ABSTRACT

SURFACE CLASSIFICATION AND KNOWLEDGE TRANSFER ACCESSIBLE ROUTING SYSTEM FOR WHEELSHARE

by Valeria Igorevna Mokrenko

Current accessible navigation systems provide specialized instructions for users with varying mobility conditions. Data gathered from smartphone sensors or crowd-sourcing can pinpoint the location of barriers, obstacles and crosswalks that may cause a safety hazard for individuals using wheelchairs. However, the lack of available, current data from GIS and GPS maps limits the understanding of specific features in the environment. Also, there is no "one-size-fits-all" definition for accessibility as different features in the environment can affect navigation differently depending on the user's ability and preference. We develop WheelShare, the first machine learning enabled, accessible routing system that classifies surfaces from vibration data contributed from users. We achieve up to 94.46% accuracy for surface classification using manual wheelchair vibration sensor data. We also applied knowledge transfer to artificial, neural network models by optimizing a subset of layers and training using power wheelchair data. Our experiments show that training using minimal power wheelchair data can transfer surface classification knowledge with 90.02% accuracy without resampling methods and has statistically significant improvements over training using minimal power wheelchair data. We argue that the lack of completeness in crowd-sourced surface features can be addressed by the inclusion of machine learning techniques in accessible navigation systems.

SURFACE CLASSIFICATION AND KNOWLEDGE TRANSFER ACCESSIBLE ROUTING SYSTEM FOR WHEELSHARE

A Thesis

Submitted to the

Faculty of Miami University

in partial fulfillment of

the requirements for the degree of

Master of Science

by

Valeria Igorevna Mokrenko

Miami University

Oxford, Ohio

2020

Advisor: Vaskar Raychoudhury

Reader: Md Osman Gani

Reader: Karen Davis

©2020 Valeria Igorevna Mokrenko

This Thesis titled

SURFACE CLASSIFICATION AND KNOWLEDGE TRANSFER ACCESSIBLE ROUTING SYSTEM FOR WHEELSHARE

by

Valeria Igorevna Mokrenko

has been approved for publication by

The College of Engineering and Computing

and

The Department of Computer Science & Software Engineering

Vaskar Raychoudhury

Md Osman Gani

Karen Davis

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my deep gratitude to Dr. Raychoudhury my research supervisor for his guidance, great engagement and useful critiques of this research work. Also, I would like to send my regards to my committee member and domain expert Dr. Gani for assisting me with the research experiments and his useful suggestions for my algorithms.
Special thanks should be given to the determined, focused and reliable students who I've mentored to assist with the WheelShare application development: Ce Zhang, Zheng Cao, Junwei Li and Mingyuan Zhuang.
I would also like to extend my thanks to the technicians for providing the resources to run computations on the Miami Redhawk cluster.
Finally, I give a big thanks to my parents for their continued support and encouragement.

Chapter 1

Introduction

Travel on urban streets and especially in unfamiliar surroundings can lead to challenges for wheelchairs users because of the dynamic nature and unknown locations of various obstacles and barriers in the built environment. Previous research has concentrated on finding and tagging barriers of accessibility along the route through crowd-sourcing, GIS or GPS modeling and through smartphone sensors. However, the pedestrian adheres to various criteria that vary from individual to individual. For example, a path that can be traveled by an power wheelchair may not be so easily traversed by a manual wheelchair because of several different characteristics which affect the speed and friction of the wheels with the surface. Recent interest has been on how to collect more information to decrease wheelchair-related accidents and reduce challenges in navigation. These include the creation of robotic wheelchairs, which can assist with traction and climbing curbs, as well as "Smart" wheelchairs, which have additional sensors for mapping, localization, and obstacle or cliff detection [8, 9, 10, 11, 12]. Though this data is valuable for the user, it is relatively difficult use in route planning due to the requirement of expensive sensors for the average user. Therefore, our interest in the research is for the use of inexpensive sensors that are already present in the commonly used smartphones for extracting information about the relevant features for the generation of accessible routes that are present in the built environment. Section 1.1 explains the motivation for recommending routes that consider wheeled mobility. Section 1.2 describes the challenges that are important to address in accessible navigation. Section 1.3 explain the contributions that are made in this thesis. Section 1.4 describes how the thesis is organized. Section 1.5 includes publications and submitted works that resulted from this work.

1.1 Motivation

For a wheelchair user, an unfavorable route with many barriers and obstacles can quickly become too overwhelming and lead to the abandonment of travel plans. Also, the navigational and cognitive barriers of identifying a suitable route may be too overwhelming, time consuming or even impossible due to the lack of information about what features are present in the location. For this reason, finding the best route to a destination is not just preferable, but crucial for those who have certain injuries or poor health and need assistance while using wheelchairs in the community. The motivation for this work is based on the idea that people with disabilities require equal access to all of the common points of interests, such as parks, bus stops, sidewalks and public landmarks. The overall need for this project is based on the fact that people with disabilities need fundamental and equal access to all public areas, such as, public buildings, parks, roads and sidewalks and even beaches. The American Community Survey (ACS) estimated in 2018 that the overall number of people with an

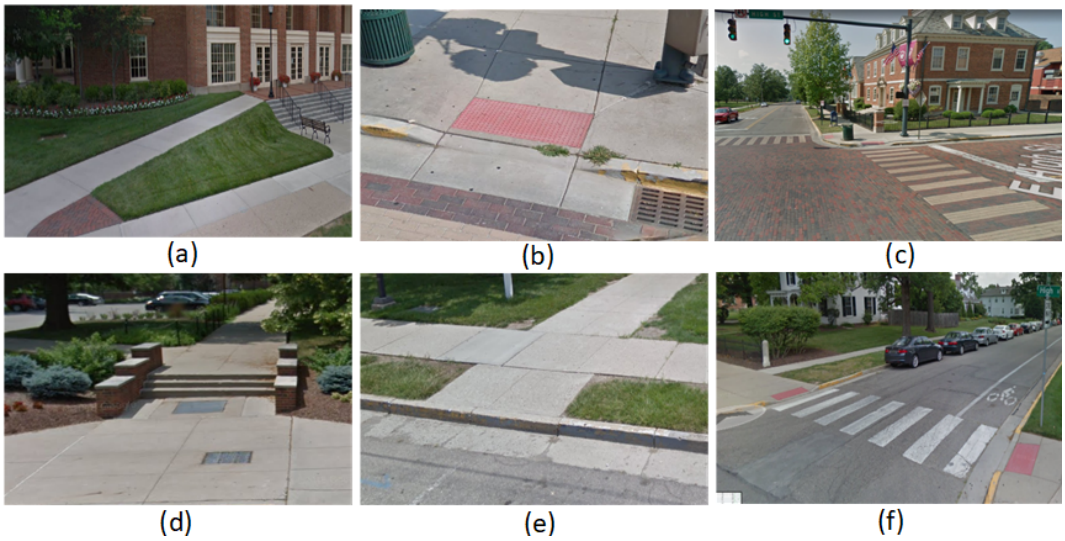

Figure 1.1: (a) Building Ramp vs (d) Stairs, (b) Curb Ramp vs (e) Vertical Drop (c) Crosswalk with Pedestrian Traffic Signal vs (f) Crosswalk without Pedestrian Traffic Signal

ambulatory disability is 6.8% of the total United States population [1]. Also, approximately 21.4% of the older population (age 65+) are estimated to have an ambulatory disability.

Previous research in accessibility navigation aims to provide several routes that mainly consider the user safety. ADA standards have defined the safety criteria that building landmarks, sidewalks and streets need to have for accessibility [13]. The majority of the United States have taken measures to ensure compliance with these standards, such as placing curbs and access ramps, as seen in Figure 1.1.

However, natural weather can degrade a path, causing cracks, flooding, and other unwanted effects. The cracks may get the wheels of a wheelchair stuck or cause difficulty in travel. This also presents other challenges when determining accessible roads [14, 15], such as broken or uneven sidewalks, slippery cobblestone with large gaps, gibbous and uneven brick surface, as well as temporary obstacles that block the path entirely, such as construction areas and fallen trees. Figure 1.2 illustrates the problems with identifying path accessibility from the surface information that we had collected from all over the world.

The presence of stairs, pedestrian crossings with heavy traffic and various weather conditions also may add difficulty in navigation for wheelchair users. In addition, the lack of curbs or a high curb height may cause roads to be completely inaccessible. These various obstacles can prove hazardous for wheelchair users who aren't knowledgeable about the built environment and may not have the advantage of route planning before going to their desired location.

Because information on a path is not always available from government or city records, it falls to each individual's contributed efforts, or crowd-sourcing, to bring updated information for path selection. Crowd-sourcing has been utilized to detect and identify the barriers to accessibility in public areas [16, 17]. In addition, sensor data collected from accelerometers and gyroscopes, [18, 19, 20, 21], as well as GPS and Geographic Information Systems (GIS) has led to increased

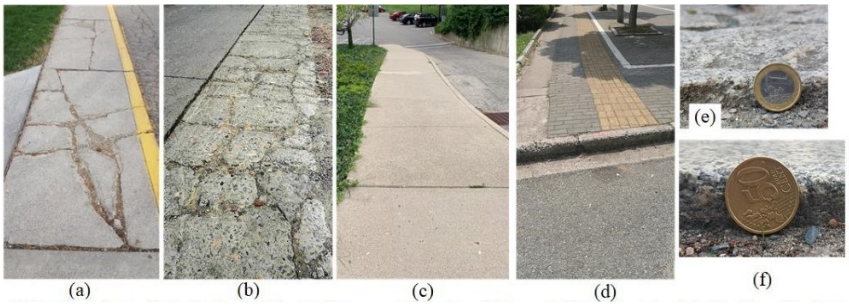

Figure 1.2: Inaccessible Path Features in Various Built Environments - (a) Broken sidewalk in Oxford, OH, (b) Narrow and broken pedestrian road in Yu Yao, Zhejiang, China, (c) High Slope (9-12°) in Oxford, OH, (d) Uneven sidewalk in Dresden, Germany, (e) Cobblestone Squares in Mannheim, Germany, (f) Curb without Access Ramp in Yu Yao, Zhejian, China

interest in the research community on how to understand, aggregate and use this knowledge for creation of accessible routes. The surface type and slope are also agreed to be major features when making a decision on an accessible route since they affects wheeled mobility greatly. The motivation behind the current accessible routing and navigation systems is to enable wheelchair users to arrive to their desired location safely.

1.2 Challenges

Accessible routing and navigation is crucial to solving the problems that arise when mobility is inhibited. In current accessible routing recommendation systems, there is a lack of completeness in systems that use crowdsourced data, since these systems heavily rely on accuracy and trust from other users for understand the built environment. Accessible routing algorithms would give inefficient, not valid, and possibly unsafe route recommendations if the knowledge about the environment is not up-to-date or falsely provided. Therefore, collecting accessible and inaccessible path features is necessary avoid selecting ill-suited routes for the user.

The first challenge is to address the lack of a large-scale repository for route generation and to determine how to collect suitable data for accessible feature detection. Large amounts of information about various surface and slope data are not available for the environments that are considered in this work.

The second challenge is that although proposed routing systems consider specific user needs such as age, weight and physical fitness, there is not enough research on the specific nature of accessible surfaces and slopes in a systematic manner. One current research [22] also uses sensor data to predict these features. However, they do not present how this sensor data collected from multiple individuals can be used to understand the accessibility of an urban feature.

The third challenge is that often times accessible routing and navigation systems are limited to a small, local environment for their knowledge base, but many similar surfaces are present in multiple cities. Therefore, the widespread use of an application is limited by its use in only a handful of cities. The system must be trained using several machine learning algorithms in order to not only

recognize accessible urban features, but also learn the mobility context. Then, the system may apply this learned knowledge to many different environments with similar characteristics.
Our fourth and final challenge is that machine learning algorithms are trained on specific knowledge from one domain. Given the large variety of built environments, including types of wheelchairs, differences in user abilities and endurance levels, and weather conditions, it is almost impossible to test every combination of these parameters using individual experiments or to train machine learning classifiers separately every time. There are no existing methods which aim to transfer knowledge of accessibility. Therefore, the difficult is how to use data collected from one type of sensor-augmented wheelchair for a particular environment in order to predict the accessibility of unknown paths while using another type of wheelchair.

1.3 Contributions

The overall objective of this thesis is to provide a solution to the challenges that we found in accessible navigation and routing systems. propose the Wheelshare system to recommend accessible routes and use machine learning to classify surfaces types based on the vibration pattern gathered from experimentation, and explore methods of knowledge transfer using a model trained on data acquired from a different device. The accumulation of this work can elevate the knowledge for how machines recognize a route for accessibility and provide assistance for navigation. Several research contributions are derived from the main objective:

1. We propose WheelShare, the first machine learning based accessible routing and navigation system, which recommends routes for wheelchair users based on surface classification. This system includes two big contributions to the research:

   (a) Accessible routes are identified based on surface types and slope identified from vibration patterns.

   (b) Data aggregated from many individual experiments is contributed to a large, open source data bank that is shared with the research community.

2. Evaluate the performance of various machine learning algorithms on their ability to classify surface types from the vibration pattern generated by manual wheelchair movement through different surfaces found in the built environment.

3. Apply knowledge transfer from a model trained from sensor data from the manual wheelchair domain to a model trained on power wheelchair data.

These three contributions are necessary to fill the current research gap in accessible navigation. The system also uses a new routing algorithm to solve the challenges of accessibility route recommendation and includes a scoring method based on accessibility of surface types in the environment. The majority of the data used to train the models and initialize our system is collected from experiments contributed by 35 participants using a manual and power wheelchair. We are able to achieve up to 94.46% accuracy in surface classification with sensor data from manual

wheelchair experiments. Using knowledge transfer to artificial, neural networks, we also show that up to 90.02% accuracy can be achieved which has statically significant improvements as opposed to training using only minimal power wheelchair data.

1.4 Organization

This thesis is organized into the following sections:

- **Chapter 2 (Background and Related Works):** This chapter gives a basic introduction to machine learning, description of recommendation systems and approaches to route calculation. The related works includes a summary of existing navigation systems that consider mobility concerns and includes accessibility parameters that are being considered for route generation. This section also includes a description of the usage of different assistive technologies by current users.
- **Chapter 3 (System Framework):** This chapter discusses the system framework for an accessible routing application.
- **Chapter 4 (Core Machine Learning Algorithms):** This chapter provides an explanation and justification of the machine learning algorithms that are used in surface classification.
- **Chapter 5 (Implementation and Results):** This chapter discusses the experimental design, including the implicit data collection of surfaces for both manual and power wheelchair devices. In addition, the results include performance metrics from training each machine learning model on manual wheelchair surface data. This includes creation of deep learning networks, which can be used to predict sequential classification problems. Results from the application of transfer knowledge to a model using sensor data acquired from power wheelchair devices is also included in this chapter.
- **Chapter 6 (Accessible Route Generation):**, This chapter explains the procedure for assigning scores to particular surfaces using vibration data detected from sensors and the accessible route algorithms that are used to recommend routes.
- **Chapter 7 (Conclusion and Future Works):** This chapter finalizes the thesis includes a discussion of future research focus areas, such as improvements to the system framework and addressing limitations of this work.

1.5 Publications from this Thesis

C1 Md Osman Gani, V. Raychoudhury, J. Edinger, V. Mokrenko, Z. Cao and C. Zhang, "Smart Surface Classification for Accessible Routing through Built Environment - A Crowd-sourced Approach", Accepted in 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys 2019). (Core Rank A).

C2 V. Mokrenko, C. Zhang, V. Raychoudhury, J. Edinger, R. Smith and M. Gani, "A Transfer Learning Approach to Surface Detection for Accessible Routing for Wheelchair Users", Submitted for review in the 18th ACM Conference on Embedded Networked Sensor Systems (Sensys), Yokohama, Japan, Nov 16-19, 2020. (Core Rank A).

J1 Md Osman Gani, V. Raychoudhury, J. Edinger, V. Mokrenko, J. Edinger and R. Smith, "Accessible Routing for Wheelchair users through Surface-induced Vibration Analysis", Submitted for review in the ACM Transactions on Accessible Computing (TACCESS). (Core Rank A).

Chapter 2

Background & Related Work

The design of the roads and sidewalks today pose limitations for people using assistive devices. Much attention in the research on assistive technologies is on how to ensure the concept of universal design, which is "the concept of designing all products and the built environment to be aesthetic and usable to the greatest extent possible by everyone, regardless of their age, ability, or status in life," as coined by Ronald Mace. This was later pioneered to be the concept of free accessibility for the disabled community in Designing for the Disabled (1963) by Goldsmith [23], who is notably known for the idea of a ramped sidewalk curb that is a customary feature in the built environment. The Department of Justice provides the Standards for Accessible Design which detail exactly the permitted surfaces for the construction and alteration of different types of surfaces and structures, such as those near ramps, pools, and stairways, [24]. These standards sets a minimum requirement for walking surfaces and on other steeper sloping components such as ramps, and curb ramps that may exist. Because of the natural degradation of roads due to natural and man-made constructions, it is often common to find paths that do not comply with the regulations. The difficulty for finding a route that is accessible is increased by the position of crosswalks, cracks and unexpected obstructions. The additional problem comes with the lack of data that is complete and valid about the road conditions, which affects the accuracy of real-time navigation systems that assist wheelchair users

The following background describes the origin and use of artificial intelligence in recommendation systems. We also describe several types of algorithms in machine learning and algorithms for route calculation. The related work section describes some enabling work that provides a baseline for our accessible routing and navigation system, WheelShare. This section also describes previous and current work for navigational systems and how they generate accessible routes. We also discuss the current trends in accessibility devices in United States as well as the differences between a manual and a powered wheelchair. Finally, we provide a summary for all of the systems and explain how WheelShare differs.

2.1 Machine Learning Algorithms and Applications

This section explains several methods for machine learning (ML) and how ML algorithms are used to train systems. It also describes several algorithms that are used in general recommendation systems as well as the types of algorithms that are generally used in calculating routes.

2.1.1 Learning Styles in Machine Learning

The term "deep learning" does not only refer to learning in the neuroscientific understanding, but more general to learning that consists of multiple levels. Deep learning algorithms aim to mimic learning in AI in the way that we currently understand how biological learning occurs for humans. Deep learning is a part of machine learning that aims to create computer systems that solve a problem that requires intelligence. This section provides a short summary for four learning methods: supervised, unsupervised, semi-supervised learning and transfer learning.

Supervised Learning

Supervised learning algorithms simply associate a new input with output according to a training set of inputs and outputs. The format of the output is predictable because it is provided to the algorithm. The goal is to find the probability distribution of $P(y|x)$ or probability of output y given input x. The best example are support vector machines (SVM) that operate on a linear function $w^T x + b$. The purpose of SVM is to predict when a class exists that is positive or negative. SVMs can be improved using kernel machines, which show that the algorithm can be written in terms of a dot product. However, the cost of evaluation is linear with the number of training examples. K-nearest is another supervised algorithm which breaks the input space by assessing the k-nearest neighbors to the input set in the training data to produce the average of the output values. Decision tree also breaks the input by creating each node as one region of the input space and the children of the node are each a sub region. When prior knowledge exists for what the output values should look like, then this environment works best using supervised algorithms. Ultimately, the output data is labeled by a function that is learned through classification.

Unsupervised Learning

The goal of unsupervised learning is predict the structure of data given the lack of labeled data [25]. The "best" representation of data means that all the information about the input must be saved and the output representation should adhere to some conditions that make it at least as accessible as the input. The various examples that fall into unsupervised learning are density estimation, denoising data and drawing distribution samples. Representational learning algorithms may consist of representing data through lower-dimensional representations, sparse representations and independent representations. Low-dimensional representations seek to represent the input narrowly, often preserving the majority of the information. Sparse representations put the data set into a representation with entries mostly zero while at the same time increasing the dimensions of the representation. Independent representations represent the data distribution in a way that the dimensions are statistically independent.

Semi-Supervised Learning

Semi-supervised learning combines supervised and unsupervised learning features. Unlabeled input data as well as labeled training data is required to understand $P(y|x)$. The goal of semi-supervised learning is similar to that of unsupervised in the sense that the system must learn the

representation $h = f(x)$ and produce output that has similar representations through classification. Typically there would be a supervised learning step that preprocesses the data and and a classifier that uses unsupervised learning to make a general understand of what data should be classified together.

Rather than treating unsupervised and supervised as separate components in semi-supervised learning, it is possible to create a generative model of either learning method that consists of similar parameters as a discriminative model $P(y|x)$ [25]. Throughout the learning, there is some control over how much the generative criterion is considered, that way the criterion may be more or less supervised or unsupervised. Most real world problems adhere to semi-supervised learning because of the lack of labeled input data.

Transfer Learning

Transfer learning is the improved rate of learning a new task through existing knowledge from a related, already learned task. The application of transfer learning is popular in tasks such as computer vision and image processing. Transfer learning utilizes knowledge such as features and weights from a previously trained model. This technique is useful when there exists a task related to an existing model that has been trained with a lot of data for testing and validating and the model features are general to be suitable for transfer. When there is not much data, transfer learning still assists with creating models for tasks in different feature spaces. If labeled data exists for both the target and source domain, the model can solve the source task and be used in a new problem in the target domain called multi-task learning. There are essentially three types of transfer learning methods: inductive, transductive and unsupervised Learning. While inductive learning requires for there to be at least some labeled data available in the target domain, transductive requires only labeled data from the existing task. For inductive learning, the existing and the target task can be different from each other and therefore decisions are made on the inductive biases from the existing source data. Unsupervised learning assumes different tasks but both domains are similar and thus can use clustering methods to transfer learning. One form of transfer learning, referred to as domain adaptation, assumes that the new task is the same as the old task, but the input distribution is different [26]. In this case, the data from the first distribution or domain is extracted and used to make predictions for a different domain. This approach is useful when there are features used in different domains that correspond to factors that are present in multiple domains.

Some of the approaches to transfer learning include:

- **Feature-representation Transfer**: Since domains tend to be different between tasks, transfer requires identifying what specific features or "denominators" from learning tasks and how would their meaning assist in understanding the relation between the source and target features. Its applications include face recognition applications with underrepresented data [27].

- **Instance Transfer**: Important instances from source data are identified through boost instance weighing [28] or elimination [29] and used for transfer.

- **Model Transfer**: Given that a highly accurate predictive model from the source domain has been learned, the model can be adapted to solve the target task given data from the target domain. This approach assumes similar inductive bias for both the source and target tasks. This approach is useful when considering storage capacity or applications when the original data is no longer needed [30].

2.1.2 Algorithms Pertaining to Recommender Systems

Recommendation Systems (RS) do not apply learning through reasoning but rather can classify the information based on Machine Learning (ML) algorithms. Although RSs are just one application of Artificial Intelligence, the interest to improve recommendations has led to the development of a large number of Machine Learning strategies. A systematic review was made to determine what ML algorithms were used in RS, domain and implementation of the study and possible problems or questions with the work [31]. The 26 research papers found to use ML were ranked lowest to highest in the implementation method for their RS.

Category	**Total Papers**
Bayesian	7
Decision Tree	5
Matrix factorization-based	4
Neighbor-based	4
Neural Network	4
Rule Learning	4
Ensemble	3
Gradient descent-based	3
Kernel methods	3
Clustering	2
Associative classification	1
Bandit	1
Lazy Learning	1
Regularization methods	1
Topic Independent Sorting Algorithm	1

Table 2.1: Popular ML Algorithms in RS Research

It is important to note that the research papers had proposed one or more ML algorithms and that each algorithm is classified as supervised, unsupervised, semi-supervised or reinforced learning. Therefore, based on Table 2.1, the types of ML algorithms that are most commonly used to train classifiers in RS are Bayesian Networks and Decision Tree. The reason for this is presumed that both algorithms of have similar calculations of low complexity and difficulty in implementation [31]. However, decision trees are used in supervised learning and therefore have limited applications in decision making where the function may change in a small region.

This is because decision trees break the input into smaller regions and depending on how many leaves were in the tree, then that many training examples need to be produced [25]. Bayesian networks can provide learning between nodes where a relationship can clearly be defined, however much more data and processing is required to determine the relationship of high dimensional data. Furthermore, supervised learning had 156 ML algorithms, whereas unsupervised algorithms had 46 ML algorithms used in RS [31]. Also, there had only been one semi-supervised learning algorithm and 2 reinforcement learning algorithms used in RS. Also, 22 ML algorithms were proposed for Ensemble, 22 for K Means, 20 for Support Vector Machines, and 14 for both Bayesian and Decision Tree. However, this encompasses the wide array of RS systems and does not only the RS systems for route planning, but also for many other applications.

2.1.3 Approaches to Route Calculation

Most self-adaptive route navigation systems use a graph-based method for calculating the optimal route between a current and a destination node. Graph-based algorithms represent a graph G with n nodes that have labeled and unlabeled data and the edge between them is a measure of how closely the two nodes are related. Typically, there is an initial computation to obtain edge weights that do not change. More complex algorithms use an optimization factor to look across many other types of criteria in order to formulate a path. This section describes current types of algorithms used in route calculation.

One of the most well-known algorithms for route planning is Dijkstra's shortest path. The algorithm works as such: first the node is chosen based on how short their distances are from the source. Once a node is visited, the edges that come from this node and the other nodes are relaxed. The size of the search space is O(n) and n/2 nodes are visited on average [32]. Other algorithms were made to speedup the route calculation process. Priority queues were used which improved the running time from O(n^2) to O($nlogn$). Bidirectional search is used in most advanced algorithms because it can compute the shortest path from the source and simultaneously from the target node. A* search algorithm took the idea that the next node should lead closer to the target node and so route planning between a node and the target destination is mapped based on the Euclidean distance and the average speed of the fastest road in the network [32]. The first approach for a faster routing algorithm represented a graph in multiple levels [33]. An overlay graph G_1 that consist of vertices V_1 and edges E_1 and E_1 represents the shortest paths in G that does not consist of V_1. However, if this algorithm produces a large number of graph levels, the higher levels would consist of more vertices. This would occur in large networks due to the large amounts of possible paths that a user may take.

Though these algorithms provide a base understanding of routing calculation, current navigational systems use more complex algorithms for route planning. The most common approach is to not consider the shortcuts from side streets unless the target node is nearby [34]. The concept is similar to that of Highway Hierarchies (HH), in which after a certain amount of nodes were traveled to, only the main highway is considered based on a heuristic. Highway Hierarchies follow the bidirectional Dijkstra approach except that the edges that stray from the main highway network are not considered some distance from the source or destination node [35]. This approach can handle the largest road networks with ease because preprocessing occurs for a limited number of

local edges around the starting node. Current routing techniques use an improvement to highway hierarchical method called Contraction Hierarchies (CH) [36], which simplifies the approach and improves querying. This algorithm orders the importance of nodes in a priority queue, which is based on the edge difference (i.e. how many edges are produced if a node is contracted). Once a node is contracted, the shortest path is still kept among the rest of the nodes. Therefore, shortcuts are introduced where appropriate, which is between nodes of a higher order. On mobile route planning, CH are preferred because of the complex amount of parameters that cannot all be considered before navigation and the important nodes do not need to be preprocessed again in the middle of the trip unless the user takes a turn that goes outside of the suggested route. CHs are useful in accessible routing applications since they can lead to a quick calculation of multiple short paths for comparison and allow for quick re-routing if the user desires an alternate path from their current location.

Currently, some algorithms attempt to find optimal routes from the source to the destination based on optimization factors. Typically, the route would have some cost awareness, such as whether to prioritize speed, safety of travel or shortest distance and compute the routes based on the current data received from a query. It becomes necessary to gain data from users, geographical maps, and other information such as current tolls and closed construction roads in order to retrieve the information necessary to compute the optimized route for a constantly evolving environment. One algorithm that is known for also working in a dynamic setting is the ALT algorithm and has been mentioned in standard route planning algorithms sources, such as Delling et al [37]. This algorithm is a combination of a bidirectional A* search, landmarks and triangular inequality, and was first proposed by Goldberg et al [38]. It takes into consideration the landmark bounds and, unlike static search, can perform accurately without preprocessing data as long as the edge weight stays above the initial value. The ALT algorithm consists of nodes called landmarks and uses triangle inequality to make potential edge weights. The down side to ALT is that that potential for the forward search as well as the backward search are consistent. Also the lower bounds depend on the quality of the landmarks and thus a heuristic must be used for choosing alternative landmarks. It is also not efficient, unless landmark data can be stored in a way that makes computing the potential faster and retrieving the landmark faster. Preprocessing of data can also cost performance because it consists of two steps: landmark selection and calculation of the distance labels. The eager dynamic ALT requires less overhead and is more memory efficient because successors of the parent node is found by checking if the source nodes from edges e of n holds that $d(s)+w(e) = d(n)$. If it does then s source node is a parent of the node n. The other variant is a lazy dynamic ALT that increases search space because preprocessing doesn't occur unless the cost of an edge becomes lower than the initial cost. Since an edge e can increase in cost, the weight of this edge increases in the graph of reduced costs thereby increasing the length of the shortest path [37]. This can occur when obstacles or real-time events occur that affect the accessibility of an edge, thereby increasing the edge weight costs. The scenario of an edge weight becoming less than the initial cost is unlikely, given the initial cost representing an empty road. However, because increasing information can affect the cost during preprocessing, there is a chance this value becomes lower than the initial cost. This is because new roads that are discovered can become new nodes along the shortest path with unpredictable values.

Another version of ALT is the time-dependent algorithm, in which bidirectional search isn't allowed. The unidirectional ALT can only perform a forward search and the behavior of the algorithm follows that of Dijkstra's algorithm. The processing is stopped once the target node is reached. During preprocessing, the minimum weight of each edge is used to create the distance labels, i.e. the potentials. This is added to the edge weight of each node in order to determine the priority of the nodes. The findings reported by Delling et al show that time-dependent model performed better than the time-independent in dynamic scenarios [37].

ALT algorithm and its variants are powerful for computing route calculation in a dynamic scenario. Traditional routing algorithms are currently used in route calculation and involve pre-processing in order to gain information about existing paths and locations of buildings. Pedestrian navigation occurs on a dynamic setting, therefore in order to make personalized routing decisions, edge weights should be routinely updated based on the nodes acquired in real-time.

2.2 Related Work

Current research into finding accessible routes for users requires familiarization with the various features of the path. In one study conducted by Meyers et al. [39], 28 adult wheelchair users identified the absence of ramps and sidewalk curb cuts, uneven sidewalk surfaces and temporary obstructions to limit accessibility, while locations with elevators and crosswalks tend to be more preferred. This section discusses existing accessibility systems that utilize algorithms and methods for data collection to recommend dynamic routes for users. Also, this section discusses current tendencies in assistive device usage in the United States for various age groups.

2.2.1 Enabling Work

The creation of an application for an accessible route for wheelchair users has been proposed through WheelShare [4]. This application would use crowd-sensing data about the surface vibrations and aims to accomplish an application that is scalable, objective and dynamic. Specialized routes for people using wheelchair devices need to consider knowledge about sidewalk surfaces, slopes, pedestrian crossings and curbs. The intended goal is to design the algorithm which would allow for determining which routes are accessible for the user.

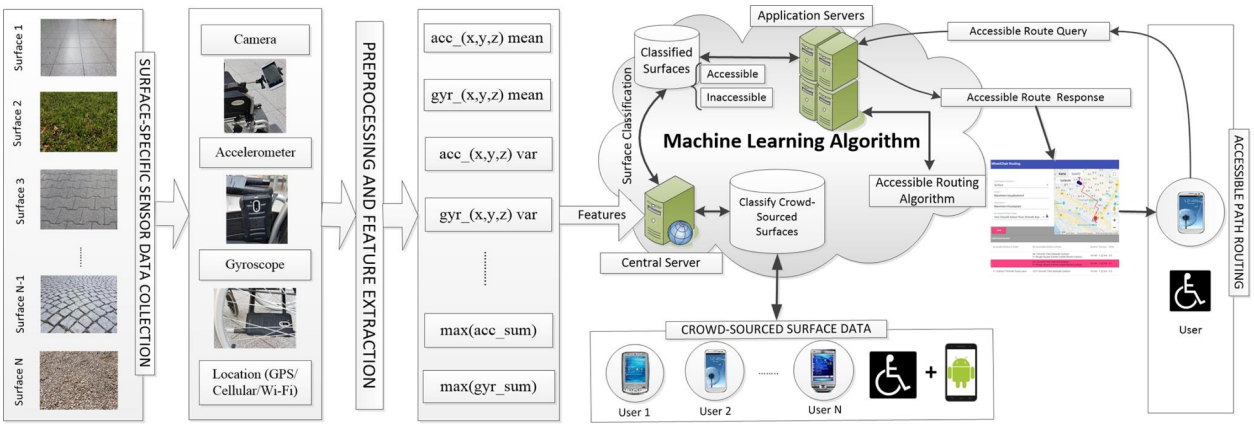

Figure 2.1: WheelShare routing system [4], published with permission.

The WheelShare system uses machine learning algorithms to classify the surfaces as accessible or inaccessible. The vibration data for each surface is collected from the accelerometer and gyroscope, and then it is geo-tagged before placing it to the classifier. The algorithm would find the best route through the paths and sidewalks which are accessible. The accessible paths are added on top of a map by making Rest API calls to Google Maps which allows for live traffic updates. Each route has a utility value based on whether it is accessible, what waypoints on that route were accessible and inaccessible, and also the total travel time and distance.

The application enables crowdsourcing, so that information from users on surface can be collected and then classified on vibration. The higher spike would indicate a rougher, less accessible terrain. The utility function is applied on each route in order to determine the score for the accessible waypoints in that route and the penalty score for the waypoints that are inaccessible.

2.2.2 The Contributing Navigation Systems

Personalized navigational systems are context-based routing planning systems, which create routes depending on the desires and needs of the user. The dependence of user preferences can be used to create optimal routes by considering the path as context dependent over the space of the whole route [40]. However, this routing approach does not allow for pedestrians to navigate spaces where the paths may be less clear, such as a forest or a park. Other approaches to solving this problem has been to consider the route as a series of waypoints where moving from one point to the next is determined by a stage of closeness between the user and the destination waypoint [41]. Much headway has been made to personalize routes, so that pedestrians can choose which path to travel with little to no interruption of the decision process during navigation. Some systems allow users to enter an efficiency value for their route, so that the system decides whether to continue using this path or to create a new one from the current map [42]. However, due to the lack of knowledge on the types of surfaces, slopes and obstacles of a route, this navigational system did not account for real-time adaptability if information was inaccurate or not up to date with the map. This led to research on how to determine the obstacles in the path and gain knowledge of preferred routes through systems such as crowd-sourced accessible route recommenders and mobility assistants.

Navigation Through Third-Party Software

Current dynamic routing systems such as Google Maps use context awareness to recognize the current, nearby nodes from the source to the destination. However, routes are not considered for accessibility and are determined based on getting to the destination in the shortest time. Though Google does not readily provide their dynamic route algorithm to the public, they do provide ways to interact with the algorithm through API. This is especially useful when creating waypoints while routing, so that the route adapts to the user in real-time. Alexandra Hofmann provides a documentation of service providers who created dynamic routing algorithms and how users may interact with them as well as which ones include the creation of waypoints [43]. Dennis Wagner's thesis [44] also explains the usefulness of gyroscope and accelerometer sensors to detect curbs and surfaces and considers his application useful for accessibility system.

The several dynamic algorithms that exist for route calculation utilize the API for service providers because they can provide more accurate, real-time information about accessible roads and construction information. The dynamic algorithm proposed by Hofmann calculates the routes by the controller calling a method in the RouteCalculationManager, which makes an API call to Google Maps [43]. The routes are sorted by a utility value, which consider the duration, distance and costs of the route. However the intention is to create a route with cheaper fuel prices, and therefore the limitation is that multi-dynamic parameters such as traffic flow and speed limits are not considered.

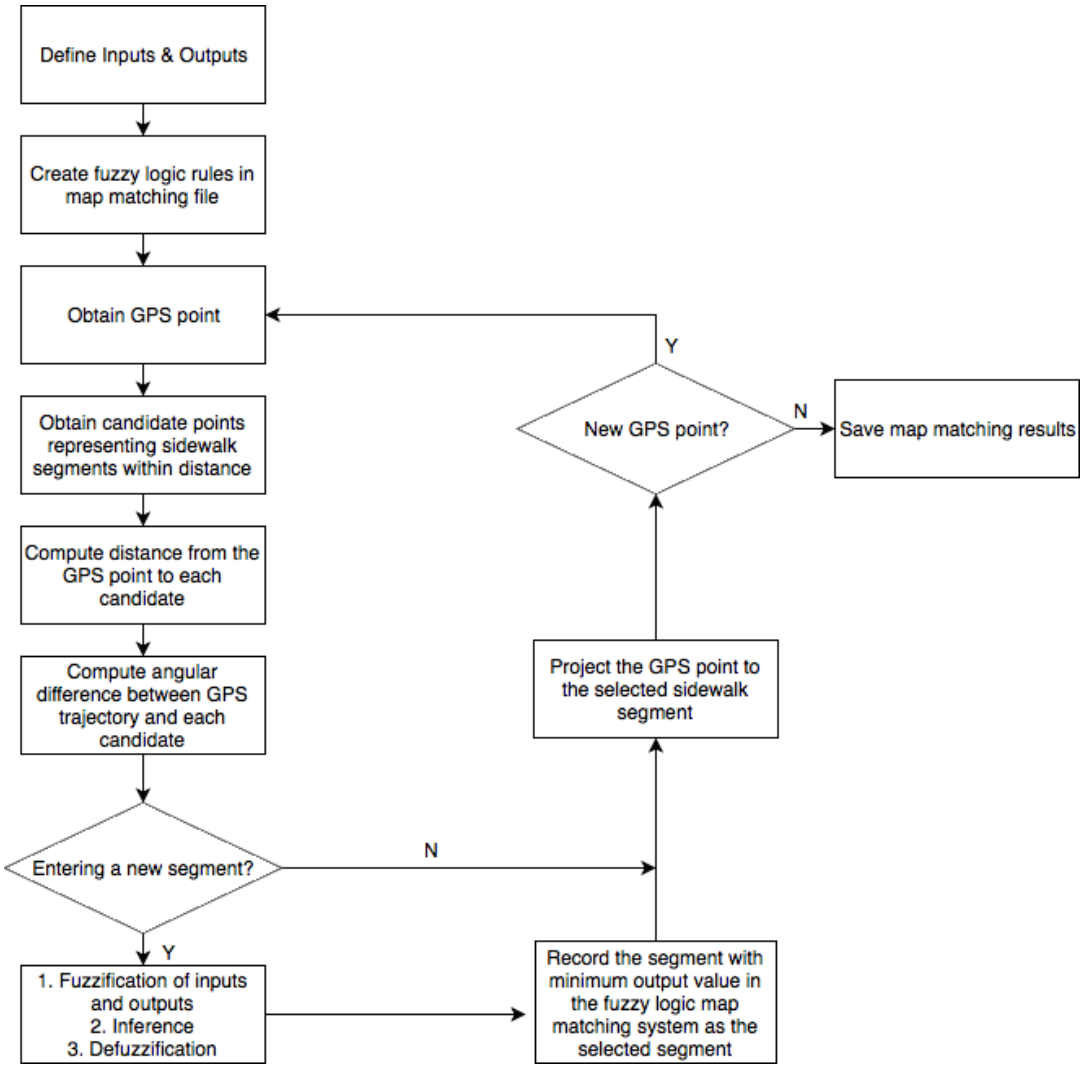

Figure 2.2: Fuzzy Logic Map Matching for Pedestrian Map Matching using GPS tracking. Adapted from [5].

Furthermore, systems based off this approach like Path 2.0 [45] already exist and leverage the useful information from smartphone sensors to obtain information on road accessibility and can use the information from Google Maps API. An alternative approach was to use fuzzy logic to predict wheelchair movements in the case that satellite data did not exist [5] (see Figure 2.2). Both of these systems are GPS based wheelchair systems and require some information about the wheelchair itself in order to make accurate predictions with the third party software.

Routing for User Preference

Several tools exist which collect and model user collected data about point of interest (POI) features or places where the trip begins or ends. For example, applications that aim to predict the safety of a city use user provided data through crowdsourcing and machine learning algorithms to create

a map that informs users of potential safety hazards. Several of these tools include SketchFactor [46] and Place Pulse [47]. There are also many systems that focus on creating a personalized route that fits one or several categories. One approach focuses on creating personalized routes that appeals to human preferences of quietness, beauty and happiness [48]. This research uses data from crowd-sourcing on two streets in London where locations that were voted on to be the most pleasant are indicated on a graph. Locations are scored based on the likelihood that it will be visited given of how pleasant it is based on one of the human preferences. For example, the probability $p(happiness|go)$ equals $k * h_i^3$, where h_i is the crowdsourced happiness score for cell i and $k = 1/(max\{h_i^3\}\forall_i)$. The best route is selected by first choosing the all shortest paths up to 10^6 paths using Eppstein's algorithm. Then, the average rank for all the locations are found for some amount of paths less than or equal to 10^6 paths. The path with the best rank is then selected. The same is done for the quiet, beauty and happiness dictated graphs in order to get the best path for that preference. What is also interesting is that the recommended paths are 12% longer than shortest paths and the cost of travel time decreases exponentially the longer the path is. The authors believe this is because users may choose several different paths if the path is short, but a longer distance route has fewer deviated routes from the source and destination. However, this perception of happy, quiet and beauty of paths varies from individual to individual. Similarly another work [49] analyzes crowdsourced data with the regions that consist of objects such as barriers and landmarks and creates a recommended route that prioritizes safety. This is accomplished through a Vector-based Diffusion and Interpolation Matrix (VDIM), which is an algorithm that creates a rating matrix of features in a spatial region and then is used to create a route according to which features correspond with safety.

2.2.3 Navigation Systems for Accessibility

Collaborative and hybrid filtering RSs are a type of personalized navigational system that allow for crowdsourcing. The purpose of crowdsourcing is to improve the quality of life for users because it allows for personalized routing. For example, users can attach pictures and comments of the route and possible obstacles that others should be aware of, so that accessible locations can be determined. There are many different kinds of crowdsourced systems that collect user data [50, 19]. One such system can use a smartphone application for acquiring photos and other information from the user, such as a barrier on a sidewalk [51]. The information obtained is then geo-tagged so that others can create their own routes for accessibility through collaboration with the obtained knowledge.

Routing based on User Mobility

Several systems have also employed the Level-of-Service (LOS) model, which refers to assessing the quality of traffic and transportation with the goal to increase efficiency by examining parameters such as vehicle speed, traffic, and density. This model has also been applied to accessibility routing systems, with MAGUS being the first navigational system that set a standard for the application of a comprehensive LOS model for wheelchair users [52]. MAGUS tested route accessibility in Northampton area in the UK and required users to enter their age and weight as well as any feedback that they had on the sidewalk parameters. These parameters include slope, surface type, and curb

cuts, and then the impedance score for each segment was calculated using mathematical models. MAGUS made route recommendation based on minimum barriers, shortest distance, fewest slopes and tough surface terrain as well as limiting the number of road crossings. Other techniques were used in Karimanzira et al [53] in Georgenthal, Germany that used ML algorithms for creating routes for visual/limb/hearing impaired by the use of a fuzzy decision system. The intention of the system was to increase navigation for users and for planning trips. However, Sobek and Miller explain the LOS model is costly to implement and users would have to spend much of their time inputting data making MAGUS unfeasible as a real-time application [54]. Instead, they propose U-Access, a web based system that provides navigation for users that are either peripatetic (do not require an aid), aided mobility, and wheelchair users. Because it is an open system that uses the World Wide Web for reporting results, it functions as a mobility assistant. It creates efficient route recommendation while being less expensive and not requiring users to have specialized software. Another system has simplified the software further by giving users the ability to identify with certain groups that share preferences for what type of mobility is required [55]. Then, users would generate their own LOS model that would consider user annotations from that group about specific route patterns. Other data that is geo-tagged is also considered based on public knowledge. This allows for customized routes for the mobility impaired, increasing availability and accuracy of route planning. Not only does this system leverage that pedestrians do not follow strict routing patterns and can take shortcuts or go around obstacles, but also that users have different preferences. Therefore a route may be different for users that have different levels of physical abilities. Based on this conclusion, U-Access approach to accessible navigation is a feasible approach to route navigation.

Multicriteria Routes

Several other systems were created that used georeferenced data to indicate barriers and cross roads on the map. In 2008, RouteCheckr, a client/server system that allows users to collaborate data annotation and personalized routing decisions, was created for the intention to improve accessibility [56]. Wheelchair users would annotate information including points of interest, specific environment features such as places where the wheelchair can be better oriented or turned around, locations of obstacles or barriers, and be able to rate the safety of a route. RouteCheckr also proposes two algorithms for calculation of the route: one being a static multicriteria routing and the other a personalized multicriteria routing.

The static approach assigns a vector of criteria to each path section and calculates the cost of path using a cost function which is based on weighted addition. Personalized multicriteria routing is based on using equally weighted and normalized criteria values based on user rating on the specific path. The system allows for annotations by those in a specific group and does not allow for convergence of annotations. For example, if there was a lower curb then people from multiple groups would have to annotate this for the system to calculate the accessible route for each group.

U-Access similarly did create personalized routes and offered routes for mobile, aided and wheelchair users [54]. At the time, accessibility routing applications were minimal and often on commercial systems that required expensive and complex software to run on. One example is the ArcView 3.x which required geographic information system (GIS) proprietary software on the client side. U-Access, on the other hand, created a universal application that uses GIS to maintain

the data, transformed it into a SVG file, parsed for three networks that cover the varying physical abilities, then delivered as an HTML file to the World Wide Web. The algorithm that was used for route calculation is Dijkstra's shortest path. U-Access considers many different features of environmental objects that can affect route calculation depending on the physical ability of the user. U-Access is the interface between the urban environment and the users. The user can only belong in one of three categories of physical ability. Other systems [57, 58, 59, 60, 61] use GIS for navigation for a dynamic setting. There can be multiple attributes that affect route calculation in the environment. GIS navigation is still actively used in navigation systems today.

Debate on Validity of Crowd-sourced Information

Other systems propose notifying the user of barriers using the GPS sensors on their phone. A mobile system application for the elderly allows for content submission, in which a user can specify a title, comment, tags and time and take a picture of the barrier or obstacle, which become available for other users that go to that location [62, 63]. This system uses a Bayesian network to choose which content to alert to the user when they walk on a route that has many submissions. For example, a wheelchair pedestrian may need an alert for where the escalator is, but a person who is working out may want to use the stairs instead. In this way, the system considers the purpose for using the routing application very important. Another important contribution to crowd-sourcing is the ability to interact with Google Street View in order to identify bus stops for blind pedestrians. Kotaro Hara et al proposed a tool called Bus Stop CSI that with 82.5% accuracy can predict bus stop landmarks using Google Street View [64]. The research points to the future ability to predict other landmarks along a route. However, they pointed out that Google Maps API does not always offer accurate locations for landmarks which made their 2D map confusing. Therefore, this error may be mitigated using OpenStreetMap, which is a collection of collaborative geo-data [65]. Though OpenStreetMap allows for crowdsourcing of landmarks, the addition of 3D details on objects and surfaces is still being developed. Furthermore, Mobasheri et al mentions that there is a lack of completeness of data about sidewalks in OpenStreetMap which limits the accuracy of routing applications who use the software [66]. Therefore, there is limitations for wheelchair accessibility of streets that consist of ill-suited surface types. This may be propagated through the ability to add images of the street surfaces, though this still heavily relies on trust from other users to provide accurate and updated information. WheelMap is an interface based on OpenStreetMap that allows for crowdsourced information about the accessibility of certain obstacles, landmarks and locations of traffic lights and crosswalks [67]. Though users can contribute the rank of accessibility and describe certain obstacles at a destination point, it does not allow for barriers or labels for obstacles in between the point of origin and the point of destination.

Personalized Paths for Users

Some research has also been done to increase accessibility for wheelchair users indoors as well as outdoors. Mobile Pervasive Accessibility Social Sensing (mPASS), is a system that creates personalized routing paths based on data gathered from crowdsourcing as well as sensory data from geo-referencing [68]. Catia Prandi et al details that the biggest issue with accessibility systems

is the lack of real-time navigation due to there never being enough information about a route that allows for efficient mapping services. Routing algorithms would become ineffective if the data set is not valid because users would be led across unsafe or non-existing paths. They also advocate for complete data so that the user and the algorithm can know as much information as possible to make decisions on where to go. The three modes of gathering data is through a smartphone, including gyroscope, accelerometer and GPS data, through crowdsourcing including textual and multimedia (pictures and videos) of environment objects, and official reviews from organizations and powers of authority. The prototype of the system can perform the following tasks:

- construct a profile for the users
- allows users to insert a report
- sends notifications on barriers and accessible facilities on the path
- allows users to view previous report logs
- display reports localized in Google Maps
- finds the best route

The research following the paper introducing mPASS illustrates a more simplistic figure of the system architecture [6] (see Figure 2.3).

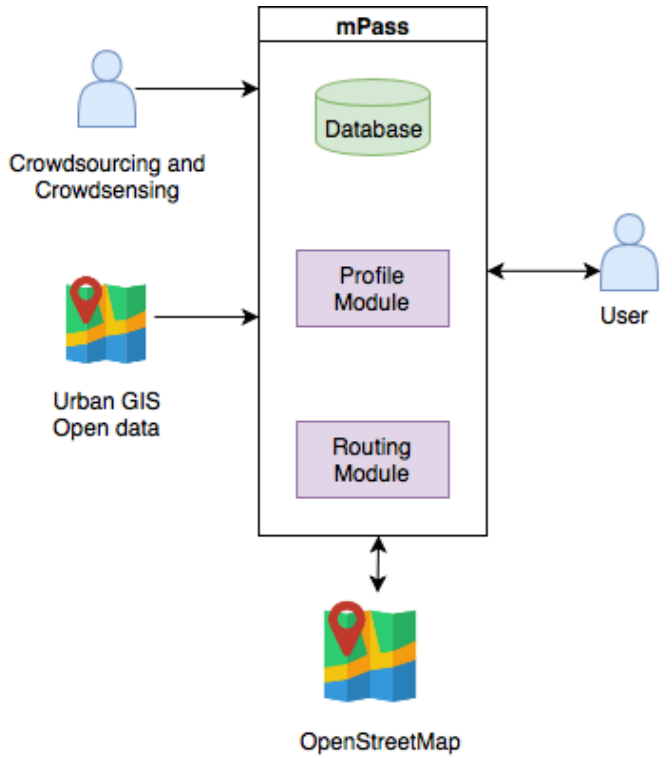

Figure 2.3: mPass system architecture adapted from [6].

Silvia Mirri et al asked 60 European users on what their preferences are on personalized paths. Around 88% has stated that they would be willing to go on a path that is 30% longer in order to avoid a barrier or one that is the best route for their needs [6]. In addition, there were suggestions for mPASS to provide more information about the surfaces and pavements, in particular recording the dimensions and positions of the curvature or uneveness of the road as well as possibility of crossing or avoiding this obstacle. Furthermore, about 73% of the pedestrians had preferences on safe paths that suited their abilities [6]. However, some systems [69, 70] have used cost-aware analysis techniques to evaluate the accessibility of a city. This involves ranking stations in their ability to offer accessible navigation for various modes of transportation and using network analysis to calculate the total travel time of accessible navigation.

eNav [71] is another proposed system that uses WheelMap crowdsourcing information to determine the POI objects, or destination nodes that are accessible for wheelchair users. The paper compares routes that are on the shortest path by using a modified A* algorithm to rank the route that is most energy efficient. The way a route is discovered to be energy efficient depend on the surface friction, the factor of incline for the surface type, the distance traveled, and the consumption of energy obtained from sensors placed on the wheelchair. The system provides REST API calls for clients to request routes and, based on certain parameters as well as crowdsourcing and map sources, provides users with routes.

Similarly, a context-aware trip planning system for increasing mobility in cities has also been created [7] and is shown in Figure 2.4.

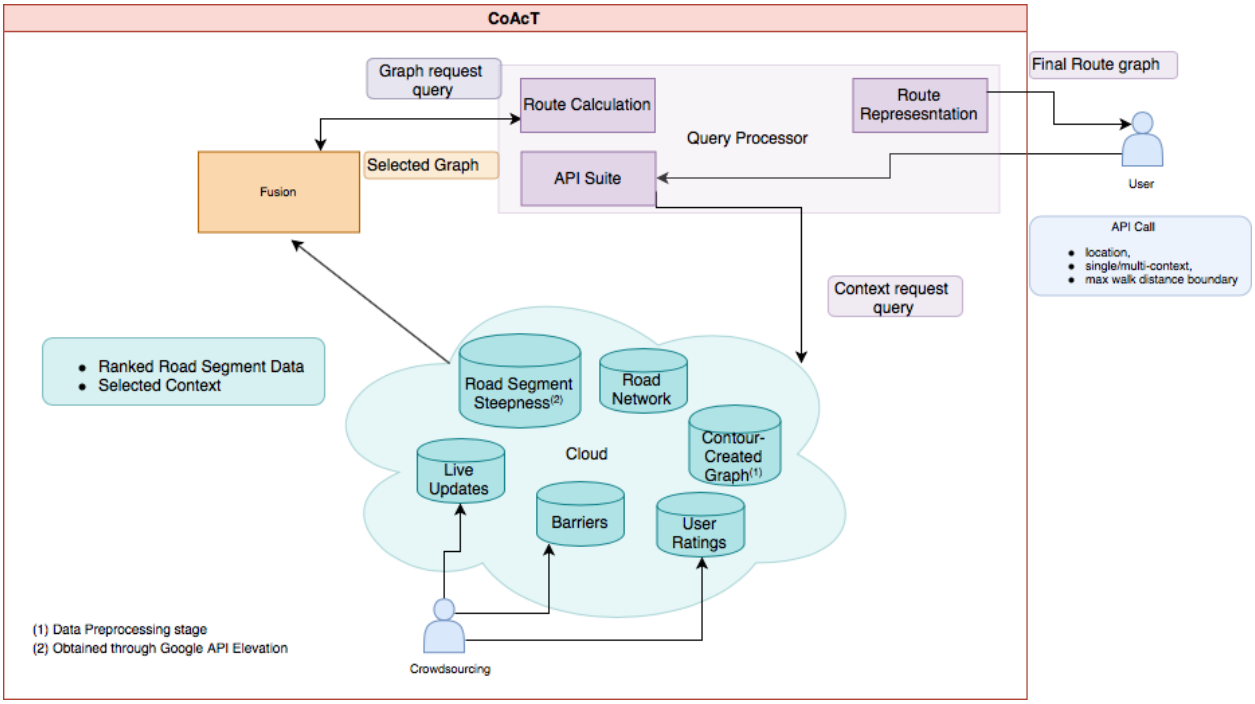

Figure 2.4: Rahaman's system framework adapted from [7].

The framework created by Rahaman has been applied in the context of trip planning for mobile

transportation vehicles, such as taxis, cars and buses, as well as to accessible navigation. Though previous methods have attempted users to define a standard rank for accessibility for a route, obstacles may be accessible depending on the physical ability of the pedestrian. Therefore, no standard rank for accessibility exists because it varies from individual to individual. The thesis introduces the scenario of context-awareness, or factors may limit mobility, and what solutions may be made to increase mobility. The thesis mentions that steepness, presence of stairs, the weather and road crossing are all factors that play a role for a wheelchair user while routing, whereas ramps and lifts play a factor at the end destination. The built framework creates a contextual data collection method that stores real-time updates, physical barriers and ratings from crowd-sourced methods and a fusion and query processor that takes in the coordinates of the source and destination locations and returns a visual map with the trip planned routes. Therefore, it is a hybrid system because it uses adaptable feature extraction to detect slope through Google's API elevation, and includes crowd-sourcing to create personalized paths. Because shortest path algorithms do not always allow for mobility, the A* algorithm is computed three times, once to obtain the shortest path, once to create the steepness road network and once more for road cross sections and road-contour sections to obtain the steepness contour. The contour sections are gathered from merging Open Street Map roads and data obtained from Srtm2Osm using JOSM (a cross-platform OSM editor) that can create elevated contours, so that the steepness of a road may be determined from the latitude and longitude values. The new crossing points found from merging the streets and lines and the edges creates the contour-based graph and then an XML file can be generated. Much of this work focused on three accessibility measures: horizontal distance, vertical distance and maximal slope. Slope is mainly a concern for wheelchair users, however it may be less of a concern to strollers due to the aided assistance of someone pushing the stroller. Therefore, there is still a problem of using CAPRA [72] in a multi-dimensional environment, especially one where the classifier has no knowledge of the mobility context data and thus no ability to make an accurate, predictive route that is personalized.

Handicapped individuals have a diverse set of needs while navigating through the built environment. Currently, there is no support for such individuals in the widely available navigational systems; however, there has been increasing interest in the research community to detect urban features in the environment. These features are necessary for the creation of personalized and accessible routes in pedestrian navigation. The latest development in assistive navigation is of a module called SmartWheels [22], which aims to identify the occurrence of urban features (such as a curb ramp), though inertial sensors that exist in a smartphone that automatically collect data, so that it may be used to compute the route for other users. This module is part of the assistive navigation system called Moving Wheels, which grants users the ability to select their preferences on what urban features they encounter during navigation. The system benefits from utilizing crowd-sourced data from users with disabilities to detect features such as steps, ramps, the type of road, uneven features such as potholes, the existence of movement aids and tramway tracks. However, the surface is only considered to be part of three classes: smooth, asphalt or dirt. Also, the paper establishes that there are two classes of wheelchairs that are used for navigation: electric and traditional. However, large collection and aggregation of data on urban features can prove difficult since it has been shown there is a larger preference for using a manual wheelchair over a power wheelchair due to it being less costly and more widely available [2]. Due to the large number of data that may be

acquired from end-users on manual wheelchairs, it becomes necessary to find a solution that may aggregate existing knowledge on features in the environment in order to assist route navigation for users operating other devices.

Navigation Systems Summary

The following table shows accessible navigation systems and the types of parameters considered as well as contributions to research. These include path algorithms, the usage of machine learning techniques and utilizing contributions from users to make routing recommendations.

Accessible Systems	Identifying Accessibility Issues (Users and Barriers) 1. Using Mobility and Vital Sign Sensors 2. Self-reporting or crowd-sourced 3. Both	1. Indoor only 2. Outdoor only 3. Both	Using ML for Surface Classification?	Developing an App or Device for Navigation?	Client Interactivity 1. Web based 2. Mobile phone based 3. Both	Route Calculation Strategies	User and accessible parameters considered
MAGUS [73], 2000	2, Accessible Paths are gathered from authorities and systematic street inventories	2	No, Focused on assigning scores to barriers that are found in urban environments	Yes, Creating a GIS based navigation device	1	Routes are assigned a score depending on the wheelchair considerations and the nature of the parameter	Types of wheelchair: Manual self-propelled, Manual Assisted, Powered Barriers: Steps, High curbs, Deep gutters, Gravel surfaces, Lack of dropped curbs, Narrow pavements, Steep gradients, Poor path maintenance
U-ACCESS [54], 2006	2, Path knowledge from GPS geo-referencing, campus information from University of Utah dept. and CDS accessible building entries	2	No, Using collected data on paths to create a map of accessible edges using ArcGIS	Yes, Creating a support tool for navigation around barriers in urban environments	1	Accessible shortest paths use a variation of Dijkstra using a hash map data structure to determine adjacent nodes.	Types of considered devices: Manual Assisted wheelchair, Electric, Use of walking aid, Peripatetic (no assistance) Barriers: Curb cuts height and width, slope width and direction, sidewalk width and step height, Handicap entrance, Handicap parking
RouteCheckr [56], 2008	2, Created a system that allows users to collaborate data and make personalized routing decisions	2	No, Uses a multi criteria routing algorithm	Yes, created a prototype client/server system with a personalized routing engine, a UI annotation interface and a profile UI for user preference routes	1	Multi-criteria route algorithm is a modified Dijkstra which uses safety and length criteria when calculating the cost of the path. The path is weighted along with user rankings	Non-specific (arbitrary) accessible criteria
PAMs [74], 2013	2, System includes a user interface, a routing and audio module, a database and map functions on interactive maps gathered from other Universities	3	No, Using the preexisting maps to generate accessible routes	No, creating a map overlay using Google Map API	3	Routing module has shortest path and personalized wheelchair routes based on preferences for type of wheelchair, age, fitness level and sidewalk parameters	User ability: Mobility impaired, Vision impaired, Peripatetic Barriers: Ramps, Surface type and condition, Traffic, Number of steps, Width and distance of route, Accessibility of certain locations (restrooms, elevators, landmarks), curb cuts, sidewalk traffic, steps
mPASS [68], 2014	3, Uses sensor data and crowd-sourcing to create personalized, geo-referenced routes for urban environment navigation	3	No, path information is used to personalize navigation by categorizing ratings of paths depending on the user's profile	Yes, the app allows for profile configuration, report on path condition, receive notification on presence of barriers and to display the report in Google Maps	2	Route is chosen based on user preferences. Users choose neutral, like, dislike or avoid on barriers and the profiler chooses paths that have the most like, go around the dislike	Accelerometer and gyroscope sensors used to detect: Traffic lights, Zebra crossings, Ramps, Curb cuts, Stair presence Others were found by user generated reports: Surfaces, Characteristic of sidewalks, Parking spaces, Obstructions

Table 2.2: Characteristics of accessible navigation system research (2000-2014)

Accessible Systems	Identifying Accessibility Issues (Users and Barriers) 1. Using Mobility and Vital Sign Sensors 2. Self-reporting or crowd-sourced 3. Both	1. Indoor only 2. Outdoor only 3. Both	Using ML for Surface Classification?	Developing an App or Device for Navigation?	Client Interactivity 1. Web based 2. Mobile phone based 3. Both	Route Calculation Strategies	User and accessible parameters considered
UPWFS [61], 2015	2, Proposes a method on intelligent routes based on self-reported parameters found from existing research	2	No, Evaluated criteria for wayfinding based on length, accident safety, social safety, difficulty and attraction	No, proposed a framework based on multi-criteria to understand the effect of context in path wayfinding	1	Each criteria has a measurement fraction based on the contributing factors and the impendance score is calculated and placed into a cost matrix. The application of Dijkstra on the matrix derives the best path	User characteristics: Age, Gender Accessible factors: Slope, Pedestrian crossing, Sidewalk existence, Width Crowds, Stairs, Pedestrian overpass Lighted paths, Length
CAPRA [72], 2017	2, Uses crowd-sourced information about distance and slope to create optimal routes to trade-off between path length and accessibility	2	No, Compares route algorithms that minimize total distance, vertical distance and slope with shortest path	No, proposes a route recommendation algorithm and evaluates total vertical distance and slope parameters	1	Use a multi-objective A* algorithm to recommend calculate routes	1. Total Vertical distance 2. Maximum Slope
eNav [71], 2017	3, Sensors on wheelchair collect energy consumption to determine incline and friction and crowd-sourcing for barriers, surface and points of interest	3	No, Uses crowd-sourced data to find the best energy-efficient route	Yes, web-based clients allow users to enter parameters for the wheelchair to calculate a shortest and efficient route	3	Uses a A* algorithm with a heuristic for energy efficiency	Parameters: Barriers, Surfaces, Inclines Knowledge gathered from crowd-sourced data: Points of Interest, Energy Efficiency Altitude, Elevation
Moving Wheels [22], 2019	1, inertial sensors detect urban features about the environment from users	2	Yes, uses sensor data on the wheelchair to detect the still, obstacles and gait of the environment	Yes, users confirm their desired route suggested by a mobile application	2	Multiple routes are chosen based on user preferences. Specific urban features along the route are chosen to be completely avoided or avoided if possible	Types of wheelchair: Electric, Self-propelled, Electric-propelled Urban features: Step height and direction, Ramp height and direction, Uneven road height, Tramway track presence, Smooth, asphalt, or dirt road type, Still, Types of gait
WheelShare [4], 2019	3, Sensors attached to the wheelchair detect features and crowd-sourced information for identifying barriers, points of interests, and elevation	3	Yes, uses machine learning to identify environment features for recommending routes	Yes, multiple accessible path routes are retrieved from a remote server and is displayed on a mobile application	2	Uses a utility function to assign a score for accessible waypoints and a penalty for inaccessible waypoints	Types of wheelchair: Manual self-propelled, Electric Knowledge gathered from crowd-sourced data: Slope height and direction, Surface type, Curbs, Barriers, Stair presence, Narrow pavements, Accessibility of points of interests, road characteristics

Table 2.3: Characteristics of accessible navigation system research (2015-2019)

2.2.4 Assistive Devices in Mobility Navigation

For people using wheelchair on a daily basis, their quality of life is dependent on being mobile. A large portion of accessibility routing depends on the needs of the individuals. Although there is shared knowledge about the environment, this knowledge may become outdated and may even be impassable for a user using a different accessibility device to navigate. Also, a user may own multiple devices that are used depending on the length of travel and the characteristics of the built environment. This section explains the selection and preferences of accessibility devices by different age groups.

Types of Accessibility Devices

There are many devices which assist in navigation. Manual wheelchairs are lightweight, require little maintenance and can be transported without specialized devices. Mobility scooters and power wheelchairs enable users to travel on longer journeys and go through obstacles or terrain that may have been difficult using a manual wheelchair. Individuals with obesity, weak upper limbs or trunk stability, and cardiopulmonary disease may benefit through the use of power devices. A user may also use a mobility scooter as a replacement for a car or used for just short trips. They are also relatively inexpensive and are available in many cities.

According to the World Health Organization, approximately 1 billion people need assistive technologies, which include people with chronic conditions, communicable diseases, with disabilities and the elderly. Of the 75 million people who may need a wheelchair, 5-15% of the world population has one [75]. In the United States alone, approximately 1.5% of the population aged 15 years and over use a wheelchair. In the United States and around the world, the number using mobility scooters is increasing, but there is no evidence to suggest that the number of people who have difficulty walking has increased. The latest American Community Survey (ACS) that was conducted by the United States Census Bureau [1] in 2018 shows the number of individuals who have ambulatory disabilities, or those that have difficulty walking and using the stairs and is shown in Table 2.4.

	Total		With an Ambulatory Disability		Percent with Disability	
	Estimate	Margin of Error	Estimate	Margin of Error	Estimate	Margin of Error
Total Civilian population	322,249,485	+/-15,307	20,655,956	+/-96,831	6.8%	+/-0.1
Population under 18 years	53,578,007	+/-31,844	327,366	+/-10,932	0.6%	+/-0.1
Population 18 to 64 years	197,888,546	+/-26,117	9,399,859	+/-66,055	4.8%	+/-0.1
Population 18 to 34 years	74,168,432	+/-45,231	934,643	+/-17,792	1.3%	+/-0.1
Population 35 to 64 years	123,720,114	+/-51,097	8,465,216	+/-64,037	6.8%	+/-0.1
Population 65 years and over	51,137,346	+/-22,004	10,928,731	+/-57,517	21.4%	+/-0.1
Population 65 to 74 years	30,226,493	+/-22,963	4,450,622	+/-36,123	14.7%	+/-0.1
Population 75 years and over	20,910,853	+/-17,765	6,478,109	+/-39,297	31.0%	+/-0.2

Table 2.4: Sampling of Individuals in the United States with Ambulatory Disabilities in 2018. [1]

A survey of 66 scooter users suggests that the main reason for using a scooter is to enhance and not increase mobility [76]. However, 70% of those who use electric wheelchairs and scooters are nonelderly and about 54.9% who use mobility devices are nonelderly [77]. A current survey that was taken in 2018, in which the average age of participants is 54.3 years of age, shows a higher percentage of assistive device usage [2]. In the survey, participants were allowed to choose more than one device, with the majority using a manual wheelchair at 58.3%, followed by a power wheelchair at 47.1% as shown in Table 2.5.

		Number of Participants	Percentage of Participants (%)
Total		1022	100
Using Assistive Devices	Manual Wheelchair	596	58.3
	Power Wheelchair	481	47.1
	Scooter	98	9.6
	Lower extremity prosthesis	49	4.8
	Lower extremity orthosis (brace)	134	13.1
	Assisted Device (e.g., cane, crutch, and walker)	387	37.9
	Other	99	9.7

Table 2.5: Survey of Users of Assistive Devices in the United States in 2018 [2]

The National Health and Aging Trends Study conducts annual, in-person interviews on disabled individuals aging 65 years and older in the United States [3]. The National Health and Aging Trends Study (NHATS) is sponsored by the National Institute on Aging (grant number NIA U01AG032947) through a cooperative agreement with the Johns Hopkins Bloomberg School of Public Health. Participants of the NHATS study were asked which accessible devices did they use in the past month and responses are indicated in Table 2.6. About 38% had reported at least using one type of accessible device recently.

Accessible Devices	Total Responses	Confirmed Using Device(s)	Denied Using Device(s)	Inapplicable	Don't Know	Missing
Cane	5547	1205	908	3321	1	112
Walker	5547	1132	982	3321	0	112
Wheelchair	5547	695	1419	3321	0	112
Scooter	5546	128	1984	3321	1	112
Any Listed Above	5547	2114	3210	105	6	112

Table 2.6: Number of Participants (age 65 or over) who have used an Accessible Device in the last month according to the National Health and Aging Trends Study (2018). [3]

The rise of canes in the elderly may be due to the aesthetic look of devices and the idea of a social acceptance of disability [78]. One may consider a cane is less disability looking that a wheelchair or a manual wheelchair to be less disabling than a powered wheelchair. A device can not only provide a sense of security, autonomy and competence but also affect the physiology of the individual. A once passable obstacle now impassable can lead an individual to have a sense of embarrassment or limitation. If they are prescribed an assisting device, the factors affecting their prescription include their diagnosis, cognitive ability, physical ability, age, weight, interest, the environment they are using the device, the time spent in the device, independence needs, transportation, other medical issues and insurance [79]. Therefore, individuals would choose a device that best fit their personal, economic and social needs.

Chapter 3

System Framework

The WheelShare system utilizes a unique, crowd-sourcing technique to collect location-tagged accessibility information from urban areas in order to identify accessible features and recommend personalized routes in the built environment. Mascetti et al. [22] reports that current accessible systems are designed for specific geographical locations, even though many similar features, such as sidewalks and curbs are found in every city. To address this concern, the WheelShare system includes machine learning in order to learn about various types of surfaces gathered from multiple smartphones with different sampling frequencies from different parts of the world. One of the biggest contributions of WheelShare is to contribute information about buildings, roads and urban features to a global, open source map. Users may participate in crowd-sourcing data about various features or use the navigation feature for planning their day-to-day routine. Once users are registered to the application, they can choose to be placed in one or both groups as contributors and route requesters.

In order to aid users with providing accurate and reliable knowledge about the environment, a system framework and application for WheelShare was proposed in [4]. Part of this framework is extended into the newer framework that is used for the basis of the WheelShare application. This application is currently being developed in Miami University for Android devices based on the framework shown in Figure 3.1. While offering multiple routes is useful for the user, generating them is non-trivial. Multiple routes are recommended for wheelchair users based on the nature and types of features found in the overlay of the current environment. Currently, the application can register users, recommend multiple accessible routes in real time as well as classify the current surface the user is on using the trained classification models also proposed in this work. Route generation includes several algorithms, including the assignment of accessibility scores to features in the built environment and route selection depending on the length and score of multiple paths. This application may be used to navigate pedestrians who use a manual or powered wheelchair and those who do not. Also, accessible entrances in buildings each have one or more route to the destination location. In Section 3.1, the specific architecture from the Figure are described, as well as possible user interactions on the application.

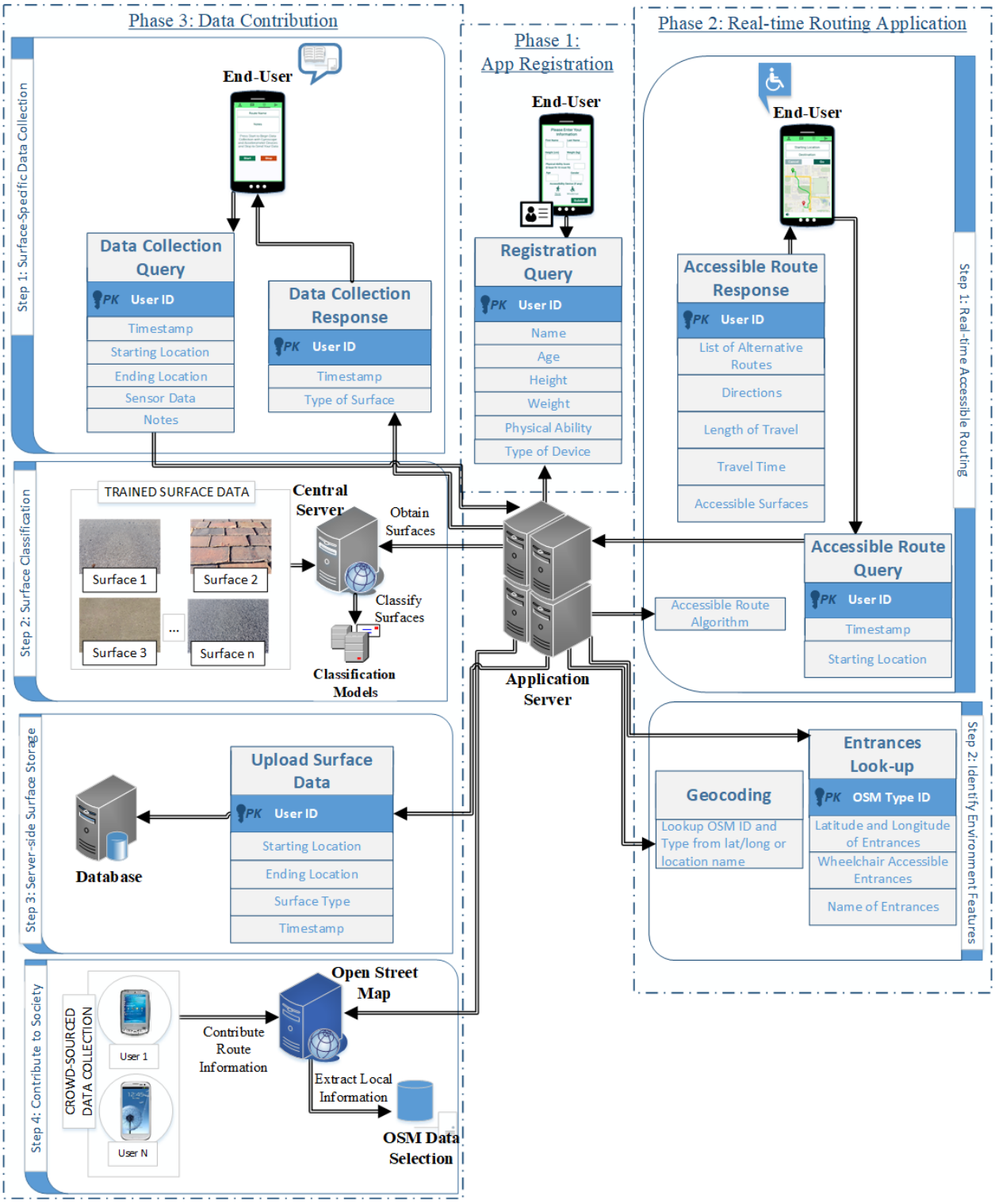

Figure 3.1: WheelShare system architecture

3.1 Architecture Elements

The system first undergoes a initial phase and a live phase, which are explained further in Section 3.2.1. Essentially, the initial phase includes sensor data from both indoors and outdoors for curbs, flat paths, as well as for various surface types. In the live phase, users are able to contribute information about map features to a large, open source map database and receive accessible route recommendations. WheelShare uses map data from Open Street Map (OSM), which allows for geocoding of location names and has the ability to tag information on nodes and ways for route planning. Local information is collected using a GPS unit and then is labeled and attributed before being uploaded for route surveying. The WheelShare uses the GraphHopper open source, routing engine, which associates GPS locations with the road graph edge that best matches the data in OSM, in order to built our accessible routing technique. The data is available under the Open Database License and cartography from our map tiles is licensed under the Creative Commons Attribution-ShareAlike 2.0 license (CC BY-SA 2.0). GraphHopper is an OSM-based external routing application because it uses OSM to interact with the road network. GraphHopper also has other important tools for accessible navigation, such as elevation data display for a particular route, which is collected from the Shuttle Radar Topography Mission. The speed and time is calculated from the elevation data and is factored in with the urban features that occur along the route. For example, the larger a curb angle is, the slower the wheelchair is assumed to move past it. The estimated travel time is also sent as a response back to the user in order for aid in choosing their preferred route. The routes that are chosen have priority of wheelchair accessibility over solely determining the shortest path to the destination.

Other popular map databases, like Google Map Maker, do not allow for free use of open source data, the data is delayed by a long review process and generally does not focus on pedestrian routing. In contrast, the GraphHopper server also has existing open source API available to interact with OSM, can import data about the environment from previous users and shares our interest to generate routes from open map data. It offers complete customization to create routing profiles and design routing algorithms. It also has a fast and efficient routing library that has been tested extensively and calculates user specific paths and travel time by factoring in elevation, speed limits and distance. A variety of routing algorithms are supported, including unidirectional Dijkstra, one-to-many Dijkstra, uni and bidirectional A*, and by default uses *Contraction Hierarchies* to find a single route from A to B. Therefore, the GraphHopper API gives certain advantages to generate accessible routes, because it is compatible with Open Street Map and can utilize our own wheelchair routing algorithm to serve mobile users. GraphHopper also uses profile aware routing by utilizing the scores of different way types, which are based on the average user vibration data, as well as tags for stairs, ramps curbs, etc. to make decisions on edge weights. Our remote server queries GraphHopper over HTTP in order to display routes through our WheelShare application. The server also processes the HTTP requests for point-to-point routing and services in identifying surfaces from sensor data collected by the user.

The WheelShare application server sends HTTP requests to the public IP address configured for a NGINX web server, which defines a GraphHopper web server as a virtual server that would process the request. The NGINX server acts as a proxy to route the client requests to the GraphHopper server. The GraphHopper web server is configured to send and receive API about routing and

data collection. In addition, GraphHopper interacts with a private web server and database on the network in the case that it receives a request for data collection. The following sections discuss the three main phases of the system: (1) user registration, (2) data contribution, (3) accessible route planning.

3.1.1 User Registration

Users first interact with the registration screen, which collects information about the user in order to personalize their navigation preferences. This page allows the user to enter their name, height, weight, physical ability, age, gender, and the assistive device they are using (if any). Depending on the user's device selection preferences, some routes may not include certain features. For example, a walking pedestrian may encounter stairwells and steep curbs, while a user with a wheelchair device would not see routes containing these features. User input details are stored locally on the android phone, so that they may be retrieved at any point after registration and displayed in the application. Users may then select how they would like to use the application as route requesters, data contributors, or both and must grant their consent for the application to use their location in order to move to the next phase. The user is tracked in real time by implementing the Location Manager to acquire GPS details from the phone. For this reason, the user has to consent to share their location before proceeding with any other function. The local SQLite database serves as a storage for user data, and it provides registered user information for the application server. The user may see their registered information at any point in the user information page. This data can be accessed anytime during routing or data collection through a toolbar feature which switches between the various functions of the application. Therefore, it is entirely possible to separately use *data collection* and *routing and navigation* (*Map My Route*) operations. After registration, the application sends user information to our secure database.

3.1.2 Data Contribution

The contributors are the most important part of our system as they provide surface vibration data collected through their smart devices attached to their wheelchairs as they travel. Contributors forward their collected geo-tagged acceleration and gyroscope data to our central server which classifies them according to the previously learned model. In the case that a user wants to participate in contributing surface or slope data, they would follow the interactions shown in Figure 3.2. On the Data Collection page, the interface communicates with the application server to show the surface type and also with a remote database to log information about what surface data was collected, who uploaded it and what is the predicted surface type from the trained model. The system uses an unobtrusive data collection method where user-permitting, the data collection takes place in the background while the user navigates through a prescribed route.

The sensors in the application begin to collect information about the movement of the phone when 'Start' is pressed. The recording does not need to be configured by the user but by default records acceleration and gyroscope data with 50Hz and GPS data with 1Hz. The data is stored on the local device until the device connects to a WiFi network where the anonymized data is uploaded to our central data server. The server pre-processes the data and classifies each data window

according to the model that has been created in the initial phase of the WheelShare system. After 'Stop', the application sends the vibration data to the GraphHopper server to receive the predicted surface type, which is displayed on the screen and uploads the data along with the surface type and the contributor details to the remote database. Since the mobile application is on an Android device, it is possible to use the API to communicate with the sensors on the phone and send sensor data by implementing a Sensor Manager and Sensor Event Listener method. This surface data is composed of a number of samples per second that is uniquely identified by an ID, a timestamp, the sensor name, and X, Y and Z coordinates. Our remote server sends this data as an HTTP POST request to a secure, private web server on the network, which processes the data by extracting the various path features including surface-induced vibration data and slope collected from the smartphone accelerometer and gyroscope sensors.

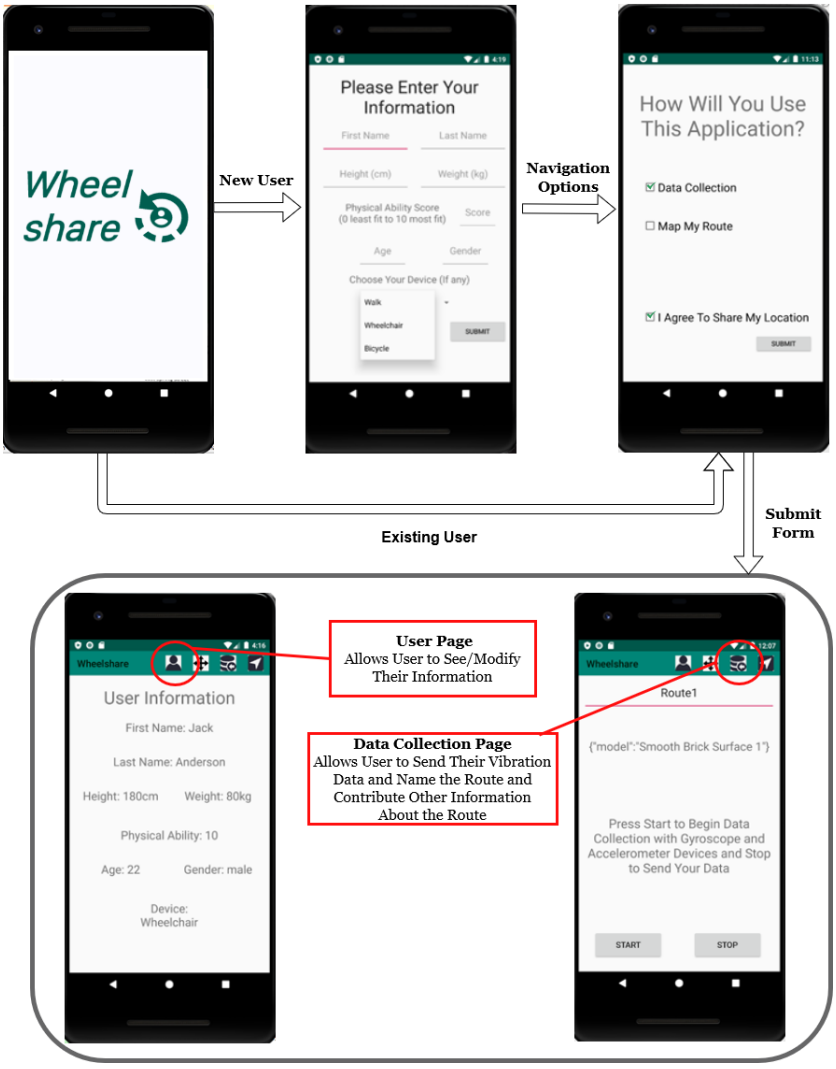

Figure 3.2: A diagram of possible user interactions with the data contribution functionality of WheelShare.

In order to learn about the built environment, several pre-trained models classify specific surface features along the route based on this contributed data from users. Only the knowledge that does not vary from individual to individual is used for classification model training, which is the surface type and slope data. The creation of the classifier is described in Chapter 5. However, the purpose of training a classifier in the algorithm is to identify accessible surfaces and curbs. Based on these features, an additional algorithm is performed in order to check whether paths fall under ADA standards [13]. Scores are assigned on different edge weights. The user is then presented with the routes chosen using the accessible routing algorithm. The target goal is to model the decision process for pedestrian navigation, which requires a decision to be made about the accessibility of the current path. This goal is modeled as a target function that is found through the application of a learning algorithm. A classifier is one type of learning function, a discrete-valued function, that is created by the learning algorithm. In a sense, a classifier is a type of hypothesis that is an approximation of the target function. It is necessary to have the knowledge about the hypothesis space so that a classifier is trained on a specific domain. A set of all hypotheses $h_1, \ldots h_n \in H$ where H is the hypothesis space allows for a target function, a hypothesis, to be found. This function represents $f(x) = y$ that accurately labels an output y to an input x and can be used to predict the surface type and slope. The intention of this algorithm is not for ensuring ADA compliance upon initial construction of paths, but from external conditions such as wind erosion, weather, floods and earthquakes that can affect slope and condition of the path. The true slope for paths often are difficult to identify since public records are often not up to date with its current condition. Several machine learning algorithms may be used to train the surface model based on known truth about accessible characteristics of the path. Through more interaction with the system, the model can gain a more objective understanding of path accessibility. Therefore, automatic detection, along with crowdsourcing, aims to solve limitations of feature detection in the built environment.

Once the top accurate model is chosen from the list, the most likely surface prediction is returned from the private server which is displayed in the Data Collection page on the application. The data collected from this screen is associated with a user id, which is sent with the identified surface type to the remote MySQL database using a JDBC connection object, and it allows the application to connect to our remote database. This information is added and modified to the existing OSM local file. Contributors to OSM would make periodic updates to ensure that knowledge about the navigation environment remains relatively current. By cross-referencing the collected information about the road with the surface found at that location on OSM as well as the timestamp when the surface data is collected, the global OSM data file is synced with the local file by verified contributors from the server side. Syncing two dumps of OSM data is relatively simple as OSM contains a general-purpose command-line Java-based OSM data tool called Osmosis which can provide periodic, daily change sets to the database. Therefore, the local file can be updated frequently with current information about the environment. After these changes are made, the new local file is used for making route planning decisions for users with multiple navigational devices.

WheelShare Remote Database

The remote database is used to maintain the OSM file for making routing decisions and understanding the user's navigation preferences. Our data server stores the surface model and as well as data

received from the contributors. Updates about the roads with accessible surfaces are used for making decisions to update the OSM local file. The updates contain tuples that store the accessibility level of the path at a certain location. The remote database of the application contains three tables: User, Roads, and Update data.

- *User table*: The 'User' table is used to save the personal information of the users. The user information is sent to the database with a unique ID (primary key) after registration.
- *Roads table*: The second table is called 'Roads', which contains the information of every road with an identified surface. Roads are represented by a Road ID, road name, the surface type, and its coordinates. After data collection occurs and a surface type has been identified by the model, this road information is saved into the database and can be later used for making route decisions.
- *Update data table*: Raw sensor data from users are uploaded to the 'Update data' table and is identified by the user unique ID. The 'Update data' table is used by verified contributors for retraining the classifier on different surface types and adding and updating tags about various obstacles or path conditions.

3.2 Accessible Route Planning

For locations starting at buildings, multiple routes are shown on the application, one route for each accessible entrance in the building. Therefore, our application assists in indoor and outdoor navigation, as it may be difficult to find accessible entrances inside buildings if no signs are posted inside the building. The process of converting an address to a latitude and longitude coordinate is called geocoding. Every building has a Type ID in the OSM database and a list of node IDs that can be tagged as entrances, stairs or elevators as well as be tagged as accessible for particular assistive devices. First, the entrances of every building in the local environment is recorded on Open Street Map by contributing users. This information can be exported as a local OSM file (in PBF format, which is a compressed, binary format of map data) and read by the server to obtain the latitude and longitude coordinates of every entrance during route generation. Reading and processing an existing local file from the OSM server reduces overhead of making API calls to OSM, which also decreases processing time. Users with physical impairments are mostly interested in short distance traveling, since they rely on a battery charge or have a limited muscle strength for manually propelling a wheelchair. Therefore, we limit the application to a local OSM file in order to decrease memory usage and battery consumption. This file can be generated for the region corresponding to the user location. After all possible accessible entrances are found for a particular building, they are place marked on the map and users may choose from any of them to reach their destination. Routes are displayed using an exported image from OSM. Once the user enters a name or address, the location points are geocoded by performing a forward search on the OSM database. The location points are sent as parameters in the HTTP GET URL request to the remote application server in order to generate multiple routes using our algorithm. The server responds with a list of multiple routes from each starting point to the destination, at which point the route data is displayed on the

application in the Directions page. This page consists of the distance until the next step, the type of surface or street name, estimated time of travel until the next step, and the specific direction to follow. The Routing Navigation page is updated with multiple routes that are identified by different color themes. The following section describe the phases of the system for suggesting accessible routes.

3.2.1 Accessible Routing Decisions

An adaptable, accessible routing algorithm considers several elements, which include the source and target destinations for the trip, information about the user, including weight, physical ability and age, wheelchair characteristics, slope measurements and surface types of the road. WheelShare suggests multiple accessible routes using a two phase approach.

Initial Phase: Pre-processing the Overlay and Training

An overlay is constructed on the map of the user surroundings. Data servers in the WheelShare system generate the overlay graph automatically and is continuously maintained and updated. The overlay consists of vertices and edges, where each vertex represents a single point on the map. A change of surface type, hence vibration pattern, can occur at junctions or pedestrian crossings as a user on a wheelchair moves along that path. This phase also includes training of the classifier. We collect geo-tagged accelerometer and gyroscope data while pushing a wheelchair over different types of common surfaces. We use the recorded sensor data to train a classifier. This classifier along with additional algorithms allows us to infer the accessibility of the surface a quantitative manner, depending on the vibration induced by it. The overlay and information about each surface is further utilized in the second phase.

Second Phase: The Live Phase

The live phase consists of contributions made by route requesters and contributors to the system. Once the central server classifies the surfaces from sensor data, the location of these classified waypoints are added to the accessibility overlay. With assistance of crowd-based data collection, the knowledge about the environment increases. Route requesters use our routing application to retrieve an accessible path to their destination. Upon receiving a user query for an accessible route, the application server executes the routing algorithms over the graph and evaluates the accessibility of multiple alternative routes. Route requesters are able to see the directions along with the map of the selected routes in the application (see Figure 3.3).
Users can select a route from a presented list of routes in the Directions page, which is then visualized on the map. The Directions page also provides textual information about each street, including the direction and time of travel, as well as important information such as the building name, presence of certain surfaces and direction information (*continue*, *turn right* or *turn left*), estimated travel time and the name of the segment (*street name*) for each segment of a route. The user chooses one of the multiple routes presented to them and follows the textual navigational directions. For each individual segments of a route, it shows the total distance, direction information. This information

is generated from the remote GraphHopper server and is received to the application at the time of route querying.

Figure 3.3: Route requester application interactions

Chapter 4

Core Machine Learning Algorithms

Machine learning algorithms are used in order to understand the structure of the data in a particular domain of knowledge. The challenge is how to use previous knowledge gained from the abundance of information provided by manual wheelchair contributors in order to gain knowledge about the environment and create personalized routes for a user with a similar mode of transportation. We know that there are factors of variation from the features extracted from vibration data. There may be generic commonalities that are both shared in the manual wheelchair (MW) and power wheelchair (PW) domains. Deep learning algorithms and the application of transfer learning have not been explored in classifiers using surface vibration data. Yet there is a remarkable opportunity for large data collection from crowdsourced methods. The availability of smartphones in the digital age led to ability to acquire a large amount of data to make accurate predictions on the accessible surfaces in the environment. Only recently have deep learning models, such as convolutional neural networks, have shown accurate classification results from data acquired from inertial sensors, such as in human fall detection [80]. Since our goal is to predict the surface using a snapshot of information taken through an interval of time from multiple sensors, we can frame the problem as multivariate time series classification problem. This is challenging given that sensors may indicate a wide variety of activity and there is no direct method of relating this data to specific surface types that a user has traveled on. Therefore, machine learning can enable a generalization of the problem in order to classify surfaces by unseen sensor data from users on the same wheelchair. This section describes the classification models trained on the wheelchair experiment data and the two approaches that we took on transfer learning involving model transfer and feature transfer.

4.1 Classification Models

The goal for this section is to investigate the performance of some of the most common classification models on MW data collected from our experiments. SmartWheels [22] has looked at flat random forest (RF), naive bayes (NB) and support vector machine classifiers (SVM). Two additional learning algorithms are presented in this work: K-Nearest Neighbor (KNN) and Decision Trees (DT). We also consider the more sophisticated deep learning classifiers: Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Long Short Term Memory Networks (LSTM). These machine learning techniques are trained to features extracted from surface data and manually labeled with the ground truth about the surface type. In this section, we describe the machine learning algorithms for each of the classifiers.

4.1.1 K-Nearest Neighbor

In KNN, predictions about the class is made through the training data input, which has the K closest training instances in the feature space [81]. The K nearest neighbor value is determined through 10-fold cross validation on the MW dataset for odd values of K from 1 to 9. A grid search is performed to choose the K value that gives the most accurate results. Though there are some methods that aims at finding the most optimal K by determining the largest local neighborhood of data points for each label [82], we do not use this approach as this may introduce bias in the data collected from sensors, which may have unpredictable distribution. Limiting the boundary size leads to a more selective decision on the surface type.

KNN consists of a training phase where the feature vectors are saved and their corresponding class label. In the classification phase, the test sample is determined by the label which occurred the most commonly among the most similar training samples. Similarity is based on the distance from the training samples to the new test data sample. We use a uniform weight function, so all the points in the neighborhood are equally weighed. The most common way to determine distance is using the standard Euclidean distance metric. If n is the total number of data points in the n-dimensional space, p and q are first and second points in each dimensional space.

$$distance_{(p,q)} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{4.1}$$

Each new point p in the test set is compared with all the training points within the specific boundary determined by the K value. p is assigned to a class with a high probability. This calculation is based on the observation that points that are close together would have the same mean. Euclidean distance determines the goodness of the estimation prediction with the true class label.

KNN is a lazy learning method, which means it keeps all of the training data. Recommendation systems often use KNN for its simplicity. Other approaches such as decision trees look at finding representations of the training data.

4.1.2 Support Vector Machines

SVM treats the input test data as an n-dimensional data and belong to different class labels. The points are assigned to a space and are mapped so their categories are split by a hyperplane with the largest distance between the nearest data points on each side. For multi-class classification, the "one-against-one" approach is used, in which a binary classifier is trained on every class label pair. This approach was proposed by Knerr et al. in 1990 [83]. At first, all the classes are candidates of the true class. An $N-1$ number of hyper planes, or decision functions, are made in the dimensional space. This is an expensive operation and has a long computational time as because if N is the number of classes, then $N*(N-1)/2$ classifiers are trained. Test data is then applied to each of the trained classifiers. Class labels are assigned using a max-wins voting strategy, so every classifier assigns a label or vote to one of the two classes. The instance is classified based on the class which received the most votes.

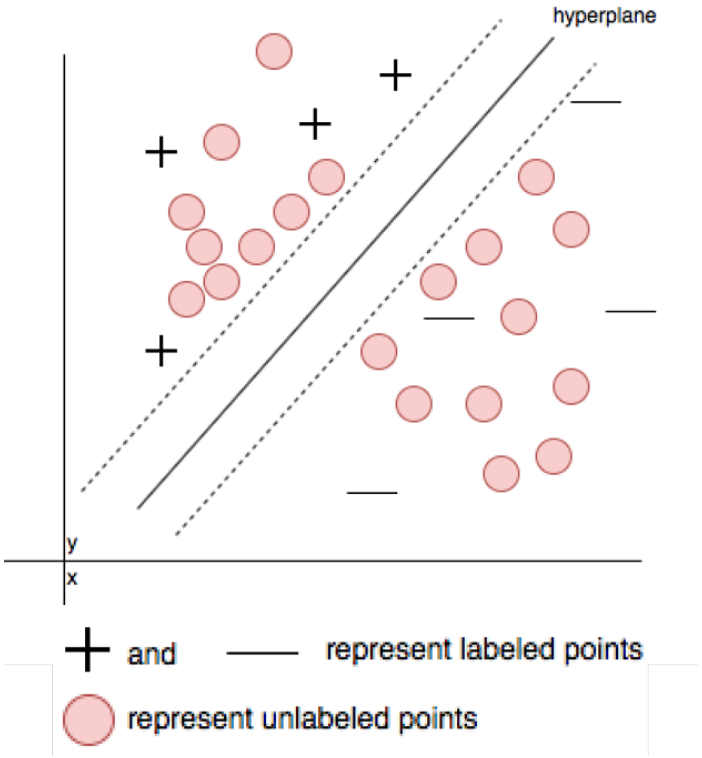

Figure 4.1: Support Vector Machine hyperplane visualization example

Figure 4.6 shows how a decision function is applied to an unlabeled, or test point. Candidates that are within the dotted region that separated positive and negative points are eliminated. Max-wins voting strategy is used to predict the class for a test point. SVMs can be improved using kernel machines, which show that the algorithm can be written in terms of a dot product. However, the cost of evaluation is linear with the number of training examples.

4.1.3 Random Forest

RFs are based on ensemble learning which combines multiple decision trees of the same type, creating a forest from training set samples [84]. Each node is split on the tree based on either all input features or a random subset of the max number of features. Random splits reduce variance of the estimator, preventing overfitting and reducing bias as there are multiple trees on a sample of the training data. Our random forest algorithm uses a random subset that is the size of the square root of the number of features in the training data and we used a threshold of 1000 trees in the forest. We also set the limited number of leafs to split an internal node to 2. The computational complexity is $T * N * log(N)$, where T is the number of trees in the ensemble and N is the number of samples. The probability for assigning a class label c to a node n for a tree t is $p_t(c|n)$. For a forest of T trees and $t \in \{1, ..., T\}$ then the predicting a label for an ensemble is $p(c|n) = \frac{1}{T} * \sum_{T}^{t=1} p_t(c|n))$.

RFs use several methods for reducing correlation between the trees in the ensemble. Decision trees are prone to sensitivity to the data, therefore one solution to this is bagging. In this way,

random sampling of the dataset across multiple trees is chosen and several different trees are formed. Also, a random subset of features is chosen for training each tree. Trees are diversified and unlikely to have similar correlations to one another.

4.1.4 Naive Bayes

NB uses conditional probability to predict the class label given that a test data point belongs to a class [85]. Naive Bayes assumes independent correlation for all the features, so the presence of one feature does not affect the presence of another. Naive Bayes formula says that given an input x being a set of dependent feature vectors and y being the true class variable then the probability of the test data features given the true class label is:

$$P(x|y) = \frac{P(x|y)P(y))}{P(x|y)P(y)dy} \tag{4.2}$$

NB works well for small training data amounts. Classification is not computationally complex as every feature distribution can have an independent, one-dimensional distribution. A Gaussian NB classifier is used for training, which makes a different assumption on the distribution of the probability. For a dependent feature vector x_i, the Gaussian probability density function is:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{1}{2}(\frac{x_i-\mu_i}{\sigma_y})^2} \tag{4.3}$$

where σ_y and μ_y represent the variance and mean for each attribute for every class that are estimated from maximum likelihood. The probability of every class can then be calculated for the test data set. The Gaussian NB classifier is simple to implement as only the mean and variance (or standard deviation) needs to be estimated.

4.1.5 Decision Trees

DTs break the input training data by creating each node as one region of the input space and the children of the node are each a sub region of that space. In a DT, every internal node represents a test case on an attribute, the branches are the outcome and leaf nodes are the predicted class label. Every feature is considered when splitting the node. The feature is chosen on the basis that results in nodes of the greatest observational separation. This results in a creation of a DT of features and labels with a formulation for a set of rules in order to make predictions. DTs are prone to overfitting as they use all of the feature data to formulate rules. Also, when considering a large number of classes, there are problems of "high-dimensionality" in the training sample and may be improved by choosing the *priori* probabilities from the data [86]. *Priori* are based on previous experience from the training data set.

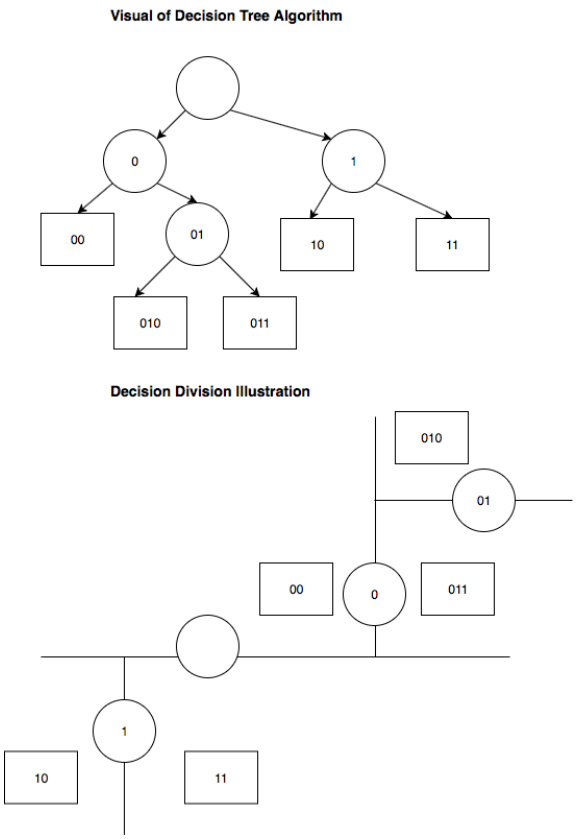

Figure 4.2: Decision tree with binary rule nodes

Figure 4.2 illustrates a decision tree. Every node of the tree is either an internal node (circle) or a leaf node (square). A binary string identifier determines the position of each node that is appended to the parent node to the (0=left direction or 1=right direction). Every leaf requires one training example and therefore the resulting illustration shown below is a piecewise-constant function, meaning every sub region consists of only one leaf.

4.1.6 Artificial Neural Networks

The training process for neural networks is quite different from standard machine learning methods. Training has a sequence of ordinate steps for tuning synaptic weights and thresholds of the neurons in order to predict the class label [87]. ANN is able to extract specific system features that are derived from the training sample data. In supervised learning, there is an adjustment procedure which changes the variables, such as learning rate, of the network after each epoch, or step in training. This procedure is based on the priori of the target labels.

The main structures of the Artificial Neural Network are:

- *Input Layer*: Data samples and features are passed as input and are normalized with an activation function.

- *Hidden Layer*: Layer of neurons are in charge of extracting patterns for learning.
- *Output Layer*: Neurons produce the final output.

We apply two different types of activation functions. Rectified Linear Unit is applied to the input and hidden layers, while softmax is applied to the output layer. Rectified Linear Unit (ReLU) are simple because what returns is a value that is inputted, so long as it is greater than zero. Softmax is used commonly in artificial neural networks for multi-classification problems and produces multiple class label outputs. Softmax calculates probabilities that a class would appear given the input from the previous hidden layer. The numbers are normalized by the exponent of the output vector, so all the values in the vector add up to one.

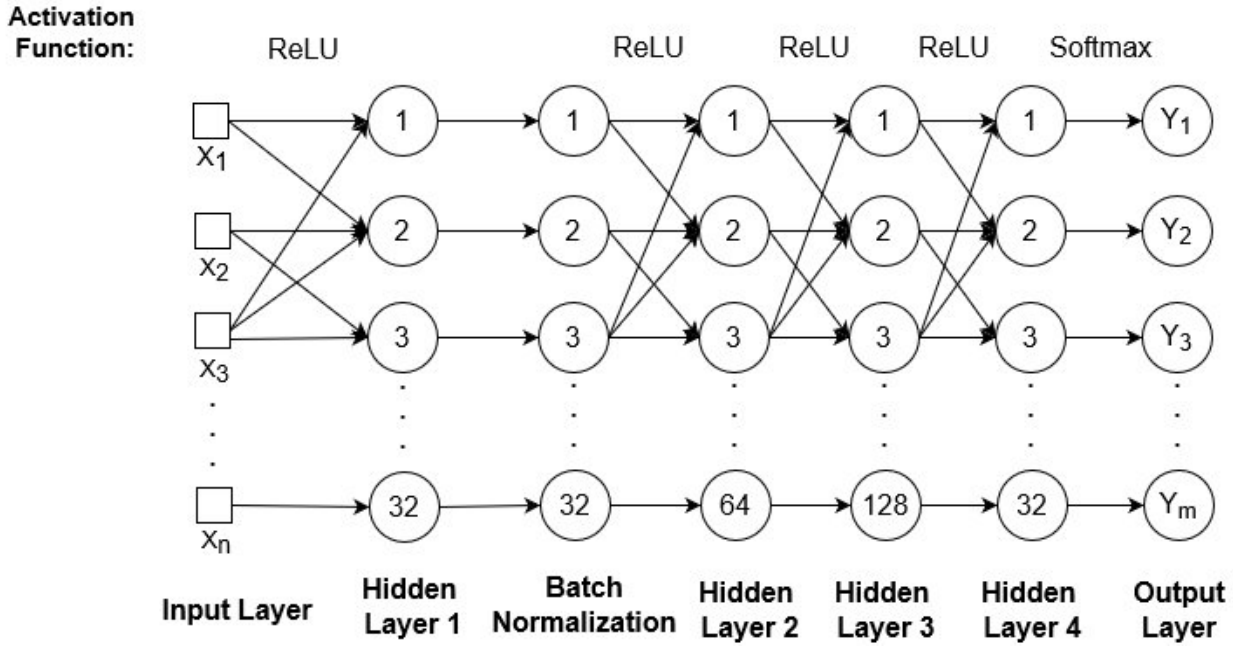

Figure 4.3: Training layers for the ANN model

Figure 4.3 shows the input, hidden and output layers applied on the training data set. This network consists of n inputs, which is the number of features, and m outputs, or the number of class labels. We apply one hidden layer before batch normalization is performed, which scales the activation results and increases stabilization of the training.

Batch Normalization

Batch normalization allows for other hidden layers to learn more independently of each other as two training parameters β and γ are adjusted for each activation function. Batch normalization is calculated by deriving the mean (μ_β) and variation (σ^2_β) of the mini batch $\beta = \{x_1 ... x_n\}$ where n is the number of activation values in the mini-batch [88]. The mean of the mini-batch is calculated

as:

$$\mu_\beta = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.4}$$

The variation for the batch is:

$$\sigma_\beta^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_\beta)^2 \tag{4.5}$$

The normalized activation values ($\hat{x_i}$) are then derived from the following:

$$\hat{x_i} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \tag{4.6}$$

The ϵ is a constant that is added for numerical stability. The normalized values are then scaled and shifted from the two trained parameters β and γ:

$$y_i = \gamma * \hat{x_i} + \beta \equiv BatchNormalization_{\gamma,\beta}(x_i) \tag{4.7}$$

This step is important for activation functions that have batch training as it leads to more efficient training. The resulting output values (y_i) are then passed to other hidden layers.

4.1.7 Convolutional Neural Networks

CNNs are similar to ANNs, but they often have additional convolutions to the ReLU hidden layers, such as pooling and may also have normalization layers. Neurons can share the particular shape of the input vector that is represented by weights. A function is applied to the input vector from the previous layer and then used in the next layer. Figure 4.4 shows the transformation of the input matrix across the various different hidden layers in the CNN model with n being the number of input feature columns and m the number of output classes.

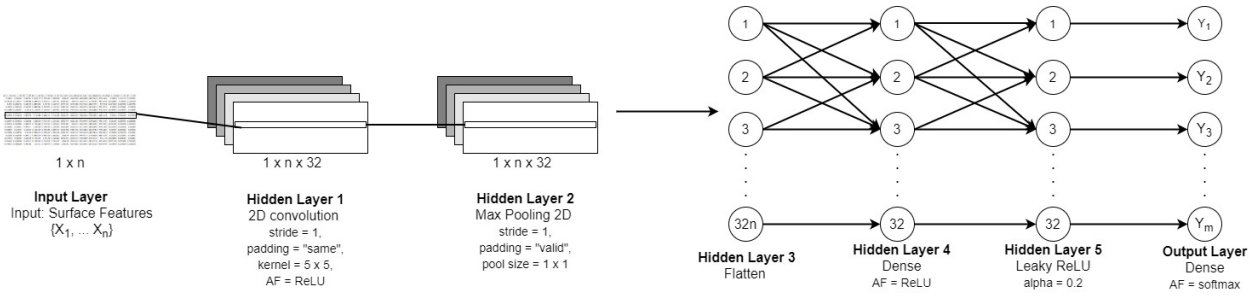

Figure 4.4: Training layers for the CNN model

Hidden Layers 1 and 2 show the representation of data in a 2-Dimensional space, which transforms to a 1-Dimensional space in the third hidden layer. The max pooling procedure decreases computation required by reducing the size of the data and the parameters and is important to

control over-fitting. The flattening procedure in hidden layer 3 is crucial to perform additional ReLU activation functions on a single feature vector. The fifth layer uses Leaky ReLU, which adds a gradient to the inactive nodes. This is to reinforce the convergence of nodes that may not be possible using a normal ReLU procedure, which maps negative nodes to zero. This layer increased performance during the cases when the optimization did not improve training.

4.1.8 Long Short Term Memory

Recently, there has been increased interest to develop Long Short Term Memory (LSTM) models for human activity recognition. [89]. LSTM is a type of recurring neural network (RNN) which avoids the vanishing gradient problem [90]. RNNs improve over Convolutional Networks in that they allow for sequences of vectors and not just a fixed amount of layers or vector size. It is a frequent choice for many classification tasks, including image recognition, sentiment analysis and language translation. The input vector can be joined with the state by learning a function to create a new state vector, which in turn is then adapted over a period of training time. The LSTM requires a feature set and the number of hidden units in the hidden layers.

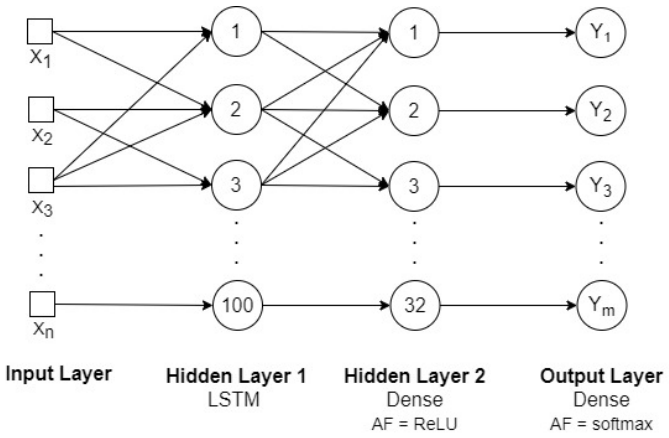

Figure 4.5: Training layers for the LSTM model

Figure 4.5 shows the layout of the LSTM training process. Our network has an input layer, two hidden layers, and an output layer. Hidden Layer 1 is self-connected that has 100 hidden units that are connected to the next corresponding layer. Hidden layers also contain memory cells, which control information and may contain gate units. Gate units help to avoid input weight conflicts and are used to decide whether to use the information in that particular memory cell for training.

4.2 Transfer Learning for Surface Classification

Our approach is to use transfer learning in order to optimize the learning performance and accelerate training of the model in order to recognize important features for route planning for powered

wheelchair users. Due to the scarcity of labeled data collected by power wheelchairs, reconstructing a new target model from a smaller amount of data alone may not lead to efficient or reliable performance, especially in deep learning models that require tens of thousands of new entries in order to tune the parameters. The purpose of transfer learning is to transform the distributions from features trained by one model from the source domain to another model in the target domain. Transferring learning from models that are trained on data collected from different mobility devices, or different distributions, is called model transfer. This type of transferring is different from traditional transfer learning frameworks like instance transfer as no training data from the source domain is used in the target domain. In order to make predictions about surfaces for users who are using a PW, we consider the problem as being in the target feature space and having different distributions than models trained on MW experiments.

Two different types of transfer learning methods are examined in this study. The first approach involves choosing a regular supervised learning algorithm that gives the highest accuracy results on the MW experiments. From training the models on the data from MW experiments, we achieved an 89.59% accuracy with training using the Random Forest model when no resampling occurred and 98.63% with resampling methods when using all features to classify each surface (results explained in Chapter 5.2). We chose to perform an existing approach on the random tree knowledge transfer proposed in Segev et al [30], which modifies the structure of the model based on the individual nodes in the existing tree. The (SER) or expansion/reduction algorithm uses the target sample data collected from PW experiments to optimize the original model trained on MW data. The approach performs a greedy search to locally optimize the tree structure to suit the target domain given a sample of sensor data collected from PW users. The application of transfer learning can lead to more scalable route calculation as source data does not need to be used after training the initial model. Since users perform in similar domains, the assumption is that the source and target domain share the same feature and label space. However, the distribution of features may be different. The recursive algorithm is described below.

Algorithm 1 Structure Expansion Reduction (SER)

Require: Node v, labeled target samples S_v^T
Ensure: Node v
procedure SER(v,S_v^T)
if $d(v) == 0$ **then**
  $v \leftarrow BuildTree(S_v^T)$
  return v
end if
for $v_i \in \{v_1, v_2, ...v_n\}$ **do**
  SER($v_i, S_{v_i}^T$)
end for
if $leafError(v, S_v^T) < subtreeError(v, S_v^T)$ **then**
  for $i \in d(v)$ **do**
    removeNode(v_i)
  end for
  $d(v) \leftarrow 0$
  $y(v) \leftarrow argmax|\{(\cdot, y) \in S_v^T\}|$
end if
  return v
end procedure

Algorithm 6 takes as input the set of nodes v from the original source forest tree and the labeled target samples. The transformation of decision trees in the forest involves an internal non-leaf tree node v that has an out degree $d(v)$ and a set of children nodes $v1, v2, ...v_{d(v)}$. If the out degree is zero, then it is a leaf node. The leaf nodes in v are expanded as a tree and the internal nodes undergo reduction. Expansion is necessary since every leaf is associated with a decision value and expanding specializes the rules on the source data to the target data. Reduction of the internal nodes is the removal of nodes that are associated with a feature, which generalizes the rule to the target domain. There are two additional errors that can occur: a subtree error is empirical error when the node v of a subtree is actually the root node, and a leaf error exists on the node v when it is being reduced to a leaf. As long as the leaf error is less than the subtree error can the subtree become a leaf node. This approach is called inductive transfer learning, as the inductive biases or assumptions about the distribution of the source data is used to transform the model to predict the labels on the unseen target data. By transforming the original tree, there exists a rule in the source tree that satisfies a rule in the transformed tree. Therefore, the transformed tree will not be much different that the original tree.

Previous research [26, 91] had identified that deep learning models to be well situated for transfer learning approaches because the aim is to learn multiple "abstract" representations based on the training criterion. These levels of abstraction are separated by the different features present in the data. Deep learning assumes that specific representations in the input data is useful for predicting class labels and understanding the input distribution. Each layer in the model consists of features that are transferred to a different domain. One approach to transfer learning [92] found that

using transferred features from a new domain to initialize an existing, trained network can increase the generalization performance to predict labels on the new target data. Our second approach to transfer learning utilizes the concept of domain adaptation by fixing or "freezing" the weights on certain layers on the network that were trained on the original source data and the unfixed layers are trained on the new data. Algorithm 7 describes the training of specific layers in the original model.

The training using Algorithm 7 is performed for the top performing ANN model, which consists of 7 layers - an input layer, four hidden layers using the Rectified Linear Unit (ReLU) activation function, one hidden layer with batch normalization and an output layer with Softmax activation. The inputs are a trained deep learning model M_S, labeled PW samples S_v^T, and the number of layers desired to fix in the trained model f. f ranges from 0 to 5 as we were interested in freezing up to 5 hidden layers. By setting each layer as not trainable, the weights are not updated when the models are retrained on the PW data. When $f = 0$, all of the weights for the layers are adapted to the new target domain, while when $f = 5$, none of the layers are adapted except the output layer and the hidden layers function. The model is then recompiled using the same Adam optimizer and loss function and fit on the PW data S_v^T.

Algorithm 2 Selective Layer Training(SLT)

Require: Trained Model M_S, labeled target samples S_v^T, number of layers to fix f
Ensure: Model M_T
procedure SLT(M_S,S_v^T, f)
// Fix the first f layers in the existing model
for i $\in f$ **do**
  M_S.layer[i] $\leftarrow$ NOT trainable;
end for
Recompile M_S using Adam optimizer and categorical crossentropy loss function
$M_T \leftarrow fitModel(M_S, S_v^T)$
return M_T
end procedure

4.2.1 Classification Training Procedure

The goal of supervised classification methods is to automatically detect a class label based on decision rules previously learned through known input patterns in training data. Classification procedures typically involve these four phases [93]:

- Data pre-processing: Removal of noise from raw data in order to make interpreted features for machine learning algorithms
- Feature extraction: Creating and choosing the particular feature columns that are meaningful for characterizing each class.
- Decision Training: Teaching the classifier to predict class labels based on a training example set.

- Validation: Evaluation of the model on test data to understand its performance on unseen labels.

Since transfer learning is a continuation of the learning procedure for multiple domains, our approach involves three additional phases:

- Model Selection: Choose the model that has the highest classification accuracy with the source data set.
- Model Transfer Algorithm: Based on the type of model, determine the appropriate algorithm for transforming the original model and apply it with features extracted from the target data set.
- Validation: Evaluation of the model using test data from the target domain.

The goal of supervised training is to automatically detect a class label based on decision rules previously learned through known input patterns in training data. The model development is split into two main phases: Source Domain Learning (SDL) which creates a model trained on unsampled data M_S and a model on resampled data rM_S, and Target Domain Learning (TDL) which uses an already trained artificial neural network M and applies the SLT technique given a specific number of fixed layers f in order to produce a newly trained model M_T. Both of the initially trained models M_S and rM_S undergo TDL and are adapted to the new PW domain in the *developModels* procedure, therefore we return additionally M_T adapted using M_S model and rM_T adapted using rM_S model. These models are then validated on test data from the specified learned domain. The procedures *SDL* and *TDL* are explained in the following subsections.

Source Domain Learning

The raw data input from MW sensors D_S undergoes data preprocessing, feature extraction, normalizing and resampling before training. The *preprocess* method inputs D_S and involves identifying and segmenting irregular vibrations that occur from the initial speed up and slow down when starting or stopping the wheelchair movement during experimentation. Since we are interested in the variation of movement and capturing the high and low peaks in the accelerometer and gyroscope data, we avoid any smoothing procedure that may cause false surface predictions. Section 5.1.2 discusses data pre-processing in more detail. In the *featureExtract* method, we used a windows-based approach with a window size of two seconds or 100 data points for acceleration and gyroscope traces to compute 22 commonly used features denoted by F_S. Section 5.1.3 discusses feature extraction. Since four different phones were affixed to all subjects in the same assigned position for all experiments and data, the values greatly depend on the lower and upper bound of the frequency range. Not only do the phones have different sampling frequency, they also have different types of sensors. The data from the sensors of each phone are transformed individually using a StandardScaler for each phone type, which subtracts the mean value for each data column and divides by the standard deviation. This section is covered in Section 5.1.4. Then, similar surface features are grouped into categories shown in Table 5.7 formed using the method described in Section 5.1.5. Then, we use a ratio of 80:20 to split F_S into $trainData$ and $testData$. Since there

is an uneven amount of instances for each surface type, it is necessary to resample the feature data in the training set in the $resample$ method. We perform oversampling on F_S to generate synthetic known class instances for each group, then undersample the majority classes using Edited Nearest Neighbor (ENN) technique to produce $rTrainData$. More about this resampling technique is described in Section 5.1.6. We initialize a dictionary D_m and rD_m to store all the trained models and their accuracy results on the test data. We then train 8 different models on each dataset: Decision Tree (DT), Random Forest (RF), K-nearest neighbor (KNN), Naive Bayes (NB), Support Vector Machines (SVM), Artificial Neural Networks (ANN), Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM). Validation includes calculating the accuracy, precision, recall and F1-score of each model. M_S and its performance macro accuracy $SMAccuracy$ are stored in D_m and rM_S and $rSMAccuracy$ are stored in rD_m. After training, we choose the ANN model M_S and rM_S and return them to *developModels* procedure.

Target Domain Learning

Raw data from PW sensors D_T is processed in the same way as D_S. Data segmentation and feature extraction is used to produce F_T, which is then normalized using StandardScaler. The surface labels from the features F_T are grouped according to the data that was used to train ANN model M. We also split the PW data into $trainData$ and $testData$ using an 80:20 split. Then, we select the number of layers to fix in the original model if we decide to perform transfer learning using the *SLT* procedure. Otherwise, we apply $trainData$ and perform model transfer using the *SER* procedure. Then, we validate the retrained model to the $testData$ from the PW domain. The scores reported by $TMAccuracy$ in the TDL procedure are reported in Table 5.12.

Develop Models

The newly trained model is returned to *developModels* procedure, which consists of two models M_S and rM_S that have theoretical knowledge about the MW domain and can predict new previously unseen sensor data. Similarly, the other two models M_T and rM_T in the procedure have their knowledge-base built on PW data.

Algorithm 3 Transfer Learning Model Development

Require: Source data D_S, Target data D_T
Ensure: Model M_S, Model rM_S, Model M_T, Model rM_T
procedure developModels (D_S, D_T)
$M_S, rM_S \leftarrow SDL(D_S)$
$M_T \leftarrow TDL(D_T, M_S)$
$rM_T \leftarrow TDL(D_T, rM_S)$
return M_S, M_T, rM_S, rM_T
end procedure
procedure SDL (D_S)
$D_S \leftarrow preprocess(D_S)$
// Extract features
$F_S \leftarrow featureExtract(D_S)$
// Normalize features
$F_S \leftarrow normalize(F_S)$
Group features F_S into surface categories
// Split the train and test values
$trainData, testData = splitTrainTest(F_S)$
//Resample known class labels
$rTrainData \leftarrow resample(trainData)$
// Training and Validation on test data
Initialize Dictionary D_m and rD_m
for $m_i \in \{m_1, m_2, ...m_n\}$ **do**
    $M_S \leftarrow trainModel(trainData, m_i)$
    $SMAccuracy \leftarrow validate(M_S, testData)$
    $D_m \leftarrow (M_S, SMAccuracy)$
    $rM_S \leftarrow trainModel(rTrainData, m_i)$
    $rSMAccuracy \leftarrow validate(rM_S, testData)$
    $rD_m \leftarrow (rM_S, rSMAccuracy)$
end for
Select ANN model M_S from D_m and rM_S from rD_m
return M_S, rM_S
end procedure
procedure TDL (D_T, M)
$D_T \leftarrow preprocess(D_T)$
$F_T \leftarrow featureExtract(D_T)$
$F_T \leftarrow normalize(F_T)$
Group features F_T into same category as M
$trainData, testData = splitTrainTest(F_T)$
Select f between 0 to 5
$M_T \leftarrow SLT(trainData, M, f)$
$TMAccuracy \leftarrow validate(M_T, testData)$
return M_T
end procedure

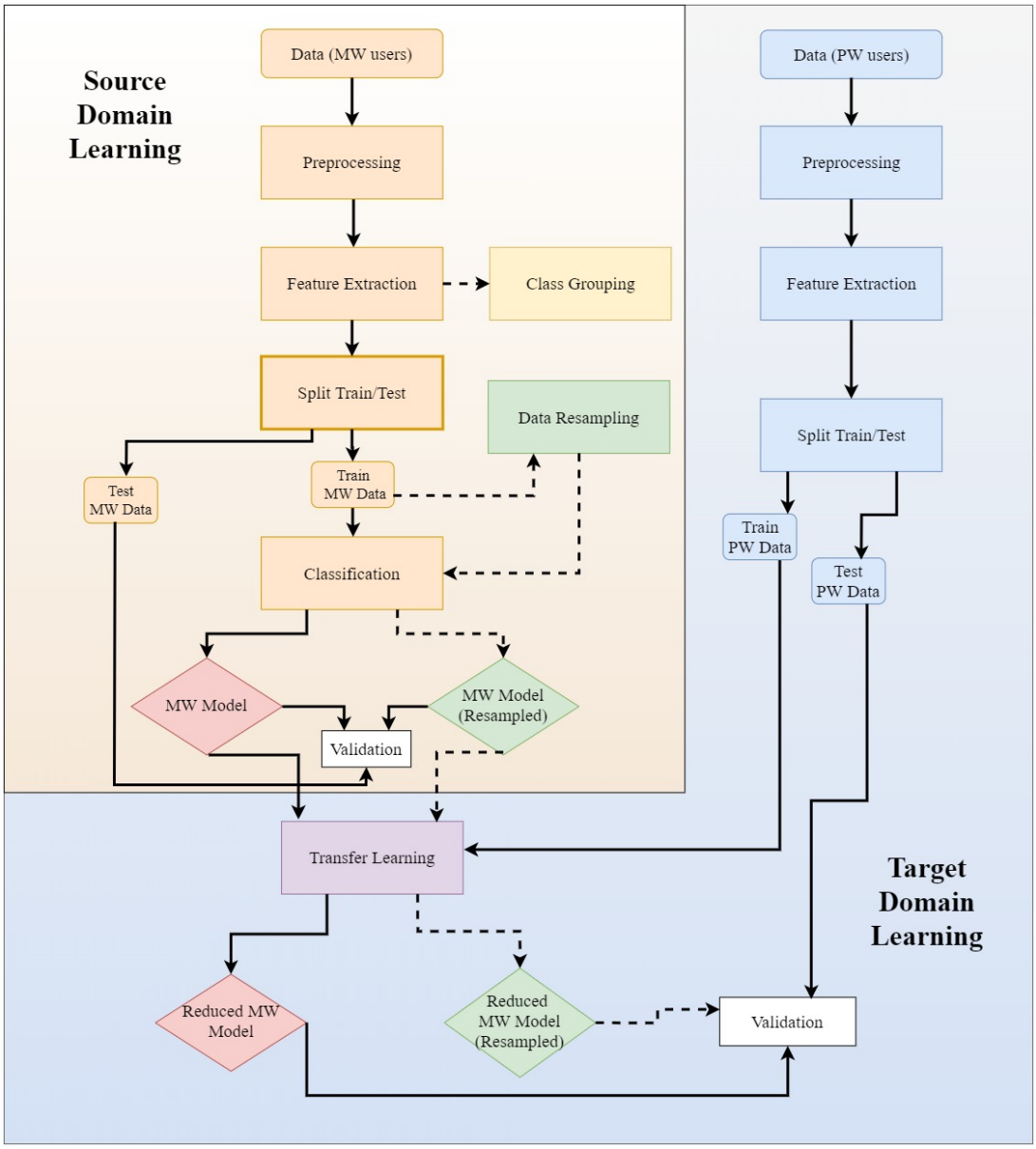

Figure 4.6: Diagram of Source Domain Learning and Target Domain Learning for Develop Models Procedure

Chapter 5

Implementation and Results

In order to analyze the accessibility of a route, the system must acquire substantial knowledge about the important features to generate the map overlay. Our hypothesis is that high classication accuracy using machine learning based algorithms can be attained if substantial data about accessible surface and curb features can be gathered from inertial sensors from smartphones attached to the user and wheelchair. Automatic feature detection using supervised machine learning techniques is the initial step to understanding the complex terrain for our accessible navigation system. The experimental setup for training the classication models are described in Section 5.1. This section includes data collection, pre-processing and normalization, feature extraction and data resampling strategies. The results from classication training include the accuracy, precision, recall and F1-Score and are explained in Section 5.2. The generation of a confusion matrix of the model with the highest testing accuracy for MW data and power wheelchair data are shown in this section. A confusion matrix provides a basis for predicting performance for input samples tested as true positives and false negatives. This section also includes performance metrics of transfer learning methods in order to analyze whether learning performance and training can be accelerated to recognize surfaces using vibration data collected from power wheelchair users. The machine learning classication approach predicts 15 surfaces from our built environment in United States and China using accelerometer and gyroscope sensors. Using this implementation, we achieved accuracy for surface prediction up to 94.46% in the manual wheelchair (MW) domain and up to 89.19% for classiers trained on the power wheelchair (PW) domain. Inter-model and inter-category analysis of the performance metrics is explained in the Discussion section 5.3. This section also explains limitations of the experiment and some future alterations and considerations.

5.1 Experimental Design

This section explains the procedure for training a model to recognize a surface in a quantitative manner. Features are extracted from time-dependent data gathered from accelerometer and gyroscope sensors attached to different types of wheelchairs. Different machine learning algorithms are trained on specific input data from manual wheelchair experiments and the known surface type in order to generate predictions to new input data. The procedure for data collection experiments for users operating manual and power wheelchair devices is also explained in this section. Resampling using SMOTE oversampling and Edited Nearest Neighbor undersampling, as well as grouping of similar surfaces, are also examined in order to analyze whether it has significant improvement in training and predicting the surface type.

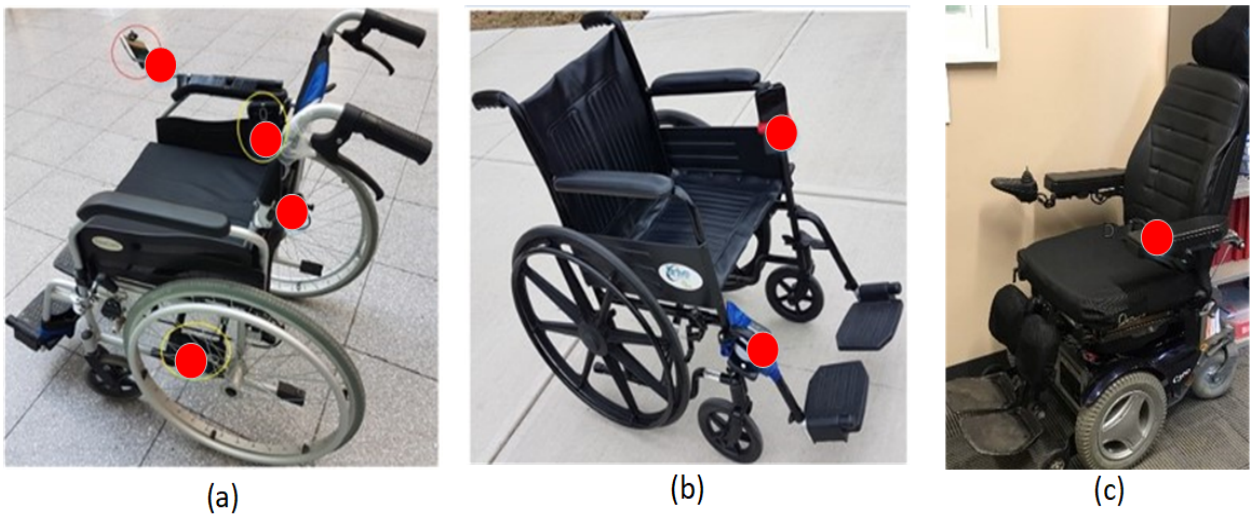

Figure 5.1: Manual Wheelchairs used in (a) USA, (b) China. Power Wheelchair used in (c) USA.

5.1.1 Surface Data Collection Experiments

Smartphones have the ability to capture accelerometer and gyroscope data from inertial sensors and are almost always present on a user. We interviewed an airport wheelchair assistance personnel to find out where a wheelchair user typically keeps a smartphone. Typically, those using electric or manual wheelchairs have an attachment to the chair that holds a phone in place or have the phone in their pocket or on their lap. There are pouches on the rear end of some wheelchairs, but are normally only used to carry tools or a bag. Therefore, it was desirable to choose to attach one phone to the side of the wheelchair and to ask the participants to place the other phone on top of their lap. We would like to mention that we do have the requisite IRB approval for the use of human subjects for our research.

Data collection using the manual wheelchair occurred on 14 days between December 2018 to January 2020. 22 participants in total operated a manual wheelchair in the campus of Miami University and in the town of Oxford, OH, USA and 6 participants near Shanghai in China. Over 400 experiments were performed on surface data collection. For PW experiments, data collection occurred also in the Miami University campus and in Oxford, OH for a total of 7 participants. The data was collected from four different smartphones of ranging sampling frequencies: iPhone X, Samsung Galaxy S9+, Samsung Galaxy S7 edge, and Samsung Galaxy J7. The S7 phone and S9 were alternated in the same spot, but always attached to the right side of the wheelchair from the sitting position, while the J7 and iPhone X remained on the lap of the participant during the experiment, attached to their right leg. The location of the phones and the three devices used for data collection are shown in Figure 5.1. The placement of phones at various positions allows for a better understanding of how sensor data differs according to their placement on the wheelchair. The recordings may also be affected by the proximity of the phone to the ground and the direction it faces.

In order to interact with the gyroscope and accelerometer sensors in the phones, we used two

different applications to record, label and export data for each experiment. The three Samsung smart phones used the application "Sensor Log", which allows for a label to be associated with one or multiple sensors. There is a round button on the home page and when pressed, the associated sensors begin recording data until that button is pressed again. Data from the sensors is stored in the SQLite local database on the phone and can be exported in EML format to the cloud. The iPhone X used the application "Sensor Play", which also has the option for using multiple sensors to collect data. "Sensor Play" allowed for sampling rate, or the average number of samples acquired in 1 second, to span from 1 Hz to 100 Hz and is exclusive to iPhones. The "Sensor Log" application allows for up to 500 Hz depending on the phone type. The S7 and the S9 have a sampling rate of approximately 500 Hz and the iPhone X and J7 have a sampling frequency of 100 Hz. At least one phone is always present on the wheelchair or user during every experiment.

An accelerometer would measure the change in velocity at one axis, which includes the force due to gravity. The Samsung and iOS devices both have a three-axis accelerometer represented by x, y and z axes. For the Android device, when the phone is not accelerating and is laying flat on a table, the accelerometer shows a magnitude of g = 9.81 m/s^2. Moving the device up will increase the z value by the forces due to gravity plus the acceleration of the device. iOS devices have accelerometers that measure the increments of acceleration due to gravity, so the value 1 represents 9.81 m/s^2. The application would receive information through listening to the accelerometer sensor, which includes the forces of acceleration due to gravity. The accelerometer sensor is useful in determining where the phone is relative to the gravity of the earth.

Gyroscopes measure the rate of rotation around the spatial axis represented by x, y and z in radians per second. Depending on the direction of rotation, the rotation values may be positive or negative for each value. The application would listen to the gyroscope sensors and then multiply the time by the rotational speed reported to determine the amount of rotation.

Primary Device Selection

Due to easy usability, increased availability and popularity of manual wheelchairs, this type of device was selected initially for performing experiments. There are essentially three different types of manual wheelchairs that could be chosen [79]. A standard wheelchair (weighing 40-65 lbs) is for short-term or temporary use and has limited customization or adjustment. Lightweight wheelchairs (28-36 lbs) are more adjustable and can be transported easily but are not meant for everyday use. Ultra lightweight wheelchairs (20-30 lbs) offers frame adjust-ability and is made of aluminum or titanium and is quite durable. The chosen primary device is a standard wheelchair and has the specifications shown in Table 5.1.

Secondary Device Selection

The decision to choose a power wheelchair as the secondary device resulted from its similar commonality as the manual wheelchair and similar popularity in the United States among middle-aged wheelchair users [2]. A power wheelchair also falls under the category of an power wheelchair, which is different from traditional wheelchairs. This is because rather than the user manually propelling the wheels with his/her arms, an power wheelchair is equipped with a motor and

Table 5.1: Selected Standard Wheelchair Device Specifications

Type of Device	Manual Wheelchair
Weight	40 lbs
Maximum Weight Capacity	350 lbs
Product Dimensions	40.75 x 25.50 x 37.20 Inches
Main Wheel Type	Solid Rubber

Table 5.2: Selected Target Device Specifications for USA Data Collection

Type of Device	Power Wheelchair
Name	Permobil C300
Weight	260 lbs
Maximum Weight Capacity	265 lbs
Product Dimensions	42.5 x 24.25 x 42.5 Inches
Main Wheel Type	Air pressurized
Range	16 miles
Max Speed Forward	5 mph
Minimum Turning Radius	25 Inches
Slope Capability	6 degrees

batteries usually underneath the seat. Current power wheelchairs allow for custom speed setting at the cost of more battery consumption. Scooters may not be able to detect various urban features from sensors alone, since the device is designed for outdoor terrain only and consists of a number suspension components. Power wheelchairs allow for maneuverability both indoors and outdoors and allows for a more diverse surface data needed for classifier training. The power wheelchair details are shown in Table 5.2.

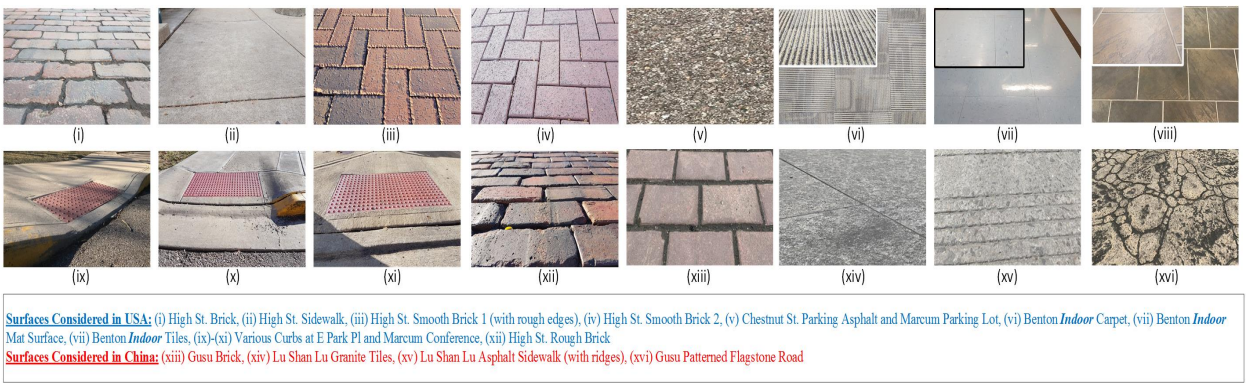

Figure 5.2: Various Surface Types Collected in Oxford, Ohio and China

Performing Data Collection

The sensor data collected during each experiment is continuous and consists of the name of the sensor and the X, Y and Z directions that are collected at a specific timestamp. The following information is obtained for each experiment:

1. Date of the Experiment
2. Name of the Participant
3. Type of Wheelchair
4. Experiment Number
5. Surface Type of the Experiment

There may be up to 100 different X, Y and Z directions in a single timestamp measured in milliseconds. Each participant is required to continuously move the wheelchair on a specific surface for three minutes long while two sensors are continuously collecting the vibration information at each timestamp. The only exception to the three minute duration is on curbs, where the time taken is from the point where the participant starts to move forward to the time where the participant stops at the bottom or at the top of the curb height completely. Figure 5.2 shows the different types of surfaces that were collected around the different parts of Oxford, OH as well as locations around China.

The majority of surface data collection experiments occurred in the United States using the manual wheelchair device shown in Figure 5.1 (a). Data collection occurred only for one day in China. The exact amount of experiments that occurred for each surface is shown in Table A.1. The various curbs in Figure 5.2 (ix-xi) are considered in terms of user position and direction of travel. So, going up the curb or "Up Curb" produces different vibration than while going down the curb or "Down curb". In total, there are 13 different surfaces (i) to (viii) and (xii) to (xvi) and two curb features, and they comprise 15 different surface groups (A to O) as listed in Table 5.7. Participants contributed once to each surface experiment, except curbs at which up to five different experiments were performed with the same participant.

Experiments using the PW device are also performed on surfaces around Oxford, OH and the total number of experiments is in Table A.2. All PW data collection experiments used the Samsung S7 and J7 sensors. Since the majority of this research focuses on classifying surfaces on data collected from manual wheelchair, the total data from PW experiments is significantly smaller and some surfaces were not able to be included due to the impact of COVID-19 on experimental days. The total count for instances that contain features for MW and PW experiments is shown in Table 5.3. This table uses the surface categories that are further described in Section 5.1.5 and are referenced in Table 5.7. The desired number of PW features is 25% of MW features as approximately having 1000 instances for each surface type provides enough target data if transfer learning is to be performed on the more data-driven deep learning models. Overall, the number of desired PW data has been attained, but some surfaces were not collected due to either too little number of instances collected in the manual wheelchair or lack of access to that surface. Therefore, we only consider 9 different categories of indoor and outdoor surfaces in the analysis of our PW classification performance.

	Surface Categories														
Data	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
MW	1666	4524	2085	232	4976	3672	3221	3366	184	185	1383	554	569	554	456
Desired PW (25% of MW)	417	1131	522	58	1244	918	806	842	46	47	346	0	0	0	0
PW	0	1096	853	0	1105	1293	1274	1031	59	56	1098	0	0	0	0

Table 5.3: Number of surface category instances in MW and PW datasets.

Decisions Made During Experimentation

The biggest challenge when conducting the experiment is to have control over the degrees of freedom, so that the variance between and within classes is not biased. Researchers have to make sure that the participants operate the device at a relatively constant speed that captures the vibrations of the wheelchair at the same point in time. It is difficult to keep the environmental conditions the same for all experiments, especially if the same surfaces are collected by users on different days. In this section, we discuss some issues with data collection and how we addressed them.

Participant Awareness of Study: Users are aware of the presence of sensors on the wheelchair and the goal of the research. However, we did not share with the participants the specific features that are being collected. In this way, adapting to the task was mitigated by the participant's interaction with various surfaces. No surface was repeated by a participant except the movement of up and down on a curb. This was done because moving over a curb often results in radical movement and thus capturing more information can reveal how a user is moving over this curb.

Movement Adaptation Over Time: Since the movement and speed of the wheelchair may vary based on the experience of the participant using the wheelchair, all of the participants in the experiment were asked whether they had operated a wheelchair before. Those that had no previous experience were given several minutes to practice operating it and shown to push the wheelchair at approximately 2 mph. The data may also change over time given that as users became more

comfortable with the experiment, they tended to go faster or exhibit a variation of movement. However, the experiments were very short and so we only received a snapshot of the user movements. The conductor of the experiment typically walked alongside the participant and ensured that the speed was relatively constant. Due to budgetary and time constraints, we were not able to find participants who use a wheelchair on a regular basis. Understanding how to operate and become accustomed to the wheelchair is important to the study, since it takes time to understand how to move using an assistive device efficiently. Wheelchair users normally start learning how to operate a wheelchair indoors before using it outdoors. For this reason, we offered users a choice to practice using the device before the experiment began for both indoor and outdoor environments.

Physical Ability: All 22 participants were asked on their physical ability and only two had temporary health conditions, which may have restricted their ability to push the wheelchair at a consistent effort. Given that the users were expected to propel the wheelchair at the same speed for several minutes, there was also expected to be fatigue that may also influence the data. During the data pre-processing stage, each data set is viewed manually and irregularities were segmented from the dataset. Therefore, if there is an unusual spike or trail in the data from irregular movement of the phone, this is eliminated prior to training.

5.1.2 Data Pre-Processing

Essentially, experiments from all of the phones undergo cleaning by identifying irregular vibrations or a change in speed that occurs from the initial speed up and slow down when participants begin and end the experiment. We found that segmentation of data at specific timestamps, where the user travels at a relatively constant speed along the surface allows for the highest preservation of vibration data. Some videos were also taken and used to cross-examine the timestamps with those occurring in the dataset in order to understand where data segmenting and cleaning may be performed. An example for identifying and cleaning the vibrations can be seen in Figure 5.3. Only the data that is not shaded is used for feature extraction and classification. The initial and ending irregular vibrations are caused by the phone being shifted in the starting and stopping of the experiment.

We predict that an uneven surface may produce more vibration in the wheelchair than a smooth surface. We are interested in the variation of movement and capturing the high and low peaks in the accelerometer and gyroscope data. Therefore, any form of smoothing, which is relatively common for removing noise from sensors [94], is avoided so that the highest information is available for surface recognition. Similar surfaces would have homogeneous sensor data and their features may be extracted for classification and grouped together. Figures B.1 and B.2 show examples of the X, Y and Z surface vibration for each experiment involving manual wheelchair users that is collected over a short period of time. Note that the X-value tends to be relatively linear except in experiments involving curbs. We are examining 16 different surfaces, three of which are similar curbs, but we consider curb data in terms of direction of movement at the time. Therefore, we view going up a curb as a different surface than going down the curb. The features from the three curbs receive the same label since the curbs are constructed very similarly as seen in Figure 5.2, but depending on whether the participant is moving up or down the curb, we consider it as a different surface category. Therefore, 15 different surface categories (comprised of 13 surfaces and 3 curbs)

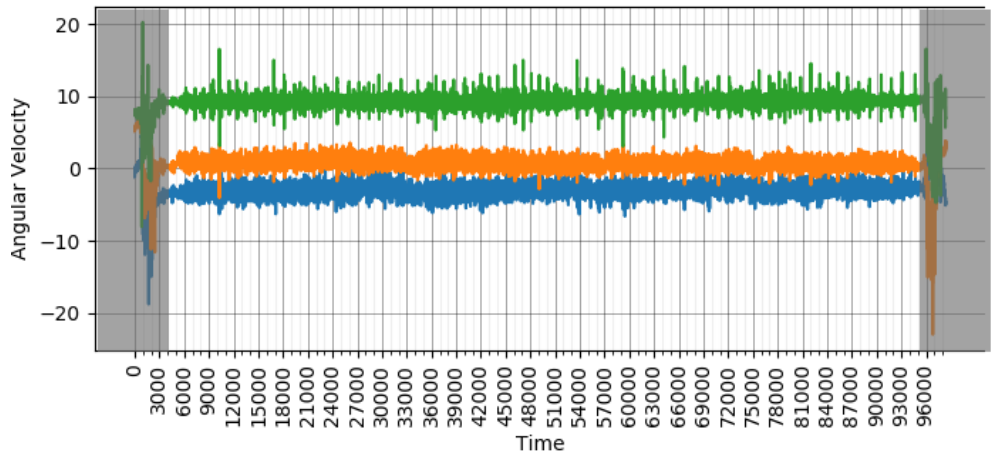

Figure 5.3: Example of segmenting gyroscope sidewalk data from irregular vibrations acquired during data experimentation.

are collected across USA and China with 3 different manual and power wheelchairs. Similarly, Figures B.3 and B.4 are examples of sensor vibrations for the 9 surface categories involving PW data collection. PW experiments involve only one curb for data, as opposed to three for manual wheelchair data collection.

5.1.3 Feature Extraction

In order to perform classification, meaningful attributes are extracted from the data in order to characterize each class. Selecting the features greatly affects the classification performance of the model and identifying accessible surfaces. A windows-based approach is used for feature extraction, which uses a window size of two seconds or 100 data points for acceleration and gyroscope traces. Given the differences in sampling rate, the window size varies for each data set obtained from each phone. For curb data experiments, given the short duration of the experiment, the whole size of the frame is used for feature extraction. During each window size, 22 commonly used features are computed and shown in Table 5.4. F1 has all computed features and is expected to give the most accurate prediction after machine learning algorithms are applied. The data collection experiment had smartphones placed in the same position on the user and wheelchair, so the mean value for x, y and z are used as features for F1. However, we argue that it is possible for contributors to not place the smartphones in the same orientation as the phones used in baseline collection. There is no guarantee that in live production sensors give consistent orientation data and are physically placed in the same location used during training. Therefore, F2 does not include the features that depend on a single direction. The intention for including F2 is to see if including only multi-directional features significantly reduces the accuracy of the predictions.

	Feature	F1	F2
1	mean(acc x)	•	
2	mean(acc y)	•	
3	mean(acc z)	•	
4	var(acc x)	•	
5	var(acc y)	•	
6	var(acc z)	•	
7	mean(sum(acc))	•	•
8	mean(sum(abs(acc)))	•	•
9	var(sum(acc))	•	•
10	var(sum(abs(acc)))	•	•
11	max(sum(abs(acc)))	•	•
12	mean(gyr x)	•	
13	mean(gyr y)	•	
14	mean(gyr z)	•	
15	var(gyr x)	•	
16	var(gyr y)	•	
17	var(gyr z)	•	
18	mean(sum(gyr))	•	•
19	mean(sum(abs(gyr)))	•	•
20	var(sum(gyr))	•	•
21	var(sum(abs(gyr)))	•	•
22	max(sum(abs(gyr)))	•	•
$\sum$	**Number of features**	**22**	**10**

Table 5.4: Features extracted from acceleration and gyroscope data. Different sets of features have been selected for F2.

5.1.4 Normalization

The raw signal sensors from each of the phones are transmitted using a tri-axial accelerometer and gyroscope. Since four different phones were used for our data collection (not more than 2 at a time) [see Figure 5.1 for their usual positions] the values greatly depend on the lower and upper bound of the frequency range. Not only do the phones have different sampling frequency, they also have different types of sensors.

One of the main requirements of the system is to perform robust classification, regardless of the hardware specifications and sampling rate. The variation of data which may introduce noise specific to the phone. Sensors vary in quality and correctness, since the common mobile devices are strongly affected by errors which affect the accuracy of the readings [95]. Therefore these errors are most commonly addressed by scaling the norm of the individual sensors [93]. The data from the sensors are transformed using a StandardScaler for each phone type, which subtracts the mean value for each data column and divides by the standard deviation. Therefore the distribution of all sensor data columns will have a mean value of 0 and standard deviation of 1.

5.1.5 Surface Label Grouping

We performed multiple different combinations for the 15 surfaces based on several key property questions. The assumption is that small differences in the vibration patterns between two or more classes would not affect their accessibility very much. Also, this would lessen the complexity of multi-class classification and may lead to more accurate identification provided that similar surfaces exhibit similar characteristics. Models that were trained on combinations of surface classes may also become more useful in routing applications, since there is a large number of independent surfaces in the built environment.

The property questions that were asked can be seen in Table 5.5. Based on these observations, we can make a prediction on the grouping of similar surface classes. However, we limit the prediction to only group classes from United States provided that there exists data from the same phone for both classes. Similar responses show a possible grouping of High St. Brick Table and High St. Smooth Brick 2, High St. Sidewalk and High St. Smooth Brick 1, the three indoor surfaces, and the direction of the curbs.

In order to test our assumption that there is no significant difference in the means of these surface groupings, we performed a Welch T-Test for unequal variances for each small iteration of time for surface data from the Samsung S7 phone, which collected the most data. The results are shown in Table 5.6. Since the data is a time series, the test is performed on approximately 2 seconds of data for all surface pairings except for curbs, for a period of 70 seconds and then the average p-value, T-statistic and degree of freedom is computed. For curb data, the whole time interval is used, so only one T-Test is computed, due to the relatively small data size. The average p-value, T-statistic and degree of freedom of the intervals is reported at a 95% confidence interval. The Welch t-test works for unequal variances in data, but the assumption of normality remains, since the data comes from the same phone. The t-statistic is reported below.

Surfaces	Properties				
	Visible Differences for All Values in Acceleration?	**Y Mostly Greater than Z in Acceleration?**	**Regular Sharp Y rotations?**	**Even Surface?**	**Regular Smooth Gaps?**
High St. Brick	No	Yes	No	Yes	Yes
High St. Sidewalk	Yes	No	No	Yes	Yes
High St. Smooth Brick 1	Yes	No	No	Yes	Yes
High St. Smooth Brick 2	No	Yes	No	Yes	Yes
Parking Lot Asphalt	No	No	No	Yes	None
Benton Indoor Carpet	No	Yes	Yes	Yes	None
Benton Indoor Mat Surface	No	Yes	Yes	Yes	None
Benton Indoor Tiles	Yes	Yes	Yes	Yes	Yes
High St. Rough Brick	No	No	No	No	None
E. Park Plaza Curb (Going Up)	Yes	No	No	No	None
E. Park Plaza Curb (Going Down)	Yes	No	No	No	No
Brick Surface in Gusu	No	Yes	No	Yes	Yes
Granite Tile	No	Yes	No	Yes	Yes
Asphalt Sidewalk (w/ ridges)	No	Yes	No	Yes	Yes
Patterned Flagstone Road	No	Yes	No	No	No

Table 5.5: The properties assigned for the organization of class groups.

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \tag{5.1}$$

where X_i, s_i and N_i is the sample mean, standard deviation and sample size of the class i. The null hypothesis is that the population mean and standard deviation of one group is equal to the second group. The p-value is the probability that the result obtained is the one that is observed if the null hypothesis is not rejected. If the calculated p-value is less than 0.05, then we can reject the null hypothesis that there is no statistical difference in the means for each group. From the results, the average P-value is greater than 0.05 for all groups, therefore there is not enough evidence to reject the null hypothesis for the two tailed t-test for each group pairing. Therefore, the groups that were created are not statistically different from each other and thus are viable for surface classification pairing.

Table 5.7 shows the organization of the classes into the different groups. S5 includes all of the class groups formed in S2, S3 and S4. We refer to a surface category as containing one or more surface types that are collected in the experiment. Simplifying the number of classes the model has to identify may increase the classification performance. The property questions that were asked describe accessibility characteristics about the surface and thus allow for a better understanding about the structures of multiple classes and their similarities.

Group Placement Considered	Surfaces		Sensor	Average P-Value	Average Degree of Freedom	Average T-statistic
S2-B	High St. Smooth Brick 1	Sidewalk Surface	Accelerometer	0.9146	3.9104	-0.0479
	High St. Smooth Brick 1	Sidewalk Surface	Gyroscope	0.5570	2.8135	-0.3223
S2-A	High St. Smooth Brick 2	High St. Brick Surface	Accelerometer	0.7373	3.4787	-0.1312
	High St. Smooth Brick 2	High St. Brick Surface	Gyroscope	0.5054	3.0132	0.0860
S4-F	Benton Indoor Carpet	Benton Indoor Mat	Accelerometer	0.9559	3.9866	-0.0240
	Benton Indoor Carpet	Benton Indoor Mat	Gyroscope	0.4966	2.9103	0.0481
	Benton Indoor Carpet	Benton Indoor Tiles	Accelerometer	0.8819	3.9415	-0.1586
	Benton Indoor Carpet	Benton Indoor Tiles	Gyroscope	0.4910	2.7509	0.1339
	Benton Indoor Mat	Benton Indoor Tiles	Accelerometer	0.8979	3.9387	-0.1337
	Benton Indoor Mat	Benton Indoor Tiles	Gyroscope	0.4564	2.9055	0.0065
S3-I	Down Curb 2	Up Curb 2	Accelerometer	0.8082	3.7198	-0.2606
	Down Curb 2	Up Curb 2	Gyroscope	0.4332	3.2303	-0.8933
	Down Curb 3	Up Curb 3	Accelerometer	0.6410	3.6745	0.5071
	Down Curb 3	Up Curb 3	Gyroscope	0.6624	2.6245	0.4896

Table 5.6: The Welch T-Test Performed on Intervals for Each Considered Class Group Pairing

5.1.6 Data Resampling

From the initial performance results with unbalanced feature data for each surface class S1 to S5, we sought to optimize the results in the ROC space through resampling strategies. Data balancing uses a combination of undersampling the majority case and then oversampling in order to balance class distribution. This resampling technique has been suggested in handling imbalanced machine learning data by the original authors of SMOTE resampling [96]. SMOTE minority class is originally chosen at random. The new synthetic sample is made from one of the K nearest neighbors at random and is formed in the feature space. As a result, there would be more distinct decision regions for minority classes and would increase classifier performance. Since there is an uneven amount of instances for each surface category, it is necessary to resample the feature data in the training set. Using the resampling approach conducted in [22] on sensor data, we have decided to perform a combination of SMOTE oversampling of the minority classes and then Edited Nearest Neighbor (ENN) undersampling of the majority class. The ENN method removes each instance of the synthetic sample for a feature class if it does not agree with $k = 3$ neighbors [97]. Using a combination of the two allows for the reduction in learn time for model training as compared to solely oversampling the data, which is why it is explored in surface detection. Identification of new surface classes will require a more balanced class distribution. Each group classification has its own ratio of feature classes that is in the minority (lacking representation) and majority (high representation over the distribution of all the classes). So in order to apply resampling, we perform

Table 5.7: The original 15 categories for surfaces (S1) merged into 13 categories (S2), 14 categories (S3), 13 categories (S4) and 10 categories (S5).

Surfaces		**S1**	**S2**	**S3**	**S4**	**S5**
1	High St. Brick	A	A	A	A	A
2	High St. Sidewalk	B	B	B	B	B
3	High St. Smooth Brick 1	C	B	C	C	B
4	High St. Smooth Brick 2	D	A	D	D	A
5	Parking Lot Asphalt	E	C	E	E	C
6	Benton Indoor Carpet	F	D	F	F	D
7	Benton Indoor Mat Surface	G	E	G	F	D
8	Benton Indoor Tiles	H	F	H	F	D
9	Up Curb	I	G	I	G	E
10	Down Curb	J	H	I	H	E
11	High St. Rough Brick	K	I	J	I	F
12	Gusu Brick	L	J	K	J	G
13	Granite Tile	M	K	L	K	H
14	Asphalt Ridged Sidewalk	N	L	M	L	I
15	Patterned Flagstone Road	O	M	N	M	J
Total Number of Categories		15	13	14	13	10

Note that we use S1 labeling procedure to refer to surfaces for other categories.

a grid search for calculating the K nearest neighbors for the minority classes in each class group for the SMOTE method. Then, we evaluate the performance by comparing the highest predicted class label with the ground truth and compute the accuracy, precision, recall, F1-score and ROC OVR metrics of the top two accurate models using a 3 time repeated stratified K-fold cross validation of 10 folds. The odd values of k are chosen between 1 to 9. Finding the k value is necessary to stabilize the stochastic nature of the SMOTE algorithm. From the initial experiments with the unsampled dataset, the random forest classifier had the highest accuracy for all group classes for F1 and F2 (as shown in Table 5.9). The decision tree classifier was second to random forest as it had the second highest accuracy for 3/5 cases in F1 and 4/5 cases in F2. Therefore, we train both classifiers using data with a particular k value for SMOTE oversampling and ENN undersampling.

The process for choosing K is shown in Figure 5.4. In order to determine K, the highest metrics among the values that were chosen is the macro F1-score, ROC OVR and accuracy. First, the max value among the K-values is chosen for the three metrics. If two out of the three metrics have the highest values for one K-value, then the performance differences is computed for all three

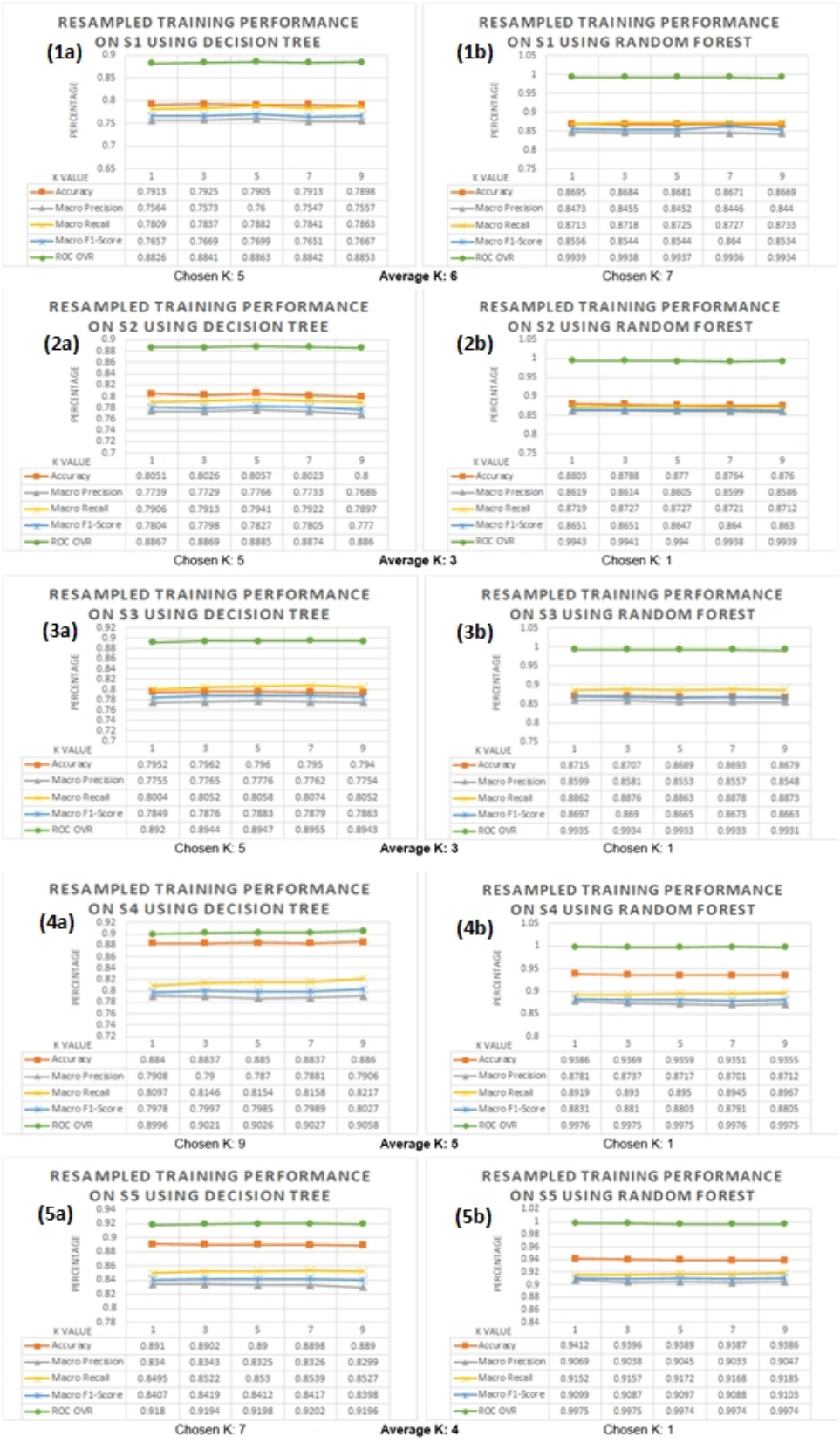

Figure 5.4: Determining K value for SMOTE resampling

Group	Resampled?	Surfaces														
		A	D	B	C	E	F	G	H	I	J	K	L	M	N	O
S1	No	1350	193	3628	1655	3952	2915	2607	2709	149	151	1092	444	468	437	351
	Yes	1060	63	2560	1054	2969	1337	1109	1566	149	30	807	159	392	151	62
S2	No	1543		5283		3952	2915	2607	2709	149	151	1092	444	468	437	351
	Yes	1123		3867		2969	1337	1109	1566	149	30	807	159	392	151	62
S3	No	1350	193	3628	1655	3952	2915	2607	2709	300		1092	444	468	437	351
	Yes	1060	193	2560	1054	2969	1337	1109	1566	290		807	159	392	151	62
S4	No	1350	193	3628	1655	3952	8231			149	151	1092	444	468	437	351
	Yes	1060	63	2560	1054	2969	7510			149	30	807	159	392	151	62
S5	No	1543		5283		3952	8231			300		1092	444	468	437	351
	Yes	1123		3867		2969	7510			300		807	159	392	151	62

Table 5.8: Number of Manual Wheelchair instances that are resampled for Categories S1 to S5.

metrics between the two K-values. If the sum of the differences for the two highest metrics for the first K-value is greater than the difference for the third metric for the second K-value, then the first K-value is chosen. This is also performed in the case when the metrics have max values at different k values. For example in the decision tree case for S5-F1, accuracy is the highest for the k-value of 1 at 0.891, F1-score is the highest for k=3 at 0.8419 and ROC OVR is the highest for k=7 at 0.9202. Subtracting the difference with accuracy from the other K-values gives 0.0008 difference to k=3 and 0.0012 to k=7. Performing the same proceudre with F1-score gives 0.0012 to k=1 and 0.0002 to k=7, and for ROC AVR gives 0.0022 to k=1 and 0.0008 to k=3. The sum for going to each k value is 0.0034 to k=1, 0.0016 to k=3 and 0.0014 to k=7, so we choose the k-value to be 7 since it produces the lowest performance loss. By looking at the differences, this eliminates bias from choosing a k-value based on a single performance metric. We do not include specifically precision or recall in the decision as the F1-score is the harmonic mean of class-wise arithmetic precision and recall. The ROC is the area under the curve of each class label against the rest and reports an aggregated performance of each of the classes. Considering the sensitivity and specificity of the models as well as the quality of the predictions is deemed at a higher precedence than just detecting accuracy, since given a large variability of the surface data, it is important for the model to consider these metrics to avoid classification errors. This technique may reduce the problems of over-fitting noise data by limiting the number of K neighbors observed to create new samples. Table 5.8 shows the number of manual wheelchair instances for each group. Resampling using Edited Nearest Neighbors and SMOTE oversampling creates artificial instances in order to balance the number of instances and improve accuracy of class predictions.

Resampling created artificial instances from the unsampled manual wheelchair dataset, thereby providing more data necessary for increasing performance of nonlinear algorithms and deep learning networks. An example comparison of the increase of distribution size for each class after resampling for S1 is shown in Figure 5.5.

5.2 Results

The application of machine learning models on manual wheelchair data involves the creation of eight classifiers: KNN, RF, DT, SVM, NB and the three deep learning models ANN, CNN and

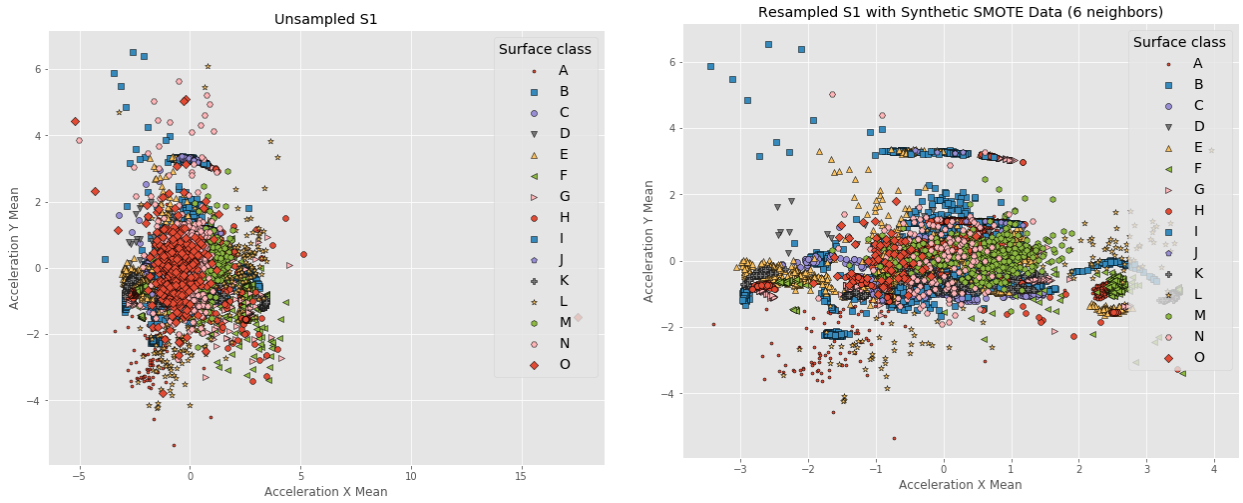

Figure 5.5: The creation of synthetic instances for each surface for S1.

LSTM. Training was performed on all 22 features in the case of F1, and the 10 selected features in F2. The merged surface classes (S2-S5) had also been used for training and evaluation. The split number for training and testing data is 80:20. Section 5.2.1 describes the evaluation for manual wheelchair data. Also, the results from the transfer learning of PW experiments from MW knowledge is described in Section 5.2.2. We utilize two different types of transfer learning approaches: model transfer and feature transfer. By observing the different models trained on unsampled and resampled data using the combined SMOTE and Edited Nearest Neighbors, we can identify whether transfer learning led to a statistically significant improvement in accuracy for model learning on the PW domain using statistical tests.

5.2.1 Evaluation Using Manual Wheelchair Data

Training for ANN went for 10 epochs, CNN went for 25 epochs and for 23 epochs for LSTM since the loss and accuracy values are consistently the same after this value as seen in Figure 5.6. The results from training on unsampled data for all of the models are shown in Table 5.9. Figure 5.7 shows the confusion matrix of the most accurate model for unsampled data, which is the RF with F1-S5 grouping, which has an accuracy of 94.46%. Training using resampled data for all of the models are shown in Table 5.10. Figure 5.8 shows the confusion matrix of the most accurate model for resampled data training, which is the RF with F1-F5 grouping at 91.21%.

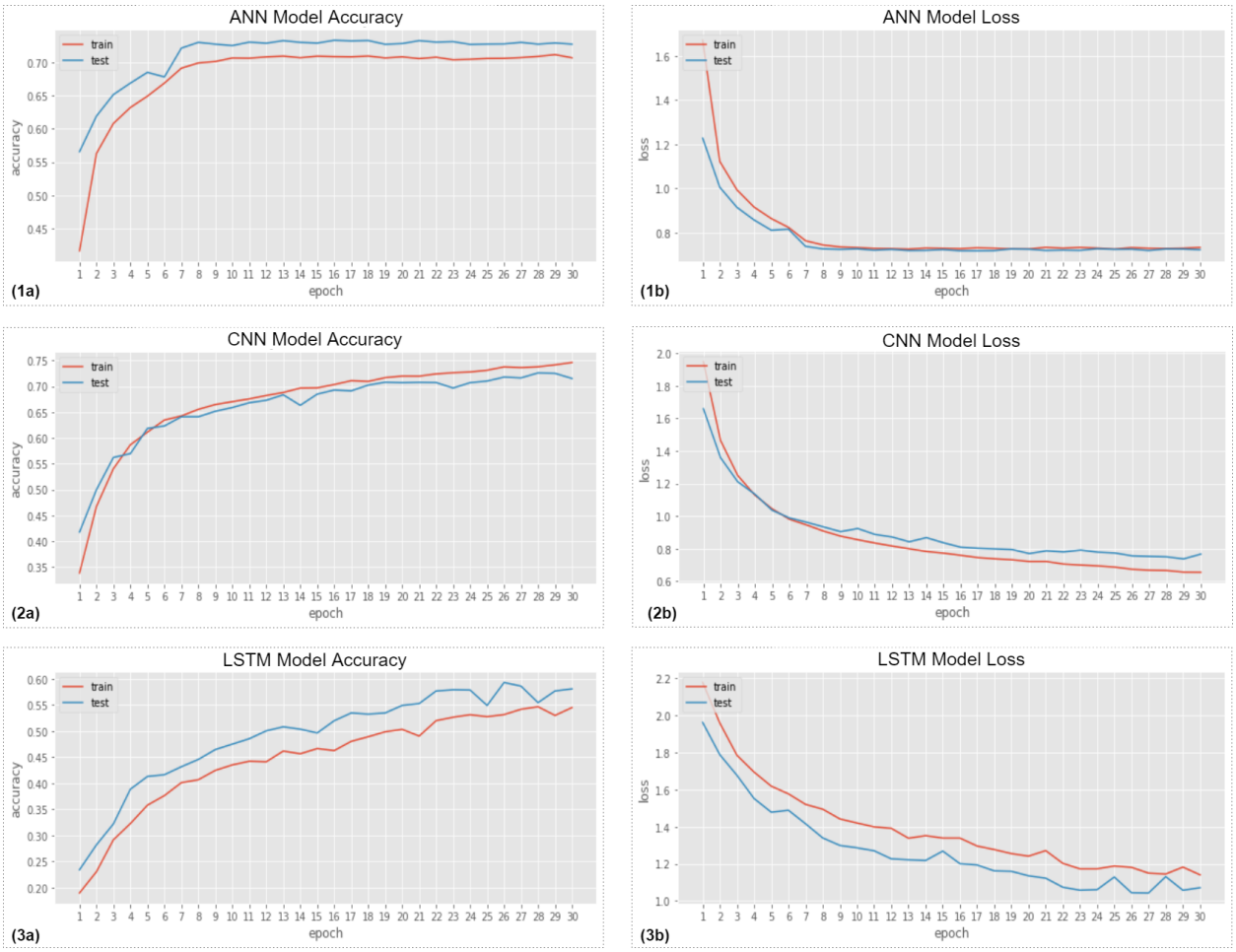

Figure 5.6: Train and test duration from 1 to 30 epochs for ANN accuracy (1a) and loss (1b), CNN accuracy (2a) and loss (2b) and LSTM accuracy (3a) and loss (3b)

Features		F1				F2			
Metrics		Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
KNN	S1	0.7952	0.7704	0.7531	0.7575	0.7119	0.6889	0.6831	0.6839
	S2	0.8017	0.7818	0.7652	0.7686	0.7215	0.7086	0.6969	0.7003
	S3	0.7984	0.7923	0.7748	0.779	0.7159	0.7121	0.7072	0.708
	S4	0.8867	0.7982	0.7807	0.7853	0.8087	0.7077	0.7022	0.6999
	S5	0.8954	0.8511	0.8339	0.8372	0.8201	0.7661	0.7582	0.7594
SVM	S1	0.6363	0.7393	0.6496	0.6705	0.4336	0.5385	0.4587	0.4651
	S2	0.6336	0.7388	0.6631	0.6812	0.4406	0.5647	0.4698	0.4752
	S3	0.6386	0.7498	0.6653	0.6809	0.4379	0.5491	0.4771	0.4811
	S4	0.7532	0.7921	0.6768	0.712	0.5936	0.6073	0.4941	0.5184
	S5	0.7653	0.8316	0.7378	0.7682	0.6028	0.6712	0.5663	0.5933
RF	S1	0.8959	0.8784	0.8697	0.873	0.8053	0.7605	0.7614	0.7604
	S2	0.8988	0.8814	0.8774	0.8786	0.8109	0.775	0.7725	0.7731
	S3	0.8965	0.8841	0.8729	0.8773	0.808	0.7826	0.7826	0.7821
	S4	0.9379	0.8881	0.8758	0.8805	0.8802	0.7829	0.7805	0.7808
	S5	**0.9446**	0.9127	0.9059	0.9083	0.8912	0.8335	0.8276	0.8296
NB	S1	0.2742	0.343	0.3523	0.2728	0.1334	0.2014	0.2206	0.1405
	S2	0.27	0.3636	0.3468	0.281	0.1336	0.2019	0.2296	0.1474
	S3	0.2772	0.3459	0.3548	0.2704	0.1366	0.2006	0.2301	0.1428
	S4	0.2571	0.3206	0.3753	0.2711	0.1314	0.2232	0.2536	0.1582
	S5	0.3382	0.3625	0.3843	0.299	0.1359	0.2717	0.2872	0.1767
DT	S1	0.8123	0.7697	0.7705	0.7688	0.7195	0.67	0.6714	0.6701
	S2	0.8104	0.7844	0.7826	0.7826	0.7182	0.6889	0.6831	0.6858
	S3	0.81	0.7769	0.7836	0.7793	0.7197	0.6759	0.6759	0.6755
	S4	0.8768	0.7627	0.7663	0.7638	0.8219	0.7143	0.7161	0.7152
	S5	0.8865	0.8216	0.812	0.8165	0.827	0.7616	0.763	0.7621
ANN	S1	0.7381	0.7392	0.7098	0.7193	0.6258	0.6384	0.6292	0.6283
	S2	0.762	0.7567	0.7506	0.7493	0.6256	0.6741	0.6622	0.6623
	S3	0.7389	0.756	0.7474	0.7459	0.5561	0.6039	0.5764	0.5824
	S4	0.8471	0.7895	0.771	0.7753	0.7287	0.679	0.6498	0.6502
	S5	0.8453	0.8277	0.8146	0.8198	0.7486	0.753	0.7404	0.7381
CNN	S1	0.6348	0.735	0.6772	0.6685	0.3453	0.635	0.4438	0.4578
	S2	0.6659	0.7633	0.7135	0.7112	0.3623	0.6556	0.4599	0.4773
	S3	0.6124	0.7788	0.6596	0.6771	0.3636	0.6649	0.4683	0.4832
	S4	0.7917	0.7806	0.7436	0.7443	0.7887	0.6602	0.5178	0.539
	S5	0.8243	0.8259	0.8097	0.8096	0.6638	0.7312	0.6415	0.647
LSTM	S1	0.6439	0.5607	0.5385	0.538	0.5425	0.5222	0.5194	0.5033
	S2	0.6587	0.6277	0.6283	0.6208	0.5362	0.5171	0.5022	0.487
	S3	0.6567	0.6358	0.6214	0.6227	0.5226	0.5392	0.5225	0.5197
	S4	0.7816	0.6999	0.6689	0.6792	0.6755	0.5873	0.5394	0.5332
	S5	0.7749	0.7262	0.6691	0.6856	0.6764	0.6418	0.6095	0.616

Table 5.9: Classification performance metrics on unsampled manual wheelchair data.

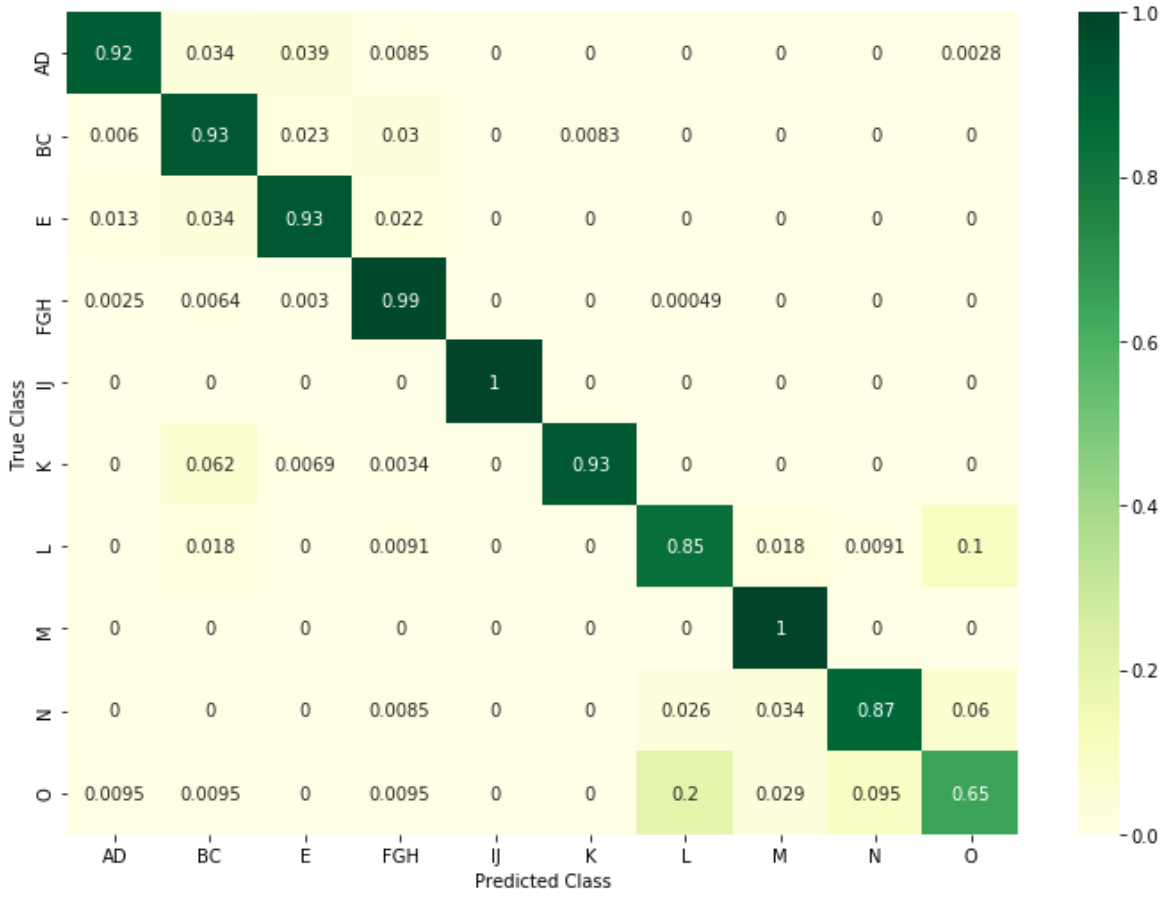

Figure 5.7: The confusion matrix for the top performing RF model on S5 unsampled data.

Features		F1				F2			
Metrics		**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
KNN	**S1**	0.7438	0.7094	0.6542	0.6534	0.667	0.6592	0.6098	0.6005
	S2	0.7535	0.7194	0.6682	0.6648	0.6763	0.6847	0.6217	0.6145
	S3	0.7503	0.7407	0.7189	0.713	0.6725	0.6677	0.6619	0.6528
	S4	0.8482	0.7463	0.6693	0.6762	0.7899	0.7108	0.6362	0.6344
	S5	0.8632	0.8196	0.7527	0.7649	0.8044	0.7593	0.7103	0.7224
SVM	**S1**	0.5907	0.6521	0.5457	0.5456	0.3938	0.4609	0.3824	0.358
	S2	0.5865	0.6766	0.5441	0.546	0.4017	0.4919	0.3778	0.3532
	S3	0.5959	0.6692	0.587	0.5938	0.4007	0.5358	0.4218	0.4155
	S4	0.7206	0.7225	0.5631	0.5926	0.5623	0.5521	0.4135	0.4264
	S5	0.7403	0.8069	0.6387	0.6832	0.5869	0.694	0.5001	0.533
RF	**S1**	0.8266	0.7991	0.7549	0.7507	0.7423	0.7011	0.6774	0.6697
	S2	0.8373	0.8196	0.7707	0.7656	0.7533	0.725	0.695	0.6873
	S3	0.8344	0.8131	0.8217	0.8078	0.7481	0.7248	0.7415	0.7235
	S4	0.8992	0.8224	0.7621	0.7622	0.8458	0.7363	0.6973	0.6952
	S5	**0.9121**	0.8727	0.8385	0.8468	0.8617	0.7972	0.7763	0.7791
NB	**S1**	0.2684	0.3894	0.3604	0.3249	0.1887	0.2589	0.2944	0.2158
	S2	0.2619	0.4007	0.3799	0.3342	0.2266	0.3343	0.3224	0.2551
	S3	0.2725	0.4121	0.3789	0.3435	0.1936	0.2678	0.299	0.2291
	S4	0.3104	0.3928	0.4529	0.3675	0.1793	0.3051	0.3595	0.2176
	S5	0.3571	0.4656	0.4946	0.424	0.2201	0.3937	0.4362	0.2904
DT	**S1**	0.7519	0.7175	0.6921	0.6925	0.6797	0.6378	0.6242	0.6219
	S2	0.7691	0.743	0.7076	0.7081	0.6958	0.6643	0.6475	0.638
	S3	0.7599	0.7259	0.7223	0.717	0.682	0.6584	0.6697	0.6542
	S4	0.8491	0.7129	0.6977	0.697	0.8091	0.6858	0.6597	0.6622
	S5	0.8692	0.7959	0.781	0.7817	0.8176	0.7316	0.723	0.7215
ANN	**S1**	0.7027	0.6483	0.6138	0.6048	0.5872	0.6136	0.5597	0.5553
	S2	0.7181	0.6947	0.6912	0.6393	0.6088	0.6335	0.5965	0.5755
	S3	0.7139	0.7089	0.693	0.6863	0.5518	0.6201	0.573	0.5761
	S4	0.8104	0.6954	0.628	0.6231	0.7329	0.6865	0.5972	0.5898
	S5	0.84	0.8084	0.7522	0.7631	0.7349	0.7258	0.6916	0.6963
CNN	**S1**	0.6086	0.6557	0.588	0.5696	0.355	0.6121	0.4383	0.4426
	S2	0.652	0.7106	0.642	0.6315	0.4035	0.6361	0.4595	0.4601
	S3	0.6516	0.7233	0.6841	0.6809	0.3764	0.6095	0.4659	0.4796
	S4	0.7975	0.7595	0.6791	0.6903	0.5737	0.58	0.4801	0.4793
	S5	0.8326	0.8244	0.7714	0.7819	0.6261	0.7183	0.578	0.6074
LSTM	**S1**	0.1871	0.1069	0.0676	0.0227	0.1858	0.0568	0.067	0.022
	S2	0.1111	0.0086	0.0769	0.0154	0.192	0.0454	0.081	0.0327
	S3	0.1853	0.0133	0.0714	0.0224	0.137	0.01	0.0714	0.0172
	S4	0.137	0.01	0.0714	0.0172	0.367	0.0282	0.0769	0.0413
	S5	0.367	0.0367	0.1	0.0537	0.367	0.0367	0.1	0.0537

Table 5.10: Classification performance metrics on resampled manual wheelchair data using Edited Nearest Neighbors.

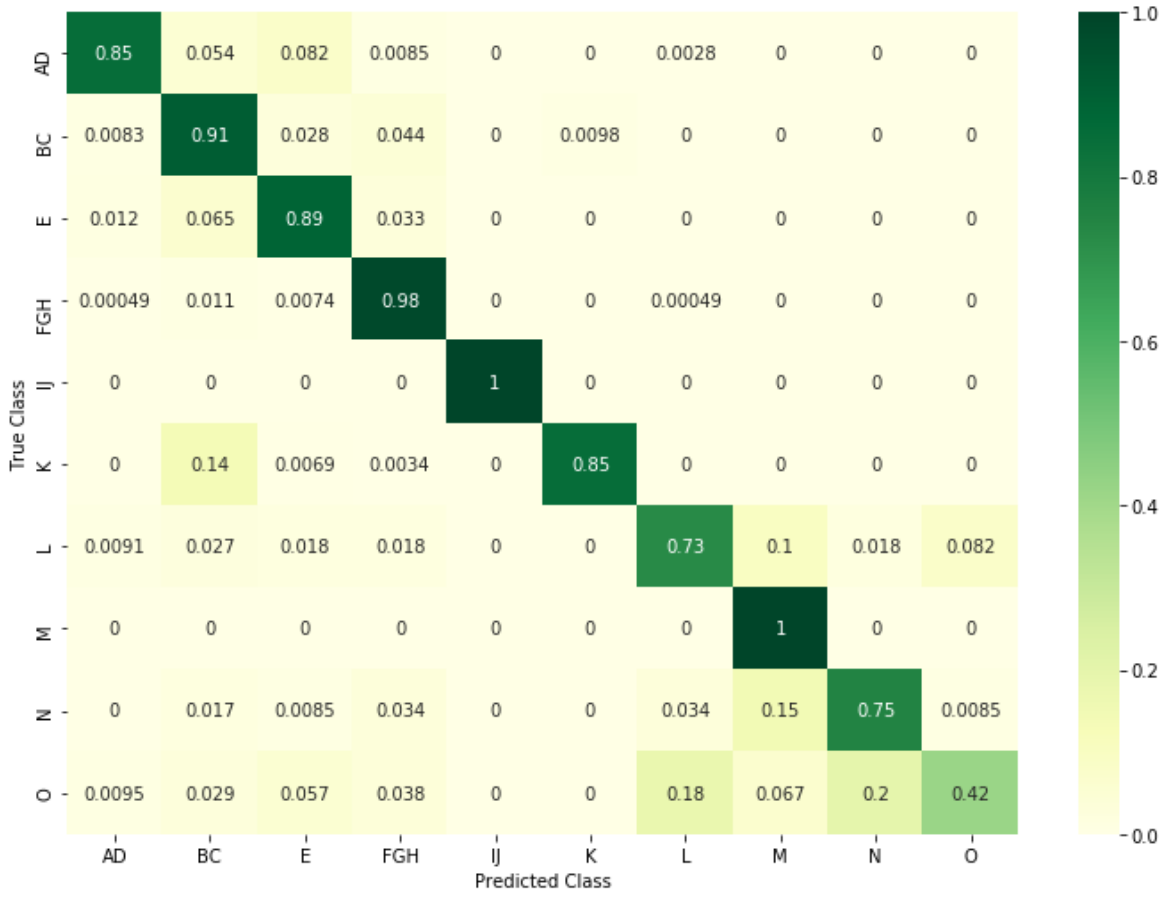

Figure 5.8: The confusion matrix for the top performing RF model on S5 resampled data.

5.2.2 Evaluation Using Power Wheelchair Data

The model transfer learning was performed on the models trained on S1 data and used all of the features (F1) for training. In order to compare the classification accuracy of the individual surface categories, all of the features are selected for training as they resulted in higher classification accuracy results. Nonlinear algorithms such as random forest, and especially deep learning models often require tens of thousands of data in order to tune all of the parameters in the learning algorithm and gain enough knowledge on the particular domain to make accurate predictions. In our experiment, MW data consists of 27,627 instances of surface features, whereas PW data has 7,865 instances or about 28.47% that of MW data. Table 5.11 shows the results for an independent T-test performed on the accuracy scores from 10 different ANN models trained on MW-only and PW-only data. We only observe S1, S3 and S4 since S2 and S5 includes a grouping of surfaces which were not collected during PW experiments. This statistical test is performed since the models are trained on two different samples collected from different devices. Since the p-value for all experiments is less than 0.05, the accuracy scores of both models are statistically significant. Therefore, we employ transfer learning techniques in order to improve learning in the target PW domain by transferring the knowledge already learned in the source MW domain.

Group		**Training Method**		**P-Value**	**Degrees of Freedom**	**T-Statistic**
Unsampled	**S1**	MW only	PW only	0.000255	16.214	4.6566
	S3	MW only	PW only	$3.28*10^{-6}$	16.046	-6.9401
	S4	MW only	PW only	$4.29*10^{-9}$	16.516	-11.134
Resampled	**S1**	MW only	PW only	$7.05*10^{-10}$	15.716	-13.117
	S3	MW only	PW only	$4.76*10^{-9}$	16.562	-11.033
	S4	MW only	PW only	$4.99*10^{-9}$	10.76	-16.668

Table 5.11: Independent samples T-test results for the MW-only and PW-only model accuracy scores

In the first attempt at transfer learning, we used the SER algorithm to perform model transfer on the random forest model, which resulted in the highest classification accuracy using unsampled and resampled MW data across all other models. However, this approach resulted in a target model that gave only a 23.04% accuracy when it was created from the existing model trained on unsampled MW data. Since these results are less than random guess, we used a feature transfer learning approach on the ANN model, which gave the highest accuracy among all the deep learning models at 73.81% when trained on unsampled S1 MW data and 70.27% trained on resampled S1 MW data. We perform a comparison of the two standalone models each trained on MW data and PW data respectively with the models that transfer their knowledge. Up to five layers of the ANN model are kept fixed, excluding the output layer. By fixing specific layers, their weights do not need to be updated on the new data and may decrease the training time on the PW domain. The accuracy scores are reported for all of the training methods in Table 5.12.

Based on the results in Table 5.12 the highest performing transfer learning method is one that has zero layers fixed and uses S4 grouping with an average accuracy of 88.41% for 10 different model

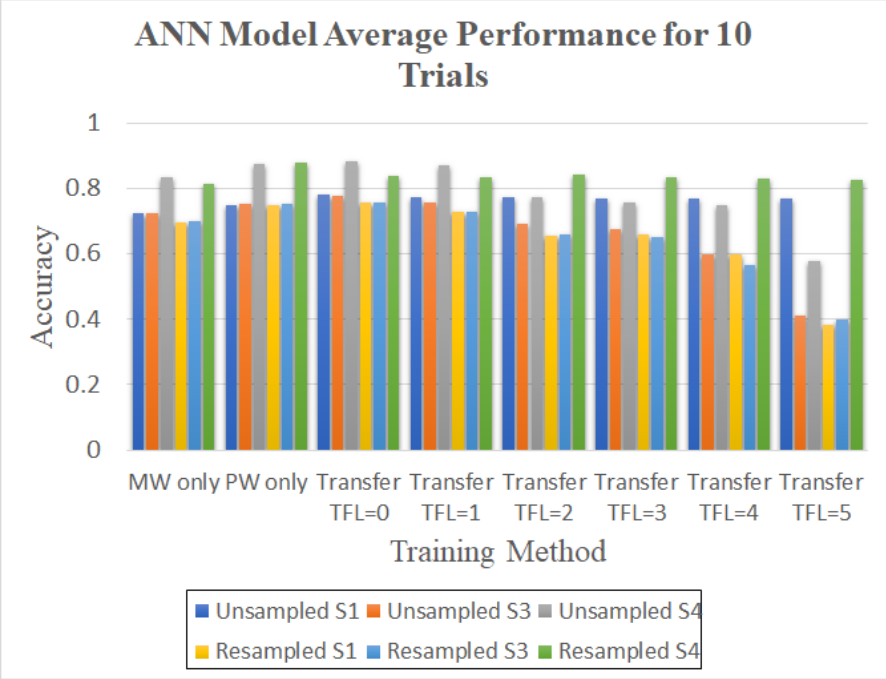

Figure 5.9: Mean Accuracy results for the Modeling Strategies shown in Table 5.12.

Performance Metric	**Category**		**MW only**	**PW only**	**Transfer TFL=0 (0 Layer Fixed)**	**Transfer TFL=1 (1 Layer Fixed)**	**Transfer TFL=2 (2 Layers Fixed)**	**Transfer TFL=3 (3 Layers Fixed)**	**Transfer TFL=4 (4 Layers Fixed)**	**Transfer TFL=5 (5 Layers Fixed)**
Accuracy	Unsampled	S1	0.7245	0.7474	**0.78**	0.7723	0.772	0.7696	0.7705	0.7676
		S3	0.7253	0.753	**0.7786**	0.7569	0.6919	0.676	0.5972	0.4117
		S4	0.8358	0.875	**0.8841**	0.8687	0.7718	0.7583	0.7474	0.5777
		S6	0.838	0.881	**0.8941**	0.8821	0.84	0.8266	0.7826	0.6067
	Resampled	S1	0.6953	0.7469	**0.7571**	0.7264	0.6568	0.6575	0.5969	0.3846
		S3	0.6996	0.7544	**0.7574**	0.7292	0.6601	0.6496	0.5642	0.3978
		S4	0.8128	**0.8779**	0.8391	0.8339	0.8431	0.8322	0.8307	0.8267
STD	Unsampled	S1	0.012	0.0085	0.0067	0.0038	0.0022	0.0047	0.004	0.0022
		S3	0.0068	0.0098	0.0065	0.0073	0.0096	0.0162	0.0114	0.006
		S4	0.0085	0.0063	0.0056	0.0055	0.0191	0.0159	0.0073	0.0025
		S6	0.0068	0.0068	0.0041	0.0062	0.0083	0.0089	0.0068	0.0053
	Resampled	S1	0.0066	0.0098	0.0081	0.0073	0.011	0.0115	0.0081	0.0041
		S3	0.012	0.0089	0.0069	0.0049	0.0078	0.0127	0.0091	0.0033
		S4	0.0035	0.0112	0.0049	0.0029	0.0045	0.0044	0.0025	0.0014

Table 5.12: Mean Accuracy and STD results fAcross 10 Trained ANN models with Unsampled and Resampled MW Model Transfer.

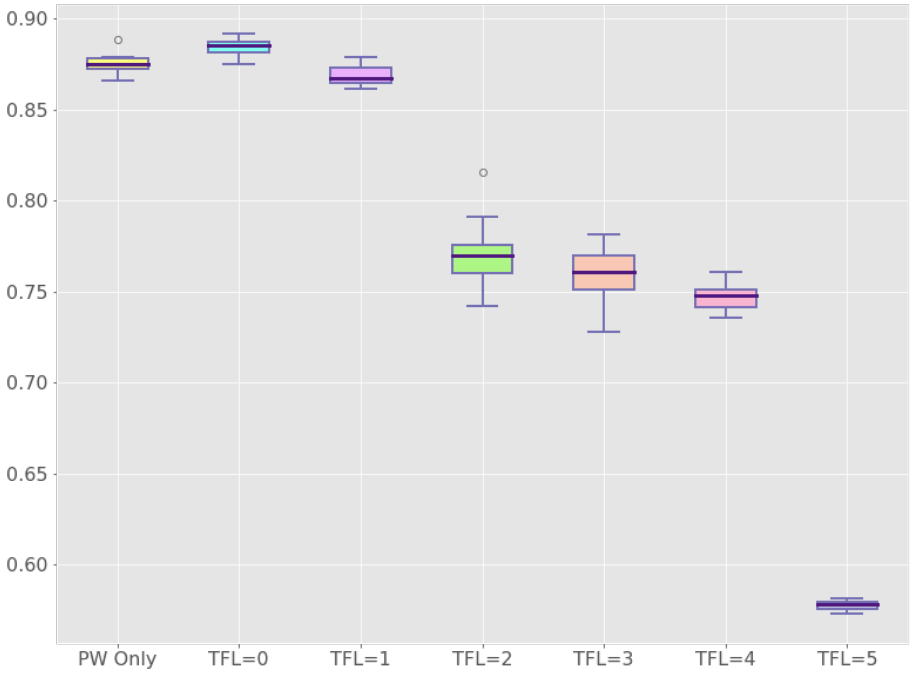

Figure 5.10: Accuracy distributions for the PW model training performed for the S4 Unsampled Group.

training trials. Figure 5.10 shows the distribution of accuracy scores for the training technique that produced the highest average accuracy. We performed a paired, two-sided T-Test with the 10 accuracy results from training using minimal PW-only data, which essentially is a weight initialization scheme, and the results from transfer learning model with no modifications to the existing model (TFL = 0). We assume a null hypothesis that the true difference of the means for the two groups is zero. Since the same PW data is being applied to different training techniques, the two observations come from the "control" and "experimental" conditions. The results of the paired test are shown in Table 5.13. For training using unsampled MW data, we can reject the null hypothesis since the P-value is less than 0.05. However, when resampling did occur on the data then only for S1 and S3 can we not conclude that a statistical difference exists in the means. Note that we do include an additional category in this analysis (S6) which is a merging of the two groups (S3) and (S4). The reasoning for this additional grouping is explained at the end of this section.

Group		Training Method		P-Value	Degrees of Freedom	T-Statistic
Unsampled	S1	Transfer TFL=0	PW only	$\mathbf{7.88}*10^{-7}$	9	11.967
	S3	Transfer TFL=0	PW only	$\mathbf{7.77}*10^{-5}$	9	6.8154
	S4	Transfer TFL=0	PW only	**0.007954**	9	3.3936
	S6	Transfer TFL=0	PW only	$\mathbf{3.27}*10^{-5}$	9	7.6163
Resampled	S1	Transfer TFL=0	PW only	0.0741	9	2.0205
	S3	Transfer TFL=0	PW only	0.1832	9	1.442
	S4	Transfer TFL=0	PW only	$\mathbf{2.4}*10^{-7}$	9	-13.749

Table 5.13: Paired T-Test results for test accuracy scores with PW-only model and highest accurate transfer learning models.

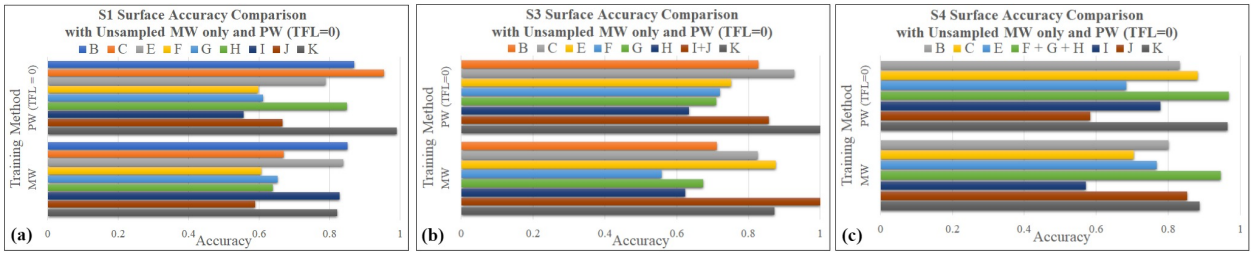

Figure 5.11: Classification accuracy for each surface category for groups (a) S1, (b) S3, (c) S4

Accuracy For Surface Prediction

For the S1 group, the highest classification accuracy occurred when zero layers are fixed to the model that is trained on PW data, which has an accuracy of 79.02%. in S1 with 79.02% accuracy. The S3 and S4 group also received the highest accuracy using transfer learning at 78.96% and 89.19% respectively. All of these results occurred without resampling strategies. The accuracy for each surface category that uses SLT Transfer Learning technique are shown in Figure 5.11((a)-(c)) which corresponds with the results in Table 5.14, Table 5.15 and Table 5.16.

Additional Experiment for S3 and S4 Grouping

Due to S3 and S4 grouping both providing statistically significant accuracy for unsampled PW-only and Transfer-TFL=0, we decided to merge these groups and combine the indoor surfaces and the curb directions into two different categories. This additional grouping we call S6. We wanted to see if we can produce a more accurate surface prediction with this group (see Table 5.12). We performed this without resampling because we found that a high classification can result without needing resampling. We achieved up to 84.69% on MW-only with an average of 83.8% and up to 89.26% on PW-only with an average of 88.1% using this grouping method. When we used transfer learning methods, we achieved a 1.31% improvement over training using only PW data and up to 90.02% accuracy or an average of 89.41% accuracy. This provided a statistically significant improvement as the P-Value of $3.27 * 10^{-5}$ is much less than 0.05 according to Table 5.13.

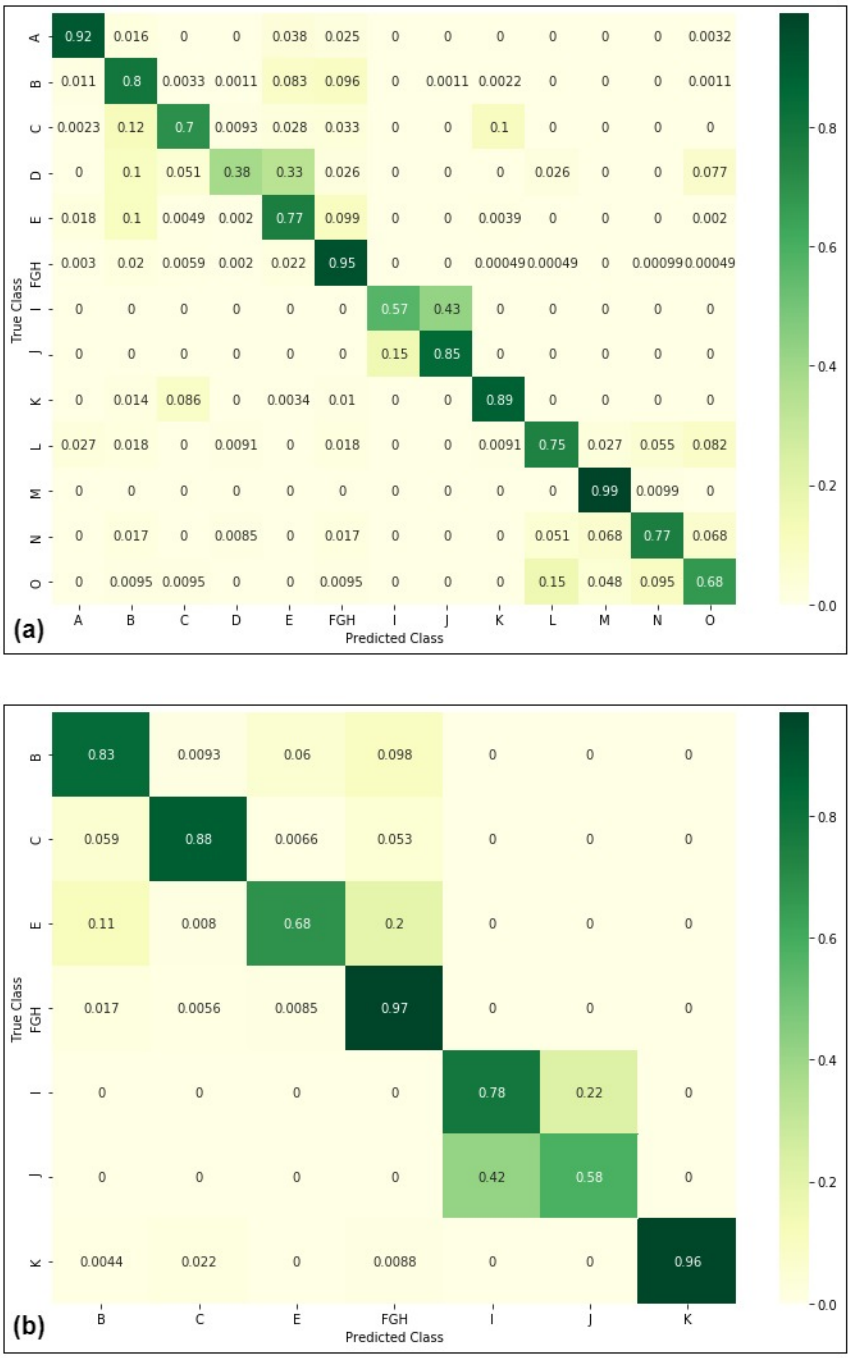

Figure 5.12: The confusion matrix for S4 ANN MW model (a) that is used to train the PW model confusion matrix (b).

	Unsampled		Resampled	
Surfaces	**MW**	**PW (TFL=0)**	**MW**	**PW (TFL=0)**
B	0.8504	0.8698	0.7879	0.8465
C	0.6698	0.9539	0.7837	0.9276
E	0.8379	0.788	0.7764	0.776
F	0.6063	0.5969	0.5733	0.6434
G	0.6531	0.6111	0.6857	0.6548
H	0.6377	0.8492	0.6317	0.6583
I	0.8286	0.5556	0.9714	0.6667
J	0.5882	0.6667	0.0588	0.25
K	0.8213	0.991	0.8041	0.9735
Model Accuracy	**0.7381**	**0.7902**	**0.7041**	**0.768**

Table 5.14: S1 group model performance comparison on different surfaces for trained MW-only and PW ANN models.

	Unsampled		Resampled	
Surfaces	**MW**	**PW (TFL=0)**	**MW**	**PW (TFL=0)**
B	0.7121	0.8279	0.7835	0.8233
C	0.8256	0.9276	0.7605	0.9474
E	0.876	0.752	0.8271	0.688
F	0.5588	0.7209	0.576	0.6318
G	0.6743	0.7103	0.6156	0.6825
H	0.624	0.6332	0.6377	0.7236
I+J	1	0.8571	0.9855	0.9524
K	0.8729	1	0.8247	0.9602
Model Accuracy	**0.7389**	**0.7896**	**0.7188**	**0.7686**

Table 5.15: S3 group model performance comparison on different surfaces for trained MW-only and PW ANN models.

	Unsampled		Resampled	
Surfaces	**MW**	**PW (TFL=0)**	**MW**	**PW Only**
B	0.8013	0.8326	0.8103	0.8419
C	0.7047	0.8816	0.7233	0.9013
E	0.7686	0.684	0.7363	0.688
F+G+H	0.9453	0.969	0.9586	0.9535
I	0.5714	0.7778	0.9429	0.5556
J	0.8529	0.5833	0.0883	0.5833
K	0.8866	0.9646	0.8282	0.9646
Model Accuracy	**0.8471**	**0.8919**	**0.8185**	**0.8875**

Table 5.16: S4 group model performance comparison on different surfaces for trained MW-only and PW ANN models.

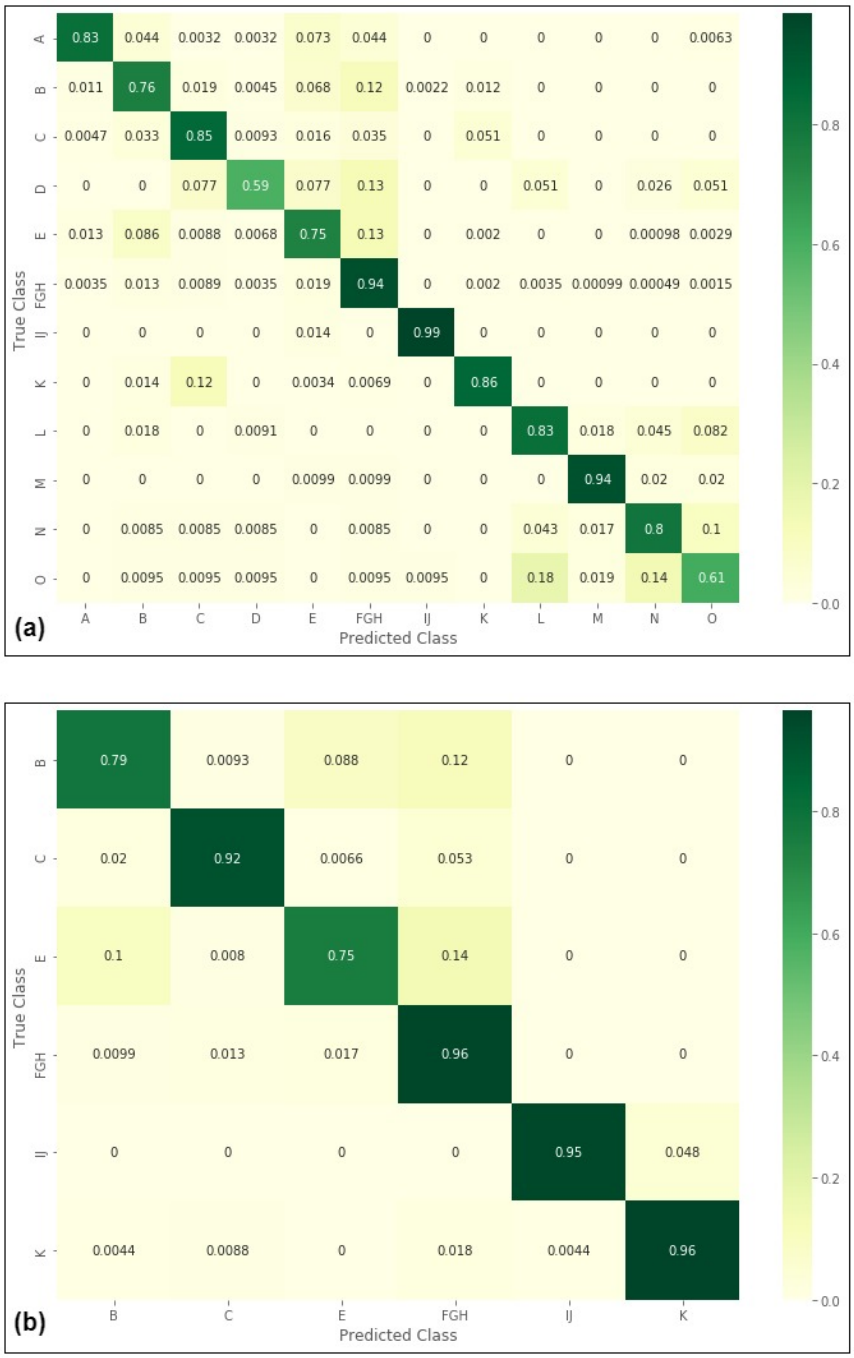

Figure 5.13: The confusion matrix for S6 ANN MW model (a) that is used to train the PW model confusion matrix (b).

	Unsampled	
Surfaces	MW	PW (TFL=0)
B	0.7645	0.786
C	0.8512	0.9211
E	0.752	0.752
F+G+H	0.9438	0.9605
I+J	0.9855	0.9524
K	0.8557	0.9746
Model Accuracy	**0.8469**	**0.9002**

Table 5.17: S6 group model performance comparison on different surfaces for trained MW-only and PW ANN models.

5.3 Discussion

The discourse on the design of accessible navigational systems pinpoints the necessity for considering the user preferences and abilities in order to recommend personalized routes that would suit their needs. However, the question still remains as to how to define accessibility within the mobility context. Existing literature shows differences in route calculation, the process of collecting information about various users and barriers to accessibility in the built environment and different ideas about what kind of accessible criteria exists in the mobility context.

Early research greatly relied on self-reported or crowd-sourced user efforts in order to deliver information about the built environment. One of the first navigational systems, MAGUS [52], greatly relied on specific authorities and systematic street inventories in order to understand the presence of certain obstacles, such as curbs, deep gutters and narrow pavements, and intended to assign scores depending on the nature of the parameter and the type of wheelchair involved. U-Access [54] also utilized official campus data and GPS geo-referencing to understand the specific height and width of various steps, sidewalks, curb cuts and location of accessible entrances. However, interest in personalizing navigation had led to the increase of crowd-sourcing data from the ordinary user. For the next decade, systems would involve user collaboration of data by including a user interface and a database for storing the inputted information [56, 74]. In some systems, users are even given the ability to add photos and comments about the various obstructions along their path [51]. This granted more knowledge about what accessible parameters existed in the environment that were outside of official records. In addition, users have also expressed a desire to be able to see information about surfaces, curbs and unevenness of the road in accessible systems [6]. Therefore, route selection has become more personalized, since route creation heavily depended on the contributions from the user and often chosen based on specific user preferences. The mPass system had users choose whether they liked, disliked, want to avoid or were neutral to various different types of urban features that may be encountered along their path [68]. Other systems aimed to create personalized routes that appeals to human preferences of quietness, beauty and happiness [48] and other systems prioritized safety [49]. After 2017, CAPRA also showed that using contour information from topographical maps can determine a path with the consideration of distance and then is evaluated for accessibility by using total vertical distance and maximum

slope [72]. However, the major problem that accessibility systems encounter is that without automated techniques such as machine learning methods, there is always a problem of accuracy when making routing decisions. Systems that are based on crowdsourcing alone have the potential of recommending unsafe routes or lead users along a non-existing road, since the knowledge base is solely reliant on user contributions. Even large map datasets such as Open Street Map lack information completeness as surfaces and objects are still limited in quality and 3D rendering [66]. Machine learning assists with data classification into two or more groups based on specific feature characteristics.

Computational scientists are tempted to design and develop multiple 'accessible' routes for users and then select their most preferred choice. However, accessibility is extremely critical as well as personal to the target users. For example, a building that is completely accessible for visually impaired users may not be accessible for wheelchair users given the presence of specific building features. Also, user endurance is important for determining accessibility, as surfaces and slopes that were easily accessible before may not be accessible all of the time. This case may happen if the user experiences fatigue, especially for longer distances. Slippery conditions due to rain or snow can also affect the accessibility of certain slopes and surfaces. Therefore, when designing systems to recommend accessible routes, systems must consider the mobility context by learning the domain knowledge. This consists of training machine learning models on various features extracted from the users and the built environment and then weighing and scoring road segments in order to make the ideal route decisions. These features may consist of personal capabilities of the users, their endurance, the nature of their mobility aid device, weather condition, along with knowledge from crowd-sourced data such as slope height and direction of travel, surface types, narrow pavements, and presence of accessible points of interests.

We examine different classifiers in order to address the problem of surface detection in a dynamically changing environment. The results from the machine learning models are necessary for filling the gap in the knowledge base that can affect the accuracy of accessible routing algorithms. Since routing engines make decisions based on the types of features, it is important to have a model that has a high classification accuracy in predicting the type of surface. A large contribution of the WheelShare routing engine is to utilize the results from trained, supervised models in order to understand how a user would move about the built environment within the specific mobility context. This is also represented by the nature of their mobility aid device.

Model Performance Using Grouping, Feature Selection and Resampling Methods

From observing the unsampled MW experiment results on F1 features for S1 groups shown in Table 5.9, Random Forest had the highest classification at 89.59% accuracy. Random Forest produced the highest accuracy when using F1 features and S5 grouping at 94.46%, precision at 91.27%, recall at 90.59% and F1-score at 90.83%. It also had the highest accuracy using F2 features at 89.12% with precision 83.35%, recall 82.76%, and F1-score 82.96%. RF had the highest average accuracy across all groups as well, with 91.474% using F1 features and 83.912% using F2 features. Grouping using S4 improved the accuracy of the RF classifier by 4.2% using F1 features and 7.49% using F2 features. Grouping with S5 had similar improvements by 4.87% using F1 features and 8.59% using F2 features. It does seem that grouping affects using F2 features slightly more. We

found that the average improvement for models using S4 grouping is 8.77% and S5 grouping of 10.55% using F1 features. These percentages improve by 5.12% to 13.89% when S4 is performed using F2 features. Similarly, the performance improvement increases by 2.56% to 13.11% when S5 is performed using F2 features as compared to S1 grouping. Therefore, we can conclude that grouping does seem to affect performance more when less features are being used for training. The highest improvement in performance occurred with S4 grouping using F2 features to CNN algorithm by 44.34% from 34.53% with S1 grouping to 78.87%.

Figures 5.7 and 5.8 show the RF that had the highest classification with unsampled MW data, F1 features and S5 grouping. These results show that both models had difficulty with surface O, or the Patterned Flagstone Road, which is often mistook to either be surface L, the Gusu Brick or N, the Asphalt Ridged Sidewalk. However, the China surfaces are from 6 users and are based on only the sensors from one phone and there is considerable less number of instances for the China surfaces compared to US surfaces as seen in Table 5.3, which can affect the test accuracy. Both models had over 85% accuracy for all of the US surface categories. In the future, these similarities may be explored further by grouping the China surfaces together and recording classification accuracy of the model on this particular category.

For ANN, which had the highest classification accuracy among deep learning models, the highest classification accuracy occurred with the S4 group at 84.71% accuracy using unsampled data. However, the results only differed with S5 by 0.18%. Also, the precision rate for S5 is 3.82% higher than S4 for ANN, and recall and the F1-score see similar improvements with 2.36% increase and 4.45% respectively. When performing resampling, ANN has higher accuracy with S5 grouping as compared with S4 by 2.2%. It's also important to note that when grouping unsampled surfaces to S4, training the CNN classifier with F2 features gave the closest accuracy results to F1. The accuracy for S4-F2 is at 78.87%, while the S4-F1 is at 79.17%. Therefore, reduction of features and S4 grouping does not affect classification accuracy with CNN as strongly as the other models. Therefore we can conclude that S4 and S5 grouping both greatly improved the performance of the deep learning models.

When resampling is applied, RF had the highest classification accuracy using F1 features at 91.21% with S5 group, 87.27% precision, 83.85% recall and 84.68% F1-score as shown in Table 5.10. RF also had the highest accuracy using F2 features at 86.17% accuracy, 79.72% precision, 77.63% recall and 77.91% F1-score. Resampling technique seemed to only slightly improve the accuracy rates in CNN at S3, S4 and S5 and ANN at S4 and S5 when using all features F1. When only using some of the features F2, only CNN improved accuracy in S1, S2 and S3. Both unsampled and resampled data led to a low accuracy in Naive Bayes. This performance is explainable as features extracted from the sensors are dependent on the phone placement. Therefore, NB conditional independence assumption leads to low classification accuracy. It's also important to note that the resampling technique reduced the accuracy of the LSTM model to below random guess. We believe it is because the ReLu activation function can be very unstable depending on the information and so input weights are not attuned properly to the domain.

The S1-F1 F1-score results for SVM and RF are very similar to those trained on the DS2 dataset in SmartWheels [22], which also uses integrated sensor data from smartphones, along with smartwatches. In their data, SVM achieved the lowest F1-Score at 0.6 and RF achieved the highest

score at 0.8. In our unsampled data configuration, SVM is the second lowest model with an average F1-score of 0.7026 for F1 categories and 0.5066 for F2 categories. RF similarly achieved the highest average F1-score of 0.8835 for F1 categories and 0.7852 for F2 categories. The same data resampling approach is used in SmartWheels, which increased the coarse grained (obstacle, gait and still) F1-Score by about 0.08. Our work is on surfaces and curb direction only, which would be most likely classified as fine-grained as they share the same parent label of *gait* and have only slight differences among their siblings in the SmartWheels' proposed feature label hierarchy. However, we see a reduction in performance after resampling with the F1-Score for RF S1-F1 data by 0.0697 after resampling and by 0.063 for S1-F2. However, initial attempt at resampling had increases the instances of the whole data set, which is followed by splitting into train and test data. We found that the resampling technique improved the F1-score by an average of 0.168 for models trained on F1 data and 0.1868 for models trained F2 data. However, resampling the whole dataset causes optimization bias because the test data contains resampled information that tends to overlap with the known data used to train the model. This concluded that our first reported highest accuracy with KNN at 99.7% is not correct, so we had to modify our classification procedure. Based on the resampling of the class labels, the ENN resampling technique may not be effective with surface collection where fine-grained information about the specific surface features exists.

While the results from models trained on F1 features can explain the type of surface for a particular assistive device if a sensor is attached directly to the user or the wheelchair, different locations or positions of the sensors can affect the classification. For example, SmartWheels [22] explains that the placement of a phone in a bag or pocket can cause noisy inertial data and reduce accuracy of the models. Our research is limited to the data from sensors that reflect the movements of the wheelchair and the device specifically. Future collaborators to the WheelShare system may not place phones in the same orientation or in the same location, which can present a challenge in classification. Machine learning techniques may aid with determining the position and placement of the phone during travel. Notably, F2 does not include features that depend on a single direction, and it is possible that using data from other devices, such as a smart watch, can account for user orientation. Using multiple sensors from other devices may also help to understand the variation of vibration data as the user travels on a different surface.

Transfer Learning Model Performance

The current hypothesis in deep learning research is that by designing models for the purpose of untangling the factors of variation, then different learning tasks, such as supervised and transfer learning, would become much simpler and easier to perform [91]. By utilizing deep learning models to understand multiple "abstract" representations, the approach is to utilize feature transfer to enhance learning and remove the cost of resources needed to retrain and create a new, deep learning network model. In our case, this would be extremely useful when we have a lack of completeness in the number of features in the built environment that are collected by users with one device, but have access to more data from users with a similar but different device like a manual wheelchair. When we performed the SER approach, this resulted in a target model that gave only a 23.04% accuracy when it was created from the existing model trained on unsampled MW data. We believe that this method did not produce an accurate result because there isn't

enough extensive research performing model transfer using regular classification models. Rather transfer learning traditionally is applied on deep learning models. Therefore, we opted to go with the second approach using SLT.

We were interested specifically in applying transfer learning to models trained on S1 data as we wanted to understand their performance to a surface type level. From our results, RF had the highest classification accuracy among all the models across all groups at 91.47% using resampled data and S5 grouping and 94.46% using unsampled and S5 grouping. ANN had the highest accuracy among the deep learning models at 73.81% using S1-F1. The transfer learning approach can lead to more scalable route calculation solutions since data from manual wheelchairs does not need to be used after the initial training of the model. This can also decrease computational complexity and reduce space about the feature types in the built environment, since there is no need to consider features that are only important for manual wheelchair users if the user's device is a power wheelchair. Even efficient algorithms such as SVM and Naive Bayes will have a high-cost for training time and space as the surface knowledge increases. While training on S1 data, we found that RF took about 150 times longer to train than Decision Tree, Naive Bayes and KNN and about 3 times longer than SVM. In comparison, ANN training is about 10 times faster than RF at approximately 8 seconds compared to 85 seconds for RF. ANN is also 2.4 times faster than CNN and 22.5 times faster than LSTM. Therefore, the ANN model is chosen for transfer learning as not only did it have the highest classification accuracy among the deep learning models, but also one of the fastest training times and required the lowest number of epochs to train.

Fixing layers while performing Selective Layer Training caused a subset of the original feature space to be optimized to the target PW domain. The reduction of classification accuracy performance as more layers become fixed, as shown in Figure 5.9 and Table 5.12, may be affected by the number of neurons fixed on the new data criteria. If the previous state is fixed on the weights trained on MW data and the input is PW data, then this may result in low accuracy if not enough generic features are captured during training. The quantity of MW and PW data causes classification to be particularly sensitive to the domain representation. For example, the S4 group with TFL=0 had the highest classification accuracy on average at 88.41% with unsampled data, however this accuracy dropped to 83.91% on average when resampling is applied to the MW data prior to initial training. Actually, in the later case, transfer learning did not improve performance as a solely training ANN with PW data gave an average accuracy of 87.79% or 3.88% higher than applying transfer learning. However, this was the only case where this occurred, as overall transfer learning did improve accuracy for other groups. Only when unsampled MW data is used in training did it have significant improvements over model with PW-only training as seen in Table 5.13. We can say with a 95% confidence level that the accuracy results of both methods of training are statistically significant from each other for unsampled MW training since the p-value for this group is less than 0.05. The opposite can be said for S4-Resampling as there was a statistically significant reduction in performance when transfer learning is applied to a model with S4 resampled MW data as compared to solely training the model with PW data with S4 groups.

While training occurs only with minimal PW data, based on results from Table 5.12, the PW model achieves the highest average surface classification accuracy for 10 trained models of 87.79% and an average of 74.69% using S1 grouping. The highest average accuracy improves to 88.41%

with transfer learning methods without needing to resample the original MW data and an average of 78% or a 3.26% improvement over the minimum PW-only S1 of 74.74%. So we conclude that the highest performing transfer learning model is the one that has zero fixed layers. Although these results appear to be very similar to those where the better training model technique is applied, the results of the paired t-test from Table 5.13 show that using unsampled data in MW training of the original model actually leads to a statistically significant improvement in transfer learning models using S1, S3 and S4 grouping. Furthermore, S4 grouping improves transfer learning accuracy on average by 10.41% using unsampled data and 8.2% using resampled data. Only when we use resampling and S4 grouping do we see a statistically significant improvement in models where no transfer learning occurred but we also see a significant difference in the drop of performance in the MW-only model by about 2.3% and in transfer learning for 4.5%.

The average classification accuracy was 2.97% lower when resampled data is used for transfer learning using zero fixed layers as compared when unsampled data is used for initial training. We conclude that ENN resampling technique on MW data does not improve the transfer learning accuracy to the target PW domain. For all three groups, the resampled models that were trained on MW-only and the transfer models all had lower accuracy than when no unsampling is applied prior to training. This could be because increasing the amount of instances really does not improve the knowledge about the original MW domain and thus leads to no improvement of performance in the transfer learning. However, it is possible that the resampling technique does aid with generalization of the knowledge represented in both PW and MW domains. In Table 5.12, S4-Resampled with TFL=5 reports an accuracy of 83.4%, which is the highest accuracy among all of the unsampled and resampled groups using this number of fixed layers. Similarly, computing the standard deviation of the six accuracy scores for each layer fixed, we found that S4 grouping has the lowest standard deviation among the transfer learning resampled groups at 0.0059, therefore the accuracy changes the least as the number of fixed layers increases with this grouping.

Performing the same approach to unsampled groups, we found that S1 grouping has the lowest standard deviation across the different accuracies reported as the number of layers are fixed at 0.0043. Therefore, we may conclude that even though less weights in the layers are being attuned to the new PW knowledge domain, the features are generic enough that surfaces can still be detected at a similar accuracy. The specific data from the initial source MW domain generalizes well on data from a target PW distribution. Therefore, in the cases of Unsampled S1 and Resampled S4 groups, the transfer learning models that are adapted to the PW domain share a relatively similar feature distribution as the original model adapted to the MW domain. We may be able to explore the similar features from these groups for model experiment training.

We decided to do a combined group of S3 and S4 to S6 since we were able to achieve a statistically significant improvement comparing the transfer learning model (TFL=0) and MW model using unsampled data. We were able to improve our highest average accuracy by 1% to 89.41% from 88.41% using S4 unsampled group. We designed the first ever transfer learning based surface classification for accessible route generation. We have successfully demonstrated knowledge transfer from a manual wheelchair to a power wheelchair up to 90.02% accuracy using only nominal amount of new training data.

Individual Surface Performance

The classification matrix for the source model trained on MW is shown in Figure 5.13(a), which led to the highest transfer model accuracy whose matrix is in Figure 5.13(b) and Table 5.16. From these results, we can see that both models had a relatively low classification of recognizing curb direction of movement (categories I and J) compared with other surfaces. However, the source model has a higher accuracy with recognizing curbs when the user is going down than up and the opposite is true for the target model. Actually, both models had a very similar lower accuracy for curb direction as about 57% accuracy occurred with I in the source model and a 58% accuracy with J in the target model. This may be due to the power wheelchair having a higher weight, keeping relatively constant speed due to engaged drive motors controlling movement and the presence of air pressurized tires to stabilize the movement as it travels up or down the curb. Irregular movements and possibly slower speeds from experimenters due to fatigue and specific device characteristics of manual wheelchairs, such as flat-free tires and light weight, are not that common in power wheelchair movement. Users can keep a relatively constant speed according to the automatic setting on the power wheelchair device, which is more like to be adjusted when the user goes down, making down curb direction harder to be detected for the PW model. The opposite is true for manual wheelchairs as users would likely slow down due to fatigue when traveling up the curb as opposed to going down.

The limitation of this work is that we only considered one specific curb type in order to analyze the direction of movement. From these results, it is not possible to determine which direction is easier for the model to predict as it depends on the device being used as well as the user and curb type. In the future, we may analyze other types of curbs and see whether the model can differentiate between them. However, both models trained on MW and PW data are able to predict indoor surfaces at above 94% accuracy and the rough brick surface at above 88%. According to our calculations in Table 6.1, the rough brick surface had the lowest score besides curbs in the United States surface collection experiments and we achieved a high accuracy for recognizing this important surface type in accessible navigation. In our attempts at grouping S3 and S4, the rough brick recognition improved to 97.46% and indoor surfaces can be predicted at 96.05% and curbs up to 95.24% with the trained PW model.

Based on these results, we can accept our hypothesis that high classification accuracy using machine learning based algorithms can be attained if enough data about accessible surface and curb features can be gathered from accelerometer and gyroscope sensor data. Both unsampled and resampled datasets using MW data as well as the transfer learning method of selective layer training using PW data have shown substantially greater classification accuracy than random guess. This research may aid greatly into acquiring more knowledge about important features for accessible navigation systems that greatly rely on crowd-sourcing efforts for data.

Chapter 6

Accessible Route Generation

The goal of the accessible route system is to select paths by considering the *accessibility*, *safety* and *difficulty* of the end-to-end route trip. A route is created between a pair of starting and ending vertices. The purpose is to find optimal routes from the source to the destination based on accessibility factors. A starting vertices can occur inside a building, at which multiple routes may be generated from the different accessible entrances in that building, or it may be any node on the map. Separate scores are assigned for the aforementioned perspectives which depend on the steepness of slope, presence of stairs and crosswalks (with and without pedestrian crossing signals), height of curbs and the type and nature (broken or uneven) of surfaces. In the system, each route is based on an encoder, which has a particular setting depending on the device the user has selected. These scores are used by remote GraphHopper server to select multiple accessible routes to recommend to the user. Section 6.1 explains how surfaces receive an accessibility score based on their vibration data gathered from multiple contributors. Section 6.2 is an explanation of route selection based on accessibility factors.

6.1 Assigning Accessibility Scores

Surface accessibility analysis is relatively subjective as there exists no agreement in research as to what features are important for accessible navigation. Since a 'one-size-fits-all' is not an appropriate solution for accessibility, our solution for assigning scores is intended to be generic by observing the ADA 2010 guidelines [13] mandatory for American urban planning of environments, facilities and infrastructure. The accessibility specifications for floor and ground surfaces mention that they "*must be stable, firm, and slip resistant [like, concrete, asphalt, tile, or wood and unlike gravel or grass]. Stable surfaces resist movement, while firm surfaces resist deformation by applied forces.*" Therefore, rough surfaces like cobblestones or Belgian blocks are "*unsuitable for wheeled mobility aids due to the vibrations they cause.*". However, the smoothness of the surface is also important to consider as it affects the friction contact with the wheels. Walkways and ramps should have clear width of 36 inches (915 mm) minimum. Also, the guidelines explain that widths can be different as the wheelchair turns. An accessible route makes a 180 degree turn around an element which is less than 48 inches (1220 mm) wide, clear width shall be 42 inches (1065 mm) minimum approaching the turn, 48 inches (1220 mm) minimum at the turn and 42 inches (1065 mm) minimum leaving the turn. Changes in level can be up to 1/4" without treatment or 1/2" if beveled with a slope no steeper than 1:2. Changes in level above a 1/2" must be treated as a ramp or curb ramp. Ramp specifications require a maximum of 1:12 ramp slope ratio which equals 4.8 degrees slope or one foot of wheelchair ramp for each inch of rise and a minimum of 1:20 ramp slope ratio. The width

of a road segment should be larger than 90 cm for most wheelchairs to pass through, and any less then it is inaccessible. Figure 6.1 shows this representation and a path with a width of 87 cm is not considered.

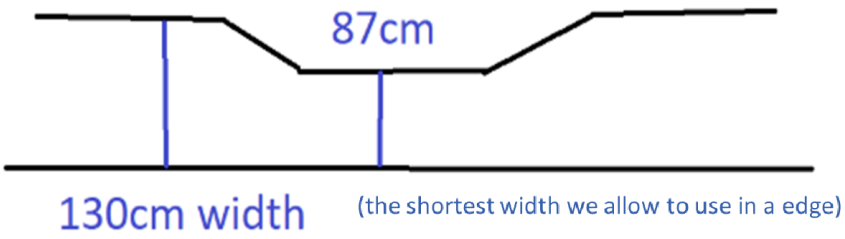

Figure 6.1: Shortest width for an edge

Indoor spaces require clear floor space, turning space, and door maneuvering clearances. Similarly, the DIN 18024-1 [98] also specify that accessible paths "*shall be easy, with low-vibration and safely accessible in each weather condition*" without any particular quantification of different parameters. By considering multiple ADA regulations, we assign accessibility scores to different surface types.

A score of 0 represents very uncomfortable and is assigned to broken and missing sidewalks. Surfaces that are marked as cobblestone, gravel and sand are also marked as inaccessible and receive a score of 0. Roads that are marked as footways, streets, pedestrian roads and residential areas are all valid road ways for accessible routing. Roads that are marked as private access or contain highways that don't have a sidewalk tag on them are not accessible, since a wheelchair is not able to share the same road as a car. Handrails, walls, stairs and turnstiles are marked as barriers, whereas curbs are marked as potential barriers for wheelchair device. The conditions for a curb to be accessible are:

- Ramps that have a slope greater than 4.8 degrees are not accessible.
- If the road contains a curb that has a raised road edge greater than 3cm or 30 mm, then it is not accessible for wheelchairs.
- The width of a road should be larger than 90 cm, otherwise it is inaccessible.
- Curbs that contain a gutter for water are also not accessible by wheelchairs.
- Curbs that are lowered or contain a ramp are accessible and are often found at pedestrian crossings.
- Other curbs that are meant for vehicles and occur on driveways are not accessible.

A road with a crosswalk receives a lower score than one with no crosswalk as it requires travel through a potentially busy road. However, crosswalks with pedestrian signals receive a slightly higher score than those without.

We've collected sensor vibration data from 15 different surfaces attached to the manual wheelchair device (see Chapter 5 for the Experimental Study and Figure 5.2 for surface images). Algorithm 4 assigns different accessible scores to road surface types. The scores for the relevant surfaces to this work are reported in Table 6.1.

First, the scaled sum acceleration values are calculated for each $surface$ collected in the list of $possibleSurfaces$, and the average and standard deviation for all the experiments are computed for that surface. These are stored in the dictionaries avg_S and std_S with the key being the $surface$. We then find the $surface$ with the lowest and highest average and round down to get the min and max value. We then add them and divide by two to get the $mean$.

A binary decision tree formed on line 14 is shown in Figure 6.2 is formed from our data collection experiments using 15 surface vibration data. The root node is formed with two branches at less than and greater than the min. We then add $mean$ to min as long as $mean + min \leq max$ and set branches at less than or greater than $min + mean$. As long as $mean > 0$, we create 2 child nodes for each n with each branch formed at less than or greater than $min + \sum_{n=1}^{\infty} mean * n$ for all $n \leq \frac{max}{mean}$. These scores provide a range set for each leaf node in the tree.

The X represents the average scaled sum of acceleration for a surface type. The surface is placed in a group formed at the leaf node. In the tree, the number of groups is the number of leaf nodes. The surfaces are assigned a score depending on the $RangeScore$, which is a reverse order list from 1 to 0 that decreases by the time step of $\frac{1}{\text{number of groups}}$. After being placed into particular groups (numbered 0 to the number of groups) formed at the leaves of the tree, surfaces are sorted according to the standard deviation of their acceleration data. After sorting, their standard deviation is rounded up to the nearest 0.1. The lower the standard deviation the surface has, the higher its score will be. Surfaces with the lowest standard deviation in their group attain the $maxScore$ or the highest score allowed by their score range. We assign scores to surfaces in each $group$ in the $decisionTree$ using preorder traversal, therefore group 0 will have the highest scored surfaces. For each $surface$ in $group$, the max score for that surface score range is reduced by the Δstd between the highest surface standard deviation and that surface standard deviation times 0.5. If the score is less than the minimum score allowed in the $scoreRange$, the surface score becomes the minimum score.

The goal of this algorithm is to assign accessible scores for the majority of users on the manual wheelchair and to create an objective understanding of accessibility for surfaces. Manual contribution of data by users is often not preferred or unclear, and it is difficult to create a system that suits all accessibility needs without creating possibly thousands of complex groups of users with specific needs. Using Algorithm 4, we are able to assign scores to the surfaces important to accessible navigation, which are shown in Table 6.1.

Algorithm 4 Assigning scores to surfaces

Obtain list of $possibleSurfaces$
Initialize Dictionary avg_S and std_S
for $(every\ surface \in\ possibleSurfaces)$ **do**
    $vibrationData \leftarrow$ List of $surface$ Scaled Sum Acceleration Values for each Experiment
    $avg \leftarrow computeAverage(vibrationData)$
    $std \leftarrow computeSTD(vibrationData)$
    $avg_S \leftarrow (surface, avg)$
    $std_S \leftarrow (surface, std)$
end for
// Find the highest and lowest surface values
$min \leftarrow roundDown(min(avg_S))$
$max \leftarrow roundDown(max(avg_S))$
$mean \leftarrow (min + max)/2$
Create a decision tree by placing decision nodes at $< min$ and $> max$ and
    at each $mean$ point between min and max
// A group is formed at each leaf node in the decision tree and consist of one or more surfaces
Assign each $surface$ in avg_S a $group$
$scoreRange \leftarrow 1/$(number of Groups)
Create reverse order list $RangeScore$ from 1 to 0 decreasing by $scoreRange$
Sort each $surface$ in each $group$ from lowest to highest STD found in std_S
Round up each STD in std_S to the nearest 0.1
// Assign scores for each surface in group starting from highest scored group 0
for $(every\ group \in\ decisionTree)$ **do**
    $maxScore \leftarrow RangeScore[group]$
    $minScore \leftarrow RangeScore[group + 1]$
    $firstSurfaceSTD \leftarrow 0$
    for $(every\ surface \in\ group)$ **do**
        $currentSurfaceSTD \leftarrow getValue(std_S, surface)$
        if First $surface$ in group **then**
            $surfaceScore \leftarrow maxScore$
            $firstSurfaceSTD \leftarrow currentSurfaceSTD$
        else
            $differenceSTD \leftarrow currentSurfaceSTD - firstSurfaceSTD$
            $surfaceScore \leftarrow maxScore - 0.5 * differenceSTD$
            if $surfaceScore < minScore$ **then**
                $sufaceScore = minScore$
            end if
        end if
    end for
end for

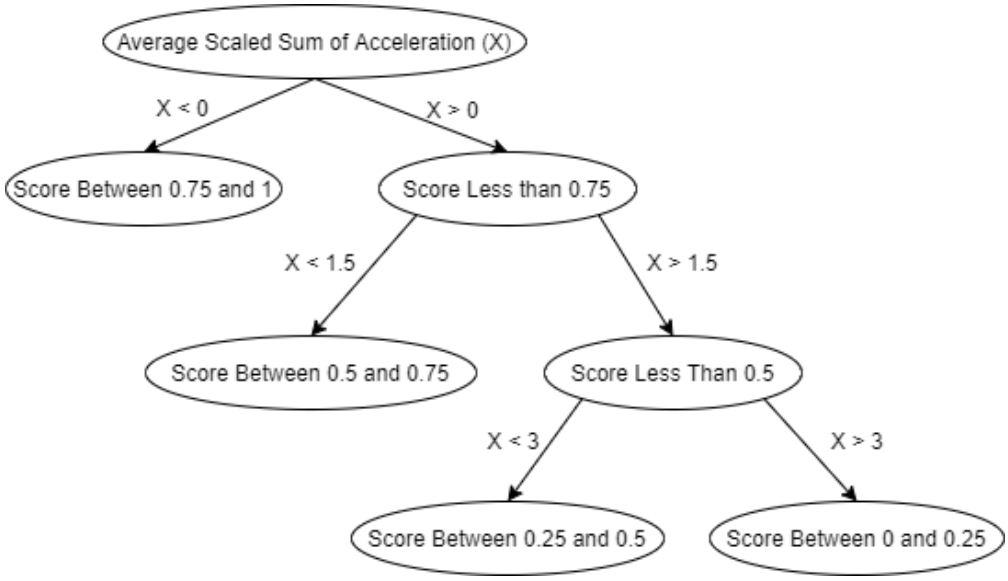

Figure 6.2: Hierarchical Decision Tree for Assigning Scores to the 15 Surfaces.

Road Surface Type	Score
High St. Smooth Brick 1	1.0/1
Benton Indoor Carpet	0.9/1
Benton Indoor Mat Surface	0.9/1
Benton Indoor Tiles	0.9/1
Asphalt Ridged Sidewalk	0.9/1
Asphalt Sidewalk	0.85/1
High St. Sidewalk	0.8/1
Patterned Flagstone Road	0.75/1
High St. Smooth Brick 2	0.75/1
Parking Lot Asphalt	0.75/1
High St. Brick	0.65/1
Brick Surface in Gusu	0.65/1
Granite Tile	0.65/1
E. Park Plaza Curb (Going Down)	0.5/1
E. Park Plaza Curb (Going Up)	0.25/1
Cobble Stones Outdoor	0.0/1
Sand Outdoor	0.0/1
Deep Gravel Outdoor	0.0/1

Table 6.1: Road Surface Score

6.2 Accessible Routing Algorithms

Accessible routes are found based on the knowledge from the built environment. As is mentioned in Chapter 3, we first generate an overlay graph on the map of the environment. The graph (G) has a set of vertices (V) and edges (E). Figure 6.3 illustrates a vertex as a red point and a blue directional arrow is the direction of travel, or the edge connecting two vertices. A vertex is set on a path when there is a change in surface type, or vibration pattern, such as in Figure 6.3(c). Each edge connects a pair of vertices together, therefore an edge represents a single surface. In a 4-way crossing, 4 vertices are made as seen in Figure 6.3(a) where there exists the starting and ending points of a pedestrian crossing. Also, vertices are set at sidewalk junctions as seen in Figure 6.3(c). Vertices are also set at accessible building entrance/exits as shown in Figure 6.3(d).

Given an input is a graph that is a set of vertices which represents accessible locations and the edge connecting those vertices represents the path, the weights are assigned for each path based on multiple standards found in the Americans with Disabilities Act (ADA) [13].

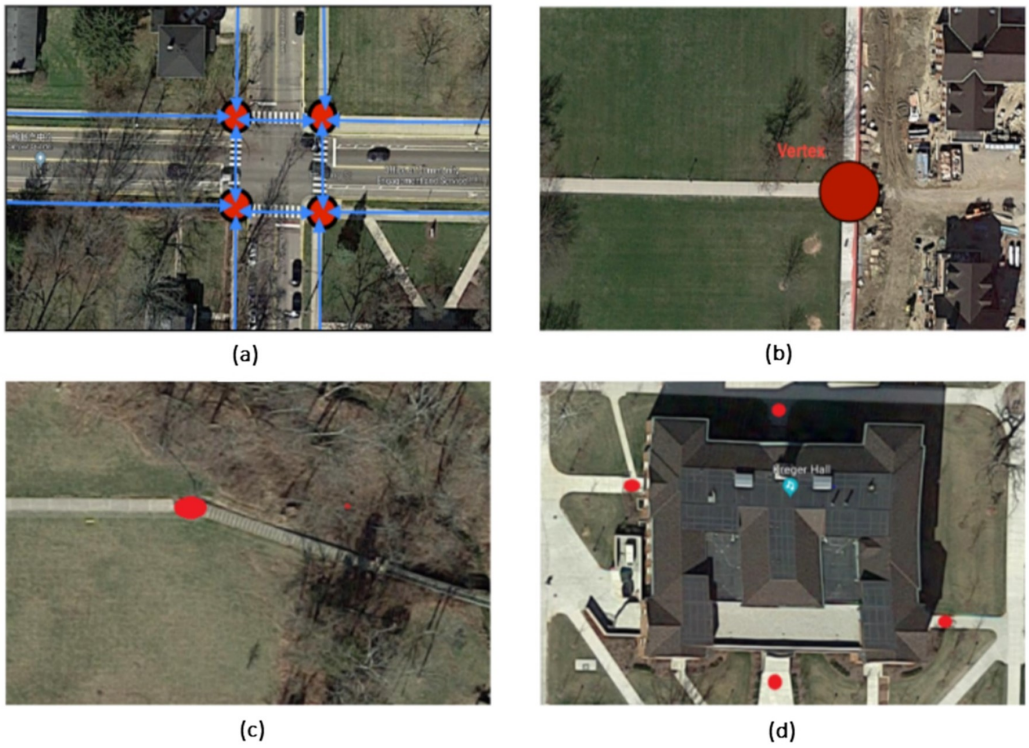

Figure 6.3: The representation of vertices and edges in the abstract built environment.

This section describes our routing algorithms after the summary of the variables used in Table 6.2. These algorithms provide optimization of the route selection process by considering the *accessibility*, *safety* and *difficulty* of the routes for wheelchair users.

Algorithm 5 illustrates how scores for all roads in a route is chosen. Once these scores are determined, they are saved into the database for route calculation. We consider the $safetyScore$, $difficultyScore$ score to be at a maximum of 100. The higher these scores are, the higher the

Variable	Meaning
Route	All connected vertices $Point_{start}$ to $Point_{end}$.
accessible	Not broken, ramp, width greater than 90cm, slope <= 4.8 degrees
slope	Slope of the edge
crossroad	Road contain a crosswalk or not
traffic	Road contain a pedestrian traffic light or not
roaddiff	Difficulty score out of 1 for different road surface
$Point_{start}$	Vertex start point of route
$Point_{end}$	Vertex end point of route
$list_{Point_x\ conn}$	Array stored connected vertices of $Point_x$
routeLength	Number possible routes
possibleRoutes	Routes already found a to b
lengthOfRoute	Length of currentRoute
sameLength	Two routes same part length

Table 6.2: Variables Used

score of the road. The $roaddiff$ is the score of the surface type of the road, whereas $slope$ is the current slope of the road. We consider safety and difficulty to vary depending if there is exists a crossroad or if this crossing tends to have busy traffic. If the road is a pedestrian crossing, the difficulty score will be lower since it doesn't receive a 30 point increase. If the road contains traffic, the safety score is reduced by 10 as compared to when there is no traffic.

Given a pair of start (A) and end (B) vertices, Algorithm 6 searches to finds all possible routes between them. In order to restrict the searching area (to keep a check on the generated route lengths), the search area is limited to the Cartesian distance between A-to-B, which is represented as the diameter of the circle (see Fig. 6.4). The diameter is incrementally updated with a specific value until at least one route is found. The circle-check is shown on the real map on Figure 6.5(a)-(b)). Algorithm 7 illustrates the circle check operation.

The generated routes are sorted by length and determined by the proportion of total length of every route. While selecting the top 10 routes we had to do away with the algorithm potentially recommending duplicated routes. If two or more routes are having a majority or about 90% of the edges in common, the routes are examined using a similarity check operation shown in Algorithm 8. The similarity check is performed between every pair of routes on an edge-to-edge matching basis. Similar routes are dropped and the top accessible 10 routes are chosen that have the same starting vertex. Scores are assigned to each edge depending on its proportion of length with respect to the length of the whole route. Consider a route with three edges and varying lengths 10m, 8m and 2m. The total length of the route therefore is 10+8+2 = 20m. The final route score = (score for first edge* 10/20) + (score for second edge * 8/20) + (score for third edge *2/20). At the end, we are able to return multiple routes with its length and score back to the user.

Algorithm 5 Determine Score for all roads in Route

scoreOfRoads = $Array(Point_{start}, Point_{end})$
for road in Route **do**
    roaddiff = GetRoadType(road)
    safetyScore = 0.0
    difficultyScore = 0.0
    if road is accessible **then**
        if crossroad==false **then**
            safetyScore = $roaddiff * 40 + ((10 - |slope|))/10 * 40) + 20$
            difficultyScore = $roaddiff * 40 + ((10 - |slope|)/10 * 30) + 30$
        else
            if traffic == true **then**
                safetyScore = $roaddiff * 40 + ((10 - |slope|)/10 * 40)$
                difficultyScore = $roaddiff * 40 + ((10 - |slope|)/10 * 30)$
            else
                safetyScore = $roaddiff * 40 + ((10 - |slope|)/10 * 40) + 10$
                difficultyScore = $roaddiff * 40 + ((10 - |slope|)/10 * 30)$
            end if
        end if
    end if
    finalScore = $safetyScore * 0.6 + difficultyScore * 0.4$
    scoreOfRoads.append(road, finalScore)
end for

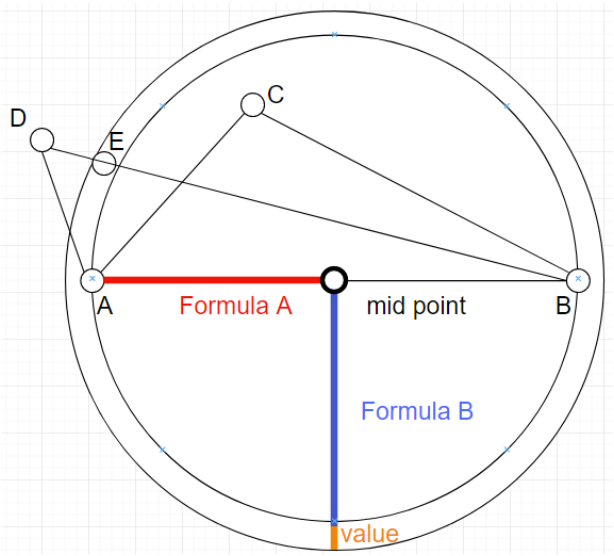

Figure 6.4: Accessible Routing Approach and Route Generation

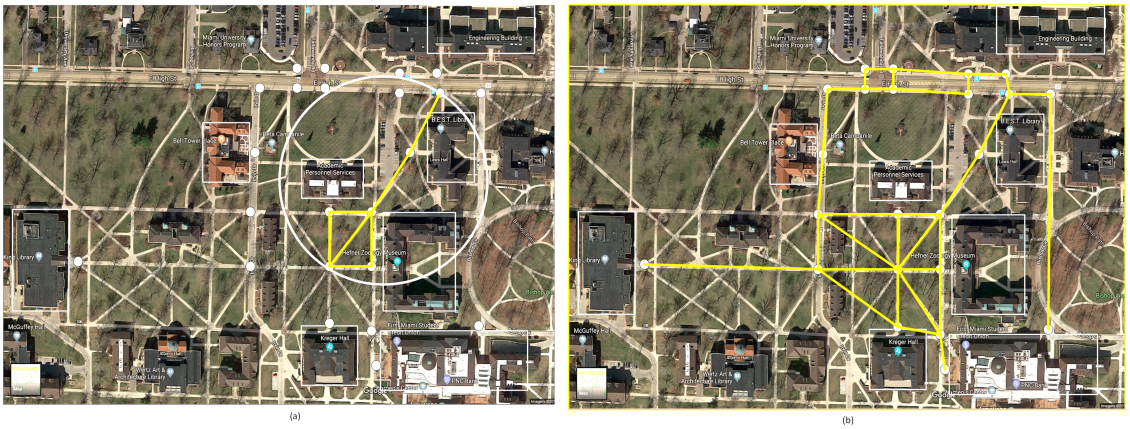

Figure 6.5: Accessible Routing Approach and Route Generation

Algorithm 6 Search Routes

```
Set Point_start as visited
routeLength = 0
possibleRoutes = Array(Point_start, Point_end)
if Point_start == Point_end then
    possibleRoutes.append(route)
    routeLength++
else
    for (int i = 0; i < length of list_Point_start conn; i++) do
        if list_Point_start conn[i] isn't null then
            if CircleCheck == True then
        Point_current = list_Point_start conn[i]
                if Point_current is not visited then Search Routes Point_current to Point_end
                end if
            end if
        end if
    end for
end if
Set Point_current as visited
```

Algorithm 7 Circle Check

Get coordinates (X_1, Y_1) from $Point_{start}$
Get coordinates (X_2, Y_2) from $Point_{end}$
Find midpoint of coordinates
Get radius of circle r
Increase range of r by a value as R
if Midpoint is within R **then**
    return true
end if
    return false

Algorithm 8 Similarity Check between Route Lengths

Obtain list of all possible routes as $possibleRoutes$
Initialize new route as $newR$
for $(every\ route \in\ possibleRoutes)$ **do**
    $lengthOfRoute, sameLength = 0$
    $currentRoute = possibleRoutes[route]$
    for $(every\ road \in\ currentRoute)$ **do**
        $currentRoad = currentRoute[road]$
        $lengthOfRoute+ = currentRoad.length$
        for $(every\ newRoad \in\ newR)$ **do**
            if $newRoad == currentRoad$ **then** $sameLength + =\ newR[newRoad].length$
            end if
        end for
    end for
    if $sameLength \div lengthOfRoute > 0.9$ **then**
        $Discard\ this\ route$
    end if
end for
$Choose\ the\ route\ with\ top\ score$

Chapter 7

Conclusion

There is a wide variety of criteria in existing literature that define accessibility. Some approach this concept more generally, as mobility assistants and aim to create a network-based accessible map that would have clear barriers and accessible points such as facilities or stairs. However, research evolved in order to recommend routes that consider specific user preferences and abilities. These crowd-sourced, accessible routing recommendation systems depend on collaborative data annotation for personalized route decisions. However, there has been no previous research on identifying surface and slopes using vibration data and it is challenging to identify inaccessible path features when this information is not readily available through official records and route generation requires a large scale repository for path features which is not readily available. There is a need for large scale data collection in order to generate accurate routes and address the issue of the lack of completeness in the knowledge base. Also, even machine learning algorithms have limitations as they are trained to understand a task in one domain. If this knowledge is to be applied for multiple mobility devices, then transferring knowledge may be necessary when there is a large decrease of information available from crowdsourcing in the new domain. The following section provides a summary of the work presented in this thesis and how we addressed this research gap.

7.1 Summary

The creation of a new accessible routing and navigation system that not only recommends routes but uses an automatic, data collection technique for tagging accessible surfaces is essential for solving the problem of the lack of completeness present in crowd-sourced data. The target goal of this accessible routing and navigation system is to model the decision making process for pedestrians and make a decision on the accessibility of a route to the user's desired location. Therefore, this system understands the mobility context of the user by learning the features extracted from surface vibrations gathered from the user's phone sensors. The identified features can then be weighted and road segments scored, so that multiple routes are generated for the user. Several learning algorithms presented in this work are able to provide a high classification accuracy for 15 different surface categories, which include Random Forest, Artificial Neural Networks, K-Nearest Neighbor and Decision Trees.

From our results, we conclude the following:

- Random Forest had the highest classification accuracy with F1 features at 94.46% and F2 at 89.12% with unsampled manual wheelchair data and S5 grouping. It also had slightly lower, but similar performance results with resampled data.

- The ANN models that are trained on manual wheelchair and power wheelchair data are able to predict the relevant surfaces for indoor navigation at above 94% accuracy. Similarly, the rough brick surface that has the lowest accessibility score in our United States data collection experiments is predicted at above 88% for both models.

- Transfer learning using zero fixed layers for training on power wheelchair data layer is the highest using S6 grouping, which is a combination of S3 and S4, and reports an average of 89.41% accuracy over 10 models or up to 90.02% accuracy.

- Transfer learning gave a statistically significant improvement over using minimal power wheelchair data for training when the source model is trained on unsampled manual wheelchair data.

- Grouping similar surfaces using S4 or S5 increases accuracy in deep learning models and does have significant improvement in manual wheelchair training overall up to 10.55% when using all of the surface features and is slightly improved up to 13.89% when using only 10 features.

- Resampling does not improve accuracy in models trained on manual wheelchair data. However, it can assist with constructing similar features that are present in manual wheelchair and power wheelchair training data.

Since the context greatly depends on the mobility device used during travel, the transfer of knowledge to a new model trained on limited power wheelchair data is also examined in this work. We realized that given the wide variety of built environments as well as natural areas, types of wheelchairs, differences in user abilities and endurance levels, and weather conditions, it is nearly impossible to test every combination of the parameters using individual experiments or to train machine learning classifiers separately every time. But, we found no existing methods can transfer their knowledge of accessibility and use data collected from one-type of sensor-augmented wheelchair for a particular environment to predict the accessibility of unknown paths while using other types of wheelchairs. Similarly, current ML algorithms work in isolation and fail to transfer learning based on cross-domain knowledge transfer. Using the selective layer training method on the the Artificial Neural Network algorithm, we achieved up to 90.02% accuracy in surface detection for the power wheelchair experiments. Based on these results, we can conclude that a subset of the feature space can be optimized for detecting surface features in the target domain, even with only 28.47% of the amount of data acquired in the source domain.

Using the quantitative analysis of accessibility criteria, the features derived from accelerometer and gyroscope sensors can not only dictate the grouping of similar surfaces, but determine their accessibility. Using the average and standard deviation from the scaled sum derived from acceleration vibration data collected from multiple phones, the road surface can be scored and used in route calculation that is suitable for the majority of the users. Based on the knowledge gathered from crowd-sourced OSM data, the ADA accessibility specifications and our routing algorithm that generates routes based on accessibility, safety and difficulty, the problem definition is addressed by our system's ability to generate the top highest weighted route from one or more starting locations.

However, these features greatly depend on phone placement in the same orientation, which may not be the case with future crowd-sensing contributions using our application. Therefore, this research is strictly limited to data that reflect the movement of the phone and wheelchair using gyroscope and accelerometer sensors.

Though this research mainly focuses on identification of specific surfaces and curbs, the framework presents a method for continuous data collection, which can aid in identifying different feature compositions in a multi-surface built environment. By understanding the user's position and direction at different locations, we are able to recognize the feature (either surface or curb), and the orientation of whether it should be approached up or down during travel. The recent research performed in SmartWheels has resulted in an urban feature hierarchy gathered from various interviews and have also utilized machine learning techniques sensor data produced by wheelchair movements for feature prediction [22]. This work can be considered complimentary to theirs as we provide an in-depth investigation into the variation of surfaces and curb direction for manual and power wheelchair accessibility. Our interest to achieve rapid prototyping of accessible routing solutions across multiple domains had led to the development of the first machine learning based, accessible routing recommendation system, WheelShare, for wheelchair users that contributes feature data to a large data bank using surface classification and knowledge transfer in order to aid further research to understand, aggregate and use this knowledge for creation of accessible routes.

7.2 Future Work

Future contributions to the WheelShare system include generating personalized, accessible routes for power wheelchair users and possibly other mobility devices, as well as additional options for route selection based on specific preference criteria. For creation of personalized routes, a future contribution to the system can place users in specific groups according to their profiles and set a minimum threshold for accessibility scores for features in the environment that are to be excluded from route generation. Since a combination of resampling and S4 grouping led to the highest classification accuracy results for both learning the source and target domain, the final objective of WheelShare is to use the models presented in this work to generate accessible routes for the users. Therefore, it is important that our path classifier is working up to the needs of the actual users before the final system testing. A new feature may give users the choice to correct the surface type if a erroneous classification is provided. In addition, a customization feature of particular routes can be useful if users prefer using a particular entrance or traveling on a specific road. We plan to improve the personalization features of the WheelShare app including additional options for route selection based on user preferences and a voice enabled navigation option. We shall also develop an iOS compatible WheelShare app for iPhone users. Once the WheelShare application is deployed on the Google Play Store, real-user testing efforts can be performed in order to increase the knowledge about the features in the built environment and to analyze the performance and usability in real world navigation. A one-way analysis of variance may be used to evaluate how the results from a user may vary among the different age groups or physical abilities. However, the variances of each of the populations would have to be equal and responses to be independent and identically distributed for the approach to be a reliable way of analysis. The continued maintenance

and development of the application may be performed by seniors participating in their capstone project.

Also, we can explore whether there are specific features that can identify user behavior in manual wheelchair data. For example, irregular, slow movements on certain curbs can indicate difficulty in travel or the decrease of wheelchair speed may indicate the user experiencing fatigue. Therefore, scores for particular curbs, road ways or sidewalks can be adjusted in order to recommend easier routes. The influence of other resampling strategies or adjustment of the K-value on Edited Nearest Neighbor technique can be examined further in order to determine if they lead to an improvement in training in the target domain. Another adjustment is to reduce the number of samples from power wheelchair to see how accuracy is affected when only a small portion of new data is contributed by the user. Where this may be useful is if a user wants to collect information through little travel in the built environment. Other similarities of surfaces in the target domain may be grouped in order to examine whether these surfaces indeed produce similar vibration for power wheelchair devices. In order to also address the limitation of phone placement on the wheelchair, the use of smartwatches and wearable technology along with machine learning techniques may aid to understand the placement of the phone on the wheelchair by understanding the user orientation. For example, if the Z-axis is positive on the smartwatch during travel and negative or zero on the phone, then the phone may be in a bag pointed straight up or laying face-down in the user's pocket. In addition, we have already begun research on standalone inertial sensors attached to the wheelchair which can provide additional magnetometer and pressure data. This may provide additional insight as to how the movement of the user changes particularly in long distance travel. We are also interested in continuous data as a wheelchair travels multiple surfaces. We may need to explore the use of geo-spatial, satellite imagery (such as Google Street view) to assist models with recognition of one or multiple surfaces.

Appendix A

Number of Data Collection Experiments

Surface Information				**Number of Experiments Per Year**			
Letter	Type	Location	Phones Involved	2018	2019	2020	**Total**
A	Brick	High St, Oxford, OH	S7, S9	2	2	0	**4**
B	Sidewalk	High St, Oxford, OH	S7, S9, J7	7	15	0	**22**
C	Smooth Brick 1	High St, Oxford, OH	S7, S9, J7	2	10	0	**12**
D	Smooth Brick 2	High St, Oxford, OH	S7, S9	2	0	0	**2**
E	Asphalt	Chestnut St, Marcum Conference, Oxford, OH	S7, S9, J7	6	15	0	**24**
F	Carpet	Benton Hall (Indoor), Oxford, OH	S7, S9, J7	0	25	0	**25**
G	Mat Surface	Benton Hall (Indoor), Oxford, OH	S7, S9, J7	0	25	0	**25**
H	Tiles	Benton Hall (Indoor), Oxford, OH	S7, S9, J7	0	26	0	**26**
I	Curb (Going Up)	Marcum Conference, E. Park Pl., Oxford, OH	S7, J7	0	129	0	**129**
J	Curb (Going Down)	Marcum Conference, E. Park Pl., Oxford, OH	S7, J7	0	131	0	**131**
K	Rough Brick	High St, Oxford, OH	S7, J7	0	10	0	**10**
L	Brick	Gusu, China	iPhone X	0	0	6	**6**
M	Granite Tile	Lu Shan Lu, China	iPhone X	0	0	6	**6**
N	Asphalt Sidewalk	Lu Shan Lu, China	iPhone X	0	0	6	**6**
O	Patterned Flagstone Road	Gusu, China	iPhone X	0	0	5	**5**

Table A.1: Manual Wheelchair Data Collection Experiments

The total number of experiments involving data collection of vibration data on phone sensors attached to the manual wheelchair is shown in Table A.1. Each experiment includes one or more phone sensors attached to the wheelchair.

Surface Information				Number of Experiments in 2020
Letter	Type	Location(s)	Phones Involved	**Total**
B	Sidewalk	High St.	S7, J7	**6**
C	Smooth Brick 1	High St.	S7, J7	**6**
E	Asphalt	Chestnut St., Marcum Parking Lot	S7, J7	**6**
F	Carpet	Inside Benton Hall	S7, J7	**7**
G	Mat Surface	Inside Benton Hall	S7, J7	**7**
H	Tiles	Inside Benton Hall	S7, J7	**7**
I	Curb (Going Up)	Marcum Conference, E Park Pl.	S7, J7	**30**
J	Curb (Going Down)	Marcum Conference, E Park Pl.	S7, J7	**28**
K	Rough Brick	High St.	S7, J7	**6**

Table A.2: Power Wheelchair Data Collection Experiments

Table A.2 shows the number of experiments performed using the power wheelchair device.

Appendix B

Surface Vibrations Collected By Phone Sensors

All of the sensor data shown below uses the Samsung Galaxy S7 edge. One experiment is shown per surface type.

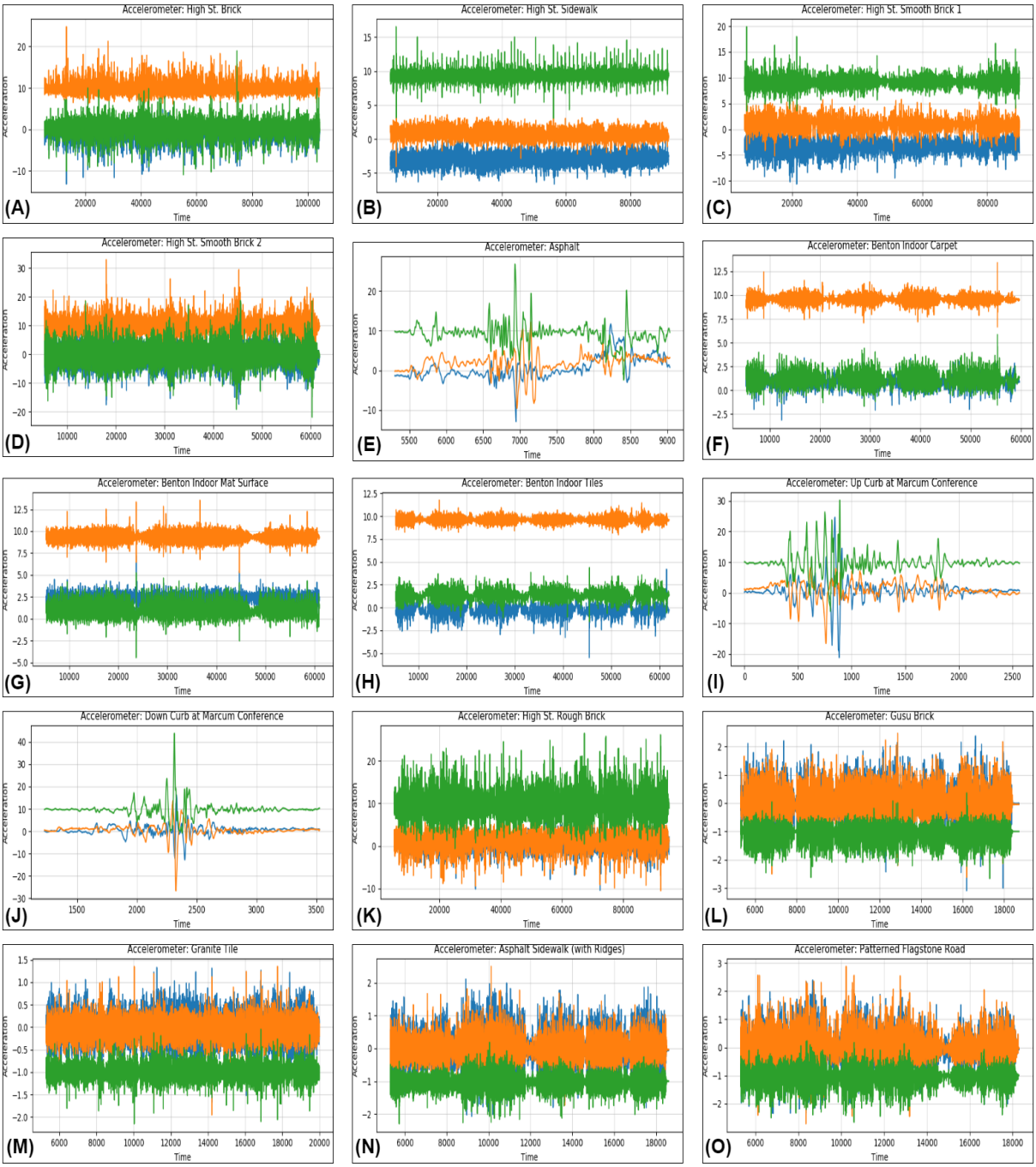

Figure B.1: Accelerometer X-Value (blue), Y-Value (orange), Z-Value (green) for the 15 Surface Categories Collected By Manual Wheelchair Experiments

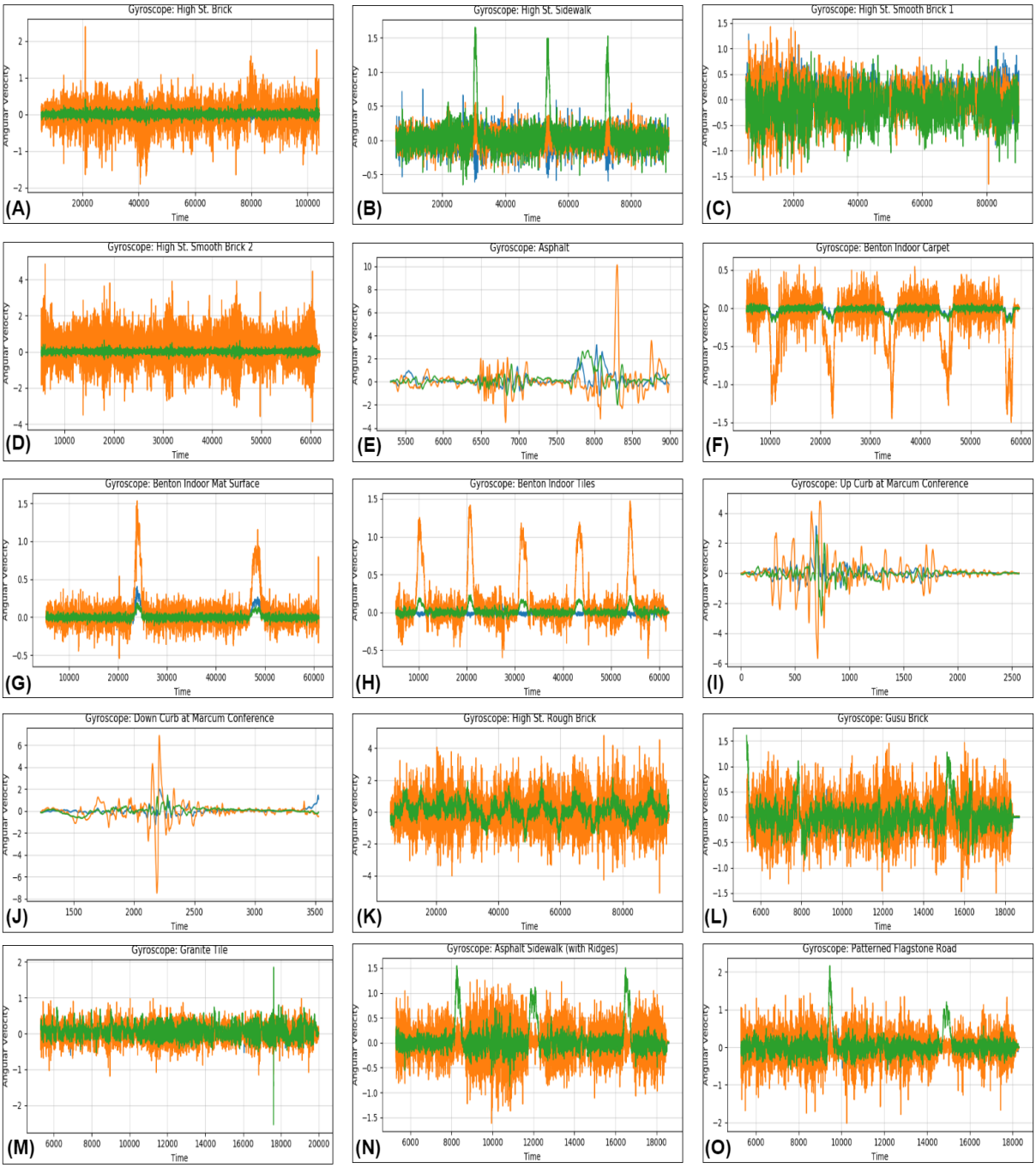

Figure B.2: Gyroscope X-Value (blue), Y-Value (orange), Z-Value (green) for the 15 Surface Categories Collected By Manual Wheelchair Experiments

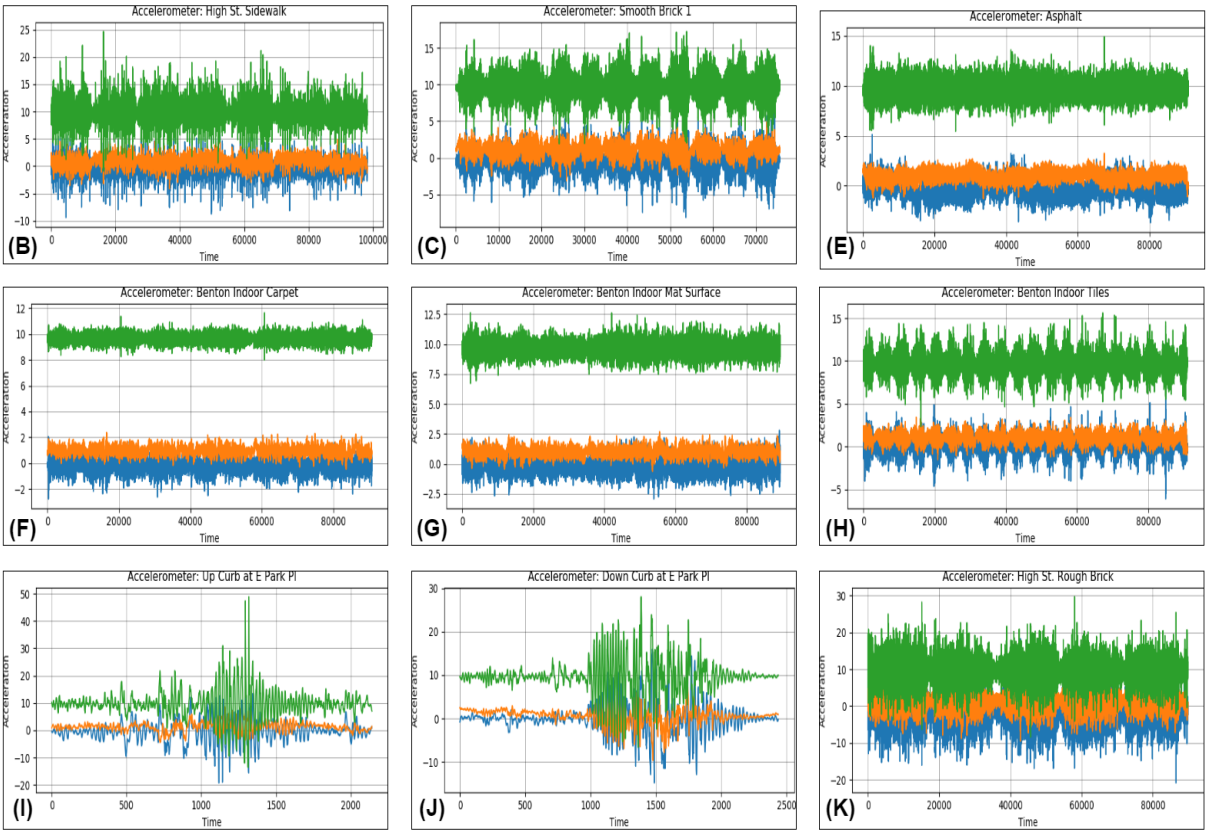

Figure B.3: Accelerometer X-Value (blue), Y-Value (orange), Z-Value (green) for the 9 Surface Categories Collected By Power Wheelchair Experiments

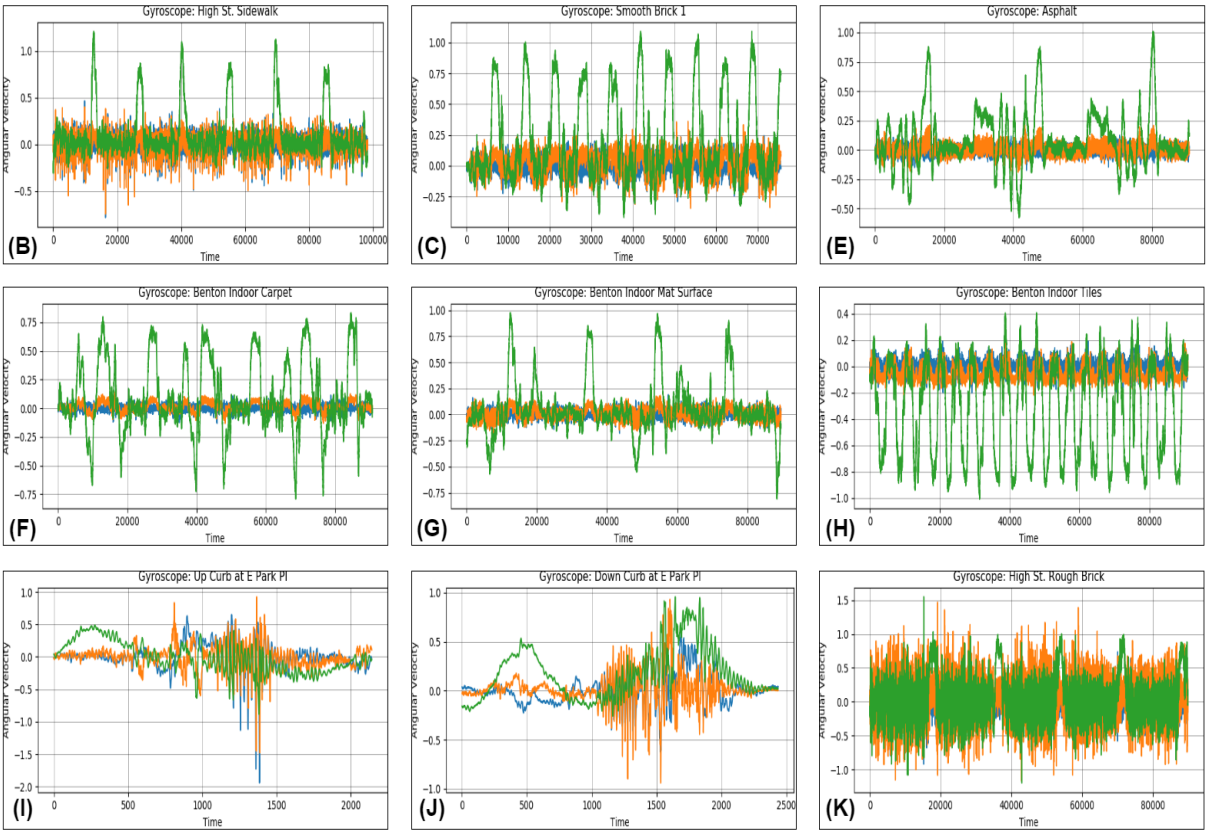

Figure B.4: Gyroscope X-Value (blue), Y-Value (orange), Z-Value (green) for the 9 Surface Categories Collected By Power Wheelchair Experiments

References

[1] United States Census Bureau. American community survey, 2018.

[2] Brad E Dicianno, James Joseph, Stacy Eckstein, Christina K Zigler, Eleanor Quinby, Mark R Schmeler, Richard M Schein, Jon Pearlman, and Rory A Cooper. The Voice of the Consumer: A Survey of Veterans and Other Users of Assistive Technology. *Military Medicine*, 183(11-12):e518–e525, 04 2018.

[3] Nhats public use data (round 8). "`www.nhats.org`", 2018.

[4] Janick Edinger, Alexandra Hofmann, Anton Wachner, Christian Becker, Vaskar Raychoudhury, and Christian Krupit. Wheelshare: Crowd-sensed surface classification for accessible routing. 2019.

[5] Ming Ren and Hassan A Karimi. A fuzzy logic map matching for wheelchair navigation. *GPS solutions*, 16(3):273–282, 2012.

[6] S. Mirri, C. Prandi, and P. Salomoni. Personalizing pedestrian accessible way-finding with mpass. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1119–1124, Jan 2016.

[7] Mohammad Saiedur Rahaman. *Context-aware Mobility Analytics and Trip Planning*. PhD thesis, RMIT University Melbourne, 2018.

[8] Erik Wästlund, Kay Sponseller, Ola Pettersson, and Anders Bared. Evaluating gaze-driven power wheelchair with navigation support for persons with disabilities. *Journal of Rehabilitation Research & Development*, 52(7), 2015.

[9] Taşkın Padır. Towards personalized smart wheelchairs: Lessons learned from discovery interviews. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5016–5019. IEEE, 2015.

[10] Brandon Daveler, Benjamin Salatin, Garrett G Grindle, Jorge Candiotti, Hongwu Wang, and Rory A Cooper. Participatory design and validation of mobility enhancement robotic wheelchair. *Journal of Rehabilitation Research & Development*, 52(6), 2015.

[11] Garrett G Grindle, Hongwu Wang, Hervens Jeannis, Emily Teodorski, and Rory A Cooper. Design and user evaluation of a wheelchair mounted robotic assisted transfer device. *BioMed research international*, 2015, 2015.

[12] Elizabeth A Madigan and Wyatt S Newman. What do users want from "smart" wheelchairs? In *NI 2012: 11th International Congress on Nursing Informatics, June 23-27, 2012, Montreal, Canada.*, volume 2012. American Medical Informatics Association, 2012.

[13] ADA. Americans with Disabilities Act (ADA) Standards for Accessible Design.

[14] Dan Ding, Bambang Parmanto, Hassan A Karimi, Duangduen Roongpiboonsopit, Gede Pramana, Thomas Conahan, and Piyawan Kasemsuppakorn. Design considerations for a personalized wheelchair navigation system. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 4790–4793. IEEE, 2007.

[15] Thorsten Völkel, Romina Kühn, and Gerhard Weber. Mobility impaired pedestrians are not cars: Requirements for the annotation of geographical data. In *International Conference on Computers for Handicapped Persons*, pages 1085–1092. Springer, 2008.

[16] Franco Zambonelli. Pervasive urban crowdsourcing: Visions and challenges. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 578–583. IEEE, 2011.

[17] Nicola Bicocchi, Alket Cecaj, Damiano Fontana, Marco Mamei, Andrea Sassi, and Franco Zambonelli. Collective awareness for human-ict collaboration in smart cities. In *22nd IEEE International WETICE Conference (WETICE 2013)*, pages 3–8. IEEE, 2013.

[18] Armir Bujari, Bogdan Licar, and Claudio E Palazzi. Movement pattern recognition through smartphone's accelerometer. In *Consumer communications and networking conference (CCNC), 2012 IEEE*, pages 502–506. IEEE, 2012.

[19] Daniel Sinkonde, Leonard Mselle, Nima Shidende, Sara Comai, and Matteo Matteucci. Developing an intelligent postgis database to support accessibility tools for urban pedestrians. *Urban Science*, 2(3):52, 2018.

[20] R. Harle. A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys Tutorials*, 15(3):1281–1293, Third 2013.

[21] Valérie Renaudin and Christophe Combettes. Magnetic, acceleration fields and gyroscope quaternion (magyq)-based attitude estimation with smartphone sensors for indoor pedestrian navigation. *Sensors*, 14(12):22864–22890, 2014.

[22] Sergio Mascetti, Gabriele Civitarese, Omar El Malak, and Claudio Bettini. Smartwheels: Detecting urban features for wheelchair users' navigation. *Pervasive and Mobile Computing*, page 101115, 2020.

[23] Selwyn Goldsmith. Designing for the disabled. 1967.

[24] Department of Justice. Ada standards for accessible design. 2010.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[26] Grégoire Mesnil, Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, Ian Goodfellow, Erick Lavoie, Xavier Muller, Guillaume Desjardins, David Warde-Farley, et al. Unsupervised and transfer learning challenge: a deep learning approach. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop-Volume 27*, pages 97–111, 2011.

[27] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for face recognition with under-represented data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, 2019.

[28] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200, 2007.

[29] Toshihiro Kamishima, Masahiro Hamasaki, and Shotaro Akaho. Trbagg: A simple transfer learning method and its application to personalization in collaborative tagging. In *2009 Ninth IEEE International Conference on Data Mining*, pages 219–228. IEEE, 2009.

[30] Noam Segev, Maayan Harel, Shie Mannor, Koby Crammer, and Ran El-Yaniv. Learn on source, refine on target: A model transfer learning framework with random forests. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1811–1824, 2016.

[31] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227, 2018.

[32] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *Algorithmics of large and complex networks*, pages 117–139. Springer, 2009.

[33] Frank Schulz, Dorothea Wagner, and Karsten Weihe. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. In *International Workshop on Algorithm Engineering*, pages 110–123. Springer, 1999.

[34] Kunihiro Ishikawa, Michima Ogawa, Shigetoshi Azuma, and Tooru Ito. Map navigation software of the electro-multivision of the'91 toyoto soarer. In *Vehicle Navigation and Information Systems Conference, 1991*, volume 2, pages 463–473. IEEE, 1991.

[35] Peter Sanders and Dominik Schultes. Highway hierarchies hasten exact shortest path queries. In *European Symposium on Algorithms*, pages 568–579. Springer, 2005.

[36] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008.

[37] Daniel Delling and Dorothea Wagner. Landmark-based routing in dynamic graphs. In *International Workshop on Experimental and Efficient Algorithms*, pages 52–65. Springer, 2007.

[38] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165. Society for Industrial and Applied Mathematics, 2005.

[39] Allan R Meyers, Jennifer J Anderson, Donald R Miller, Kathy Shipp, and Helen Hoenig. Barriers, facilitators, and access for wheelchair users: sbstantive and methodologic lessons from a pilot study of environmental effects. *Social science & medicine*, 55(8):1435–1446, 2002.

[40] Masayuki Kawabata, Ryo Nishide, Mayumi Ueda, and Shinichi Ueshima. Graph-based approach to context-adaptable pns and its application scenarios. *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1250–1250, 2005.

[41] Markus Wuersch and David Caduff. Refined route instructions using topological stages of closeness. In *W2GIS*, 2005.

[42] Karen Zita, Haigh Jonathan, Richard Shewchuk, and Manuela M Veloso. Exploiting domain geometry in analogical route planning. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(4):509–541, 1997.

[43] Alexandra Hofmann. Self-adaptive cost-aware route calculation. Master's thesis, 2018.

[44] Dennis Wagner. Requirements analysis for an accessible road map. Master's thesis, 2018.

[45] Claudio E Palazzi, Lorenzo Teodori, and Marco Roccetti. Path 2.0: A participatory system for the generation of accessible routes. In *2010 IEEE International Conference on Multimedia and Expo*, pages 1707–1711. IEEE, 2010.

[46] Sketchfactor. 2014. `http://www.sketchfactor.com/`.

[47] Philip Salesses, Katja Schechtner, and César A Hidalgo. The collaborative image of the city: mapping the inequality of urban perception. *PloS one*, 8(7):e68400, 2013.

[48] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In *Proceedings of the 25th ACM conference on Hypertext and social media*, pages 116–125. ACM, 2014.

[49] Samuel Elsmore, Irwan Fario Subastian, Flora Dilys Salim, and Margaret Hamilton. Vdim: vector-based diffusion and interpolation matrix for computing region-based crowdsourced ratings: towards safe route selection for human navigation. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia*, pages 212–215. ACM, 2014.

[50] Virgilio Gilart-Iglesias, Higinio Mora Mora, Raquel Pérez-delHoyo, and Clara García-Mayor. A computational method based on radio frequency technologies for the analysis of accessibility of disabled people in sustainable cities. 2015.

[51] Carlos Cardonha, Diego Gallo, Priscilla Avegliano, Ricardo Herrmann, Fernando Koch, and Sergio Borger. A crowdsourcing platform for the construction of accessibility maps. 05 2013.

[52] Linda Beale, Kenneth Field, David Briggs, Phil Picton, and Hugh Matthews. Mapping for wheelchair users: Route navigation in urban spaces. *The Cartographic Journal*, 43(1):68–81, 2006.

[53] D Karimanzira, P Otto, and J Wernstedt. Application of machine learning methods to route planning and navigation for disabled people. pages 366–371, 02 2006.

[54] Adam D. Sobek and Harvey J. Miller. U-access: a web-based system for routing pedestrians of differing abilities. *Journal of Geographical Systems*, 8(3):269–287, Sep 2006.

[55] Harald Holone, Gunnar Misund, and Hakon Holmstedt. Users are doing it for themselves: Pedestrian navigation with user generated content. In *The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2007)*, pages 91–99. IEEE, 2007.

[56] Thorsten Volkel and Gerhard Weber. Routecheckr: Personalized multicriteria routing for mobility impaired pedestrians. pages 185–192, 01 2008.

[57] Mei-Po Kwan and Jiyeong Lee. Geovisualization of human activity patterns using 3d gis: a time-geographic approach.

[58] Shinobu Izumi, Go Kobayashi, and Takaichi Yoshida. Route navigation method for disabled access gis in consideration of abilities and psychologies. In *2007 2nd International Conference on Digital Information Management*, volume 2, pages 519–525. IEEE, 2007.

[59] Suzanne Mavoa, Karen Witten, Tim McCreanor, and David O'sullivan. Gis based destination accessibility via public transit and walking in auckland, new zealand. *Journal of Transport Geography*, 20(1):15–22, 2012.

[60] Alistair Ford, Stuart Barr, Richard Dawson, and Philip James. Transport accessibility analysis using gis: Assessing sustainable transport in london. *ISPRS International Journal of Geo-Information*, 4(1):124–149, 2015.

[61] Mohammadreza Sahelgozin, Abolghasem Sadeghi-Niaraki, and Shokouh Dareshiri. Proposing a multi-criteria path optimization method in order to provide a ubiquitous pedestrian wayfinding service. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(1):639, 2015.

[62] Takahiro Kawamura, Keisuke Umezu, and Akihiko Ohsuga. Mobile navigation system for the elderly – preliminary experiment and evaluation. In Frode Eika Sandnes, Yan Zhang, Chunming Rong, Laurence T. Yang, and Jianhua Ma, editors, *Ubiquitous Intelligence and Computing*, pages 578–590, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[63] Keisuke Umezu, Takahiro Kawamura, and Akihiko Ohsuga. Context-based barrier notification service toward outdoor support for the elderly. *CoRR*, abs/1307.3013, 2013.

[64] Kotaro Hara, Shiri Azenkot, Megan Campbell, Cynthia L Bennett, Vicki Le, Sean Pannella, Robert Moore, Kelly Minckler, Rochelle H Ng, and Jon E Froehlich. Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view: An extended analysis. *ACM Transactions on Accessible Computing (TACCESS)*, 6(2):5, 2015.

[65] Openstreetmap. `http://planet.openstreetmap.org`.

[66] Amin Mobasheri, Yeran Sun, Lukas Loos, and Ahmed Ali. Are crowdsourced datasets suitable for specialized routing services? case study of openstreetmap for routing of people with limited mobility. *Sustainability*, 9(6):997, 2017.

[67] Amin Mobasheri, Jonas Deister, and Holger Dieterich. Wheelmap: the wheelchair accessibility crowdsourcing platform. *Open Geospatial Data, Software and Standards*, 2(1):27, Nov 2017.

[68] C. Prandi, P. Salomoni, and S. Mirri. mpass: Integrating people sensing and crowdsourcing to map urban accessibility. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 591–595, Jan 2014.

[69] Laura Ferrari, Michele Berlingerio, Francesco Calabrese, and Jon Reades. Improving the accessibility of urban transportation networks for people with disabilities. *Transportation Research Part C: Emerging Technologies*, 45:27–40, 2014.

[70] Yoshie Inada, Shinobu Izumi, Motoya Koga, and Shigehito Matsubara. Development of planning support system for welfare urban design–optimal route finding for wheelchair users. *Procedia Environmental Sciences*, 22:61–69, 2014.

[71] Dženan Džafić, Pierre Schoonbrood, Dominik Franke, and Stefan Kowalewski. *eNav: A Suitable Navigation System for the Disabled*, pages 133–150. Springer International Publishing, Cham, 2017.

[72] Mohammad Saiedur Rahaman, Yi Mei, Margaret Hamilton, and Flora D Salim. Capra: A contour-based accessible path routing algorithm. *Information Sciences*, 385:157–173, 2017.

[73] P. Picton H. Matthews, L. Beale and D. Briggs. Modelling access with gis in urban systems (magus): capturing the experiences of wheelchair users. *Area*, 35(1):34–45, 2003.

[74] Hassan A Karimi, Lei Zhang, and Jessica G Benner. Personalized accessibility maps (pams) for communities with special needs. In *International Symposium on Web and Wireless Geographical Information Systems*, pages 199–213. Springer, 2013.

[75] World Health Organization et al. Global perspectives on assistive technology: proceedings of the great consultation 2019, world health organization, geneva, switzerland, 22–23 august 2019. volume 1. 2019.

[76] Esther May, Robyne Garrett, and Alison Ballantyne. Being mobile: electric mobility-scooters and their use by older people. *Ageing & Society*, 30(7):1219–1237, 2010.

[77] Mitchell P LaPlante and Disability Statistics Center. Demographics of wheeled mobility device users. In *Conference on space requirements for wheeled mobility*. Center for Inclusive Design and Environmental Access Buffalo, New York, 2003.

[78] L. Gitlow and K. Flecky. *Assistive Technologies and Environmental Interventions in Healthcare: An Integrated Approach*. Wiley Custom Select. Wiley, 2019.

[79] Phil Chen and Gianna Rodriguez. Wheelchair and power mobility. *PMR KnowledgeNow*, Oct 2013.

[80] Guto Leoni Santos, Patricia Takako Endo, Kayo Henrique de Carvalho Monteiro, Elisson da Silva Rocha, Ivanovitch Silva, and Theo Lynn. Accelerometer-based human fall detection using convolutional neural networks. *Sensors*, 19(7):1644, 2019.

[81] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[82] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 986–996. Springer, 2003.

[83] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

[84] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[85] Aritz Perez, Pedro Larranaga, and Inaki Inza. Supervised classification with conditional gaussian networks: Increasing the structure complexity from naive bayes. *International Journal of Approximate Reasoning*, 43(1):1–25, 2006.

[86] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

[87] Ivan Nunes da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, and Silas Franco dos Reis Alves. Artificial neural network architectures and training processes. In *Artificial neural networks*, pages 21–28. Springer, 2017.

[88] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[89] Guillaume Chevalier. Lstms for human activity recognition, 2016.

[90] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[91] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36, 2012.

[92] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[93] Melania Susi, Valérie Renaudin, and Gérard Lachapelle. Motion mode recognition and step detection algorithms for mobile phone users. *Sensors*, 13(2):1539–1562, 2013.

[94] Mushfiqul Alam and Jan Rohac. Adaptive data filtering of inertial sensors with variable bandwidth. *Sensors*, 15(2):3282–3298, 2015.

[95] David Titterton, John L Weston, and John Weston. *Strapdown inertial navigation technology*, volume 17. IET, 2004.

[96] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[97] R. Alejo, J. M. Sotoca, R. M. Valdovinos, and P. Toribio. Edited nearest neighbor rule for improving neural networks classifications. In Liqing Zhang, Bao-Liang Lu, and James Kwok, editors, *Advances in Neural Networks - ISNN 2010*, pages 303–310, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[98] Barrierefreies bauen - teil 1: Straßen, plätze, wege, öffentliche verkehrs- und grünanlagen sowie spielplätze; planungsgrundlagen, January 1998.