan Carlo Rebanal
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2020
Professor Christina Fragouli, Chair

Fully autonomous vehicles are rapidly approaching realization and concerns regarding their safety and robustness are a prominent obstacle to their integration into society. A bottleneck of an autonomous vehicle's safety is its ability to self-localize under all types of conditions. Self-localization is critical for the vehicle to route to a destination, for the vehicle itself to be tracked in case of theft, and in determining local traffic laws to operate harmoniously with other traffic and pedestrians. The global positioning system (GPS) is utilized by autonomous vehicles for self-localization, but the availability and reliability of GPS is not guaranteed in all situations, such as when GPS reception is weak or when an adversary is spoofing information. We address this problem by proposing a self-localization method for autonomous vehicles that does not require GPS at localization time. The framework herein describes the representation of a road network as a graph using the data made available by OpenStreetMap, and an encoding of street segments based on detected landmark objects. We derive the current state observer transition function from the resulting graph, and employ localization methods based on set distance and minimum cost paths to determine the most probable location of the vehicle given a sequence of observed landmarks. Through simulating vehicle traveling paths along a road network generated from real data of a region of Washington D.C., we evaluate the performance of our method for varying degrees of landmark observation error and analyze the algorithm's time and memory complexity, demonstrating that the approach provides a feasible solution to the problem.

The thesis of Juan Carlo Rebanal is approved.

University of California, Los Angeles

2020

To my faithful friends,
whose support and laughter made the day-to-day manageable,

To all my teachers,
both in and out of the classroom,
who taught me the beauty of learning,
and of helping others learn,

To all my favorite creators,
whose work has pushed me to improve,
and to become more mindful, and to hope,
as well as for providing a space for solitude,

To my maternal grandparents Jun and Ofie Hidalgo,
for being exemplars of the fruits of good, honest work,

To my paternal grandparents Jose and Nora Rebanal,
whose prayers and support throughout my life
have been constant and unwavering,

And to my parents Ellen and Joseph Rebanal,
for everything.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Firstly, I want to thank Professor Christina Fragouli for introducing this thesis idea to me last year and for all her guidance and encouragement throughout the thesis. While the process was indeed challenging, Professor Fragouli helped make it so much more manageable and enjoyable thanks to her consistent and insightful feedback. I will always be thankful to her for her support throughout this thesis.

In addition, I would also like to thank Professor Paulo Tabuada for his support and guidance throughout this project. Together with Professor Fragouli, he has shared many insightful conversations which have led to developments and innovations during this project. Also, I would like to thank Professor Ankur Mehta for being supportive of helping me realize the UCLA experience I wanted by making himself and his lab members accessible.

I also want to thank Yahya Ezzeldin for being a great mentor figure to me. Yahya has helped clarify and mediate countless times, and has always been willing to help me out with technical matters, as well as just having pleasant conversations with.

I need to thank all the open source tools and resources I used throughout this project, which together expedited the realization of this thesis.

My amazing friends and family have given me constant support in even the toughest of times and I am forever thankful for them.

Lastly, I would like to thank my parents Ellen and Joseph Rebanal. Thank you for everything.

CHAPTER 1

Introduction

Autonomous vehicles are becoming closer to consumer realization than ever before, thanks to great strides being made by researchers and manufacturers alike. While a fully-realized autonomous vehicle comprises a complex, multi-faceted system, it can still only be as strong as its weakest link. The global positioning system (GPS) utilized by autonomous vehicles, among countless other devices, is critical for an autonomous vehicle to be able to localize, i.e. to find its absolute position on the planet. Localization is important for routing, [FMC13] asset tracking and for helping the vehicle confirm its position relative to pedestrians and obstacles to stay clear of. However, the availability of GPS is not guaranteed in all situations, such as when GPS reception is weak or when an adversary is spoofing information. A method for localization without requiring GPS information is essential for the overall robustness of an autonomous vehicle, and would help to expedite their transition into everyday use by consumers. We address this need with our approach which utilizes object landmark detection and planar graph representations of road networks to develop a method for self-localization of an autonomous vehicle without using GPS at localization time.

Self-localization refers to the task of an agent determining its position and orientation within some coordinate system. Autonomous vehicle self-localization is a well-researched field, rife with variety in terms of approaches to the problem, including those with roots in localization in robotics ([YCQ18], [LLZ10], [BG13]), and having different trade-offs. We focus on improving a particular subset of such approaches, namely localization approaches that rely on landmark detection, modeling the environment as a probabilistic map, or integration of multimodal sensors (GPS, inertial measurement unit, odometry, etc.). Landmark detection-based methods have roots in the simultaneous localization and mapping (SLAM) problem from indoor robot navigation and localization. Adapting such work to a large, urban area without the freedom to instantiate installations forces approaches to use naturally-occurring landmarks in urban areas such as trees and fire hydrants. Assigning posterior probabilities to various discretized regions of space in a (GPS-given) prior known search area given observations has also been proposed. Perhaps the most prominent method of localization involves using a collection of sensors equipped by a vehicle, including GPS, to localize to an root-mean square (RMS) accuracy within 10 cm of ground truth [LT10].

In this work, we propose a self-localization method for autonomous vehicles that does not require GPS at localization time. The framework herein describes the representation of a road network as a graph using the data made available by OpenStreetMap (OSM) [OSM], whose street segments are encoded based on proximal landmark objects [AZT14]. The current state observer transition function $\phi_{\mathbf{c}}$ is derived for the constructed graph [CGW91] to determine the minimum path length required to localize assuming 0 observation errors, n_0. To be able to correct up to t errors, n_t is found experimentally offline. Then, at localization time, given a sequence of observed landmarks we return the most likely location of the vehicle with flexible trade-offs between localization speed and accuracy. Our method demonstrates a novel, scalable solution to GPS-less localization, enabling localization in up to 2 street segments traversed, on the average of 1,000,000 trials, even with 5 observation errors in a 10.7 km^2 urban area.

1.1 Method summary

The method proposed in this paper aims to provide a means for the self-localization of a vehicle using landmark detection that results in building a codebook of possible walks on a graph representation of a region.

Offline, the region to be localized within is exhaustively swept for both landmark objects and features of the current street segment: for example, trees and the street segment length.

The information about these landmark features is encoded and mapped to each individual street segment. Each observed combination of landmark features is a symbol in a code alphabet.

The region of interest is represented as a graph, whose vertices are street segments and edges represent intersections spatially connecting adjacent street segments.

Then, a walk on different vertices of this city graph results in observing some sequence of landmarks, discretized as symbols, one for each street segment.

Concatenating these symbols together creates a codeword. By taking a sufficiently long codeword, the sequence of landmarks observed becomes unique, or equivalently the resulting codeword becomes uniquely decodable. Then, localization can be achieved by looking up the location of the street segment at the end of the walk.

Using this method, a vehicle can localize in a 10 km^2 area in up to 2 street segments traversed, on average, even with 5 observation errors. However, limitations of this method surface when faced with larger regions one may wish to localize within: firstly, the method's effectiveness is bottlenecked by the availability and cleanliness of OpenStreetMap and landmark location data, and secondly the offline and online components of the method will take more resources in time and memory (significantly more so for the offline component).

1.2 Related Work

Localization of an agent is a well-studied field, with applications from indoor robot navigation to autonomous vehicles, and many others in between. Here, we look at some key prior work that helps to contextualize our contributions.

1.2.1 Extracting street-level landmarks for localization

Methods involving landmark-based localization require some form of object detection. Some methods focused on employing deep learning to detect landmark objects and text ([EHH19], [KHK19], [JSV16], [WWC12], [GBI13]), whereas others focused on more classical computer vision techniques revolving around landmark extraction such as filtering, pole detection, curve estimation and stereo depth estimation ([FMC13], [LLZ10], [SGR16], [Spa15], [WYG19], [BN18], [MD04], [TB00], [HDV18]).

Some approaches in literature to the localization problem for autonomous vehicles take more direct inspiration from its roots in robotics by making use of rangefinders, such as [RSR15], [NVT08], and [VNB09].

The deep learning-based methods can use neural networks to detect landmarks, and sometimes even to estimate the pose using regression [EHH19]. Using neural networks for pose estimation has produced some good results but introduces room for regression error and requires time to train both the detector and the localizer networks. Our method determines the location of a vehicle deterministically and can be implemented in existing pipelines with deep-learning powered object detection systems. In addition, results from our method are more interpretable than those of black box AI systems, i.e. the reasoning behind why a particular location is estimated can be traced to seeing what landmarks were observed.

On the other hand, the classical computer vision methods lack the flexibility of deep learning-based methods and are consequently limited in their capacity of object detection. This is often mitigated by making use of multiple sensors or by using machine learning as part of the object detection pipeline. These multi-modal approaches inspire and inform the method proposed in this work through selecting to detect landmark features that possess desirable characteristics (explained more in Chapter 3) that can be sensed by camera as well as odometry sensors. Additionally, methods using rangefinders often are unable to detect classes of landmark objects that can vary in shape and size; object detection neural networks help with this.

1.2.2 Probabilistic approaches to localization

Prior work has approached the localization problem in a probabilistic way ([BGU16], [BI17], [LT10], [dBA19]), often employing particle filter localization. In particular, visual odometry and distributing probabilities of candidate locations was the main approach investigated in [BGU16], which found good results and localized in reasonable lengths of time. Visual odometry (the curvature of roads traversed) was the main mode of discriminating between potential paths. However, a purely probabilistic approach may result in unresolved ambiguities. If a region with identical road network lengths and configurations is traversed (e.g. parts of Manhattan, New York), visual odometry alone may not result in localization as admitted in the work. Our work employs multiple methods of helping to differentiate path segments from each other, including using landmark object detection along with odometry, and using OpenStreetMap data to break down regions into atomic street segments. These additional measures help to avoid such ambiguous cases and return a location estimate deterministically.

1.3 Contributions

Prior landmark-based localization techniques applied to autonomous vehicles rely on some type of prior GPS information or use GPS directly when observing landmarks to triangulate, determining the vehicle's absolute position. Our approach seeks to minimize usage of prior GPS information to localize, particularly during the actual time of localization. The main contributions of this work are the following:

- A novel connection between automata theory [CGW91] and coding theory
- Leveraging of this connection to design efficient, polynomial-time algorithms
- An extensive evaluation of such algorithms on real-world data

Furthermore, the proposed method allows for polynomial-time localization on average in a 10 km^2 area. The method is also deterministic in arriving at a position estimate, even in the presence of observation errors when the observed path length is sufficiently long.

This work uses the openly-available OpenStreetMap dataset and can be applied to any reasonably accurate landmark object detection pipeline to build (offline) and detect (online) landmarks when landmark data is not already available for an area.

1.4 Organization

Overall, this paper is ordered in the same fashion as the proposed method itself. That is, the chapters build upon the previous ones to outline the method.

Chapter 2 introduces essential terminology when working with graphs, a ubiquitous mathematical construct, which is used to represent the road networks we work with. The chapter also explains the data used to generate the road networks, and how it was used and cleaned for the purposes of this work.

Chapter 3 adds another critical layer to the method: the concept of landmarks as they relate to this method. We discuss our definition of landmark objects and landmark object classes, and how we arrived at said definitions. We also examine the dataset of geo-tagged landmark objects used in this work. We explain in detail how landmarks are assigned to street segments and how the road network graph is transformed to best accommodate the remainder of the approach. In addition, we look at the method behind determining the bearing of each street segment so it can be used later in the localization algorithm. We close the chapter by defining a fundamental distance metric for this paper, the well-established Hamming distance.

Chapter 4 focuses on introducing the concept of a finite automaton and relating this to the framework built up over the previous chapters. After establishing the necessary connections, we walk through the current state observer algorithm, which ultimately allows us to determine the minimum number of street segments to visit in order to achieve localization if no observation errors are expected. This chapter introduces notions of minimum localization path length that are revisited and built upon in coming chapters.

Chapter 5 rounds out everything laid out in the thesis up to this point by introducing the localization algorithms used. We introduce the classical problem of minimum cost paths and set distance that our approaches are based on, as well as explain the preparation necessary to apply them. Then, we present the pseudocode of the algorithms and provide time and complexity analysis to assess its scalability.

Chapter 6 presents experimental design, results, and discussions. We include preliminary work at the beginning of this chapter since said work was important in establishing lower performance bounds, expectations and intuitions that were put to use when developing the method later in the life of this work. Then, we present, compare and discuss results of using the main localization algorithms proposed in this paper.

We conclude this work in chapter 7 by summarizing the problem and our proposed solution, along with the key results and suggestions for future work.

CHAPTER 2

Road network graph representations

Graphs are structures that serve to abstract relationships between entities. In this work, graphs are used extensively to model arbitrary road networks. This both discretizes urban areas and captures relationships between different points in space in such a way that facilitates working with them. Literature commonly uses graphs to represent road networks.

Definition 2.1. (Graphs) A graph $\mathbf{G} = (V, E)$ is represented as a tuple of its vertices V and its edges E. Each edge $e = (v_s, v_d) \in E \subseteq V \times V$ can be described in terms of its source vertex $v_s \in V$ and destination vertex $v_d \in V$.

In particular, the graph we use is initially constructed such that vertices represent *street intersections* and edges represent *streets connecting street intersections.* Furthermore, the direction of traffic along a road can be captured by introducing directed graphs.

Definition 2.2. (Undirected and directed graphs) In an undirected graph, $(v_s, v_d) \in E \implies (v_d, v_s) \in E$. For a directed graph, there is no such implication.

In this work, we use only directed graphs. Now that vertices and intersections can be understood in isolation and as their entire respective sets, we can now look into defining relationships between these elements and interpreting them through this work's lenses.

Definition 2.3. (Incident edges) An edge e is incident to a vertex v if the source vertex of e is v.

Definition 2.4. (Adjacent vertices in directed graph) A vertex v_2 is adjacent to another vertex v_1 if there exists an edge $e = (v_1, v_2) \in E$.

Adjacent intersections indicate that there is a street segment that can be traversed to reach one from the other. More generally, from any intersection, the list of all adjacent intersections captures all the intersections one could reach using a single street segment. Using graphs this way, we can represent intricate road networks of urban areas given that we know the locations of the intersections and how they are connected to each other.

Definition 2.5. (City graph) A graph representation of a city where the vertices represent a city's intersections and the edges represent the street segments of a city that connect intersections.

In particular, this work assumes that the city graphs being used are planar graphs representing geographically adjacent locations and paths from a vertex to a non-adjacent vertex are accessible through intermediate vertices (consisting of simple intersections and no overpasses).

Definition 2.6. (Planar graph) A graph is planar if it can be drawn such that no edges intersect with each other.

However, it can be shown that the line graph (See definition 3.10) of such a city graph is not necessarily planar.

2.1 Constructing the graph of a road network

We make use of the data made available by OpenStreetMap (OSM) [OSM] to construct the graphs used in our experiments. When implementing this work's method, any reasonably reliable database of road network global information system (GIS) data can be used.

OpenStreetMap is a freely editable map of the world that is powered by a worldwide community of volunteers. The project can be thought of as a combination of Google Maps and Wikipedia. OSM has recently broken 5.6 million registered users and its data is used in dozens of projects [OSM] covering applications from routing to retrieval of information about political boundaries between countries. OSM has also been used in various literature to create maps of urban areas for localization purposes [BGU16, RSR15, FVL13, PC18].

2.1.1 Data in OpenStreetMap

The atomic element in OSM is the *node*. A node is implemented as an XML tag, which has its own set of properties such as a latitude, longitude, and unique OSM node identification number (ID). Nodes can be used in conjunction with (potentially nested) tags to specify both the location of an item of interest, and what the item is. For example, a node could represent a coffee shop by capturing its latitude and longitude, as well as the type of amenity it provides, its opening hours and even if it is wheelchair accessible. However, due to the crowdsourced nature of OSM, not every node has the same level of detail. For our work, we only use 2 basic properties of each node: their location (latitude and longitude) and whether or not they represent a point on a street.

Another fundamental structure used in OSM is the *way*. A way is an ordered list of at least 2 nodes that also has its semantic meaning assigned by an OSM user. That is to say, a way can represent a closed area such as a public park, or it can represent a street if the nodes comprising it are the intersections at the street's ends. Curvature can be approximated by a way made from nodes along a curved road that are close enough together. In this work, we are only concerned with the ways that represent streets.

2.1.2 Extracting data from OpenStreetMap

There exist various methods for extracting data from OpenStreetMap. One of the more prominent methods due to its flexibility in specifying queries and accessibility via web interface is Overpass Turbo [Ove]. Here, a user can supply an XML-style query specifying what type of data they wish to extract and over what region. Overpass Turbo is what we used to extract node data of our experiment region, an area of Washington, D.C. Other regions such as Westwood Village in Los Angeles, California had its data extracted for preliminary work.

Initially, every road segment node was captured as part of our process. However, empirical preliminary results showed that having many street segments with high variance in length led to relatively high frequency of street segments being assigned little-to-no landmarks. Additionally, this made each street segment not as informative due to each segment not

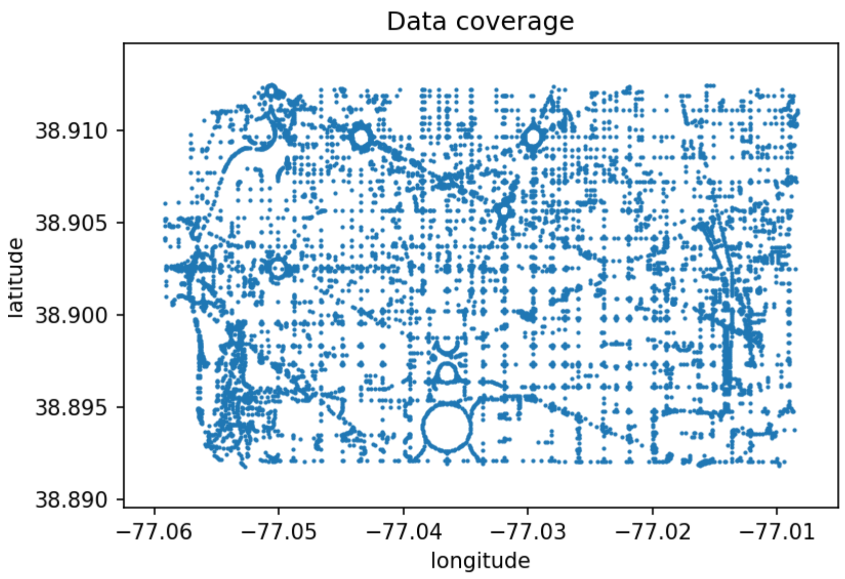


Figure 2.1: The region of Washington D.C. used in this work; the northern area surrounding the White House. Depicted here are the nodes extracted from OSM, including ways meant for vehicle travel, pedestrian/bicycle travel, and various points of interest.

necessarily having multiple turning options. In order to better facilitate a memory-efficient and faster-to-search map of landmarks, we opted to use only intersection nodes, which were more likely to have multiple turning options and would have more unique landmark assignments based on the landmarks detected when traversing them; this reduces complexity of the localization path algorithm detailed later.

In order to capture only the intersection nodes, we formulated a query to find nodes that are present in at least 2 distinct ways. By returning only these nodes, we arrived at a more minimal representation of the region that was more conducive to our localization method.

As visible in Figure 2.3, there are still some instances of false positives (intersection nodes that are not intersections) and false negatives (intersections not marked as nodes). The data was cleaned manually to try and account for these, but almost surely some will still remain.

Initially, we expected to be able to extract sufficiently dense data about locations of landmarks from OpenStreetMap. However, landmark node data that this work is concerned with were found to be sparse in the regions we were examining. Hence, the extent to which

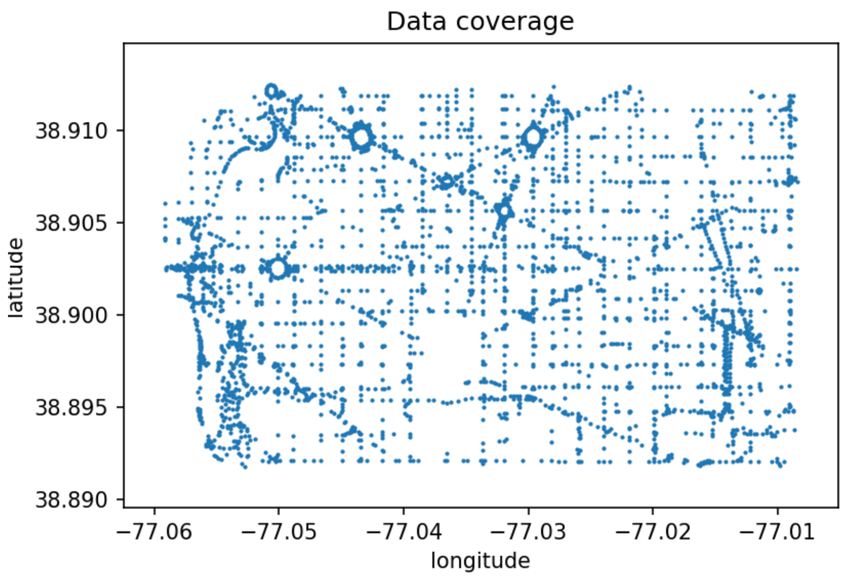


Figure 2.2: The same region of Washington D.C., but after filtering out most non-intersection nodes.

OSM data is used in this work is for intersection locations.

2.1.3 Using *igraph* to construct the graph object

The Python package *python-igraph* [pyt] was used extensively to model graphs. This package boasts plenty of useful methods for creating, modifying and visualizing graphs.

The Graph class features an iterable vertex sequence and edge sequence, which allow one to access each vertex and edge of a graph, respectively. By default, each vertex and edge has a unique ID assigned to them when they are added to the graph. Crucially, vertices and edges can each be assigned user-defined properties. Most commonly, vertices or edges are assigned appropriate *weights* for use in shortest-path algorithms, for example. However, in this work, each edge is assigned a *landmark symbol* based on the landmarks observed while traversing that edge. This is expanded upon in Chapter 3.

An existing tool [An] used to extract vertex and edge information from raw OSM data was modified to further remove false positive intersection nodes. Given the list of intersection nodes extracted from Overpass Turbo, only ways that included at least 2 intersection

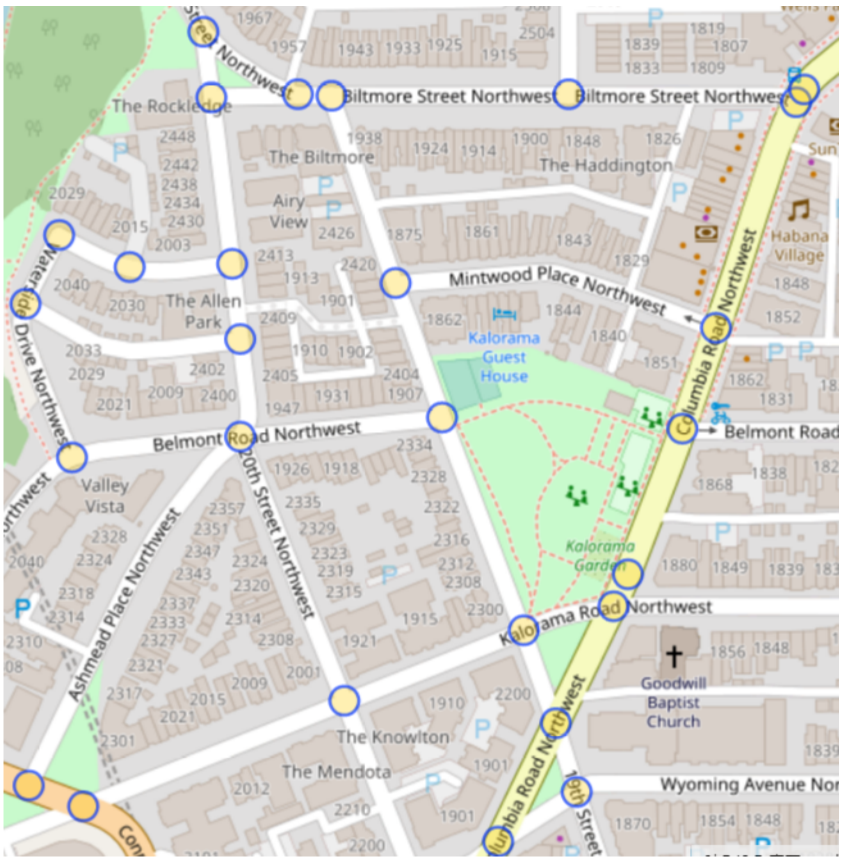


Figure 2.3: A portion of the Washington D.C. region visualized in Overpass Turbo. The circular points are intersection nodes.

nodes were left in. The tool outputted a text file containing a list of all the vertices' IDs, latitudes and longitudes, followed by a list of edges defined by their source and destination vertex, as well as whether they were bidirectional given by either 0 (not bidirectional) or 1 (bidirectional).

The python-igraph package allows for the construction of directed graphs by adding 1 edge and up to 2 vertices at a time. By parsing the road graph description produced by the previous tool, the directed graph object was able to be created where the vertices represented intersections in a road network, and edges represented the street segments connecting them.

In the next chapter, the process of labeling the edges and deriving the line graph from this graph will be examined.

2.2 Chapter summary

In this chapter, the necessary fundamentals of graph theory were defined. Following that, the dataset provided by OpenStreetMap and its selective extraction was explained and relevant parts were visualized. The construction of the graph from this data was detailed. Next, the concept of landmark assignment to edges of the graph will be introduced, and the true power of representing a road network as a graph will start to take form.

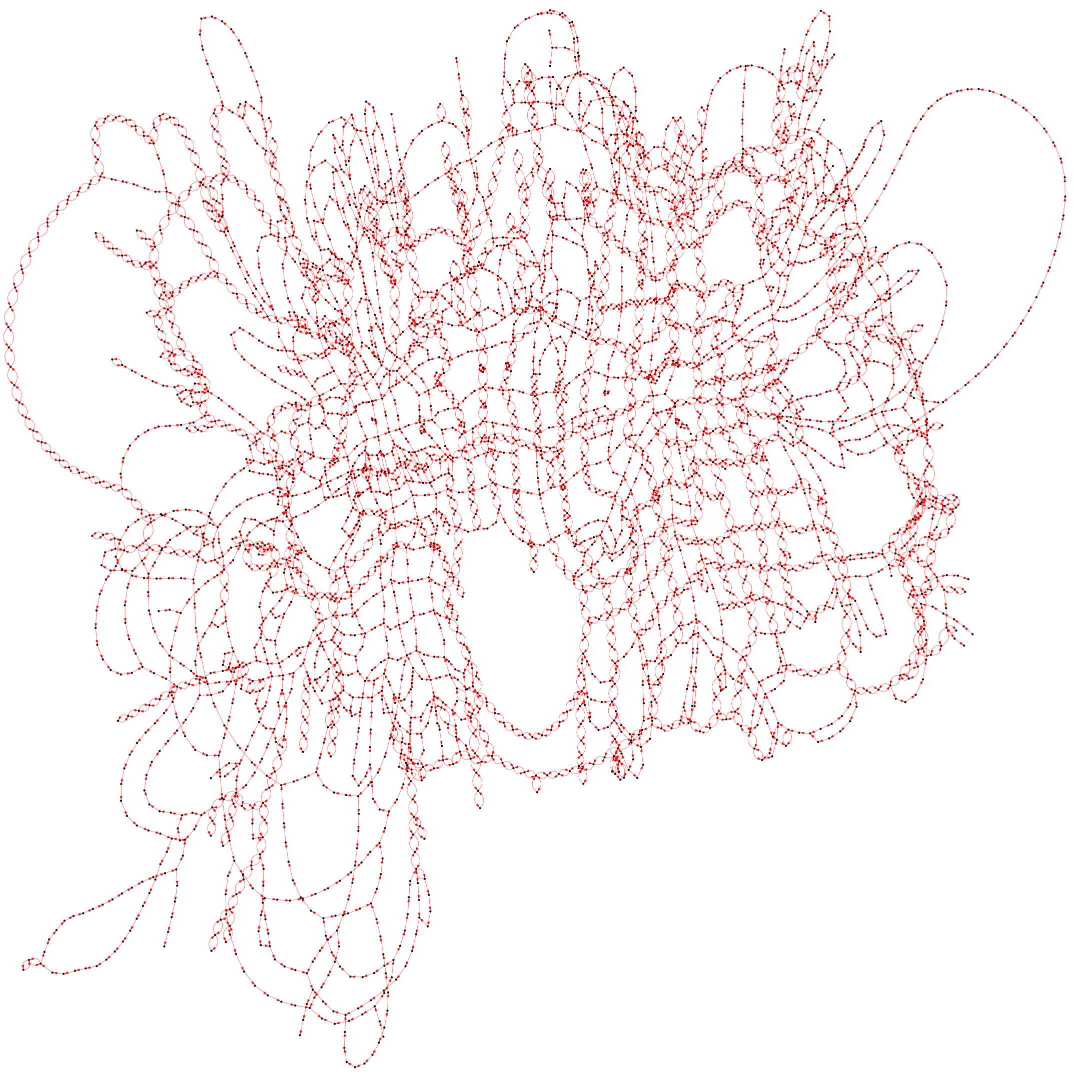


Figure 2.4: Graph of the region of Washington D.C. using Kamada Kawai layout [KK89]. Vertices are black and edges/arrows (indicating direction) are red. Representation mainly serves to depict connectivity; the locations of the vertices in this image are loosely based on their corresponding true relative locations, but the edge lengths are not to scale.

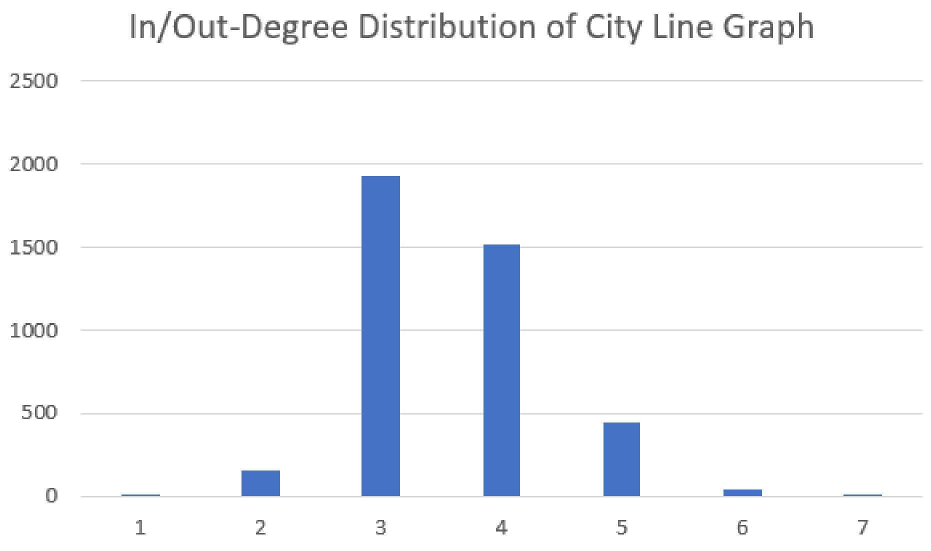


Figure 2.5: Degree distribution of the line graph resulting from the road graph. Combines in- and out-degree data. See Chapter 3 to read about line graphs.

CHAPTER 3

Landmark-based localization and relations to coding theory

3.1 Landmark-based localization

The main characterizing aspect of this work is its use of landmarks to localize, i.e. to determine where the vehicle is located. This approach is based on intuitive approaches to determining one's location from even before technology. "Meet me by the tree" or some variant of this phrase has likely been encountered at some point in the reader's lifetime.

The following are the main principles that underscore our overall method for localization using landmarks:

1. Localization can be achieved by observing a unique landmark, given that we know the absolute location of that unique landmark from a given landmark map.
2. If the landmark observed is not unique, observing more landmarks will generate a sequence of observations that may be unique among all other observation sequences of the same length. Thus, localization can then be achieved.

As an example, imagine that you wake up in an unfamiliar part of Paris, France. You intrepidly turn the corner and spot the Eiffel Tower. There's only one such tower in the world, and you can immediately tell where you are. However, if you happened to travel away from the Eiffel Tower, you might run into a coffee shop - hardly uncommon. After walking some more, you pass by two more coffee shops before reaching leather shoemaker's store. Such stores are rather uncommon, but definitely not unique. But having walked past a total of three coffee shops and arriving at a leather shoemaker's store, there's only one street in Paris with these landmarks so close together - so, at last, you determine exactly where you are.

In an effort to generalize this work as much as possible, i.e. for it to be applicable in as many locations as possible, we work with landmarks that are common in urban and suburban areas and we do not use street name signs so that our method can be applied in environments without such signs or when the signs are obscured by traffic, weather, etc.

Additionally, in the chapters to follow, observation error correction is discussed. This helps to improve the robustness of the method.

3.1.1 Characteristics of desirable landmarks

In this work, we focus on instances of classes of landmark objects as opposed to unique, named landmarks, e.g. the Eiffel Tower.

Definition 3.1. (Landmark object) An observable entity belongs to the discrete set of all observable entities X with a unique position in the region of space bounded by the road network graph $\mathbf{G}= (V, E)$.

Remark 3.1. *For this work, we focus on landmark objects that possess certain desirable characteristics.*

Based partially on literature [MD04, KHK19], landmarks used for localization share most of the following characteristics:

- **Naturalness.** The landmarks should not have to be installed (e.g. a beacon or encoded image); they should already be observable in the environment.
- **Time-invariant representation.** Observable measures of the landmark should not change significantly over time (save for, incident illumination sources from different times of day). Examples of landmarks that do not have this property include trees in regions that experience Summer, Fall, Winter and Spring, or a movie theater marquee whose contents change with the movies.

- **Time-invariant location.** Landmarks should not change their location over time. A parked car is a landmark that does not have this property [LLZ10].
- **Viewpoint invariant observability.** The landmark should be observable regardless of what orientation it is viewed from. Pole-like structures and other structures with distinct shapes have this characteristic.
- **High frequency of occurrence.** The landmarks should be common in space. They should not be sparsely distributed across the region of localization. This helps to facilitate more informative observations.
- **Belongs to a class distinct from others.** Each landmark should not be able to be confused for a different landmark class. This helps with error handling.

Identifying desirable characteristics of landmarks helps to narrow down the types of objects we can use to those that we can expect to be most helpful in localization.

3.1.2 Landmark object classes

After considering the aforementioned desirable characteristics of landmarks, we decided on what landmark object classes to use to define each street segment with.

Definition 3.2. (Landmark object class) A class, or label, $y_i \in$ the discrete set of all classes Y. Each landmark object x_i is paired with a single corresponding label y_i.

Remark 3.2. *There also exist problems where an item x_i can belong to more than 1 class, i.e. x_i can have labels $y_1, y_2, \cdots, y_k$ where $1 \leq k \leq |Y|$, referred to as multiclass classification, but that is outside the scope of this work.*

- **Fire hydrants.** These have a stand-out color as well as unique shape and height. Their main drawback in our application is that they may be obscured by unlawfully parked cars due to their short height.
- **Street lights.** Features a distinct pole-like shape that is recognizable and observable despite weather conditions or most obstructions.

- **Traffic lights.** Present at most intersections, the number of traffic lights visible at an intersection combined with some combination of other landmarks helps to distinguish busier intersections from less busy ones. Additionally, they are by design unobstructed by most traffic.

- **Traffic signs.** A broad landmark object class composed of several kinds of traffic signs such as yield signs or stop signs. Future work can discriminate between the inherent sub-classes in this object class.

- **Trash receptacles.** These have distinct shapes and appear regularly along sidewalks.

We constrain this work to only use landmark objects that belong to one of the above classes. The main bottleneck preventing the incorporation of more landmark object classes is the availability of data; with more classes of landmarks, more landmarks can be observed and used towards characterizing areas as distinct, helping to expedite the process of localization.

3.1.3 DC landmark dataset

Infrastructural data is often publicized by governmental bodies of different regions. However, there is no standardized method used to publish or describe such data, so we defer to mainly using datasets referenced by other literature.

We employ the dataset used in [AZT14] that includes the geographic information system (GIS) data of the locations of our landmark object classes of interest. The data comes in the form of keyhole markup language (KML) files that are used for geographic information visualization and annotation. Included are the coordinates in latitude and longitude for each landmark object, as well as some additional, annotative information (e.g. unique IDs, and service status at the time of data collection).

Clearly, some regions contain information for more landmarks than others. This is, again, bottlenecked by the available data. We constrain the area we try to localize within to the area that has the most overlap in coverage from all the landmark object classes.

Approaches to geo-tagging objects of interest have been proposed [AZT14] and could be

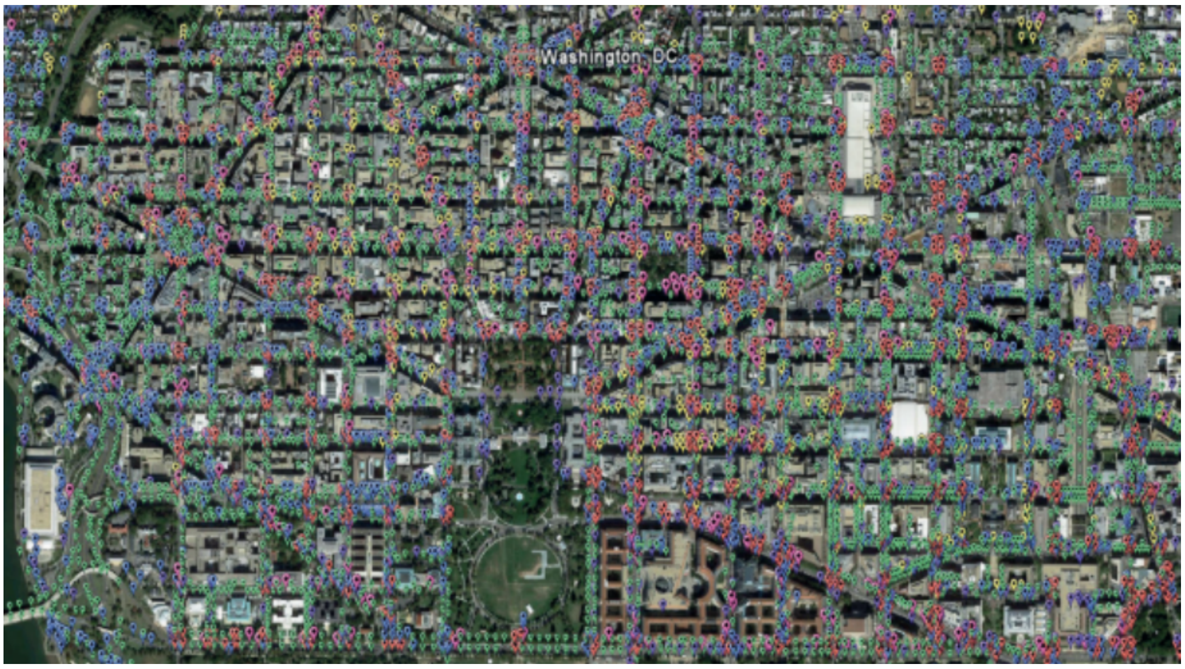


Figure 3.1: Plot of the landmark object information in a region of Washington, D.C. Different colors of plotted waypoints represent different landmark object classes. Image from [AZT14].

used to diversify both the coverage of such datasets, as well as the varieties of landmark object classes that comprise the datasets.

3.2 Landmark assignment to street segments

Given the locations of different landmark objects and a graph representation of the road network that they cover, we can begin to assign landmarks to areas of the region of interest.

At this point, we wish to build a landmark map that indicates what landmarks we expect to observe at certain locations. This landmark map would serve as the ground truth to be compared to during localization time when the vehicle would observe whatever landmarks it can to try and match the observed sequence with some sequence in the landmark map.

In order to measure the effectiveness of using the landmark map to arrive at a uniquely identifiable sequence, simulations were designed to have a random walk be performed on the road network graph. This will be the focus of this chapter; the chapter will give definitions and introduce notions that build up to the understanding of localization performance in the context of this work. To start unpacking the idea of measuring localization effectiveness, we have to define some concepts.

Definition 3.3. (Probability space) A tuple of outcomes Ω, events of interest $\mathcal{F}$, and a probability measure P together define a probability space $(\Omega, \mathcal{F}, P)$ over which observable outcomes $\omega \in \Omega$ can be mapped to events $F \in \mathcal{F}$, and each event has an associated probability measure.

Definition 3.4. (Random variable) A random variable Z is a function that maps a particular outcome ω from a sample space Ω to a real value, i.e.

$$Z : \Omega \to \mathbb{R} \tag{3.1}$$

Remark 3.3. *A random variable Z can denote the probability of an event z occurring with the notation $\mathbb{P}(Z = z)$. In this work, events z consist of different types of landmark observations.*

Definition 3.5. (Code) A code C is a function that maps an input from an alphabet X to another alphabet Σ, i.e.

$$C : X \to \Sigma \tag{3.2}$$

Remark 3.4. *We can think of the input alphabet of a code as a random variable Z. The random variable here can represent the possible landmark observations we can make. Hence, a code C maps possible landmark observations to a symbol $C(z) \in \Sigma$.*

Definition 3.6. (Non-singular code) A code C_{NS} is non-singular if every element of an alphabet $x \in X$ maps to a distinct $C(x) \in \Sigma$, where Σ is a set of sets that each correspond to mappings $C(x)$ for some x.

$$C_{NS} : X \to \Sigma,\ |C_{NS}(x_i)| = 1,\ 1 \leq i \leq |X| \tag{3.3}$$

Definition 3.7. (Extension of a code) The extension of a code C^* is the mapping from finite-length strings (where strings are concatenations of individual symbols) from an input alphabet X to finite-length strings of another alphabet Σ. Let $x_1 x_2 \cdots x_n$ be the concatenation of words from an input alphabet, and $C(x_1)C(x_2)\cdots C(x_n)$ be the concatenation of the corresponding mappings. Then,

$$C^* : C(x_1 x_2 \cdots x_n) \to C(x_1)C(x_2)\cdots C(x_n) \tag{3.4}$$

Definition 3.8. (Unique decodability) A code C_{UD} is uniquely decodable if its extension is non-singular.

Remark 3.5. *In other words, if a code is uniquely decodable, then any concatenation of a finite number of input alphabet symbols maps to a single, unique corresponding codeword from the set of all finite-length output alphabet strings made from Σ, defined as Σ^*. This is the generalization of non-singular codes to work for inputs of any finite length.*

This work focuses on creating a uniquely decodable code mapping landmark observations to their corresponding codewords. The landmark map is populated with the codewords corresponding to each landmark observation. Then, during localization time (and without GPS), observations are made and their corresponding codewords are essentially compared to the landmark map of known codewords to search for a unique, (as close as possible to) perfect match. Because the code used is uniquely decodable, any finite length sequence of landmark observations maps to only one unique codeword and the brunt of the work can be focused on searching for sequences that match the observed codeword sequence.

Each vertex would then be assigned a codeword based on its corresponding landmark observation.

Definition 3.9. (Landmark code) A landmark code C_L is a uniquely decodable code that maps landmark observations $L_i^{obs} \in L$ to an alphabet Σ.

$$C_L : L \to \Sigma \tag{3.5}$$

3.2.1 Road network graph to line graph transformation

Our initial approach of assigning landmark observations was to assign landmark observations to intersections based on the proximity of landmark objects to that intersection. However,

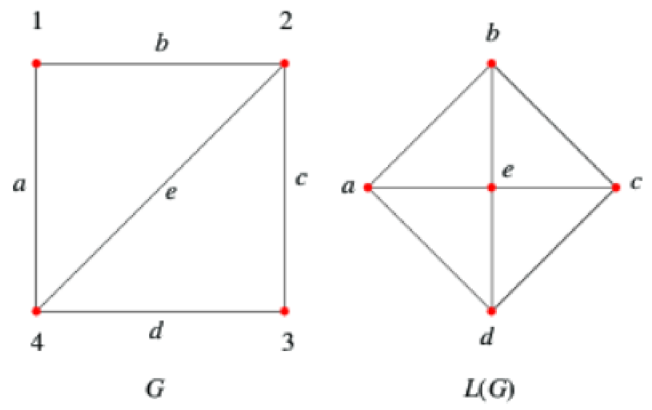


Figure 3.2: Example of a graph G and its corresponding line graph L(G). [Wei]

one fundamental problem with the approach is that a significant portion of landmark object data ends up not being used; only landmark objects close to any given intersection are used in encoding the corresponding vertex's landmark observation. Hence, we transformed the road graph to better facilitate the expected observation of more unique sequences.

Definition 3.10. (Line graph) The line graph $L(\mathbf{G}) = (V_L, E_L)$ of a graph $\mathbf{G} = (V, E)$ has a vertex v_L for each edge $e \in E$. Two vertices v_{L_i}, v_{L_j} are adjacent if and only if their corresponding edges in $\mathbf{G}$ are incident to the same vertex in $\mathbf{G}$.

Remark 3.6. *The line graph of the original road network graph has new interpretations of its vertices and edges. The line graph's vertices are now the street segments connecting the intersections, and the line graph's edges now represent the intersections, i.e. adjacency of one street segment to another.*

Definition 3.11. (Convex combinations) A convex combination of points is a linear combination of points

$$\sum_{i=1}^{k} \alpha_i x_i \tag{3.6}$$

such that $\alpha_i \geq 0$ for $1 \leq i \leq k$ and $\sum_{i=1}^{k} \alpha_i = 1$.

Definition 3.12. (Convex hull) The convex hull Conv(S) of a set of points S is the set of all convex combinations of points in $x_i \in S$.

$$\mathrm{Conv}(S) = \left\{ \sum_{i=1}^{|S|} \alpha_i x_i : \alpha_i \geq 0 \, \forall \, i \, \wedge \sum_{i=1}^{k} \alpha_i = 1 \right\} \tag{3.7}$$

Definition 3.13. (Landmark observation: road network line graph) For the road network line graph $L(G) = (V_L, E_L)$ from the road network graph $\mathbf{G} = (V, E)$, a landmark observation L_i^{obs} made along a street segment $v_{L_i} \in V_L$ corresponds to an edge $e_k \in E$. That (directed) edge e_k is incident to a source vertex v_1 and a destination vertex v_2, both $\in V$, which correspond to intersections with locations $v_{1_{lat}}, v_{1_{lon}}$ and $v_{2_{lat}}, v_{2_{lon}}$, respectively.

Next, define the vector

$$\vec{s} = (v_{2_{lat}} - v_{1_{lat}})\hat{x} + (v_{2_{lon}} - v_{1_{lon}})\hat{y} \tag{3.8}$$

Correspondingly, let the vector perpendicular to $\vec{s}$ be defined as $\vec{s_\perp}$. Let $||\vec{a}||$ denote the L2-norm of $\vec{a}$, w be the desired width of the bounding box, and c be the scalar constant scaling factor $\frac{2w}{||\vec{s_\perp}||}$.

Then, define the set of landmark region bounding points

$$\begin{aligned} B = \{&(v_{1_x} + c\vec{s_\perp}_x, v_{1_y} + c\vec{s_\perp}_y), \\ &(v_{1_x} - c\vec{s_\perp}_x, v_{1_y} - c\vec{s_\perp}_y), \\ &(v_{2_x} + c\vec{s_\perp}_x, v_{2_y} + c\vec{s_\perp}_y), \\ &(v_{2_x} - c\vec{s_\perp}_x, v_{2_y} - c\vec{s_\perp}_y)\} \end{aligned}$$

where each element $b_i \in B$ is a point in space $\in \mathbb{R}^2$ and width of the rectangular landmark bounding region $w \in \mathbb{R}$. Then, the landmark observation L_i^{obs} for a street segment v_{L_i} is

$$L_i^{obs} = \{L_j : L_j \in \mathrm{Conv}(B)\} \tag{3.9}$$

Remark 3.7. *The landmark observation is the set of all landmark objects that fall within a rectangular region whose corners are defined by points that are a set distance away from the vertices of an edge in the directions perpendicular to the path from source to destination vertex.*

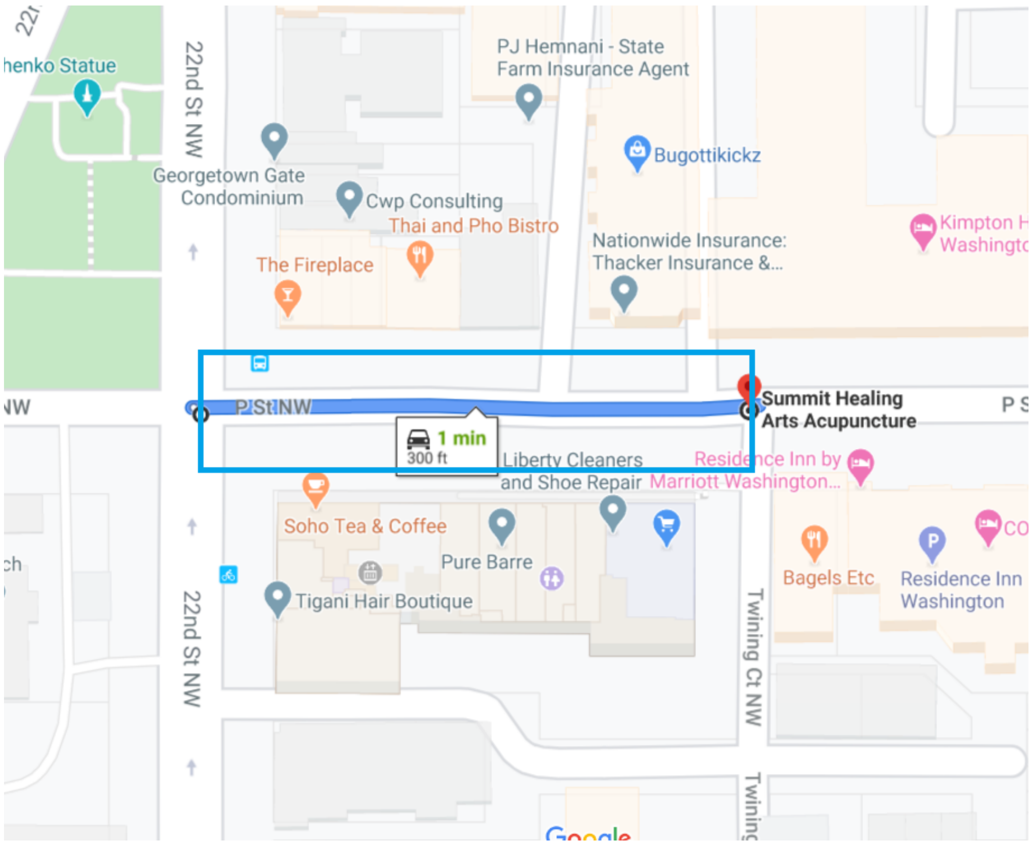

22nd St NW
PJ Hemnani - State Farm Insurance Agent
Bugottikickz
Georgetown Gate Condominium
Cwp Consulting
Thai and Pho Bistro
The Fireplace
Nationwide Insurance: Thacker Insurance &...
P St NW
1 min
300 ft
Summit Healing Arts Acupuncture
Liberty Cleaners and Shoe Repair
Residence Inn by Marriott Washington...
Soho Tea & Coffee
Pure Barre
Tigani Hair Boutique
Bagels Etc
Residence Inn Washington
Twining Ct NW
Google

Figure 3.3: Example of a rectangle generated with the above method given two real data points from the dataset. Landmarks assigned to the shown street segment are those that are bounded by the rectangle depicted. Image generated from Google Maps.

Defining landmark observations based on street segments as vertices allows more landmarks to be used (more information) in encoding street segments, in turn resulting in more unique codewords and faster localization.

3.2.2 Landmark bounding box assignment

In assigning landmark observations to street segments, a landmark bounding region needs to be defined for each street segment. Landmark objects L_j each have their positions represented as $(L_{j_{lon}}, L_{j_{lat}}) \in \mathbb{R}^2$ as does each intersection v_i. However, because longitude and latitude are angular measurements, these points in space need to first be transformed to represent the appropriate distance to be used in landmark observation assignment. To do this, we employ the Haversine formula to obtain a reasonably accurate measure of the shortest distance between two points on a sphere.

Definition 3.14. (Haversine function) For a central angle θ, the Haversine function $\text{hav}(\theta)$ is defined as:

$$\text{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2} \tag{3.10}$$

Definition 3.15. (Haversine formula) Let $\theta = \frac{d}{R}$ be the central angle between any two points p_1 and p_2 on a sphere, where a central angle is the angle between two points whose apex is at the center of the sphere and each point is a distance equal to the sphere's radius away from the apex. Let the distance along the sphere between p_1 and p_2 and the radius of the Earth be d and R, respectively.

Let the longitude and latitude of the points be $p_{i_{lon}}$ and $p_{i_{lat}}$, respectively, for $i = 1, 2$. Then, the Haversine formula is defined as

$$\text{hav}(\theta) = \text{hav}(p_{2_{lat}} - p_{1_{lat}}) + \cos(p_{2_{lat}})\cos(p_{1_{lat}})\text{hav}(p_{2_{lon}} - p_{2_{lon}}) \tag{3.11}$$

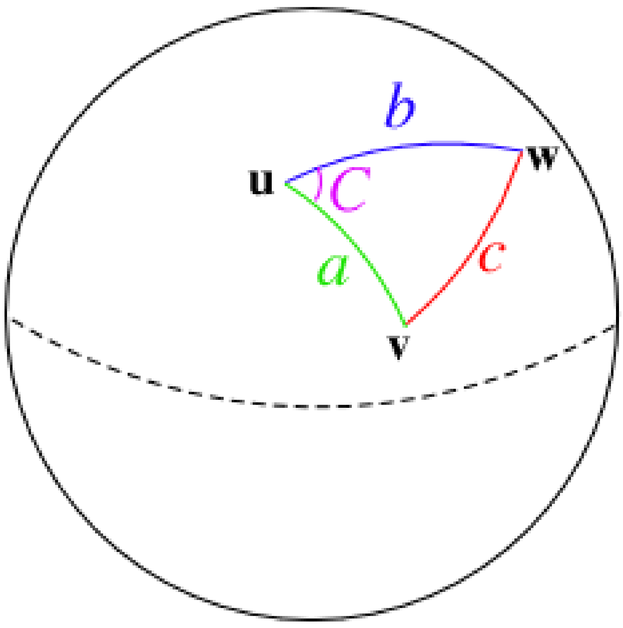


Figure 3.4: Illustration of Haversine function as applied to a sphere's surface. [Wik]

Expanding, the distance d can be obtained with

$$\begin{aligned} d &= R\,\text{archav}(\text{hav}(\theta)) \qquad (3.12)\\ &= 2R\arcsin\left(\sqrt{\text{hav}(p_{2_{lat}} - p_{1_{lat}}) + \cos(p_{2_{lat}})\cos(p_{1_{lat}})\text{hav}(p_{2_{lon}} - p_{2_{lon}})}\right) \qquad (3.13)\\ &= 2R\arcsin\left(\sqrt{\sin^2\left(\frac{p_{2_{lat}} - p_{1_{lat}}}{2}\right) + \cos(p_{2_{lat}})\cos(p_{1_{lat}})\sin^2\left(\frac{p_{2_{lon}} - p_{2_{lon}}}{2}\right)}\right) \qquad (3.14)\end{aligned}$$

The calculated distance along the Earth's surface is a reasonably accurate approximation used in finding locations of the landmark bounding region corners.

It is worth noting there are some cases where two adjacent intersections are not connected by a straight or almost straight road, but rather by a curved road. This is something that can be addressed by comparing the intersections-only-vertices map with the all-vertices map and leaving in vertices from the latter that allow for a series of straight-line segments between all intersections. The idea is to approximate a curve using several straight line segments. This improvement will become important when the method is to be used in a live simulation, but for the purposes of exploring the proposed concept, can be left for future work.

3.2.3 Street segment bearings

So far, only the idea of assigning landmark observations to street segments has been discussed. We now introduce an additional informational feature that can be tied to each street segment, further helping to make landmark observation sequences more unique as well as critically provide a means for modeling the road network line graph with varied sets of inputs depending on the intersection one is currently visiting.

We attempt to realize the idea of discerning possible turn direction from a given intersection by assigning each street segment a bearing direction. The procedure is to find the difference of the vectors given by a destination vertex v_d and a source vertex v_s, then assigning the resulting bearing to the corresponding edge v_{L_i} based on the angle interval it falls within.

Definition 3.16. (Street segment bearing) For each street segment $v_{L_i} \in V_L$, the corresponding directed edge $e \in E$ from the original road network graph has an associated source vertex v_s and destination vertex v_d. The direction of the path from v_s to v_d is obtained from $\vec{b} = v_d - v_s = (\vec{b}_x, \vec{b}_y) \in \mathbb{R}^2$. The resulting angle θ can be obtained using $\theta = \tan^{-1}\left(\frac{\vec{b}_y}{\vec{b}_x}\right)$. Then, the bearing L_i^b represented as a string is assigned to street segment v_{L_i} is piecewise defined as

$$L_i^b = \begin{cases} \text{East} & \theta \in [\frac{15\pi}{8}, 2\pi] \cup [0, \frac{\pi}{8}) \\ \text{Northeast} & \theta \in [\frac{\pi}{8}, \frac{3\pi}{8}) \\ \text{North} & \theta \in [\frac{3\pi}{8}, \frac{5\pi}{8}) \\ \text{Northwest} & \theta \in [\frac{5\pi}{8}, \frac{7\pi}{8}) \\ \text{West} & \theta \in [\frac{7\pi}{8}, \frac{9\pi}{8}) \\ \text{Southwest} & \theta \in [\frac{9\pi}{8}, \frac{11\pi}{8}) \\ \text{South} & \theta \in [\frac{11\pi}{8}, \frac{13\pi}{8}) \\ \text{Southeast} & \theta \in [\frac{13\pi}{8}, \frac{15\pi}{8}) \end{cases}$$

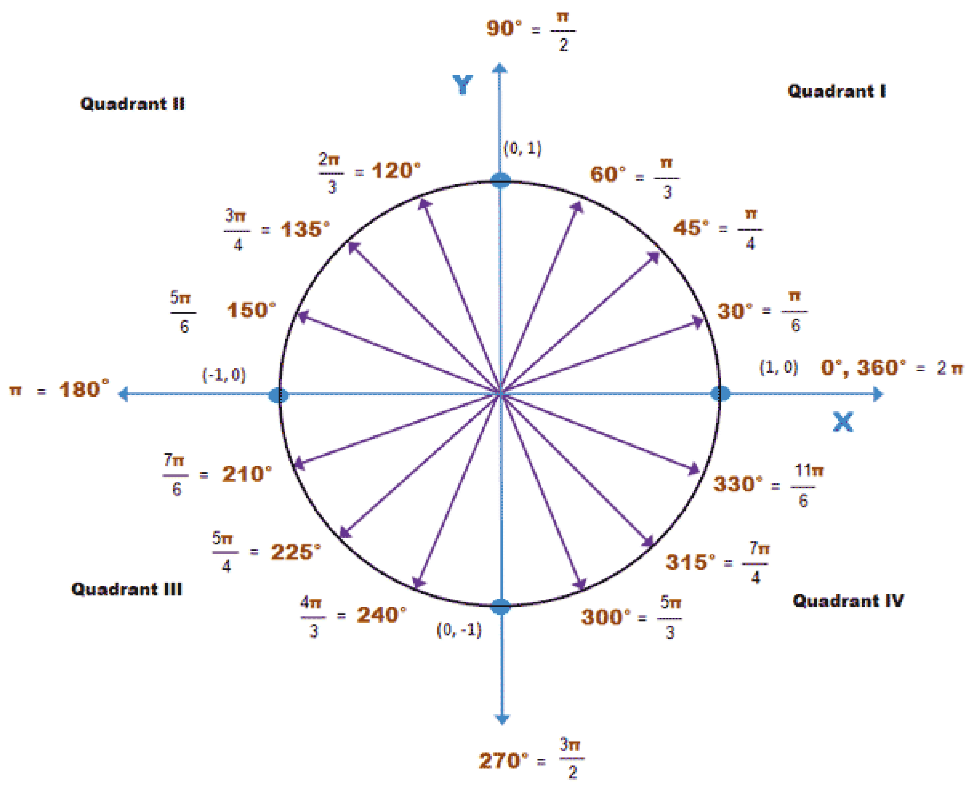


Figure 3.5: Divisions of the unit circle depicting the intervals for bearing assignment. [Shu]

Now, each street segment $v_{L_i} \in V_L$ has assigned to it a landmark observation L_i^{obs} and bearing L_i^b. The bearing of the street segment serves simultaneously as an additional landmark object (that is, a feature used to discriminate it from other street segments) as well as a model for each street segment having its own inputs to determine which street segment to reach next.

The formulation of the road network line graph as a finite state automaton is explored in Chapter 4.

The number of intervals available for a difference vector to be binned into is arbitrary and can be as granular as one's implementation allows.

Although rare, it is possible for a street segment to be adjacent to multiple street segments that each have the same bearing. This can be alleviated by making the bearing assignment intervals more granular, else it can serve to simulate an additional complexity of systems where the input provided could result in transitioning to more than one possible state.

3.3 Landmark code

The landmark code L_C maps landmark observations $L_i^{obs} \in L$ to an alphabet Σ. For preliminary work, we employed a simple to implement and understand landmark code.

Definition 3.17. (Naïve landmark code) Each landmark object class $y_i \in Y$,

$$Y = \begin{Bmatrix} y_1 & = \text{fire hydrant} \\ y_2 & = \text{street light} \\ y_3 & = \text{traffic light} \\ y_4 & = \text{traffic sign} \\ y_5 & = \text{trash receptacle} \end{Bmatrix}$$

is mapped to a single corresponding class code $c_i \in L'_C$,

$$L'_C = \begin{Bmatrix} c_1 & = \mathrm{A} \\ c_2 & = \mathrm{B} \\ c_3 & = \mathrm{C} \\ c_4 & = \mathrm{D} \\ c_5 & = \mathrm{E} \end{Bmatrix}$$

and each landmark observation L_i^{obs} is the set of all landmark objects L_j that fall within the corresponding region. Let the sets $L_{i_z}^{obs}$ for $1 \leq z \leq |Y| = 5$ be the partitions of the landmark observation set into subsets containing only landmark objects of class y_z such that

$$\bigcup_{z=1}^{|Y|} L_{i_z}^{obs} = \mathbf{U} \tag{3.15}$$

$$L_{i_m}^{obs} \bigcap L_{i_n}^{obs} = \emptyset \; \forall \, m \neq n, \; 1 \leq m, n \leq |Y| \tag{3.16}$$

where $\mathbf{U}$ and $\emptyset$ are the universal set and empty set, respectively. Then, let Z be the set of all z that denote a landmark class set $L_{i_z}^{obs}$ with $|L_{i_z}^{obs}| > 0$.

The resulting codeword $L_C(L_i^{obs})$ is initialized as a null string. Then, symbols are appended to it according to

$$L_C(L_i^{obs}) = (c_{z=1} \, |L_{i_1}^{obs}|) \, (c_{z=2} \, |L_{i_2}^{obs}|) \; \cdots \; (c_{z=|Y|} \, |L_{i_{|Y|}}^{obs}|) \; \forall \, z \in Z \tag{3.17}$$

where $(a)(b)$ denotes concatenation.

Remark 3.8. *For example, a landmark observation of 2 fire hydrants, 2 street lights and 9 traffic signs is encoded as A2B2D9. A landmark observation with no landmarks would have a null codeword.*

This code is easy to interpret and implement, and is clearly not focused on optimizing for compression or storage of the codewords and the resulting sequences. When scaling up this approach, it will become increasingly important to encode the observations efficiently.

One simple solution could be to simply enumerate all unique codes and to store a hash table with keys as landmark configurations and values as an integer indicating the symbol.

The focus of the code is to enable fast similarity search so that sequences of codewords can be matched with a landmark map database quickly.

Methods for making localization faster, involving set distance, are discussed in later chapters.

3.3.1 Augmenting landmark diversity

An additional component of a landmark symbol can be the absolute bearing of a street segment. This helps to augment the uniqueness of landmarks, and to compensate for regions that lack landmark data in the dataset. In this work, the absolute bearing is used as part of the landmark.

Another landmark is the length of the road segment, quantized to some bin of length of reasonable robustness to autonomous vehicle odometry measurements. In this work, the road length rounded to the nearest multiple of 5 meters is used as a landmark.

Lastly, yet another landmark that can be discerned is whether or not the street the vehicle is on is a residential street. This is detectable through various cues that can be picked up by an onboard camera.

3.4 Handling observation errors

A practical consideration is to be mindful of observation errors that may occur at localization time. While the first line of defense of making landmark objects observable was in selecting the landmark object classes with that in mind, in a real situation there may be any number of factors that hinder landmarks' observability. Factors ranging from obstructions, construction blocking off parts of street segments, adversarial attacks, the object detection agent (such as a convolutional neural network) making a detection error, or the camera hardware of the vehicle being damaged can all result in observation errors. Here, we begin to formalize some ideas of how to combat such difficulties.

3.4.1 Hamming distance

When comparing codewords, there is a clear intuition as to determining that 1111 is *closer* to 1110 than 0000; the former is more similar to 1111 than the latter. It is useful to have a defined distance metric for quantifying the similarity of codewords. Hamming distance formalizes one approach to such quantification.

Definition 3.18. (Hamming distance) The Hamming distance d_H between 2 codewords x and y of equal length is defined as the number of symbols that differ between them.

$$d_H = |\{i \in \{1, \ldots, |x| \ : \ x_i \neq y_i\}\}| \tag{3.18}$$

This is an important building block for ideas introduced in chapter 5 concerning determining the localization path.

3.4.2 Hamming bound

In the context of observing landmarks, there exists plenty of inherent sources of error that could result in errant observations manifested as an invalid codeword.

In visiting a new street segment, the vehicle could observe codeword $C \notin \Sigma$. The absence of C from Σ is sufficient to detect an errant observation. In this case, the true codeword can be derived from potentially any of the previous street segment's neighbors.

The Hamming distance is a useful quantity in determining conditions for codes such that a given number of errors is guaranteed to be correctable.

Definition 3.19. (Hamming bound) The Hamming bound provides the minimum Hamming distance needed between all pairs of codewords in a code to guarantee correctability of up to t errors. Such Hamming distance must be

$$d_H \geq 2t + 1 \tag{3.19}$$

for all pairs of codewords in the code in order to correct up to t errors.

3.5 Chapter summary

In this chapter, we introduced concepts of what constitutes a landmark and a landmark observation in this work. The types of landmarks used were explained and the dataset from which the relevant information was extracted was discussed. We outlined the benefit of transforming the original road network graph representation to that of its corresponding line graph and explained how landmark observations and street segment bearings were assigned to their street segments. The landmark code used in this work was defined and some thoughts were shared on other approaches to designing the code. Fundamental notions of distance of codewords were introduced through the concept of Hamming distance, which will be built upon in further chapters. This chapter focused on explaining the framework that the landmarks in this work exist in, and provides an important basis for the following chapters.

CHAPTER 4

Current state observer algorithm

Until now, we have laid the foundations for representing an urban area as a road network line graph, with each of its street segments connecting intersections being assigned a codeword resulting from a mapping of observed landmarks. In this chapter, we begin to exploit the framework presented so far to be able to determine the bounds on the number of street segments one needs to visit in order to achieve localization given a sequence of landmark observations.

4.1 Definitions

Here, we begin to lay out a representation of a finite automaton as defined in [CGW91] so that we can relate this concept to the problem.

Definition 4.1. (Finite automaton) Finite automata are quintuples that represent a set of atomic elements of a finite-state system, complete with transition functions and output functions.

$$M(X,U,Y,\phi,\eta) : \text{finite automaton} \tag{4.1}$$

$$X : \text{finite set of states } x \tag{4.2}$$

$$U : \text{finite set of inputs } u \tag{4.3}$$

$$Y : \text{finite set of outputs } y \tag{4.4}$$

$$\phi : \text{transition function } X \times U \to X \tag{4.5}$$

$$\eta : \text{output function } X \to Y \tag{4.6}$$

The set of states X in the context of this problem are the *street segments*, or the *vertices* of the road network line graph. This is a natural choice since now each street segment can be tied to its own inputs and outputs.

For any such state x_i, some set of inputs $u_i \subseteq U$ are possible. In the context of the problem, inputs are *possible turn types.* In other words, for a given street segment, there are turns it can make onto adjacent street segments where each turn is one of the bearings defined in Definition 3.16, such as North or Northwest. Relating this idea to finite state machines, it can be imagined as different inputs making for different resulting states depending on the starting state and input.

The output observed by a state x_i can be thought of as the landmark observation of that state. Each state x_i has a singular output y_i that is its codeword assignment based on landmark observations along that street segment. These are similar to outputs that a system could produce to diagnose [LLH18] its state.

The transition function ϕ maps a state x_i and some action $u \in U$ to another state x_j. For this work, the input to such a transition function would be a street segment and a turn type, producing an output of what street segment one would end up on following that turn type from that initial street segment.

Lastly, the output function η maps a state x_i to its corresponding output y_i, or in the context of this problem, a street segment to its corresponding landmark codeword.

Collecting these re-interpretations together, we can define:

Definition 4.2. (Finite automaton for localization) In the context of this work's localization problem, we define a finite automaton for localization to be

$$\rho(S, T, \Sigma, \phi(s,t), \eta(s)) : \text{finite automaton for localization} \tag{4.7}$$

$$S : \text{finite set of street segments } s \tag{4.8}$$

$$T : \text{finite set of turn types } t \tag{4.9}$$

$$\Sigma : \text{finite set of landmark codes } L_C \tag{4.10}$$

$$\phi(s,t) : \text{transition function } S \times T \to S \tag{4.11}$$

$$\eta(s) : \text{output function } S \to \Sigma \tag{4.12}$$

4.2 Current state observer algorithm

A fundamental difficulty of localizing the vehicle based on a landmark observation is that it may be that more than one location features that landmark observation. This can remain true even for sequences of landmark observations. The current state observer algorithm can provide information that is very useful for being able to localize: lower and upper bounds on the number of observations needed to localize. As seen in the localization algorithm in chapter 5, this information is indeed critical to implementing a solution that can even handle some observation errors.

Essentially, the current state observer algorithm defines a current state observer function ϕ_c which can be implemented as a hash map that maps sets of states and an input/output tuple to another set of states. This allows for O(1) lookup time after the generation of the hash map during pre-processing (before localization time).

We will now present the pseudocode of the algorithm.

Definition 4.3. (Stack) A stack is a data structure that supports a number of standard operations, but for this work only *pushing* and *popping* are relevant. Stacks operate under first-in, last-out order (or equivalently, last-out, first-in), which is to say that the first elements to enter a stack are the last to leave the stack. The operation of inserting an element to a stack is a *push*, and removing an element is called a *pop*. Popping an element from the stack returns the most recently pushed element.

Remark 4.1. *The data structure is analogous to a real-life stack of boxes; the top ones have to be removed before getting to the bottom ones.*

Algorithm 1: Current state observer algorithm

```
Result: Computes the current state observer function $\phi_c$.
Generate $\phi(s,t)$ for all street segments and turn types $s$ and $t$;
Initialize stack states = $[S]$ ;
while states is non-empty do
    $s' =$ states.pop() ;
    for $t \in T$ do
        for $L_C \in \Sigma$ do
            Initialize stack tmp = [ ] ;
            for $s \in \phi(s', t)$ do
                if $\eta(s) = L_C$ then
                    tmp.push($s$);
                end
            end
            $\phi_c(s', (t, L_C)) =$ tmp;
            if $|tmp| > 1$ then
                states.push(tmp);
            end
        end
    end
end
for $s \in S$ do
    for $t \in T$ do
        $s' = \phi(s, t)$ ;
        $L_C = \eta(s')$ ;
        $\phi_c(s, (t, L_C)) = s'$;
    end
end
```

In summary, this algorithm starts by determining what states can be reached from any other state given any pair of turn type and landmark code. For each pair of turn type and landmark code, we can deduce our current state to have gone from the set of all states to a subset of all states. This process is repeated for each new subset of states, each subset of states of size greater than 1 representing an ambiguity in not knowing the single state one is in based on some pair of turn type and landmark code while starting from some other subset of states. Eventually, the first part of the algorithm terminates when the states stack is empty, or equivalently, when there exists a $\phi_c(s, (t, L_C))$ of size 1 for every subset of states with a predecessor in ϕ_c, i.e. for every subset of states s' for which $\phi_c^{-1}(s')$ is defined.

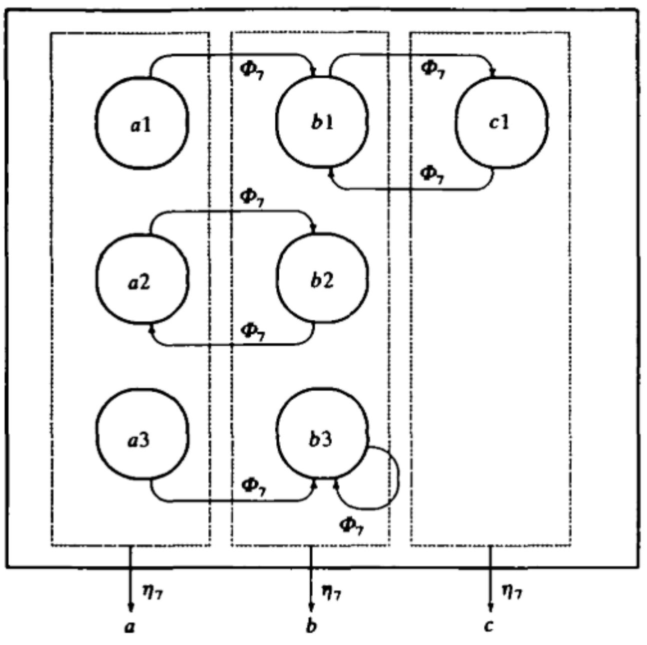
The second part of the algorithm iterates through all the pairs of singleton states and turn types, generating $\phi_c(s, (t, L_C))$ for each of them.

The end result is that $\phi_c(s, (t, L_C))$ can now map any street segment s with a feasible turn type t and landmark code L_C pair to a subset of states s' which can then be iteratively fed into ϕ_c until a singleton state is reached, i.e. one can only reach one state from a given subset of states with a given turn type and landmark code. By taking the number of iterations required to arrive at a singleton state output of ϕ_c given a starting state s, taken over all pairs (t, L_C), one can determine the minimum (and maximum) number of street segments required to be visited from any street segment in order to localize (assuming the vehicle traverses the perfect path by taking the right sequence of turns).

For example, in Figure 4.2 starting from the set of all states, one can determine based on the first observation $\in \{a, b, c\}$ that they must be in one of the subsets $\{a1, a2, a3\}$, $\{b1, b2, b3\}$, or $\{c1\}$, respectively. Given an initial observation of a or b the minimum remaining steps to localize is 1, but if a c is observed then the state is immediately known and the minimum steps required to localize is 0.

4.3 Minimum localization path length n_t

A quantity of interest is the minimum path length required to be able to localize, whether or not in the face of observation errors. This informs several key components of the localization algorithm discussed in the next chapter. First, we examine n_0, the minimum localization



$$\mathcal{M}_7 = \left\{ \begin{array}{l} X_7\colon \{a1, a2, a3, b1, b2, b3, c1\} \\ Y_7\colon \{a, b, c\} \\ \Phi_7\colon \left\{ \begin{array}{l} a1 \mapsto b1,\ b1 \mapsto c1,\ c1 \mapsto b1, \\ a2 \mapsto b2,\ b2 \mapsto a2, \\ a3 \mapsto b3,\ b3 \mapsto b3 \end{array} \right\} \\ \eta_7\colon \left\{ \begin{array}{l} a1 \mapsto a,\ a2 \mapsto a,\ a3 \mapsto a, \\ b1 \mapsto b,\ b2 \mapsto b,\ b3 \mapsto b, \\ c1 \mapsto c \end{array} \right\} \end{array} \right\}$$

Figure 4.1: Example diagram showing an automaton M7, from which its CSO can be derived. [CGW91]

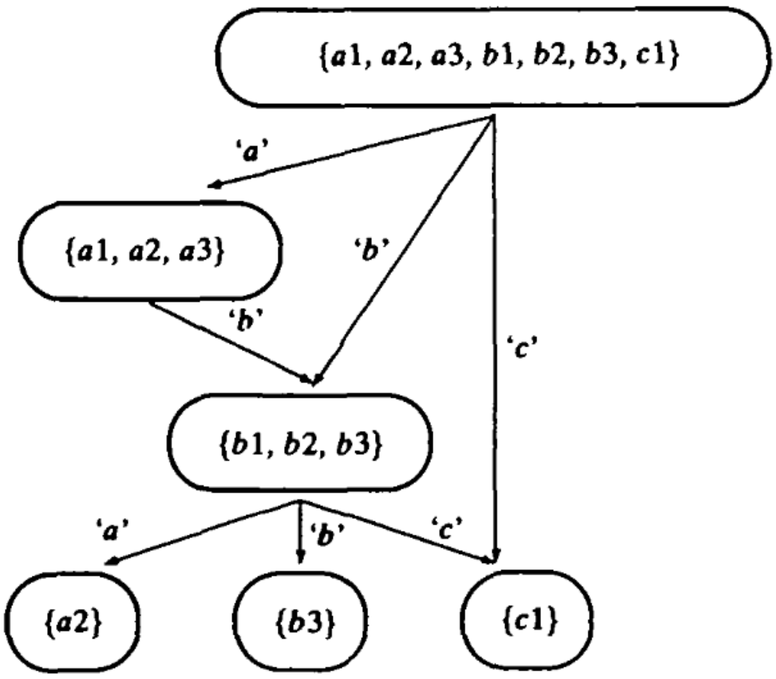


Figure 4.2: Example diagram depicting CSO algorithm output when used on automaton M7 in Figure 4.1. [CGW91]

path length when there are no errors.

Definition 4.4. (Minimum localization path length $n_t|_{t=0}$) Given an observed landmark code L_C and the current state observer ϕ_c for a finite automaton for localization ρ (and the assumption that no observation errors occur), the minimum localization path p_{min}'s length $n_0 = |p_{min}|$ is the minimum depth tree that is an induced subgraph of the graph generated by ϕ_c whose root is the node of this generated graph represented by the subset of states given by the first observed landmark L_C.

From the current state observer, n_0 can be calculated by running a minimum cost path algorithm on the current state observer's graph representation. However, for $t > 0$ errors, calculation of n_t requires more than just the current state observer.

4.4 Chapter summary

In this chapter, we formulated the road network as a finite automaton in order to apply the current state observer algorithm on the road network line graph representation developed. We walked through the comparisons of the original, abstract formulation of a finite automaton to this specific problem. The current state observer algorithm's pseudocode was presented along with a summary of the essentials of how it works and what it is used for. A brief example was shown to illustrate how the minimum localization path length can be determined given some initial observations. In the next chapter, we will bring together everything laid out in this work so far to explain our method for landmark-based localization without using GPS at localization time.

CHAPTER 5

Localization algorithm

The previous chapters laid the groundwork for understanding and implementing the localization algorithm proposed here. We begin by providing context to our contribution by explaining the vanilla minimum cost path problem and notions of set distance. These form the foundations of the two different approaches we attempt. Then, the encoding and decoding of paths that exploits landmark configuration information will be presented.

To contextualize this chapter, we begin with a theorem.

Theorem 5.1. *Given that all codewords of length n_t in an alphabet are uniquely decodable, and that V_{city}, E_{city} are the sets of vertices (street segments) and edges (intersections) in a graph representation of a city, respectively, there exists an algorithm with time complexity $O(|V_{city}| \log |V_{city}| + |E_{city}| \log |V_{city}|)$ that enables the unique decoding of an observed codeword represented as a path in the city graph.*

5.1 Minimum cost path problem

Traditionally, a graph can have *costs* associated with its edges. Intuitively, for a road network graph with cities represented as vertices and highways between them represented as edges, the cost of an edge can be analogous to the distance between the cities it connects. Thus, it can be desirable to determine the minimum cost path between two given points.

Definition 5.2. (Minimum cost path problem) Given a graph (directed or not) with costs for each edge, the minimum cost path problem is concerned with finding the path between a source vertex v_s and destination vertex v_d such that the cost of edges used in the path between the two vertices is the minimum among all paths between the two vertices.

$$\arg\min_{e' \in E} \sum_{i=1}^{|e'|} c_i \ , \text{ edge with cost } c_i \in e' \tag{5.1}$$

Here, $e' \in E$ are all the feasible sequences of edges e' in the set of all edges E, and c_i is the cost of the edges in e'.

This is a classic problem and has had many approaches proposed for solving it given different sets of constraints.

In the context of this problem, we attempt to apply the minimum cost path problem as a decoder to the problem of choosing the encoding with the minimum error among all encoding, i.e. the most likely encoding given the observed landmark code sequence.

5.2 Set-based code

Before addressing the construction of the graph to be used for localization, we introduce the definition of modified set distance.

Definition 5.3. (Modified set distance) Building upon Hamming distance d_H from equation (3.18), modified set distance d_{S_1,S_2} measures the minimum Hamming distance between any pair of elements where each element belongs to a different set S_1 or S_2:

$$d_{S_1,S_2} = \min_{p_{1_i}, p_{2_j} \in S_1 \times S_2} d_H(p_{1_i}, p_{2_j}) \tag{5.2}$$

for all $p_{1_i} \in S_1$ and all $p_{2_j} \in S_2$, $i = 1, 2, \ldots, |S_1|$, $j = 1, 2, \ldots, |S_2|$.

This quantity is not strictly a distance metric because $d_{S_1,S_2} = 0$ does not imply that $S_1 = S_2$; it can only be said that there exists an element that is in both S_1 and S_2. For the purposes of this work, we interpret this to mean that the sets are effectively identical because observing a path that is in both sets results in ambiguity during decoding, preventing localization.

Using modified set distance (from here on referred to as set distance), we can create a new set-based code C_S with as many codewords as vertices in the line graph $L(\mathbf{G})$.

By leveraging the fact that we only care about determining the position of the vehicle, and not the path it took to get there, we can use set distance and a set-based code to enable efficient online localization.

Definition 5.4. (Set-based code) The set-based code C_S consists of $|V_L|$ set codewords c_i, which are each non-empty sets consisting of length n path codewords $L_C(L_j^{obs})_i$. Each path and its corresponding landmark observation L_j belong to a set codeword c_i and ends at vertex v_i.

It follows that each pair of set codewords has their respective set distance. By ensuring that the length of path codewords in each set codeword is large enough such that the set distance between all pairs of set codewords is at least $2t+1$, by the Hamming bound, paths with up to t errant observations can be corrected and localization can be achieved.

If the minimum set distance among the set-based code is $d^*_{S_1,S_2}$, there does not need to be any Hamming distance constraints between pairs of path codewords that belong to the same set codeword. Intuitively, this is because we do not care to distinguish between paths to the same destination. This helps to lower the minimum path length required to achieve desirable error-correcting capabilities compared to if every pair of codewords were required to have some minimum Hamming distance.

Put another way, if we can determine that the path we observed belongs to a particular set codeword, we can determine the location of the vehicle.

5.3 Constructing a graph for localization

When navigating a path online, the vehicle is faced with the problem of being able to efficiently compare the path it observes to the offline-prepared alphabet of valid paths so that the vehicle can localize. However, brute force methods are prohibitively time-intensive for problems of this scale and a different method needs to be employed for efficient real-time localization.

To efficiently determine the candidate paths, we employ a dense graph that can be described in terms of layers akin to common deep neural network architectures used for regression and high-level feature extraction.

The graph to be constructed can be described by the following:

Definition 5.5. (Path searching graph) The vertices of the path searching graph are organized in layers of depth. The path searching graph is directed. There is a single source vertex v_{source}, n_t layers of $|V_L|$ vertices each, and finally a single sink vertex v_{sink}. From v_{source} there is a single edge to each of the vertices in the first layer. From layer i to layer $i+1$, $i = 1, 2, \ldots, n_t - 1$ the vertices are fully connected such that each vertex has a single edge to each of the next layer's vertices. From the final layer, there is one edge from each of the final layer vertices to the sink vertex v_{sink}.

What we have seen so far is a method of encoding paths in a city as codewords made up of symbols that correspond to landmark configurations. By leaving the paths as is, we are ready to use the path searching graph to decode the paths and localize. By grouping the paths into sets based on the destination location, we can be ready to decode the paths and localize. Next, we will describe both decoding (localization) methods.

5.4 Applying minimum cost path algorithm for localization

By applying the minimum cost path algorithm to a path searching graph, we can receive the set of paths that have minimum cost, or equivalently, the set of paths that are most similar to the path that was observed at localization time. In addition, there are two potential early termination conditions:

- If any unique landmark is observed
- If any unique path is observed (consisting of landmarks that are not necessarily unique)

Algorithm 2: Min-cost path localization algorithm

```
Result: Returns estimate of current position.
Let $G$ be the path search graph;
Let $x$ be the $n_t$-length path codeword observed during localization;
Initialize $i = 1$;
for symbol $x_i \in x$ do
    for all edges in layer $i$ of $G$ with symbol $= x_i$ do
        Set edge weight $= 0$;
    end
    if all newly-zeroed edges pass through the same vertex then
        Return that vertex's location;
    end
    Increment $i$;
end
Let $p = MCP(G)$, where $MCP$ is a minimum cost path-finding algorithm;
Return $p$;
```

This algorithm finds the minimum cost paths resulting from zeroing the weights of edges in the path searching graph layer-by-layer that correspond to a symbol in the observed codeword. If the algorithm is terminated early, it's because a unique landmark was encountered at some point during the observation. Else, the resulting set of minimum cost paths is returned. If the set is a singleton, localization is achieved. This is guaranteed to be the case if there are a sufficient number of layers (n_t) to account for up to t observation errors.

5.5 Applying set-based code in localization algorithm

If the initial encoding of the paths in the localization region is processed with an additional step of grouping the paths into sets such that each set consists only of paths that end at the same location, we can try this approach to localization.

Algorithm 3: Set-based localization algorithm

```
Result: Returns estimate of current position.
Let $x$ be the $n_t$-length path codeword observed during localization;
Initialize $i = 1$;
for symbol $x_i \in x$ do
    if $x_i$ is unique then
        Return corresponding location;
    end
end
Initialize $d_{min} = \infty$;
for set in set-based code do
    for path in set do
        Calculate $d_H(path, x)$;
        Update $d_{min}$ if new $d_H < d_{min}$;
    end
end
Return location of set that gave $d_{min}$;
```

This algorithm computes the Hamming distance of the observed path and all paths. The set that gave the lowest distance is the set that should contain the observed path, and its corresponding location is returned. Determining d_{min} can be sped up significantly by terminating the calculation of d_H early if the Hamming distance reaches the previous minimum Hamming distance, moving onto the next set.

5.6 Complexity analysis

The complexity analysis will be split into two distinct halves: offline (preparations that are done once before the vehicle is to be deployed), and online (routines run multiple times as the vehicle tries to localize).

5.6.1 Offline

For both localization methods, n_t needs to first be determined. Deciding on the value of t is a design choice that should depend on factors such as the robustness of the landmark detection components of the vehicle (e.g. object detection system, odometry, etc.) and the expected amount of landmark obfuscation (a more traffic-dense area is more liable to landmarks being obfuscated).

If $t = 0$, n_0 can be determined by finding the maximum minimum path length from the CSO root node to a singleton node. Let $|V|$ be the number of vertices in the CSO graph, and $|V_{city}|$ the number of vertices in the city graph. The time complexity of such a task is the same as that of, for example, Dijkstra's shortest path algorithm applied to $|V_{city}|$ paths, $O(|V_{city}||V| \log |V| + |V_{city}||E| \log |V|)$.

For $t > 0$, n_t was found experimentally for a given region by incrementing path length and including more landmarks to help diversify the symbols collected.

Once n_t is determined for your choice of t, all paths of length n_t in the city graph of interest need to be found. This step is $O(|V_{city}|^{n_t})$ in time and memory complexity (as the number of paths of length n_t is at most $|V_{city}|^{n_t}$), but is only done once and offline. Any additional paths of length $n'_t > n_t$ can be found more quickly using previously found paths and a dynamic programming-based approach.

For the minimum-cost-path-based approach, the path searching graph needs to be stored. The memory complexity is based on the number of vertices and edges: $O(2 + n_t|V_{city}|) + O(2|V_{city}| + (n_t - 1)|V_{city}|^{|V_{city}|}) = O(|V_{city}|) + O(|V_{city}|^{|V_{city}|}) = O(|V_{city}|^{|V_{city}|})$.

For the set-based approach, negligible extra memory is needed with an implementation that assigns a set flag to each existing path based on its ending location.

5.6.2 Online

For the minimum-cost-path-based approach, finding the shortest path in the path searching graph is just a matter of applying a shortest path algorithm such as Dijkstra's algorithm, requiring time complexity of $O(|V_{city}| \log |V_{city}| + |E_{city}| \log |V_{city}|)$.

For the set-based approach, a Hamming distance calculation between the observed path and all other paths is required. This results in a worst-case time complexity of $O(n_t|V_{city}|^{n_t})$ because each Hamming distance calculation takes $O(n_t)$ for each path. However, in practice the average case is significantly faster and approaches $O(n_t|V_{city}|)$ as the minimum Hamming distance calculated approaches 0 because Hamming distance calculations can be terminated early once the Hamming distance being calculated for the current pair of codewords meets the last minimum Hamming distance calculated.

5.7 Representing observation errors in localization methods

In order to correct an observation error, the error must first be detected. This can be accomplished by recognizing that the symbol created through a street segment does not exist in the alphabet of valid symbols, or by recognizing that the newest symbol observed results in an invalid path (a path that does not exist in the map).

In the event of detecting an error, such an error is handled accordingly by each localization method.

For the minimum-cost-path method, no edge weights are zeroed for the layer corresponding to the errant symbol observed.

For the set-based method, the errant symbol is replaced by a special symbol that represents an observation error: this symbol will not have appeared as a landmark configuration anywhere else in the ground truth landmark map.

These considerations allow the algorithms to continue to function as expected even in the face of observation errors, given that n_t is sufficiently large.

5.8 Chapter summary

In this chapter, we presented the localization algorithms used which allows one to self-localize without GPS given a sequence of landmark observations, a landmark map, and its corresponding current state observer function or collection of valid, appropriate-length paths. We applied concepts of minimum cost path algorithms and set distances to demonstrate the localization algorithms. We performed a complexity analysis to demonstrate their scalability, to be applied to localizing in larger areas. We also clarified how observation errors are represented using these localization methods.

CHAPTER 6

Experiments

Having presented the method this work proposes, we now turn to explaining the design and results of experiments used in demonstrating the feasibility of our approach. We begin with preliminary experiments as they were instrumental in framing expectations and in honing intuition moving forward, and they may also be worthwhile points to fork at when extending this work. Then, we explain the experiments of the main work and detail their results.

6.1 Preliminary experiments

In this section we discuss and present some experiments done to test the feasibility of some ideas that were expanded on.

6.1.1 Experiment design and setup

A simulation was designed to gather empirical data about the effectiveness of localizing with various degrees of landmark code uniqueness across all street segments. The goal of this preliminary experiment was to evaluate the feasibility of the work, and to possibly uncover ideas, patterns or problems not yet considered.

The outline of a trial is as follows:

1. Randomly generate a directed graph. These graphs are meant to simulate different road networks.

2. Assign landmark codes to the graph edges. Again, to simulate assigning real landmark codes based on landmark observations to street segments.

3. Generate a random walk on the graph. This results in an observed sequence of landmark codes.

4. Measure the steps required to localize. Do this for each graph, for a set code uniqueness percentage.

6.1.1.1 Random directed graph generation

The Erdős–Rényi model [ER59] was used to generate the directed graphs. For a given vertex set and number of edges, each graph using the given vertices and number of edges are equally likely. This graph generation method is comparable to a uniform distribution across the set of possible graphs. In other words, given a vertex set V and fixed number of edges $|E|$, a graph $G(V, E)$ was drawn from the discrete uniform distribution $\mathcal{U}$ consisting of the set of all graphs that can be made with the given vertex set and number of edges $G_{V,|E|}$:

$$G(V, E) \sim \mathcal{U}\{G_{V,|E|}\} \tag{6.1}$$

In an attempt to match realistic degree distributions of road networks, the number of edges used was $2|V|$ where $|V|$ is the size of the vertex set (which varied with the experiments).

6.1.1.2 Landmark code assignment for simulation

To simulate varying degrees of landmark code repetition, we defined a metric to measure the notion.

Definition 6.1. (Unique code percentage) Let the probability measure $\mathbb{P}(L_{C_i})$ be defined as the probability that L_{C_i} is the landmark code of a visited vertex v_i, independent of any past visited vertices. Then let the unique code percentage U be:

$$U = \frac{|\{i : \mathbb{P}(L_{C_i}) = \frac{1}{|V|}\}|}{|V|} \times 100\% \tag{6.2}$$

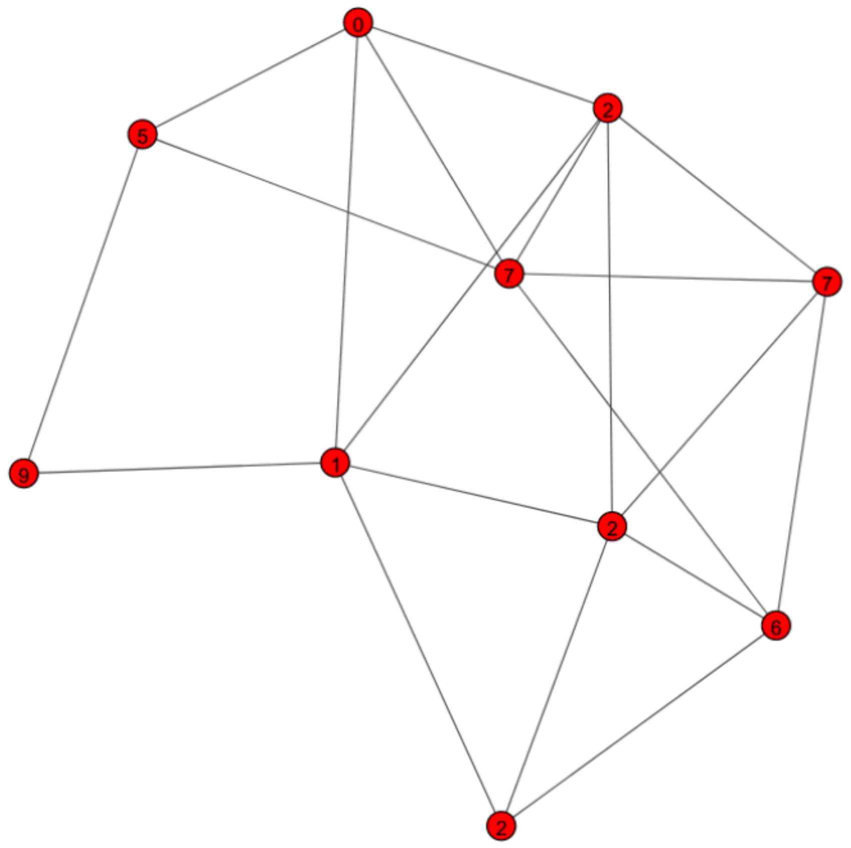


Figure 6.1: Toy example of a randomly graph with 10 vertices; assigned codewords are integers and are displayed atop the vertices. Here, $U = 50\%$.

6.1.1.3 Generating a random walk

To begin, a start vertex was chosen uniformly at random. Then, the next step in the walk was decided by choosing again uniformly at random from the vertex's incident edges. The edge was followed to the other vertex and the process was repeated until localization.

6.1.1.4 Measuring steps needed to localize

Localization here is intentionally defined in a way that is not dependent on the entire past sequence of observations, only the immediately previous step (i.e., treated as a Markov process). The purpose of this was to have this preliminary experiment serve as a lower bound of performance.

Localization in this preliminary experiment is defined as the event that a step in a walk reaches a unique code.

6.1.2 Results and discussion

For a set number of 10^4 vertices, a random walk was generated until localization was achieved for various values of U, $u \in \{95\%, 85\%, 75\%, 50\%, 40\%, 30\%, 20\%, 10\%\}$. For each value of U, 1000 trials were done and the results were averaged over the trials. The resulting histograms (with horizontal axis representing number of steps to localization) were normalized so they function as probability density functions.

From Figure 6.2, it can be seen that with at least $U = 20\%$ unique codewords, localization can be achieved within 100 steps, on 95% of the random walks on random graphs. This optimistic result indicates that once the main version of the algorithm is applied, we can expect localization performance to improve. While 100 steps is a long distance for an autonomous vehicle to have to drive, this result functions as a sort of lower bound of performance. With realistic (greater) diversity in symbols we can expect to be able to localize in fewer steps.

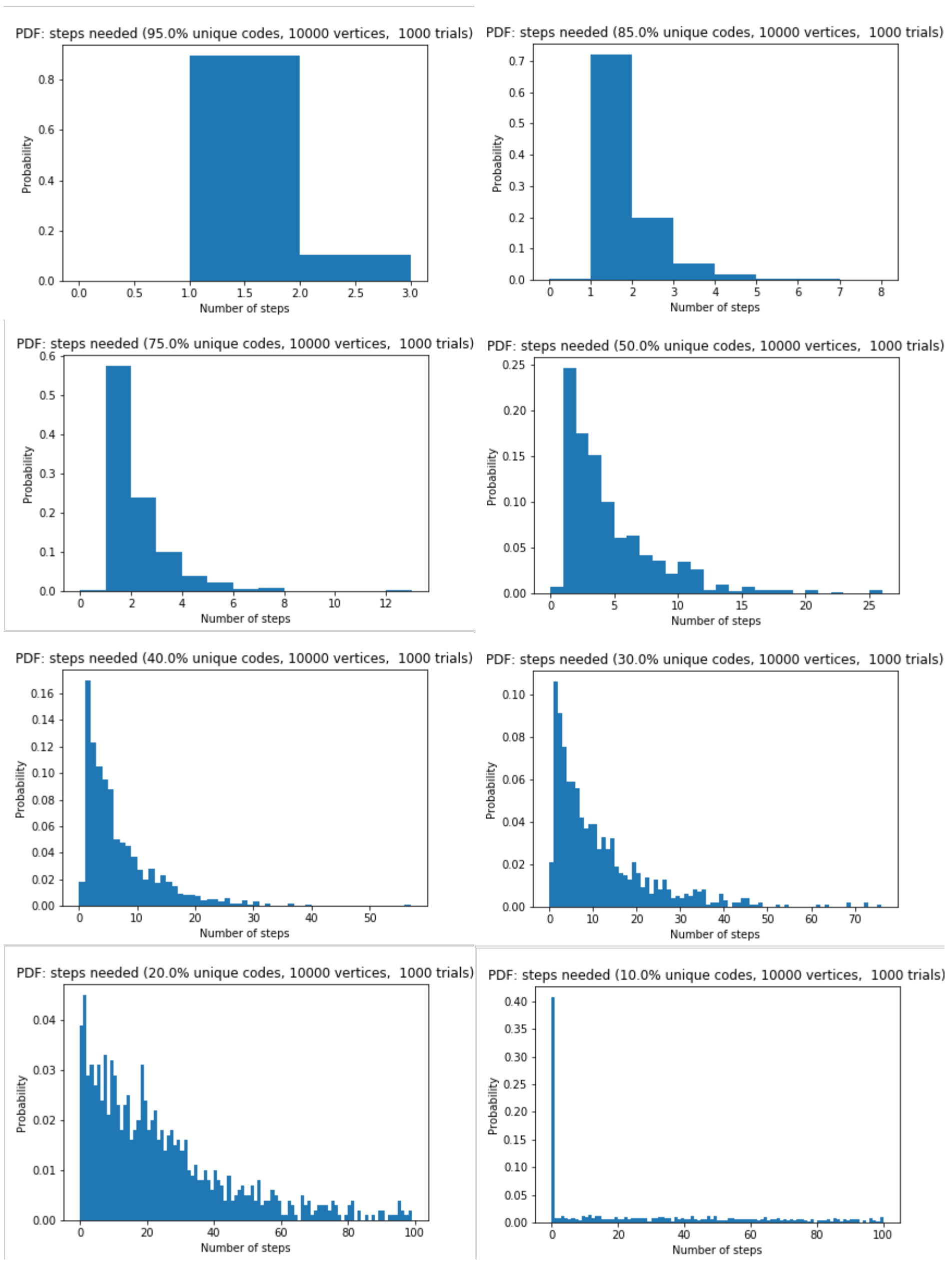


Figure 6.2: Probability density functions for varying values of U as a function of the number of steps required to localize. 1000 randomly generated graphs with 10^4 vertices each had their results averaged and plotted here.

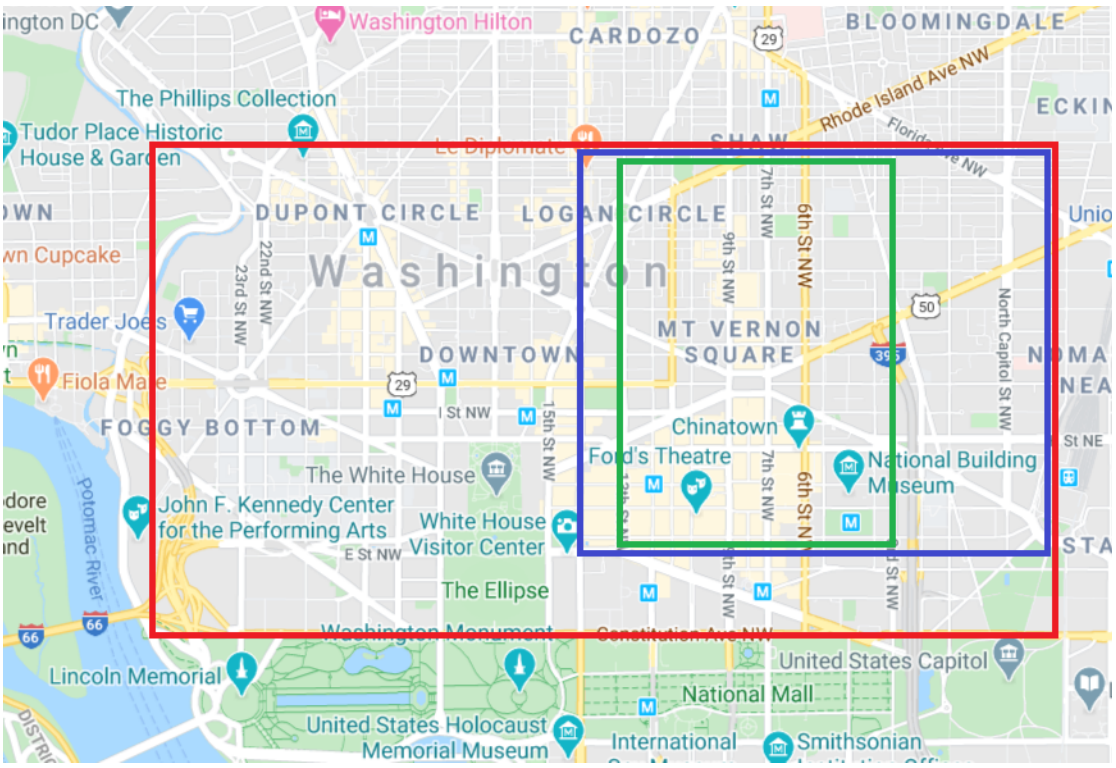


Figure 6.3: Red, blue, and green rectangles showing the regions of the large, medium, and small-sized maps, respectively. Original image capture from Google Maps.

6.2 Main experiments

The main experiments used in this work to verify the effectiveness of the proposed method consist of applying the two localization methods outlined in the previous chapter to real-world data and measuring how distance required to travel for localization is affected by the number of observation errors we wish to be tolerable.

6.2.1 Experiment design and setup

To create the landmark code, landmark location data is required. To this end, we use the dataset from [AZT14] which comprises a region of Washington, D.C. The OpenStreetMap data we use is constrained to the data that falls within the region covered by the aforementioned dataset. From this coverage, we a large rectangular area that covers the majority to be the large-size map, and take two smaller subsets to be the medium and small-sized maps.

Large	Medium	Small
10.097 km^2	3.257 km^2	2.306 km^2

Table 6.1: Area of each map region.

Large	Medium	Small
4102 nodes	1171 nodes	697 nodes

Table 6.2: Number of nodes in line graph of each map region.

Both localization methods were tested on the maps of each size, and various results were collected.

6.2.2 Results and discussion

First, we examine the results of experimentally finding n_t, the minimum path length necessary to guarantee localization even with up to t observation errors. We compare results of the set-based and minimum-cost path-based methods.

(See Figures 6.4, 6.5, 6.6) For the full- and medium-sized maps, the set-based method cleanly out-performs the minimum-cost path-based method. This is due to how the set-based method requires only a minimum set distance constraint to be met, which is not as harsh as a minimum Hamming distance constraint over all codewords, which is the case for the minimum-cost path-based method. For the smallest map size, the performance of both methods was identical.

Next, for a set codeword length of 15, a node in the full-size map was uniformly randomly selected as the start of a 15-street-segment walk. A set amount (from 0 to 5 inclusive) of random symbols in the walk were treated as observation errors in the observed codeword and localization was attempted using the set-based method. This was repeated for 1,000,000 trials for each of the amounts of errors from 0 to 5. The purpose of this experiment was to demonstrate the method's effectiveness on average (instead of worst-case, which was highlighted in the previous experiment) using available data to construct and observe landmarks.

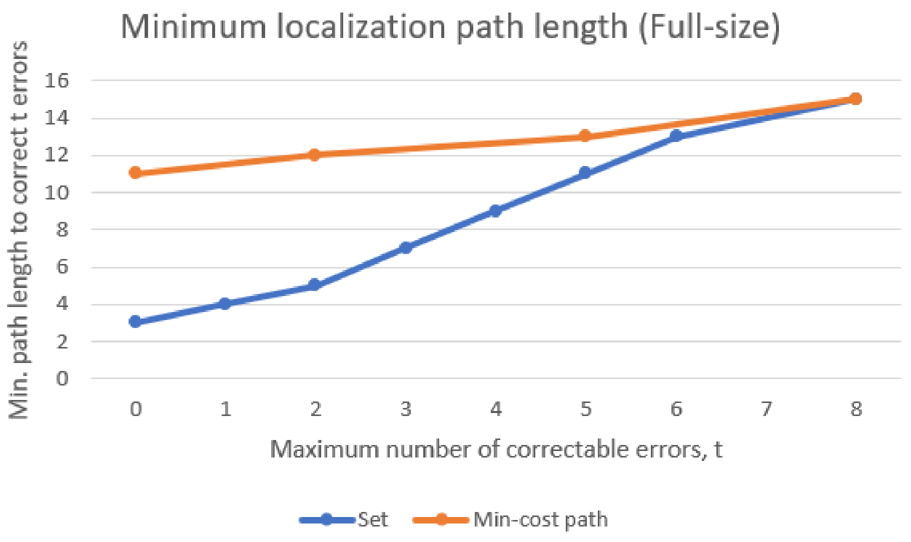


Figure 6.4: Minimum path lengths necessary for localization for different values of t (maximum observation errors to correct and still localize); full-size map.

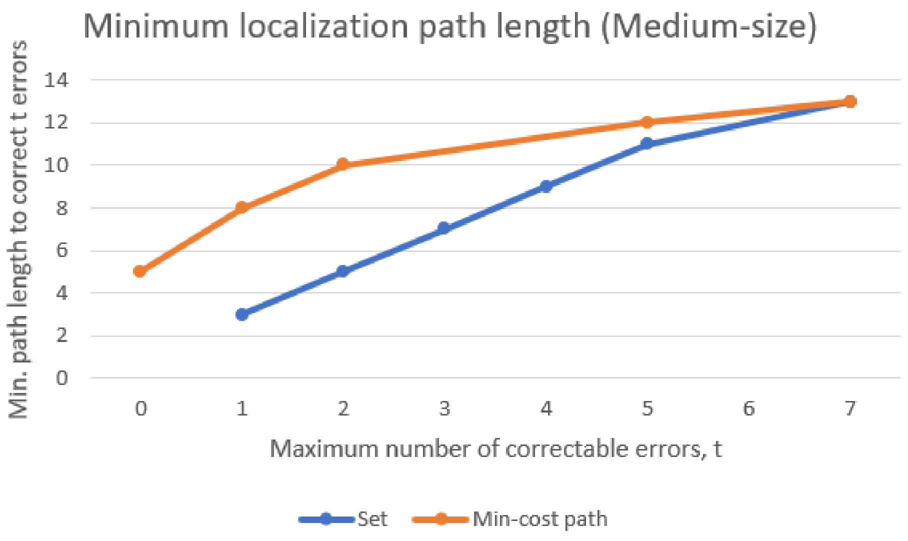


Figure 6.5: Minimum path lengths necessary for localization for different values of t (maximum observation errors to correct and still localize); med-size map.

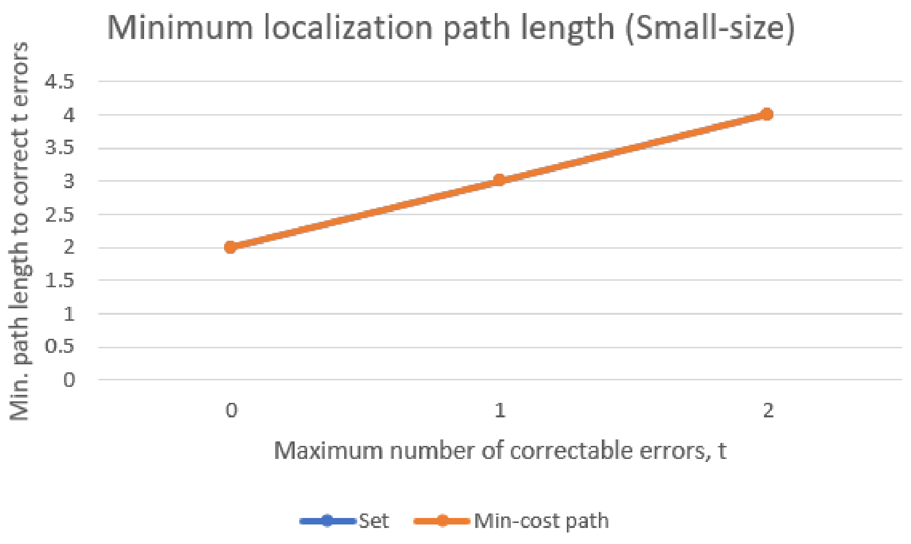


Figure 6.6: Minimum path lengths necessary for localization for different values of t (maximum observation errors to correct and still localize); small-size map.

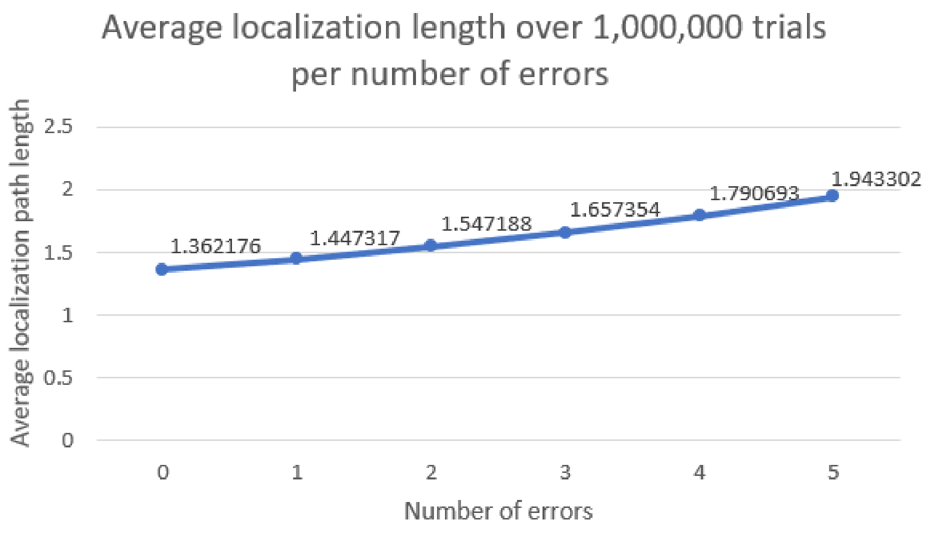


Figure 6.7: Average path length required to localize using a length 15-codeword in the full-size map. For each number of errors, 1,000,000 trials were used to obtain the average.

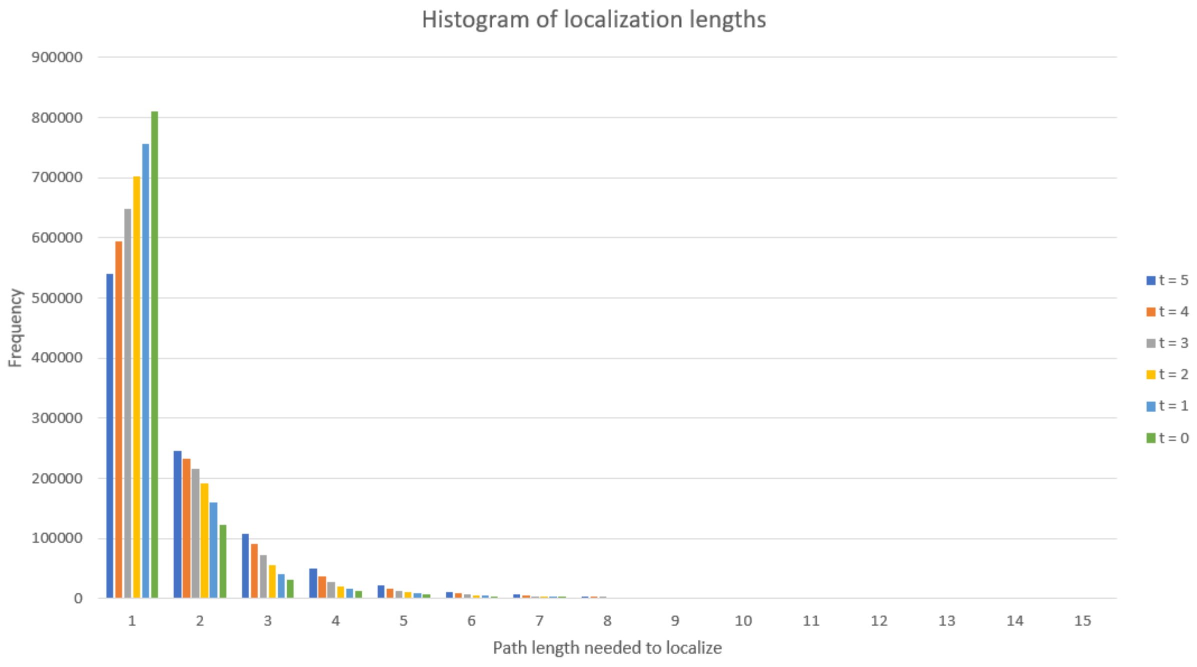


Figure 6.8: Histograms of localization path length required to localize for different numbers of errors t in the observed 15-length codeword.

(See Figure 6.7) Experimentally, for even 5 errors randomly corrupting the observed codeword, the set-based method is able to localize in less than 2 street segments traversed on average.

Lastly, the histograms over these sets of 1,000,000 trials for each number of errors t is shown to give a more full picture of how the localization lengths were distributed in practice. This is, again, on the full-sized map.

(See Figure 6.8) Less than 5% of the cases required 5 or more street segments to be traversed to localize using the set-based method.

Overall, these results using real data prepared using the pipeline outlined in this thesis proved to be effective in quickly localizing an autonomous vehicle without requiring GPS at localization time. The set-based method was shown to be able to localize faster than the minimum-cost path based method. With achievable landmark diversity and observability, these results show that this method can be used in the event of an autonomous vehicle's localization systems being compromised.

6.3 Chapter summary

This chapter presented the design and results of experiments used over the course of this work, from preliminary work to the main work that the former helped to shape. Experimental results using real data showed that the methods proposed can accurately and efficiently localize even when multiple observation errors are made.

CHAPTER 7

Conclusion

With autonomous vehicles steadily approaching full autonomy, the safety and robustness of the systems that help it make decisions are paramount. In an effort to improve their robustness, we proposed an approach to self-localization of an autonomous vehicle using landmark objection via car-mounted camera, without the need for GPS at localization time.

By representing a road network as a graph, landmark observations could be encoded and assigned to different street segments. Sequences of landmark codes can become unique when of sufficient length, as dictated by the current state observer function that can be derived for any directed graph and/or by experiment when wanting to localize even with errors.

Using both a set-based method and a minimum-cost path-based method allowed the proposed method to successfully localize using real road network and landmark data with great online efficiency and accuracy even with multiple observation errors.

Preliminary results have shown that even a lower bound of performance can achieve reasonable localization results to the tune of being able to localize in a road network with 10^4 vertices and only 20% of the landmark codes being unique, in 95% of simulated trials.

Our main results employing the localization algorithm detailed herein achieved an average localization length of 1.94 street segments traversed over 1,000,000 trials when the observed codeword is corrupted with 5 observation errors. In addition, for a region of Washington D.C. with area of over 10 km^2, our set-based method was able to correct up to 5 errors given any 11-street-segment-long sequence of observations.

Overall, the method proposed in this work demonstrates a feasible and scalable solution to localizing an autonomous vehicle without access to GPS at localization time.

One question to address in the future is how to fully exploit the fact that the original city graphs (not necessarily their line graphs) are planar graphs. A potential direction is to assign information to the facets that are formed by surrounding edges; perhaps by looking at facet-based sequences and codewords instead of street-segment-based sequences, sizeable factors of memory could be saved in shrinking the size of the landmark map and corresponding walk dictionary.

Another open question is in improving the time complexity of the offline one-time setup processes. To handle larger regions, more vertices, walks and encodings will be necessary (as will longer minimum localization lengths). Implementing and analyzing a dynamic programming-based method of increasing path lengths from prior stored paths could be important in significantly cutting down running time and memory in implementations.

REFERENCES

[An] Andreas "AndGem". "OsmToRoadGraph." Website. https://github.com/AndGem/OsmToRoadGraph.

[AZT14] Shervin Ardeshir, Amir Roshan Zamir, Alejandro Torroella, and Mubarak Shah. "GIS-assisted object detection and geospatial localization." *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **8694 LNCS**(PART 6):602–617, 2014.

[BG13] Margrit Betke and Leonid Gurvits. "Mobile Robot Localization using Landmarks." (May 1997), 2013.

[BGU16] Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. "Map-based probabilistic visual self-localization." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **38**(4):652–665, 2016.

[BI17] I. Bukhori and Z. H. Ismail. "Detection strategy for kidnapped robot problem in Monte Carlo Localization based on similarity measure of environment." *USYS 2016 - 2016 IEEE 6th International Conference on Underwater System Technology: Theory and Applications*, pp. 55–60, 2017.

[BN18] Bianca Cerasela Zelia Blaga and Sergiu Nedevschi. "A method for automatic pole detection from urban video scenes using stereo vision." *Proceedings - 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing, ICCP 2018*, pp. 293–300, 2018.

[CGW91] Peter E. Caines, Russell Greiner, and Suning Wang. "Classical and logic-based dynamic observers for finite automata." *IMA Journal of Mathematical Control and Information*, **8**(1):45–80, 1991.

[dBA19] Lucas de Paula Veronese, Claudine Badue, Fernando Auat Cheein, Jose Guivant, and Alberto Ferreira De Souza. "A single sensor system for mapping in GNSS-denied environments." *Cognitive Systems Research*, **56**:246–261, 2019.

[EHH19] Nico Engel, Stefan Hoermann, Markus Horn, Vasileios Belagiannis, and Klaus Dietmayer. "DeepLocalization: Landmark-based Self-Localization with Deep Neural Networks." 2019.

[ER59] Paul Erdős and Alfréd Rényi. "On Random Graphs I." *Publicationes mathematicae*, **6**(26):290–297, 1959.

[FMC13] Jonathan Fabrizio, Beatriz Marcotegui, Matthieu Cord, Jonathan Fabrizio, Beatriz Marcotegui, Matthieu Cord, Jonathan Fabrizio, Beatriz Marcotegui, and Matthieu Cord. "Text detection in street level images To cite this version : HAL Id : hal-00906841 Text detection in street level images." **16**(4):519–533, 2013.

[FVL13] Georgios Floros, Benito Van Der Zander, and Bastian Leibe. “OpenStreetSLAM: Global vehicle localization using OpenStreetMaps.” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1054–1059, 2013.

[GBI13] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. “Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.” pp. 1–13, 2013.

[HDV18] Simon Hecker, Dengxin Dai, and Luc Van Gool. “End-to-end learning of driving models with surround-view cameras and route planners.” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **11211 LNCS**:449–468, 2018.

[JSV16] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Reading Text in the Wild with Convolutional Neural Networks.” *International Journal of Computer Vision*, **116**(1):1–20, 2016.

[KHK19] Achim Kampker, Jonas Hatzenbuehler, Lars Klein, Mohsen Sefati, Kai D. Kreiskoether, and Denny Gert. “Concept study for vehicle self-localization using neural networks for detection of pole-like landmarks.” *Advances in Intelligent Systems and Computing*, **867**:689–705, 2019.

[KK89] Tomihisa Kamada and Satoru Kawai. “An Algorithm for Drawing General Undirected Graphs.” *Information Processing Letters*, **1**(31):7–15, 1989.

[LLH18] Stéphane Lafortune, Feng Lin, and Christoforos N. Hadjicostis. “On the history of diagnosability and opacity in discrete event systems.” *Annual Reviews in Control*, **45**:257–266, 2018.

[LLZ10] Mingyong Liu, Xiaokang Lei, Siqi Zhang, and Bingxian Mu. “Natural landmark extraction in 2D laser data based on local curvature scale for mobile robot navigation.” *2010 IEEE International Conference on Robotics and Biomimetics, ROBIO 2010*, pp. 525–530, 2010.

[LT10] Jesse Levinson and Sebastian Thrun. “RobustVehicleLocalizationinUrbanEnvironmentsUsingProbabilisticMaps.pdf.” pp. 4372–4378, 2010.

[MD04] R. Madhavan and H. F. Durrant-Whyte. “Natural landmark-based autonomous vehicle navigation.” *Robotics and Autonomous Systems*, **46**(2):79–95, 2004.

[NVT08] P. Núñez, R. Vázquez-Martín, J. C. del Toro, A. Bandera, and F. Sandoval. “Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation.” *Robotics and Autonomous Systems*, **56**(3):247–264, 2008.

[OSM] “Open Street Map.” Website. https://www.openstreetmap.org/.

[Ove] “Overpass Turbo.” Website. https://overpass-turbo.eu/.

[PC18] Pilailuck Panphattarasap and Andrew Calway. "Automated Map Reading: Image Based Localisation in 2-D Maps Using Binary Semantic Descriptors." *IEEE International Conference on Intelligent Robots and Systems*, pp. 6341–6348, 2018.

[pyt] "python-igraph." Website. https://igraph.org/python/.

[RSR15] Philipp Ruchti, Bastian Steder, Michael Ruhnke, and Wolfram Burgard. "Localization on OpenStreetMap data using a 3D laser scanner." *Proceedings - IEEE International Conference on Robotics and Automation*, **2015-June**(June):5260–5265, 2015.

[SGR16] Robert Spangenberg, Daniel Goehring, and Rául Rojas. "Pole-based localization for autonomous vehicles in urban scenarios." *IEEE International Conference on Intelligent Robots and Systems*, **2016-Novem**:2161–2166, 2016.

[Shu] W. Shujen. "Unit Circle." Website. https://www.wyzant.com/resources/lessons/math/trigonometry/unit-circle.

[Spa15] Robert Spangenberg. "Landmark-based Localization for Autonomous Vehicles." p. 115, 2015.

[TB00] Tieniu N. Tan and Keith D. Baker. "Efficient image gradient based vehicle localization." *IEEE Transactions on Image Processing*, **9**(8):1343–1356, 2000.

[VNB09] Ricardo Vázquez-Martín, Pedro Núñez, Antonio Bandera, and Francisco Sandoval. "Curvature-based environment description for robot navigation using laser range sensors." *Sensors*, **9**(8):5894–5918, 2009.

[Wei] E.W. Weisstein. "Line Graph." Website. http://mathworld.wolfram.com/LineGraph.html.

[Wik] "Haversine formula." Website. https://en.wikipedia.org/wiki/Haversine_formula.

[WWC12] Tao Wang, D J Wu, A Coates, and Andrew Y. Ng. "End-to-end text recognition with convolutional neural networks." *21st International Conference on Pattern Recognition,2012 (ICPR2012)*, (Icpr):3304–3308, 2012.

[WYG19] Lihong Weng, Ming Yang, Lindong Guo, Bing Wang, and Chunxiang Wang. "Pole-based real-time localization for autonomous driving in congested urban scenarios." *2018 IEEE International Conference on Real-Time Computing and Robotics, RCAR 2018*, pp. 96–101, 2019.

[YCQ18] Canbo Ye, Guang Chen, Sanqing Qu, Qianyi Yang, Kai Chen, Jiatong Du, and Ruien Hu. "Self-Localization of Parking Robots Using Square-Like Landmarks." 2018.