Cloud Based Smart Route System for Blinds

by

Mushfikunnabi Nijhum
17101142
Tasnim Alam
17101265

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Nijhum

Mushfikunnabi Nijhum
17101142

Tasnim

Tasnim Alam
17101265

Approval

The thesis/project titled "Cloud Based Smart Route System for Blinds" submitted by

1. Mushfikunnabi Nijhum (17101142)
2. Tasnim Alam (17101265)

Of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 6, 2021.

Examining Committee:

Supervisor:
(Member)

[illegible]

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

[illegible]
8.6.21

Arif Shakil
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

[illegible]

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

SHKazi

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Safe and independent navigation in an unfamiliar environment is a difficult task for visually impaired individuals. Existing systems are unusable because these systems are not user-friendly and also expensive. The objective of this paper is to use cloud services and path smoothing techniques to help visually impaired persons so that they can travel more efficiently and independently. For the detection of accurate and smooth paths, it is necessary to collect some path data and then compare them with one another using cloud services. Also, for a smoother path, a good path smoother algorithm is needed. For our model, we have collected path data using app called "MapMyRun". In this paper, we are going to introduce a model for blind navigation where we used our own algorithm for path smoothing and obstacle detection.

Keywords: Visually impaired, Path Smoothing, GPS, Obstacle detection, Smartphone, Cloud

Dedication

This thesis is dedicated to our beloved parents. We are very much thankful to them. Without their cooperation and mental support, we would not be able to come to this far. Thanks to them.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, our supervisor, Dr. Sadia Hamid Kazi madam for letting us do this thesis under her supervison. Thirdly, to our co-supervisor Arif Shakil sir for his continuous advice and support in our work. We found him beside us whenever we needed help. Fourthly, our friend Fatema Ahsan Meem and Mahmudul Haque for their help. We used their smartphones to collect data.
And finally to our parents without their mental support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

API	Application Programming Interface
ARM	Advanced RISC Machine
AUV	Autonomous Underwater Vehicle
$cNAS$	Cloud Navigation and Awareness Server
CPU	Central Processing Unit
$EC2$	Elastic Compute
GCE	Google Compute Engine
GCP	Google Cloud Platform
GP	Gaussian Process
GPS	Global Positioning System
$mNAS$	Mobile Navigation and Awareness Server
$NMEA$	National Marine Electronics Association
OCR	Optical Character Recognition
OST	Original Sound Track
PDF	Portable Document Format
QR	Quick Response
RAM	Random Access Memory
$RANSAC$	Random Sample Consensus
RF	Radio Frequency
$RFID$	Radio Frequency Identification
RRT	Rapid Response Team
SD	Secure Digital

$SLAM$	Simultaneous Localization and Mapping
TTS	Text To Speech
USB	Universal Serial Bus
VM	Virtual Machine
$Wi-Fi$	Wireless Fidelity

Chapter 1

Introduction

1.1 Motivation

According to the World Health Organization (WHO), the total number of visually impaired people in the world is 285 million and in Bangladesh, there are 6 million blind people [1]. These numbers are not constant and increasing day by day. Blind people cannot live a normal life like us. They face so many difficulties in their daily life to travel independently. For example, they cannot cross a road alone or cannot see the traffic lights and path holes. One of the biggest problems for visually impaired people while self-navigating is that they cannot able to detect obstacles and also they have no idea how to avoid those obstacles. Traditionally, blind people used a cane (stick) for navigation purposes and detecting obstacles but this is a very inappropriate way. They lead a very miserable life. So, this has become an important issue to help blind people to navigate independently and make their lives easier. Though there are some existing navigation systems to help visually impaired people while they are navigating but these systems are expensive and difficult to use. Moreover, in a low-income country like ours a blind person cannot afford an expensive navigation system. In this research, we are going to build a navigation system for visually impaired people which will be inexpensive, user-friendly, and flexible. Our proposed navigation system will not only guide a blind person to the right path but also suggest a perfect and smoother path. Moreover, this system will not cost so much money as it only requires a smartphone, and nowadays smartphones have become cheap, and also everyone has a smartphone with them.

1.2 Problem Statement

There are 6 million visually impaired people in Bangladesh and over 750000 people are completely blind and many more worldwide [2]. The number of people having a visual impairment is increasing year by year due to population growth and aging, behavioral and lifestyle changes, and urbanization [3]. Vision loss is the most serious disability a person can ever have. One of the biggest challenges for blind and visually impaired people is safe and efficient navigation. Therefore, they always need to depend on others for safe navigation. The reason behind visually impaired people cannot navigate independently because they fail to detect obstacles, traffic signals, and also printed materials. Blind and visually impaired people must learn how to detect outdoor obstacles and indoor stairs, also they must have knowledge about traffic signals and bus stops and most importantly about their own location. While navigating independently blind people also feel insecure. The two biggest problems blind people face while navigating independently are crossing urban pedestrians and traveling by public transport. visually impaired people depend on their past information on a situation to explore, normally finding support from control canines or the white stick, which leaves them impaired in accomplishing the wanted degree of versatility [4]. Moreover, they completely depend on other individuals to navigate in a completely unfamiliar environment. Blind people in unfamiliar situations may feel completely confused or confined [5]. These individuals can effortlessly conclusion up in dangerous circumstances as they move in obscure places [5]. There are many navigational systems developed in the last few years for visually impaired people. Lately, there has been a surprising advancement in assistive innovations helping the visually impaired and blind people in access to electronic data and communication technologies. Braille screens and per users coupled to computers, text to speech synthesizers for desktops and mobile platforms additionally various committed software applications [3]. However, Blind people discover troubles in identifying road titles, transport halt, and foot crossing [6]. To overcome this circumstance, Electronic Travel Helps has been created for outwardly impaired people [6]. This electronic framework makes a difference for the blind people while navigating and communicating [6]. But most of the systems are inaccessible and expensive for blind people. The existing system for route detection falls because of their dependence on a particular framework. Also, most of the navigation systems are developed by using high technology and are also expensive. For this reason, visually impaired people cannot use these systems because they are unfamiliar with this high technology, and also they cannot afford these expensive systems. Existing navigation systems are also unusable for many reasons like some systems cannot detect the obstacle or the type of obstacle, some are unable to give the exact location, some systems have very poor user interactive functions. For example, an ultrasonic based navigation system detects the obstacle and measures the distance between the obstacle and the user and sends this information to the user through audio but the problem of this system is it can not detect the exact location and also cannot identify the type of obstacle [1]. Moreover, there is some navigation system developed by using RFID tags to label the place's name and RFID readers. However, the problem of this system is that it is almost impossible to place an RFID tag on every street or road because it is expensive [7]. Most of the navigation systems are GPS based because GPS is easily available in today's smartphones. Despite this system's availability, it has some problems like through GPS it is hard to detect indoor location [7]. Because of these problems, we are trying to build a navigation system that will be able to assist the user with safe navigation and give the user a smoother route from the user location to his destination.

1.3 Research Objectives

As we can see in the problem statement, the hardness in a blind person's life is boundless. They always need to depend on others in their daily life. They cannot move independently. They only have to rely on their past sense of an environment. Moreover, they always need a supporting hand with them. To overcome these problems, to give them independent mobility, we are proposing a cloud-based navigation system for visually impaired people. Our system can help blind people to be independent in their day-to-day life. We have tried to make a navigation model for blind people which will guide them to their destination. Our system will not only suggest direction rather the system will detect a smoother path and suggest the user accordingly. Also, our system will send a warning to blind people through a Text-To-Speech synthesizer about an obstacle when the user will be a few meters away from the obstacle to avoid any accident. By implementing this system, we are expecting a blind person will not face any problem while traveling in an outdoor environment. They will be able to move freely. Moreover, our system will be very cost-effective as the users only need a GPS-enabled smartphone with them.

1.4 Research Methodology

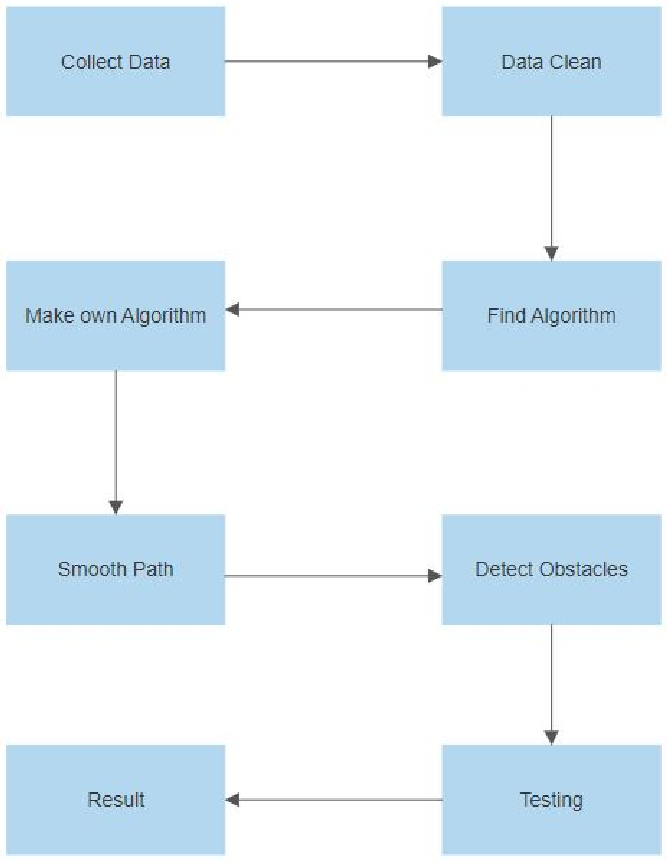


Figure 1.1: Workflow

To build the model we followed workflow shown in figure 1.1. First, we collected data. Next, we cleaned the datasets by removing unnecessary things. Later on, we started finding relevant algorithm for our model. As a result, we could not find an appropriate algorithm for the datasets we collected. Furthermore, we made our own algorithm. Then we applied path smoothing algorithm and obstacle detection algorithm on our datasets. Finally, we tested our algorithm and got the result.

1.5 Paper Organization

The rest of the paper is organized as follows:
Chapter 2 contains Related Works. Related Works basically describe the previous researches related to our model and its limitations.
Chapter 3 contains the description of Data Collection and Processing. Data Collection and processing gives an overview of how we have collected the dataset and how we processed those and what amount of data we got to conduct our research.
Chapter 4 contains the Model Selection part. We have explained different types of existing algorithms, our algorithm, and their problem. Also, the solution of our algorithm is described in this chapter.
Chapter 5 is all about the Results and Analysis of our model. In this chapter, we have shown the results of path smoothing and obstacle detection algorithm and also we given a brief statistical analysis of the result.
Chapter 6 is about uploading to the cloud. This chapter explains our plans related to uploading our model to the cloud server.
Chapter 7 contains the Conclusion. It gives a summary of our research and also shows the impact and contribution of our model in visually impaired people's life and also we have given a brief description of our Future plan for this model.

Chapter 2

Related Works

To help blind people, to give them a smart navigation system, many researches have been conducted so far by us. In [3], Strumillo, et al, stated a technology on outdoor mobility of the visually impaired people which will assist them in independent travel. The paper explained the concepts like cognitive mapping, wayfinding and navigation. Moreover, they said their model would help the blind people to be safe and skillful while travelling with sensory substitution methods and communicating navigational data which will allow a sighted human to make an accurate mental image of the environment. For environment sensing they used obstacle detectors, environment imagers and orientation & navigation systems. Furthermore, they mentioned an environment sensing module which works by sensing modality and that will acquire information about the environment. They also used electronic obstacle detectors to detect any kind of obstacle such as Ultra Cane, Bat “K” Sonar cane, MiniGuide and Palm sonar. In [8], Mishra, et al, stated a navigation system for blinds which will be operated by voice. They divided their model into two parts: Sensing of the immediate environment for the blind people while traveling and another one is navigating to a remote location. They used a 32-bit ARM processor mentioning it as a heart of their system, a GPS receiver to get the instant location in the form of longitude and latitude, a SD card to store the location information and voice data. They also used a joystick to select the direction and an ultrasonic sensor to detect obstacles. Moreover, they used Audio amplifiers to amplify the voice signal which is stored in the SD card and to make it hearable by using speakers or headphones. They claimed that their system can advise the user where s/he is currently located and can direct to a remote destination providing speech. They also concluded saying that their system could collect latitude and longitude perfectly. In [9], Coughlan, et al, described a pathfinding system for blinds based on a smartphone that will determine the user location. The smartphone will automatically detect color marker signs that are pre designed to be detected in real time. They also mentioned that the color marker signs can be detected in cluttered environments using computer vision software which will be running on the phone. The algorithm behind this detection is a novel segmentation algorithm that will locate the borders of the color marker in each image. This method will allow the system to calculate the distance of the marker from the phone. They worked on a different approach for wayfinding which is Piloting. In this approach they did not collect any GPS data to get the exact location rather they used Light beacons, RF beacons, RFID etc. to get a specific landmark. They claimed these landmarks will be detected by their programmed phone. They also used image processing to detect objects at distance which is preset to avoid blurry images using a theoretical expression that ensures the maximum angular speed of the mobile phone. In [10], Lapyko, et al, introduced a cloud based outdoor navigation system for blind and visually impaired people. The main goal of their system was to prepare a navigation system on which blind people can rely and they feel less stress while walking in the outdoors. For their system they used an android mobile phone, a L1 GPS receiver and cloud server. Initially they tested their system in three different environments. Those are: acknowledgement about traffic light, crossing road using zebra crossing and finding a bench in short distance. Through GPS cloud server will get the actual location of the user then the cloud server will notify the user whether the current navigation is safe or not. In [11], Bai, et al, proposed a system that provides a navigation system for blinds along with the capability of perceiving the outer world as much as possible to live like a normal person. In the system they stated the use of helmets molded with stereo cameras, a smartphone, web application and cloud computing. Moreover, they used deep learning algorithms to detect and recognize object, speech processing, OCR (Optical Character Recognition), Vision-based SLAM (Simultaneous Localization and Mapping) and path planning. They conducted two kinds of tests of their system, one is perception and the other one is navigation. In [12] Angin, et al, proposed a real time context aware blind navigation system using mobile cloud computing. They prepared their system for social interaction, object recognition and both outdoor and indoor navigation. They claimed that their system is context-rich, easily accessible and inexpensive. In their system they used camera, GPS & Wi-Fi location, Android mobile, Web and cloud computing resources for computation. For an outdoor navigation system, at first their system detects the user location through GPS & Google map then it captures an image by camera integrated in the user sunglass/goggles then this data transfer to the user mobile through Bluetooth. After that a computation runs in the cloud server and then the server sends the result whether the navigation is safe or not for the user. In [13], Thomas, et al, proposed a reading assistive system for visual impaired people to get easy access to printed materials and a safe navigation system. In their system they used goggles with built-in cameras inside to capture the image of the object or printed text or the road direction text. After capturing the image, they converted it into grayscale. Later on, they converted the image into PDF format and stored it into the cloud and then they used OCR to convert the text into grayscale. At last they converted the text into and they used the Android Phone's local TTS (Text to Speech) engine to read it aloud to the user. They claimed that their system is more accessible because they used Android Platform and Amazon EC2 platform as cloud components which makes their system inexpensive. In [14], Bhargava, et al, proposed a mobile-cloud based pedestrian crossing guide for blind people. In their research paper they said that they did this project because they thought that the existing guiding systems are inaccessible and insufficient for the blind people to have safe navigation. They used an android based mobile phone with integrated GPS receiver and compass and cloud server for their project. They choose android mobiles for their system because of its open architecture, accessibility and support for multitasking. At first the android application of their system determines the user's location through GPS and google map and if the application detects that user location in front of any outdoor intersection then it opens the camera of the mobile device and takes the picture and sends the picture to the cloud server. After that the server runs image processing by pedestrian algorithm then sends the result to the user. In [6], Tamizhselvi, et al, presented a prototype system of mobile public transportation assistance for visually impaired people. They introduced cloud computing in their system because mobile phones have less resources like low memory and low computational power. They used cloud computing for unlimited storage and rich resources for computations. In their system the visually impaired people first need to send the destination address to the cloud server & by the GPS of the mobile phone of the user the cloud server will locate the visually impaired people. Cloud server uses this information and its resources about transport number, road number etc. to calculate the destination and the transport information. Then the cloud server sends this information to the user. Then they used a Text to speech synthesizer tool to convert it into audio and this audio is given to the visually impaired person as an output. In [4], Angin, et al, proposed a model that would detect traffic lights for blind navigation. They proposed a mobile-cloud collaborative system for context-aware navigation where they expected to use the computational power of resources from cloud computing providers. Also, they used the wealth of location-specific resources which were available on the internet to maximize context-awareness. Primarily they just made a traffic light detector which is easy to use, portable and affordable and they claimed that their model is extensible in many ways. Their main two components were "Mobile Navigation and Awareness Server (mNAS)" and "Cloud Navigation and Awareness Server (cNAS)". The first one is any kind of smartphone device and the second one is Web Service Platform. For local navigation, local obstacle detection and avoidance they used mNAS which will interact with both the user and cloud server by providing location data to cNAS. For better accuracy they used GPS signal and Wi-Fi based location tracking. Moreover, a compass was also used in the smartphone to detect the direction that the user was facing. They used the mNAS as a server which would perform multi-tasks like guidance, warnings, and textual information. Furthermore, they stated that the mobile component will provide specific location through sensors e.g. camera and detect the traffic lights and send back to the cloud server. They also mentioned AdaBoost Machine Learning algorithm for real time object recognition. In [5], Abdullah, et al, introduced path planning and obstacle avoidance for blind people. In their system they tried to find the smoother path and obstacle free path for visually impaired people. Therefore they used A* algorithm to determine the smoother path. First of all, their navigation system takes the location of the user and then find the best smoother path from user's location to the destination by using A* algorithm.

[15], Yazici, et al, proposed a smooth path planning system for an autonomous mobile robot and that is compared to path smoothing. They introduced a concurrent path planning and smoothing approach with kinematic constraints that would reduce the waste of much power and time while the robot takes any sharp turns. For this they used A* algorithm to plan path over a directed graph. Their claim was that this algorithm would find the least cost path from start to finish. For a smoother path, they used an arc-line approach considering a sample piece of path and a mobile robot of least turning radius of 100 mm. Moreover, their approach was able to find least cost paths compared to path smoothing where a pre-planned path was chosen and then smoothed. In [16], Lijun, et al, proposed a routing optimiza-

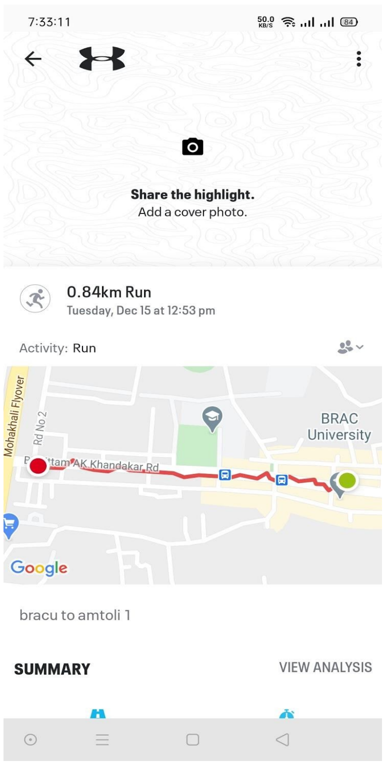
tion method using the Smooth-RRT algorithm for AUV. This algorithm is able to get a planning path without environment construction but it needs to select random points of the tree. Moreover, to improve angle factor and to find out the best practical route they used maximum rotation angle as the constraint by generating random points and selecting it as a target. Furthermore, they mentioned a greedy algorithm which they used to smooth the planning path. This algorithm will get the optimum solution by starting from the initial stage of the problem through a series of greedy choices. Their approached algorithm had higher efficiency. It could remove unnecessary points in the path. In [17], Islam, et al, presented a system in which they tried to build a system that can detect the holes of a path and give notification to visually impaired people. Blind faces many difficulties while navigating in the outdoors and indoor. Path holes are a big issue for blind people while they are walking so in this paper they tried to make a system that can alert blind people about the path holes while they are walking. Moreover, they claimed that the existing navigation system for blind people is only able to detect the obstacles in the path. However, these systems are not able to detect the holes of the path and that is the reason why they built this path hole detection project to give the assessment to the blind people while they are walking. In their system, they used the Convolution Neural Network algorithm to evaluate road surface area to detect the holes of the roads. Their proposed system is able to distinguish between the path which has path holes and which has not. In this paper, they said that they used two datasets: one of these has the images of roads without the path holes and another has images with path holes. Then they divided the total images into train and test sets to evaluate them. In addition, they used two convolution layers and two subsampling layers and they connected these layers with 2 neurons of the output layer where one neuron determines the path hole and another neuron determines the non-path hole of the image. In [18], ko, et al, proposed a vision-based wayfinding system for blind people, and their system will use situational awareness and activity-based instruction to assist the visually impaired people with their navigation. They developed a real-time wayfinding system by using smartphones. Besides, they actually developed a situation-based wayfinding system, and to recognize the situation and locates the environment they used smartphones. Therefore they used iphone6 as the smartphone which has an embedded camera, gyroscope, and accelerometer and it consists of five modules: circumstance mindfulness, question discovery, protest acknowledgment, activity-based instruction, and client direction recording. In their system, they refer situations to the sort of put where the visually impaired person is standing, and it is classified as an entryway, hallway, hall, or intersection. Furthermore, they used two types of QR codes in their system to show different types of information about the environment. One of the OR codes encodes location-specific data and another encodes directional information. At first, their system takes pictures of the path using the smartphone's camera and it handles the pictures gotten from the iPhone camera and after that, the proposed framework recognizes the current circumstance. At that point, it finds scene objects and recognizes their meaning so that it can direct the user along with a way to the target goal. In the interim, it calculates the user's movements utilizing the two inertial sensors and records the user's directions, which are utilized as a director for the return way. One of the good things about their proposed model is that it can able to interact with the user and the system is very user-friendly for blind people. When a user wants to interact with the user system the user can give input to the system by speech and then the system will recognize the speech and it will be activated. Then the system will start collecting the information from the sensors and is persistently handled, and the result is conveyed to the user by a Text-To-Speech interface. In [19], Nandini, et al, proposed an obstacle-free optimal path planning system based on greedy algorithms. Their system was able to minimize the number of turns along with the distance and they claimed it as their novelty. Moreover, the algorithm was developed to help a visually impaired person to walk in a straight path in any shopping mall and it could also detect obstacles of the shopping malls. The algorithm expects a single-floor grocery store format with racks on both sides of the way. In addition, the proposed calculation computes an ideal way from the user's current position to the chosen goal. In case a deterrent is recognized amid the route through the recommended way, the calculation reroutes the client to his goal through an unused ideal way. Furthermore, the input to the pathfinding calculation comprises a chart corresponding to the general store format, the user's current position, the user's chosen goal, and the list of deterrents. The output is the obstacle-free ideal way from the user's current position to his chosen goal within the general store. Their algorithm was implemented in C++. Also, they conducted their experiments in two ways, without obstacles, and with obstacles. For these two experiments, they used Dijkstra's algorithm and in both of the cases, they found the algorithm is capable of minimizing the cost and number of turns in the shopping mall. In [20], Bai, et al, stated a navigation system for blind people for an indoor environment using a novel wearable navigation device. They said that it is difficult to avoid obstacles while route following in an indoor environment and to overcome this issue they came up with a model that uses a dynamic sub-goal selecting technique to direct the clients to the goal and offer assistance to bypass impediments at the same time. In addition, the sensors implanted are of less cost, small volume, and simple integration, making it conceivable for the glasses to be broadly utilized as a wearable buyer gadget. Besides, they used visual SLAM to find visually challenged peoples accurately because it does not need any external hardware except a CAMERA. However, they used the A* algorithm for wayfinding. After detecting the globally shortest distance using the A* algorithm, their route following algorithm helps the blind people to follow their directions. The route following algorithm works in such a way that it takes the globally shortest route selecting some special nodes as sub-goal to guide the user. It actually compares the current path with its training data and gives the result. Furthermore, they used a fisheye and a depth camera, an ultrasonic rangefinder, an embedded CPU board, a pair of OST glasses, and an earphone to build their model. Here, to find a result, they also conducted their experiment into two parts, with obstacles, and without obstacles. In both of the experiments, they chose totally blind and partially sighted people to follow 3 different paths which they provided to their system and recorded their actual walking strategy. Finally getting their result they claimed their proposed model is capable of being a good product in the electronic travel aids market. In [21], Uddin, et al, presented a system that will help visually impaired people with the shortest pathfinding and detecting an obstacle. The goal of their project was to make the life of blind people easier and comfortable. Their system has two main functions one is responsible for providing direction and another is for detecting obstacles. They mainly used a smartphone and ultrasonic sensor in their system. In addition, the smartphone is used to give instructions to the user about location via voice command, and the ultrasonic sensor is used for detecting the obstacles in the road while the user is navigating. At first, A visually impaired people need to turn on his GPS on the mobile phone so that system can able to find his current location and then the user also need to give input to the system about his destination location through voice command, and then the system will start calculating the shortest path to the destination from the source. In addition, to calculate the shortest path they used the Dijkstra algorithm. They used the Dijkstra algorithm since Dijktra's calculation continuously considers all positive edges. Also, at the same time, the system starts its detection of obstacles in the shortest path using an ultrasonic sensor. Additionally, the distance from the obstacle to the user is counted by the microcontroller. If the sensor finds any obstacle then the system will send the information to the user's smartphone via Bluetooth module and convert this information into a voice message so the user can hear. In [22], Jose proposed a system that gives blind people a real-time path and also detects the obstacles. In this paper, they claimed that they tried to develop an algorithm that will improve the navigation system of visually impaired people. They said that their system not only detects the obstacles but also notify the user and also assists the user with the bypass. They used a stereo camera and netbook as devices in their system. They used these devices to take pictures of the path while the user is walking and then they used this image to evaluate the best path for the user by detecting the obstacles in the paths. To detect the path at first they detect the borders of the paths by using the Canny Edge detector and then they used a new version of Hough transformation to detect the real-time path for visually impaired people. For detecting the border they used the vanishing point which is the intersection point of the left and right border. Also, they try to detect the obstacle of the path by using three algorithms and if any two algorithms can find an obstacle is present then their system will conclude an obstacle is present there. The primary algorithm connected is based on the zero intersections of vertical and horizontal subsidiaries of the picture. The moment algorithm employs the Canny edge detector, isolating vertically and evenly situated edges to characterize a region where an obstacle may be. The third algorithm employs Laws' texture veils in order to confirm contrasts within the ground's surfaces. They claimed that their system is able to detect the obstacle within seconds if the distance between the obstacle and the user is 8 meters. In [23], Zeineldin, et al, introduced an algorithm that will help visually impaired people while navigating. They claimed that their proposed model will help the visually impaired people navigating independently and safely. Their proposed algorithm will detect the ground plane of the paths and also it will detect the obstacles of the paths. Firstly, they used a 3D sensor to collect data and the depth image will collect from this data. They used the Asus Xtion as the 3D sensor. This sensor provides the depth data to them with a lot of points. At that point, it is changed over to a 3D point cloud. After that, a preprocessing organization is connected to the point cloud where a passthrough channel and voxelization are utilized. Actually, in this step, they cut off some points. They do this step to reduce the complexity, processing time. The ground plane is at that point fragmented by RANSAC and evaluated to surface normals. They evaluate the surface normally first. To evaluate surface normal they evaluate a point from two neighborhoods. After done with the evaluation of surface normal they do the plane segmentation using RANSAC and surface normals. Finally, a post preparing arrangement is done for obstacle detection and clustering. To detect the obstacle in the path for blind people they eliminate the ground plane and then they assume the remaining objects as the obstacles. In [24], Ho, et al, introduced a smooth path planning process using the Voronoi diagram and composite Bezier curve algorithm. They constructed the Voronoi diagram as the global environment. In the Voronoi diagram of the piecewise straight rough path which keeps far from the obstacle is obtained by performing Dijkstra's shortest path algorithm. In addition, they used dynamic programming to subdivide the nodes on the linear path piecewise, To create a collision-free collision into control point subsequences The composite Bezier curve that meets the constraint of curvature Minimal route length and approaches. Moreover, their method has four steps. The primary step is to create a continuous piecewise straightway by piecewise straight lines. The nodes on the piecewise straight path are utilized as control points on the composite Bezier curve. The moment step is to evacuate the crowded control focus to advance and decrease the length. The third step is to subdivide the control points into subsequences such that each raised body of the subsequences does not collide with the obstacles and the bend length of the composite Bezier curve will approach the most limited. The final step is to create the composite Bezier curve by including extra control that focuses on the subsequences to meet the kinematic constraints. In [25], Yusof et al, proposed a path planning with a predetermined waypoints strategy utilizing Particle Swarm Optimization which computes the shortest path given the predetermined waypoints from the starting position to the last position. Moreover, the system offers to select the destination along with the path planning process. The users can select places from a given list of destinations that they want to travel to. Besides, the path planning process calculates way length between all hubs which includes foreordained waypoints, begin point and the last goal utilizing Euclidean distance formula. Furthermore, the possible shortest path will be optimized by the PSO algorithm by minimizing the total cost for the way length. The authors coded the path planning algorithm in MATLAB and in order to analyze the performance of the algorithm they run the simulation several times so that they can get the best value of the algorithm parameters. Again, The simulation investigations of the proposed strategy have shown promising output about an ideal route for distinctive goals. Finally, the authors assured that the results from this paper will be utilized to assist investigate the potential improvement of way direction framework for the outwardly disabled individuals by permitting them to travel freely in an unfamiliar environment. In [26], Priyasad, et al, proposed a point cloud-based algorithm for autonomous navigation of robots. They built their system because in some environments the existing knowledge about the environment is not helpful for the robots and in that case their system can help the robot in navigation. In addition, their proposed system will be competent in mapping the environment and autonomously investigating zones ideally by utilizing the generated map. At first, they get data from the Kinect sensor, and then they build a 3D environment using this data. Then they used this 3D map to discover the frontier cell and the shortest path. Then their algorithm determines the presence of a floor to navigate, section environment into small parts starting from the current area and after that create and extend a graph on traversable locales. The graph is backtracked to decide the way to wilderness obscure areas and robots are explored along the way. The same strategy is executed recursively until the entire region is mapped. In addition, their proposed algorithm builds a map of the found region whereas investigating and extracts details of the outline like recognize obstacles, planar surfaces, and slanted surfaces. In [27], Magid, et al, mentioned a path planning algorithm for robots that is based on splines and they claimed their detected path is capable of avoiding obstacles and also smooth and short. Moreover, they demonstrated the paths by an arrangement of splines characterized by a bit by bit expanding the number of bunches so that they can keep away from a troublesome enhancement over all the path's points. Their goal was to find a path from a starting point to an ending point and to fulfill their goal they designed the algorithm in such a way that can navigate a point robot in a planar known climate populated by fixed obstacles. The environment was filled with intersecting circles of different sizes. Furthermore, as they mentioned, their algorithm was able to work iteratively and at the beginning, it was able to guess the initial path and by each iteration, a new initial guess was made. Thus in each iteration, it was getting closer to the desired path. Besides their algorithm sometimes failed to find some predefined path from starting point to ending point. The reason behind this was Nelder-Mead optimization which always produces local solutions that are not globally optimized. Finally, they came up with a simple solution that can deal with such cases of potential function local minima. When the failure occurs, they already know the reason which is for local minima and they used to take a separate scale of the variables to rearrange the local minima. As a result, they found their tests in curved and straightforward conditions demonstrated quick union of the technique in all scenes inside the main phase of the calculation and the tests in difficult scenarios had some problem in getting stuck in local minimums and that was solved by their algorithm in the second stage. In [28], Lijun, et al, proposed a routing optimization method which is based on the Smooth-RRT algorithm. Basically, they tried to find a path planner for Autonomous Underwater Vehicles. They used greedy algorithms to smooth the planning way to fulfill the special necessities of the smallest distance and mobility of the AUV. Moreover, they claimed their method could quickly complete the path searching and improvise the effectiveness also could lessen the distance and this optimized path was appropriate for robot tracking. As they said, the RRT algorithm could get a planning path without developing the environment, all it needed to randomly pick a point of the tree which ensures the global optimality of the searching path. The algorithm has two phases which are the construction phase and the path query phase. In the beginning, it selects a random point and searches for the closest point. Thus it extends its node to reach the target point and gets a complete path. Furthermore, they said that they used greedy algorithms to find smoother paths as greedy algorithms find out the optimum solution using a series of greedy choices. It reduces the number of nodes and smoothes and shortens the taken path. In such a way, the energy of AUV can be saved, they claimed. In [29], Kotyan et al, proposed a help framework created utilizing object detection and profundity perceivement to explore an individual without dashing into an object. Their developed prototype can detect 90 several kinds of obstacles and can compute their distances from the user. Also, they made a navigation system which can take input from the user to where they want to go and can navigate the visually challenged person to that destination using Google Directions API. Moreover, they claimed their system has multiple features and has a high accuracy in navigation which can be sent within the wild and offer assistance to the visually impaired people in their standard of living by exploring them easily to their desired goal. They divided their prototype into three parts which are: input module, processing module, and output module. For the input module, they used a quantum USB Web Camera and a smartphone to collect real-time frames from the environment. Additionally, the smartphone is used to interact with the user by voice command. The person will give the destination information through the smartphone. Secondly, for the processing model, a Raspberry Pi and a smartphone is used. The raspberry pi will receive the input from the web camera which will be used to detect objects and will be sent to the smartphone. Furthermore, the authors used TensorFlow standard object detection and classification algorithms to detect the object in the frame. They calibrated the stereo camera using the Open CV algorithm and created a map using the images. Also, they build a very simple smartphone application that consists of only three buttons so that the user can easily give input. It takes voice speech as input and converts it into text and searches the location using Google API and replies back with voice to the user. Lastly, they claimed their proposed model could detect objects accurately and successfully. In [30], Stentz presented a new algorithm named D* which is capable of arranging ways in unknown, somewhat known, and changing situations in a proficient, ideal, and complete way. They introduced this algorithm for planning the navigation of the robot in an unknown environment. They named this algorithm as D* because it is the new version of the A*. They claimed that the algorithm can handle the total range of a priori map information, extending from total and precise map information to the absence of map data. D* is a very common algorithm and can be connected to issues in artificial intelligence other than robot movement arranging. In its most common shape, D* can control any path cost optimization issue where the cost parameters alter amid the look for the arrangement. D* is most proficient when these changes are identified close to the current beginning point in the look space, which is the case with a robot equipped with an onboard sensor. In [31], Moulton, et al, proposed a model which is suitable for user centered client server approach for the advancement of a talking GPS framework planning to fill a specialty for open air wayfinding. They claimed the work as a working model proof-of-concept framework that uses a speech-recognition speech-synthesis interface. Moreover, it includes a custom web application which needs Google maps API. Also, their claim was the system is expecting to be adaptable and extensible with additional highlights such as sensors for obstacle avoidance and access to web-based data such as climate, prepare or transport timetable data. Furthermore, the client server approach was found to be reasonable for the development of this sort of application. The voice recognition interface could give the current location and bearings to another area. Next it amplified the usefulness of the framework and provided SMS offer assistance and a bluetooth interface to other systems. Also, they used Franson GPS gate to take the raw GPS data from a connected bluetooth GPS device and filter it using the standard NMEA protocol to obtain latitude and longitude values. Using all of these they build a model and by using it visually impared person could hear the names of buildings, street intersections etc. In [32], Mekhalfi, et al, introduced a portable camera-based method to help blind people recognizing indoor objects. They proposed a scheme that is known as coarse description. This method points at growing the recognition assignment to numerous objects and, at the same time, keeping the preparation time beneath control by relinquishing some subtle elements. Also, they said the advantage of it is to increase the awareness and the discernment of a blind individual to his coordinate contextual environment. Moreover, there were two image multi labeling strategies to tend the coarse description issue. The primary one makes use of the Euclidean distance degree, whereas the second one depends on a semantic similarity degree modeled by the Gaussian process (GP) estimation. Furthermore, in order to attain quick computation capability, both strategies depend on a compact picture representation based on compressive detecting. The proposed technique was surveyed on two indoor datasets speaking to different indoor environments. Encouraging results were accomplished in terms of both exactness and processing time.

Chapter 3

Data Collection and Processing

3.1 Data Collection

To build our model we need path coordinates (longitude, latitude). We collected our own data using Mobile Phone and an android app named “MapMyRun”. The app gives longitude and latitude in every 1-meter distance. We started walking from point A holding the mobile phone in hand keeping the app running. While walking we had tried to avoided several obstacles by either turning to right or left. At point B we stopped. Our goal was to detect the curves while avoiding an obstacle. Moreover, we did this ten times and got ten different datasets for a particular route. We choose three different routes, are BRAC University to Amtoli Signal, Gulshan 1 Police Plaza, and Banani Signal to Banani Lake and we got a total of 30 sets of data. Next, we collected data of a single walk of each route which we used to test our algorithm. All these data will be uploaded to a central cloud platform.


7:33:11

Share the highlight.
Add a cover photo.

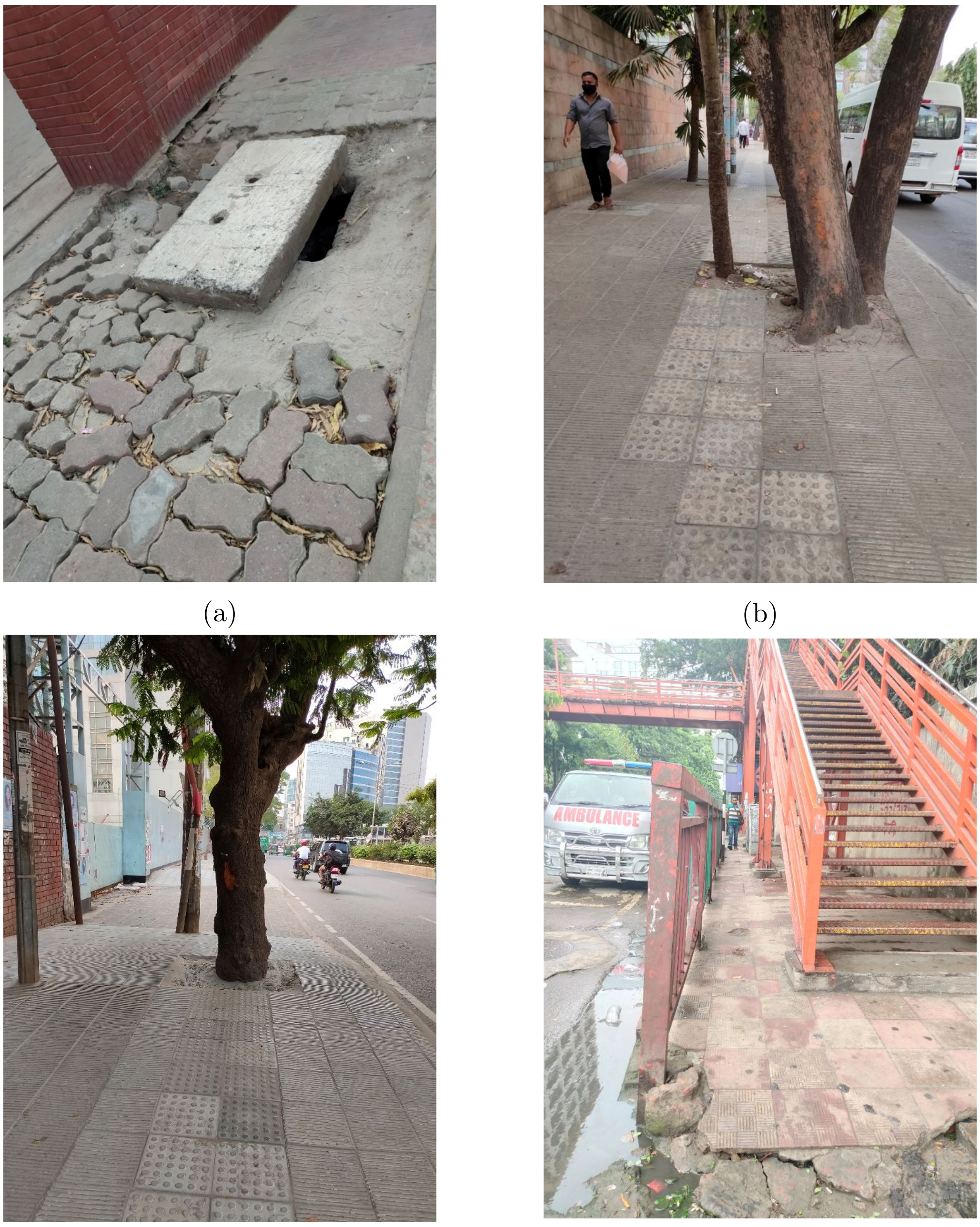
0.84km Run
Tuesday, Dec 15 at 12:53 pm

Activity: Run

bracu to amtoli 1

SUMMARY
VIEW ANALYSIS

Figure 3.1: MapMyRun App Interface



(a) (b) (c) (d)

Figure 3.2: Real-life images of Obstacles

More pictures are shared in 7.3.

3.2 Data Export

We exported the data from the app as a GPX file. A GPX file is a GPS data file recorded in the GPS Exchange format, which is a widely used open standard. It includes landmarks, paths, and tracks, as well as longitude and latitude position data. We used a Google Chrome extension named “MapMyRun GPX Exporter”. It allowed us to download a walk activity from the app. Next, from the GPX file, we exported the data to a CSV file. For exporting to a CSV file we used a python library named *gpx_csv_converted*. There we got the columns named longitude, latitude, timestamp, and elevation.

3.3 Data Cleaning

As we mentioned in Section 3.2 we have got columns such as longitude, latitude, timestamp, and elevation. For our model, we do not need timestamps and elevation. So, we dropped those 2 columns and kept only latitude and longitude. Additionally, there was no NaN value in our dataset. We combined all the ten sets of data in one CSV file. Thus we got 3 different CSV files for our three chosen routes. There were twenty columns in each CSV file. Every two columns were containing a single walk data with longitude and latitude which are our x and y. After cleaning the datasets, we got an average of 700+ rows of data per route. For a 100-meter walk, we got 100+ rows of data.

3.4 Data Type

Our data sets have only numeric and continuous data. These are unsupervised data as there are no output variables to predict. These data will be used to find relationships between data points.

3.5 Preliminary Analysis

We used the python *matplotlib.pyplot* library to plot our coordinates. Also, we have read the CSV file using the *pandas* library.

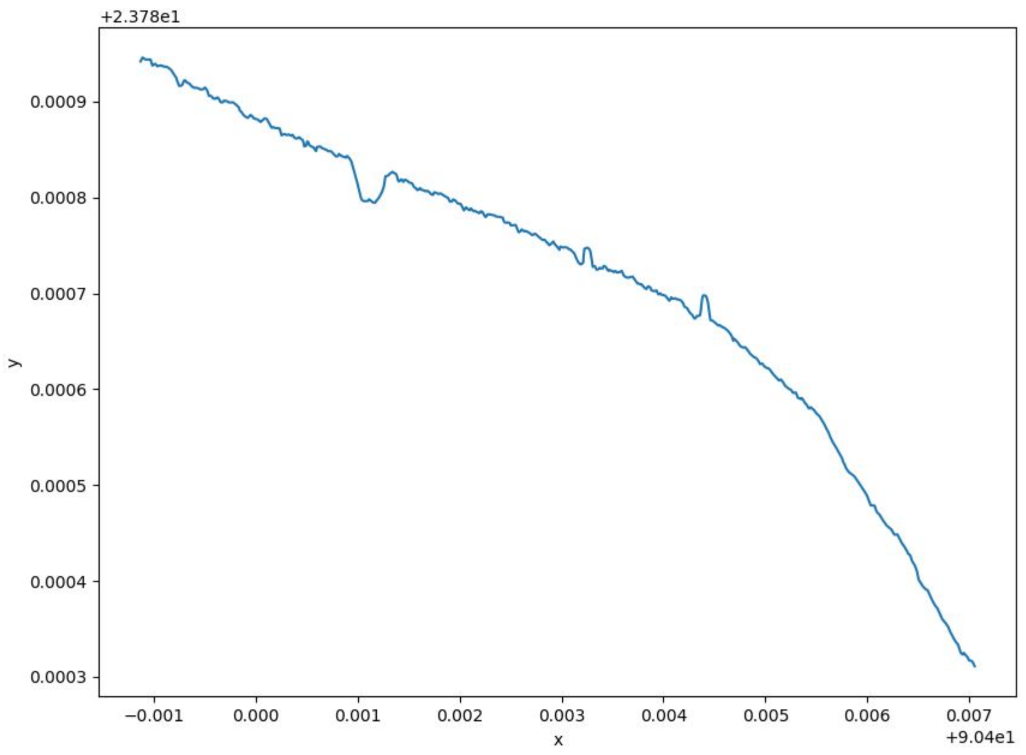


Figure 3.3: A single walk after plotting

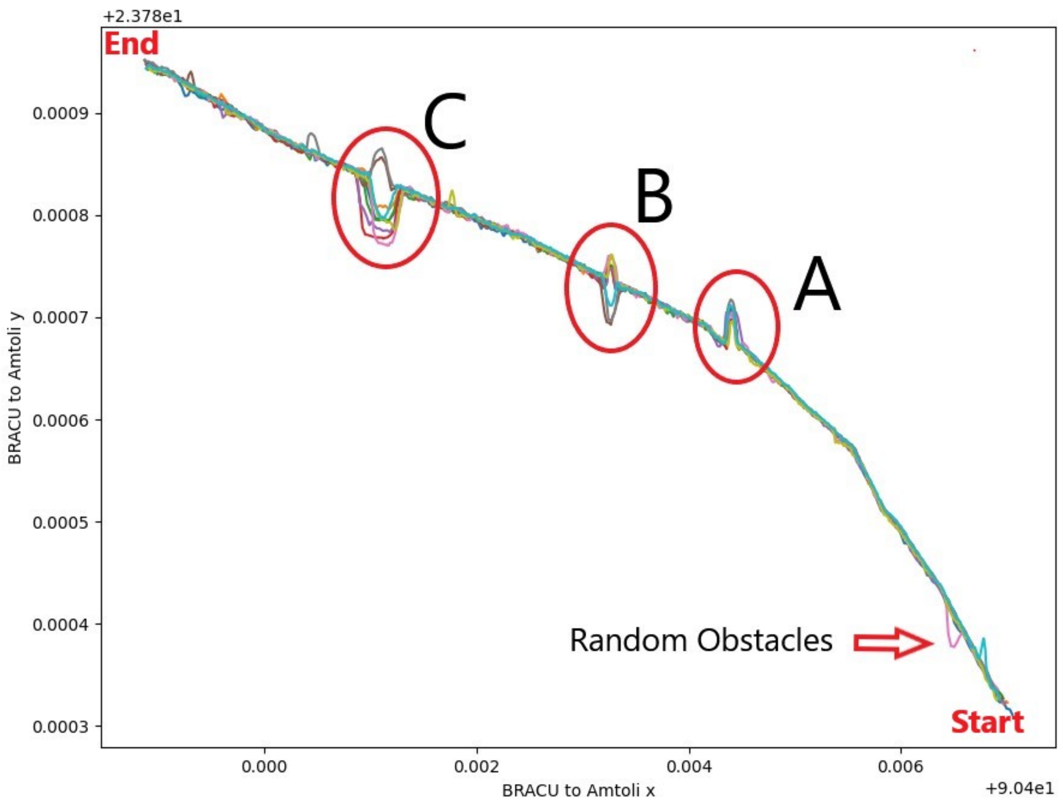


Figure 3.4: BRACU to Amtoli all Paths after Plotting

3.6 Preliminary Analysis Result

In the figure 3.4 , we plotted all the ten paths of a particular route. We can see that different paths are stacking one another and created some curves. Some curves are repeated and some are random. The curves A, B, and C repeated many times. As a result, we can say that these are our permanent obstacles on the road. The other curves are random because while walking maybe we avoided a person or something that is not permanent on the road. Lastly, we can say, we could successfully collect our desire dataset for our model.

3.7 Challenges faced while collecting data

Collecting the data was a little challenging for us. The challenges we faced are:

- Corona Pandemic
- GPS Problem

Due to the corona virus pandemic, we were not being able to go outside to collect data. There was lockdown everywhere. After some days the lockdown was withdrawn partially and we could collect only 30 sets of data. Furthermore, while collecting data, we faced GPS problem. Our mobile device's GPS was not that much accurate. For example, we started walking from point A but the GPS was showing our location in some other place. To overcome this problem we changed our device to iPhone and the problem reduced greatly. Still, there were some zig-zig in the path map but we solved the problem with our algorithm.

Chapter 4

Model Selection

4.1 Existing Algorithms

To build our model we need to do two things. First, we need to smooth our paths because there were other curves that may be considered as obstacles. So we need to remove those and keep only the original obstacles' curves. Secondly, we need to detect the obstacles and give warning to the users that there is an obstacle ahead, please turn right or left. To do so there are different algorithms for path smoothing. For example Q-Learning, Cubic Spline, Dubins Curve, A* algorithm, etc. Let's have a look at these algorithms.

4.1.1 Q-Learning Algorithms

Q-Learning Algorithm is a part of Reinforcement learning. It is about taking appropriate actions to maximize rewards in a situation. It is used to discover the best possible path or solution in a specific situation. Reinforcement learning does not have any training data set. Actually, it interacts with the environment continuously and learns from experience. Q Learning is one of the basic algorithms of reinforcement learning. In every environment, there can be many different situations and each situation is called a state. Q-Learning used Q-value or learning activities to improve the learning process. Q-value is computed using the TD-update rule [33]. A Q-learning agent starts learning from an initial state and makes a number of moves from its current state to another state based on its choice of activity. At each step of the move, the specialist from a state takes an activity, watches compensation from the environment, and after that travels to another state. One of the chances that at any point of time the agent ends up in one of the ending states which means there are no possible moves left. Typically said to be the completion of an episode. Q-Learning values are defined as Q (S, A) here S is the state and A is the action taken from state S. Q-Learning value is computed from the TD-update Equation. The TD-update equation is

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma Q\left(S', A'\right) - Q(S, A)\right) \tag{4.1}$$

Here,

S = *Current State*
A = *Current Action*
R = *Reward points*
S'= *Next State*
A'= *Next Action*
γ = *Future Reward Discounting Factor*
α = *Step length for each learning*

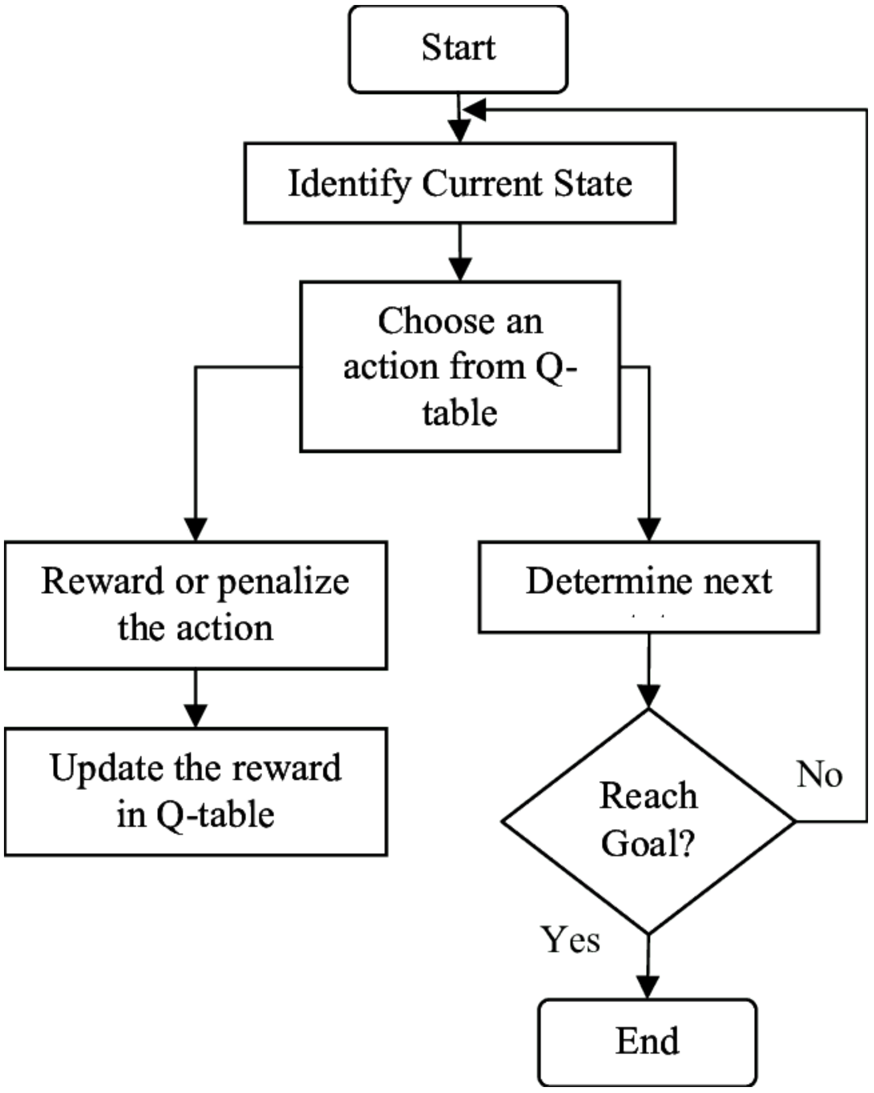
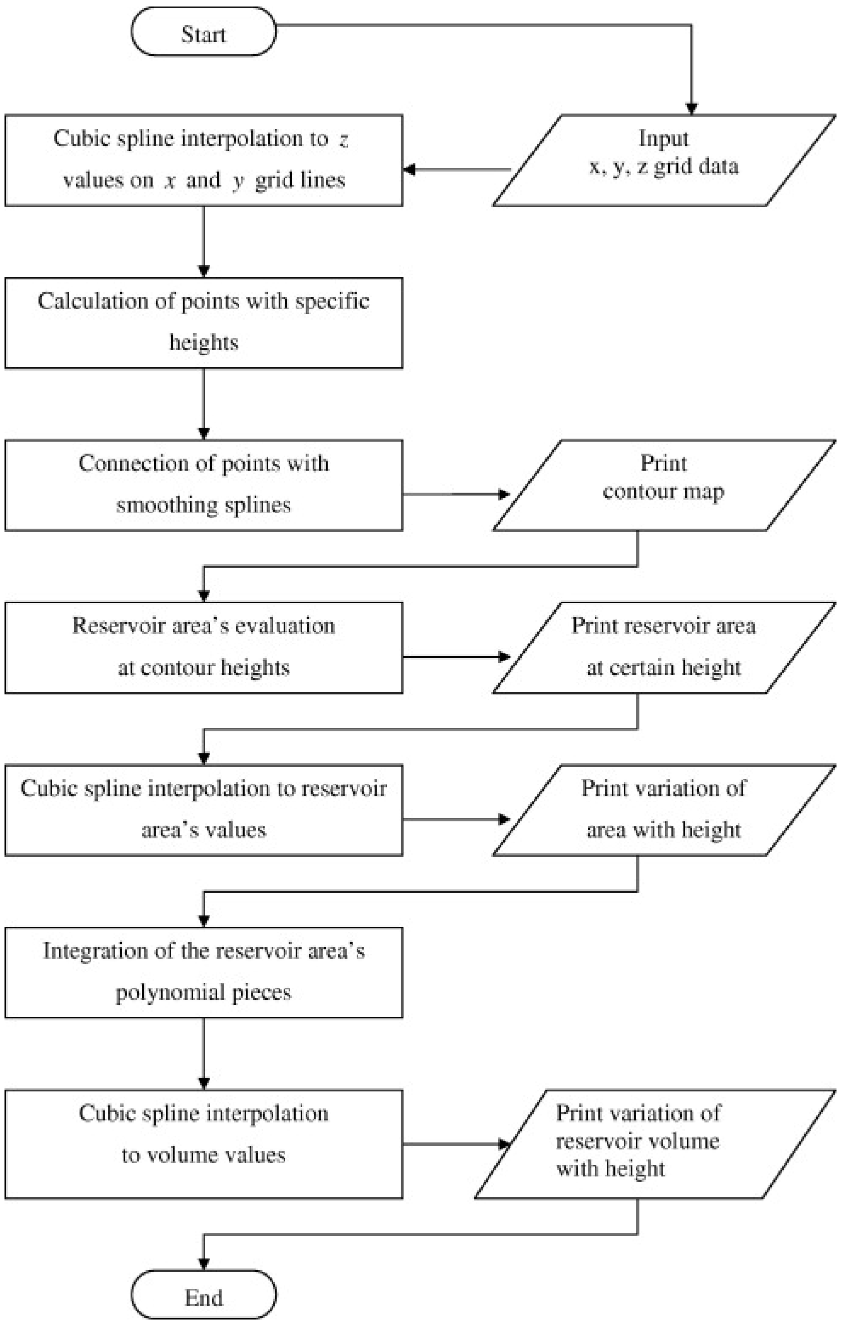

Start
Identify Current State
Choose an action from Q-table
Reward or penalize the action
Update the reward in Q-table
Determine next
Reach Goal?
No
Yes
End

Figure 4.1: Q-Learning Algorithm Flowchart

4.1.2 Cubic Spline Algorithm

Another path smoothing algorithm is cubic spline interpolation which is widely used as a path smoothing method. The spline is a piecewise polynomial function. A cubic spline is a piecewise cubic function that introduces a set of data points and ensures smoothness at the data points [34]. Cubic splines are especially significant since they are the splines of the least degree and sufficiently smooth within the presence of little curvature. Cubic spline has a low computational cost. Also, the calculation process of cubic spline is simple and it provides numerical stability. Moreover, Spline could be efficient to smooth turning curves of a path.



Start
Input
x, y, z grid data
Cubic spline interpolation to z values on x and y grid lines
Calculation of points with specific heights
Connection of points with smoothing splines
Print contour map
Reservoir area's evaluation at contour heights
Print reservoir area at certain height
Cubic spline interpolation to reservoir area's values
Print variation of area with height
Integration of the reservoir area's polynomial pieces
Cubic spline interpolation to volume values
Print variation of reservoir volume with height
End

Figure 4.2: Cubic Spline Algorithm Flowchart

4.1.3 Dubins Curve

In the Dubins curve a straight-line section and a circular arc segment are used to find the shortest smooth path for a given two-point. Dubins curves give a basic however capable method for real-time way smoothing as they are for the most part not computationally costly like Dijkstra and other algorithms [34]. Dubins curve reduces the radius of the running curve and also gives a smooth turning curve.

4.1.4 A* Algorithm

A* is one of the popular path planning algorithms. Many technologies use A* for path planning and shortest pathfinding. A* uses the path cost for computation purposes [35], [36]. When a starting node and a target node are given to A* then it computes the best shortest path based on the path cost. At each step, A* picks a node that has the lowest sum value of the path cost from the starting node to that node and the heuristic value.

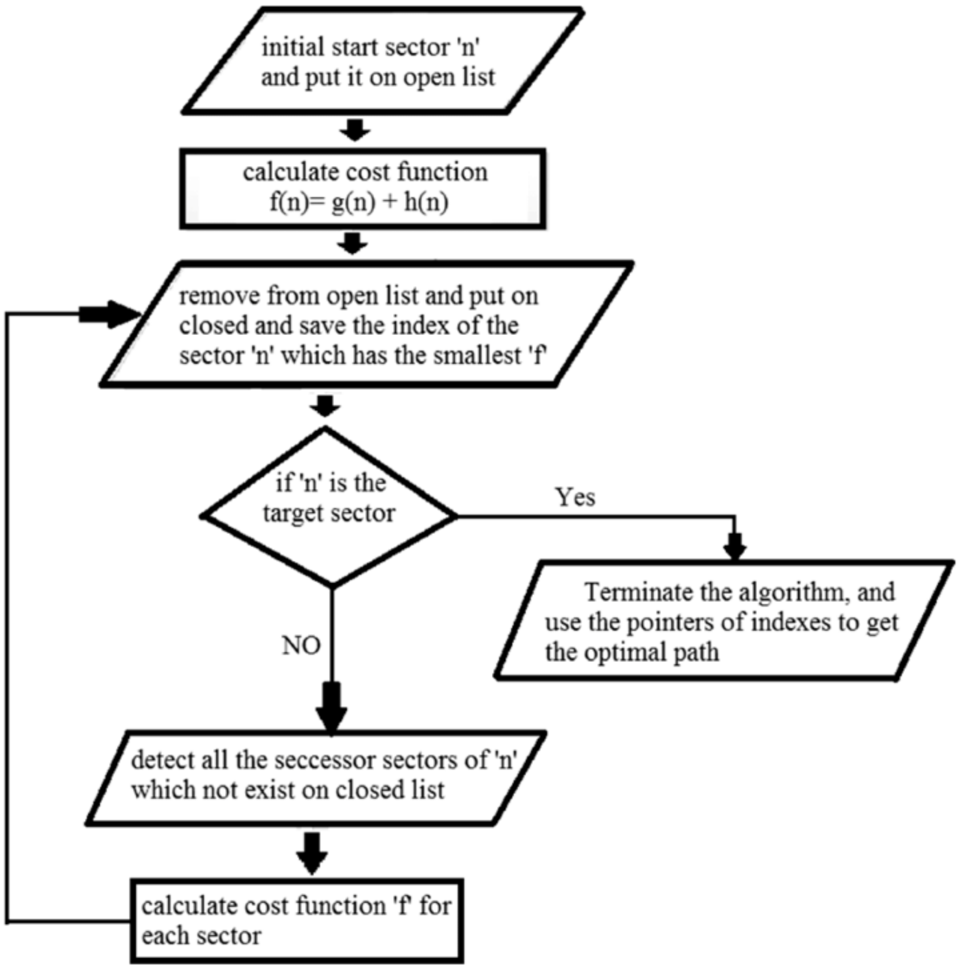

initial start sector 'n' and put it on open list
calculate cost function f(n)= g(n) + h(n)
remove from open list and put on closed and save the index of the sector 'n' which has the smallest 'f'
if 'n' is the target sector
Yes
Terminate the algorithm, and use the pointers of indexes to get the optimal path
NO
detect all the seccessor sectors of 'n' which not exist on closed list
calculate cost function 'f' for each sector

Figure 4.3: A* Algorithm Flowchart

4.2 Problems of Existing Algorithm

The algorithms mentioned in section 4.1 are used to smooth the path but we could not use those algorithms in our model. This is because these algorithms did not fit with our datasets. We needed an algorithm that can read ten different sets of data of a single route, smooths that ten sets of data, and gives us a single smoothed path as an output. In the Q-Learning algorithm, it takes a single path as an input and finds the shortest and optimal path from point A to B. Cubic Spline and Dubin's curve also takes a single dataset and smooths the path in a way that removes also our detected obstacles which we do not want. Moreover, the A* algorithm also gives an optimal path. So we could not use any of these algorithms.

4.3 Our initial approach for path smoothing and it's problem

As we could not use any of the existing algorithms, we decided to make our own algorithm. Initially, we thought we would average all the x and y values of ten datasets of a single route and the output x and y would be our new smoothed path. But there is a problem with this approach. While collecting the data we avoided some obstacle from both left and right side. Let's assume that there were five left turns and five right turns. Now if we do the average, the output will be a straight line which we do not want. We need to keep the actual obstacles and remove the random ones.

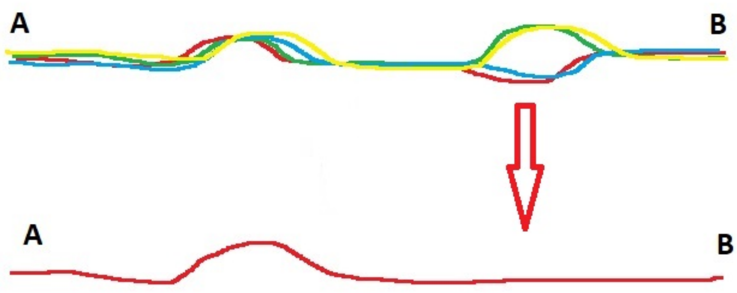


Figure 4.4: Normal Average Problem

4.4 Our final algorithm for Path Smoothing

For the problem mentioned above, we decided to fix a threshold value after analyzing the data points. There is a difference between each x-axis point. If any point difference is greater than the threshold value we have removed that particular coordinates from our datasets meaning we have dropped x and y both while doing the average.

Algorithm for path smoothing:

Algorithm 1
Pseudo code for Path Smoothing

Input: D: Raw dataset
Output: N: Single Smoothed Path (NewX,NewY)

```
 1: threshold,
 2: smoothed_x[ ]
 3: smoothed_y[ ]
 4: for each row
 5:    walkX,walkY
 6:    new_pointX[ ]
 7:    new_pointY[ ]
 8:    for (i=1 to i=10)
 9:       if difference of 2 x point is less than threshold
10:          new_pointX ← walkX(i)
11:          new_pointY ← walkY(i)
12:       end if
13:    end for
14:    smoothed_x[ ] ← (Average(new_pointX))
15:    smoothed_y[ ] ← (Average(new_pointY))
16: end for
17: export Smoothed_x, Smoothed_y to smoothed.csv
```

Here, the Algorithm 1 will read a CSV file which will have the data of our 10 walks. Each odd column will have the x and each even column will have y. Each row will contain 10 x and 10 y. Now, let's assign a threshold value, for example, $threshold = 0.00004$. If the difference between x_i and x_{i+1} is less than the threshold value the x will be stored in *new_pointX* also the y will be stored in *new_pontY*. All the other coordinates will be removed. Finally, if we do an average of these x and y we will get new x and y for each row. Thus our path will be smoothed.

4.5 Our algorithm for obstacle detection & warning

Now, our path is smoothed but we need to detect obstacles and give warning to the user accordingly. For this we created another algorithm. This algorithm will be used to identify the obstacles of the smoothed path which will be generated by Algorithm 1.

Algorithm for obstacle detection and warning:

Algorithm 2

Pseudo code for Obstacle Detection and Warning

```
Input: Smoothed Dataset and Test Data
 1: smoothed_array ← smoothed dataset
 2: test_array ← test dataset

 3: for i in range smoothed_array do
 4:     deviation_array ← smoothed_array(i) - smoothed_array(i+1)
 5: end for

 6: for i in range deviation_array do
 7:     if deviation_array[i] is larger than threshold then
 8:         obstacle_array ← smoothed_array(i)
 9:     end if
10: end for

11: for i in range test_array do
12:     for j in range obstacle_array do
13:         if test_array[i] close to obstacle_array[j]
14:             alert(Obstacle ahead)
15:         end if
16:     end for
17: end for
```

At first, we calculated the deviation of the smoothed path's y-axis points. The first for loop is running till the end of the smoothed array. Here, all the deviation values are storing in an array. Next the values are being compared with the threshold. If someone is walking straight, the y-axis will not change much. Thus the deviation will not be much from the previous point. If there is an obstacle and if anyone turns left or right, the deviation will increase from the previous point. So, we took a threshold value to compare the deviation. If the deviation is greater than the threshold, there is an obstacle at that point. So, we stored that y-axis point in another array. Next, we compared the y-axis points with our testing path data and if any point starts approaching the obstacle points it gives a warning saying whether to turn left or right.

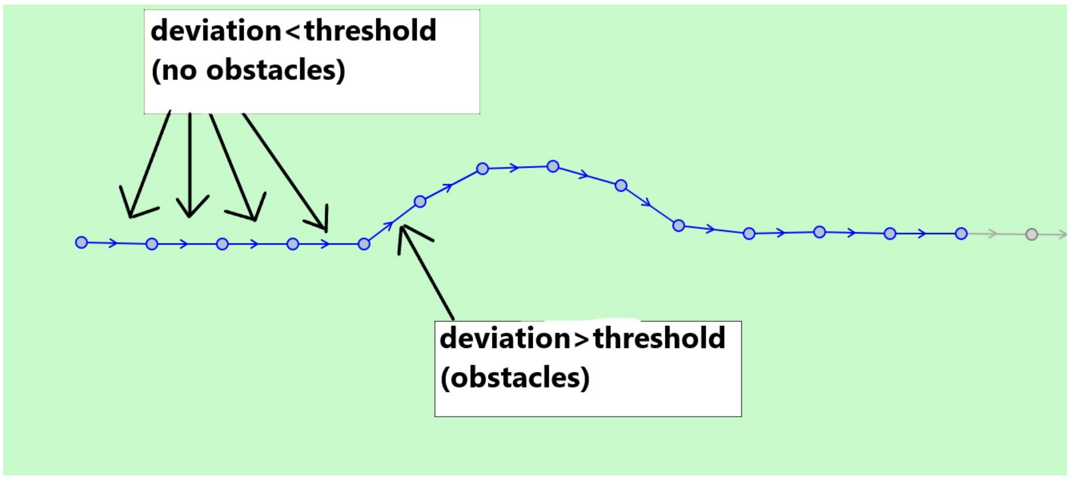


Figure 4.5: Example of detecting obstacle with deviation

These two algorithms will be applied to our datasets. First we will use Algorithm 1 on our collected datasets that we described in section 3.3. Finally the Algorithm 2 will be used to detect obstacles from the output of Algorithm 1 and to give warning comparing a new single walk data that we also discussed in section 3.1.

Chapter 5

Result and Analysis

5.1 Path Smoothing Result

As mentioned in section 4.4 we applied algorithm 1 to our collected data. The algorithm reads a CSV file that has 10 pairs of longitude and latitude. These are our 10 path coordinates of a single route. Our algorithm is able to create a smooth path. So it makes a new CSV file. The file contains the new longitude and latitude of a particular route. We did this for three routes and here are the results. The comparison before and after applying the algorithm is given below.

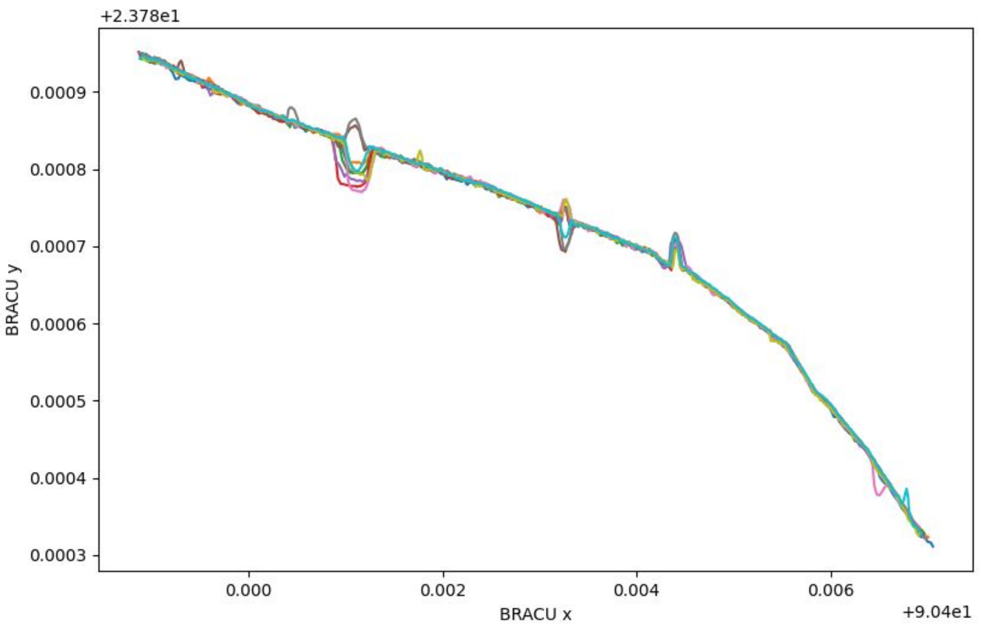


Figure 5.1: BRACU to Amtoli route before applying algorithm

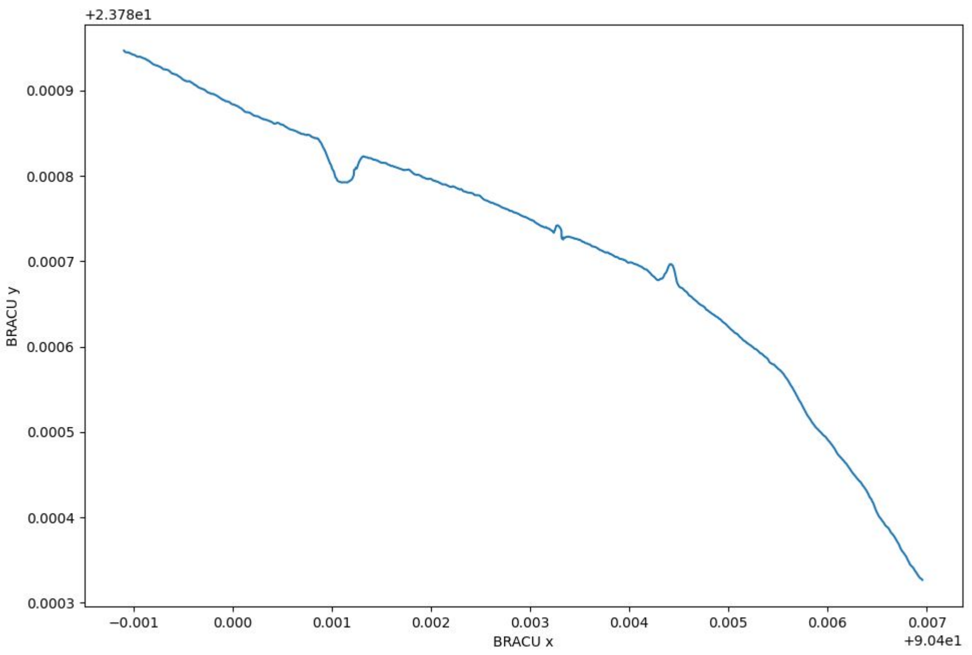


Figure 5.2: BRACU to Amtoli route after applying algorithm

In figure 5.1 we can see that except for the permanent obstacles, there are a few random curves that denote some random obstacles. Figure 5.1 also shows that two of the permanent obstacles can be avoided from the both left and right sides. Next, in figure 5.2, we can see that all the other curves and some noises are gone. Also, the problem we described in section 4.3 will be no more because in figure 5.2 we can see that, the curves have become one way. So we can say, our algorithm worked nicely on our first route. Next, we applied this algorithm in our second route that is Banani Signal to Banani Lake.

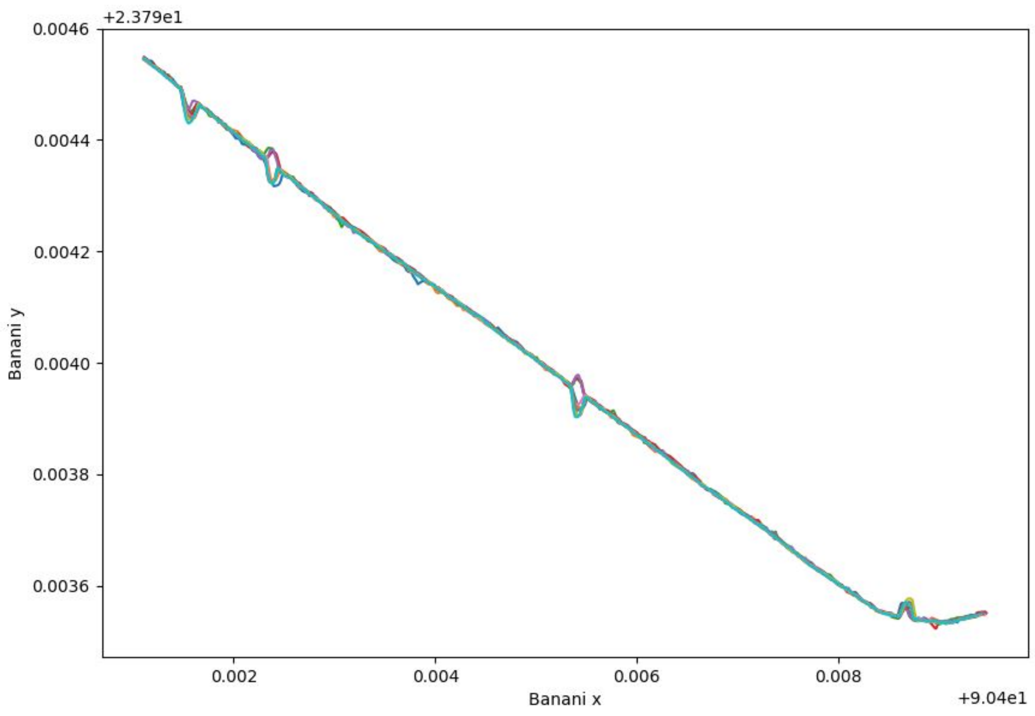


Figure 5.3: Banani Signal to Banani Lake route before applying algorithm

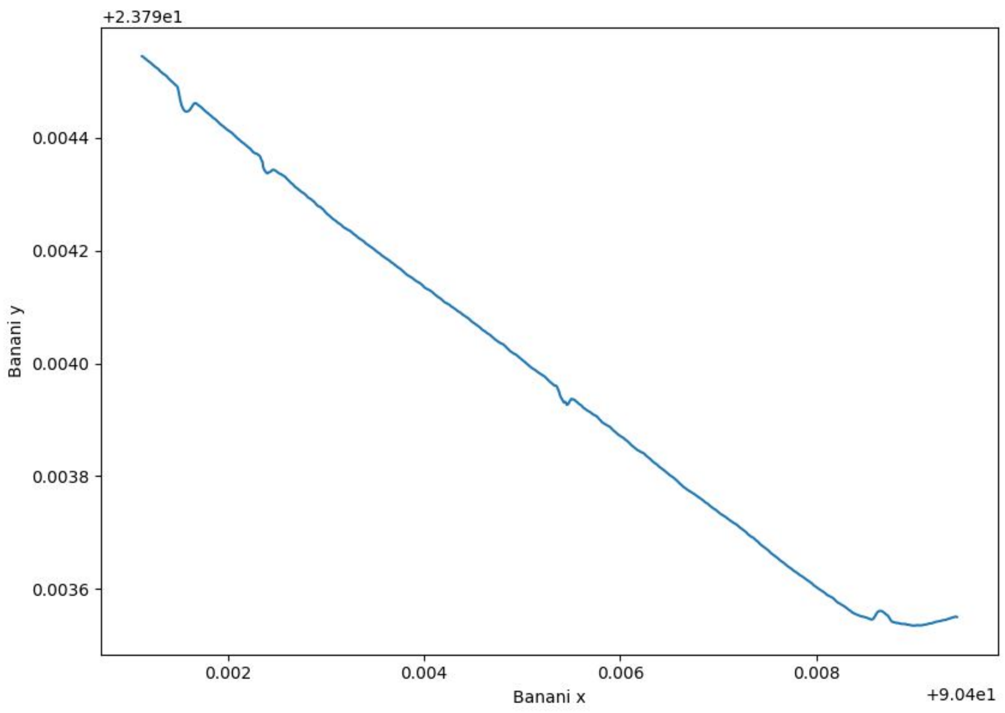


Figure 5.4: Banani Signal to Banani Lake route after applying algorithm

In figure 5.3 we can again see that a lot of noise and curves but in figure 5.2 there is none except the permanent obstacles. So our algorithm 1 aslo worked for Banani

Signal to Banani Lake route. Then, we applied Algorithm 1 on our final route that is Gulshan 1 to Police Plaza.

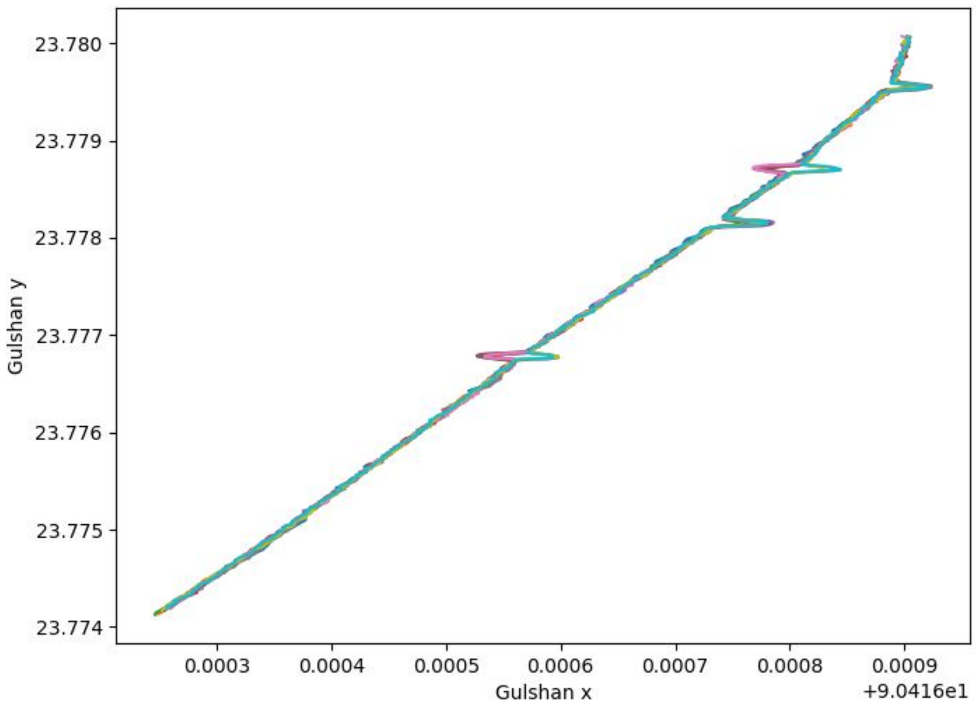


Figure 5.5: Gulshan 1 to Police Plaza route before applying algorithm

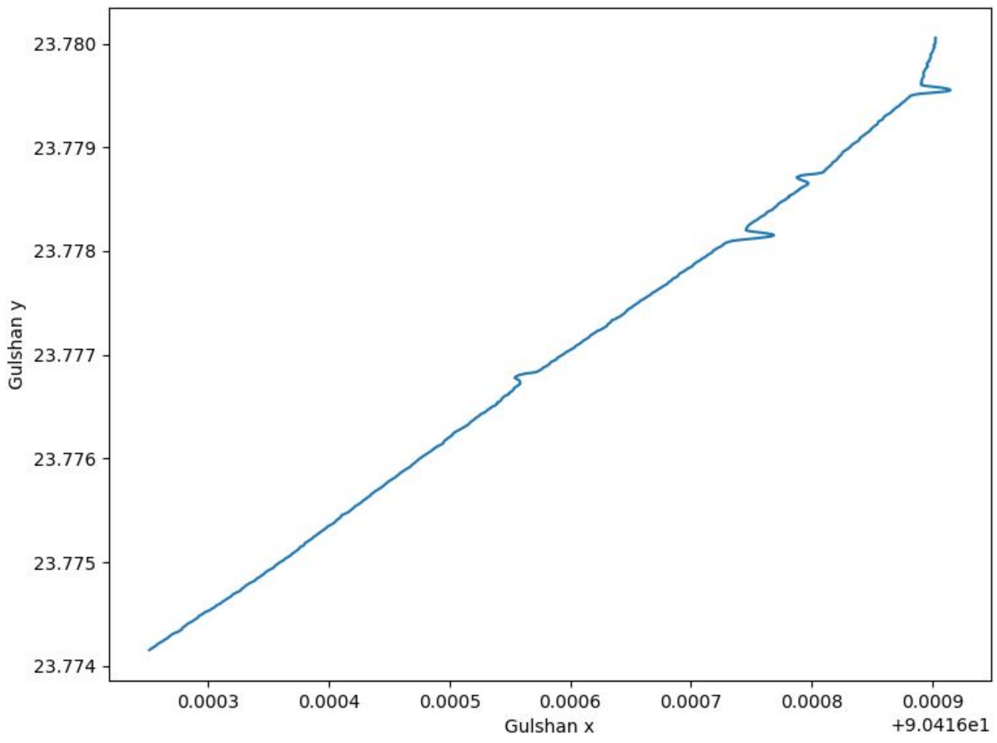


Figure 5.6: Gulshan 1 to Police Plaza route after applying algorithm

Again, in figure 5.5 there were permanent obstacles, random obstacles, and noises but in figure 5.6 those were removed by our algorithm 1 except the permanent ones.

From the figure 5.1, 5.3, 5.5, we can see, before applying the algorithm there were so many noises. In addition, there were also random obstacles in our path which we do not want to be there. After applying our new algorithm, all the other noises and false obstacles were removed which we can see in figures 5.2, 5.4, 5.6. Though there were some very small noises those can be avoided by our algorithm 2 mentioned in section 4.5. Finally, we got the original obstacles as expected. So, we can see that our algorithm 1 worked pretty well for our three selected routes that we described in section 3.1.

5.2 Obstacle detection and warning result

5.2.1 Generating Deviation

Now we need to detect the obstacles and give warning to the users. To do so, we need two paths: the smoothed paths that we got by applying algorithm 1 and a tester path that we mentioned in section 3.1. First, we read the smoothed path and the testing path in Algorithm 2. The algorithm first calculated the deviation of all y-axis of the smoothed path and stored it in an array.

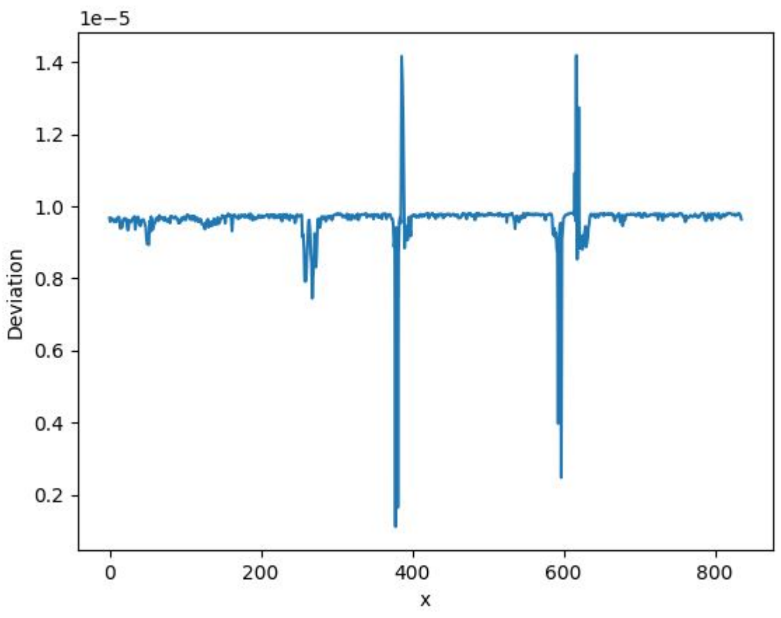


Figure 5.7: BRACU to Amtoli Smoothed Path's Deviation Spike

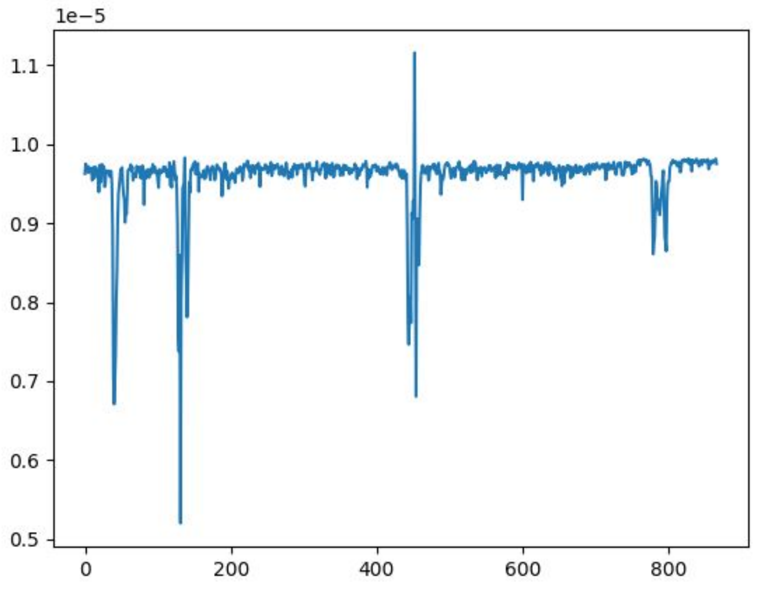


Figure 5.8: Banani Signal to Banani Lake Smoothed Path's Deviation Spike

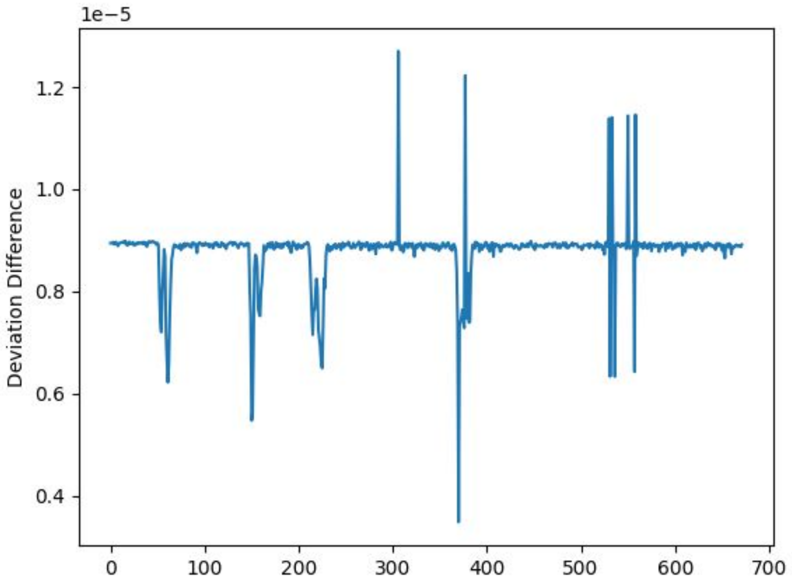
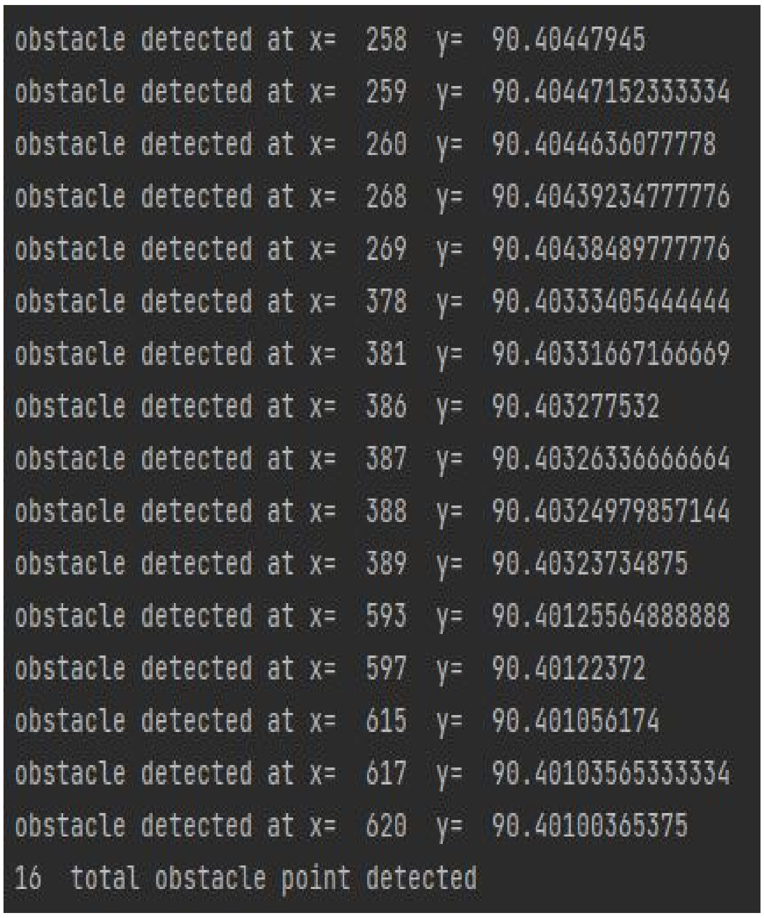


Figure 5.9: Gulshan 1 to Police Plaza Smoothed Path's Deviation Spike

Here, we plotted all the deviation of the smoothed path that we can see in figures 5.7, 5.8, 5.9. In those figures, x-axis is the row number of the smooth paths datasets and the y-axis is the calculated deviation. When we turn left or right, the difference between the two y-axis increases which we have shown in figure 4.5 and creates spikes. These spikes will be used in the next part of our algorithm.

5.2.2 Detecting Obstacles

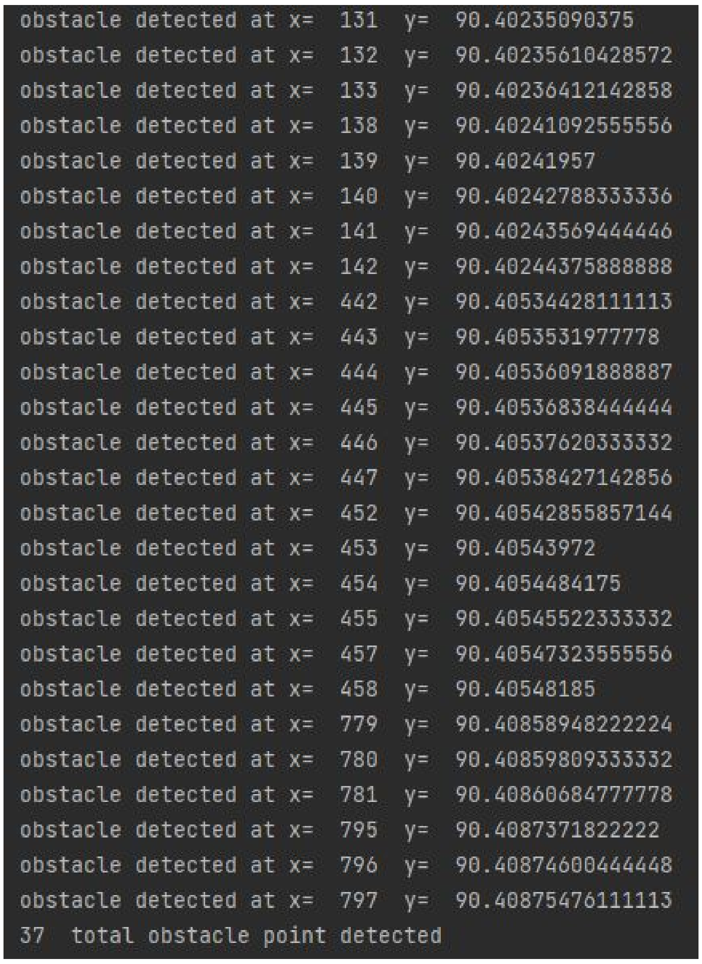
The deviation we got from the first part of our algorithm will be used to detect obstacles. In the figures of section 5.2.1, we can see some spikes. Actually, these spikes are the starting of our obstacles. The x-axis is almost flat if we consider any of the figures of section 5.2.1 and when any obstacle arrives the y-axis rises up as well as the spikes. This is because of turning left or right. The second part of Algorithm 2 will detect these obstacles. Again we applied Algorithm 2 in three different routes.



```
obstacle detected at x=  258  y=  90.40447945
obstacle detected at x=  259  y=  90.40447152333334
obstacle detected at x=  260  y=  90.4044636077778
obstacle detected at x=  268  y=  90.40439234777776
obstacle detected at x=  269  y=  90.40438489777776
obstacle detected at x=  378  y=  90.40333405444444
obstacle detected at x=  381  y=  90.40331667166669
obstacle detected at x=  386  y=  90.403277532
obstacle detected at x=  387  y=  90.40326336666664
obstacle detected at x=  388  y=  90.40324979857144
obstacle detected at x=  389  y=  90.40323734875
obstacle detected at x=  593  y=  90.40125564888888
obstacle detected at x=  597  y=  90.40122372
obstacle detected at x=  615  y=  90.401056174
obstacle detected at x=  617  y=  90.40103565333334
obstacle detected at x=  620  y=  90.40100365375
16  total obstacle point detected
```

Figure 5.10: BRACU to Amtoli Obstacle Detection Result

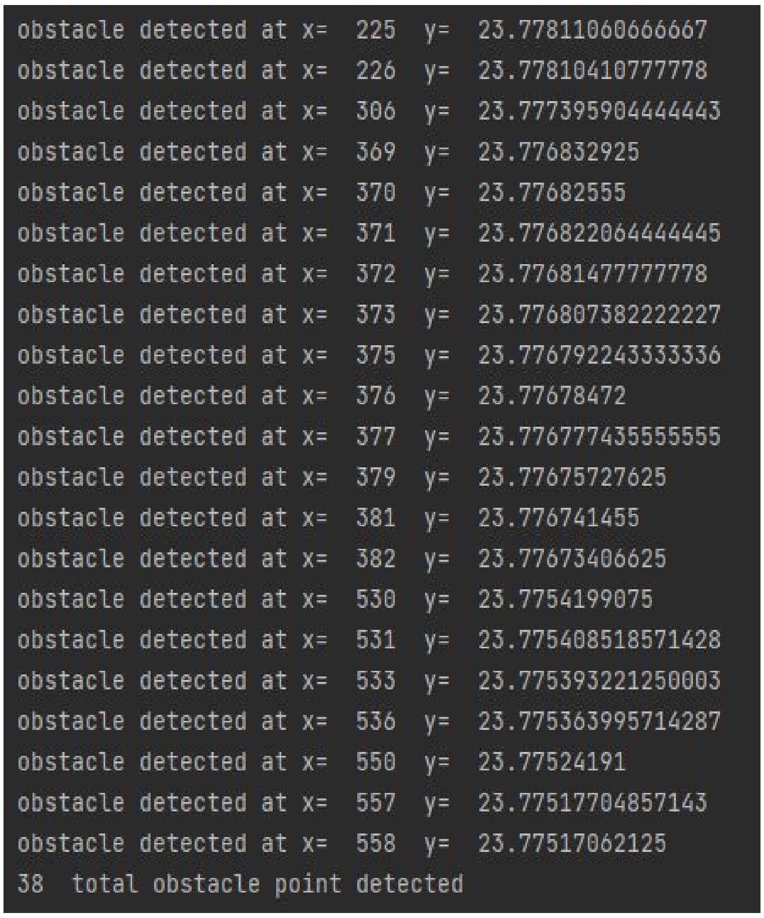
Here, 16 total obstacle points are detected. This is because when we are turning left or right the deviation keeps increasing. And as we said in section 4.5, increased deviation means a point greater than the threshold and this will be marked as an obstacle.



```
obstacle detected at x=  131  y=  90.40235090375
obstacle detected at x=  132  y=  90.40235610428572
obstacle detected at x=  133  y=  90.40236412142858
obstacle detected at x=  138  y=  90.40241092555556
obstacle detected at x=  139  y=  90.40241957
obstacle detected at x=  140  y=  90.40242788333336
obstacle detected at x=  141  y=  90.40243569444446
obstacle detected at x=  142  y=  90.40244375888888
obstacle detected at x=  442  y=  90.40534428111113
obstacle detected at x=  443  y=  90.4053531977778
obstacle detected at x=  444  y=  90.40536091888887
obstacle detected at x=  445  y=  90.40536838444444
obstacle detected at x=  446  y=  90.40537620333332
obstacle detected at x=  447  y=  90.40538427142856
obstacle detected at x=  452  y=  90.40542855857144
obstacle detected at x=  453  y=  90.40543972
obstacle detected at x=  454  y=  90.4054484175
obstacle detected at x=  455  y=  90.40545522333332
obstacle detected at x=  457  y=  90.40547323555556
obstacle detected at x=  458  y=  90.40548185
obstacle detected at x=  779  y=  90.40858948222224
obstacle detected at x=  780  y=  90.40859809333332
obstacle detected at x=  781  y=  90.40860684777778
obstacle detected at x=  795  y=  90.4087371822222
obstacle detected at x=  796  y=  90.40874600444448
obstacle detected at x=  797  y=  90.40875476111113
37  total obstacle point detected
```

Figure 5.11: Banani Signal to Banani Lake Obstacle Detection Result

In Banani Signal to Banani Lake route, 37 obstacle points are detected by our algorithm 2.



```
obstacle detected at x=  225  y=  23.77811060666667
obstacle detected at x=  226  y=  23.77810410777778
obstacle detected at x=  306  y=  23.777395904444443
obstacle detected at x=  369  y=  23.776832925
obstacle detected at x=  370  y=  23.77682555
obstacle detected at x=  371  y=  23.776822064444445
obstacle detected at x=  372  y=  23.77681477777778
obstacle detected at x=  373  y=  23.776807382222227
obstacle detected at x=  375  y=  23.776792243333336
obstacle detected at x=  376  y=  23.77678472
obstacle detected at x=  377  y=  23.776777435555555
obstacle detected at x=  379  y=  23.77675727625
obstacle detected at x=  381  y=  23.776741455
obstacle detected at x=  382  y=  23.77673406625
obstacle detected at x=  530  y=  23.7754199075
obstacle detected at x=  531  y=  23.775408518571428
obstacle detected at x=  533  y=  23.775393221250003
obstacle detected at x=  536  y=  23.775363995714287
obstacle detected at x=  550  y=  23.77524191
obstacle detected at x=  557  y=  23.77517704857143
obstacle detected at x=  558  y=  23.77517062125
38  total obstacle point detected
```

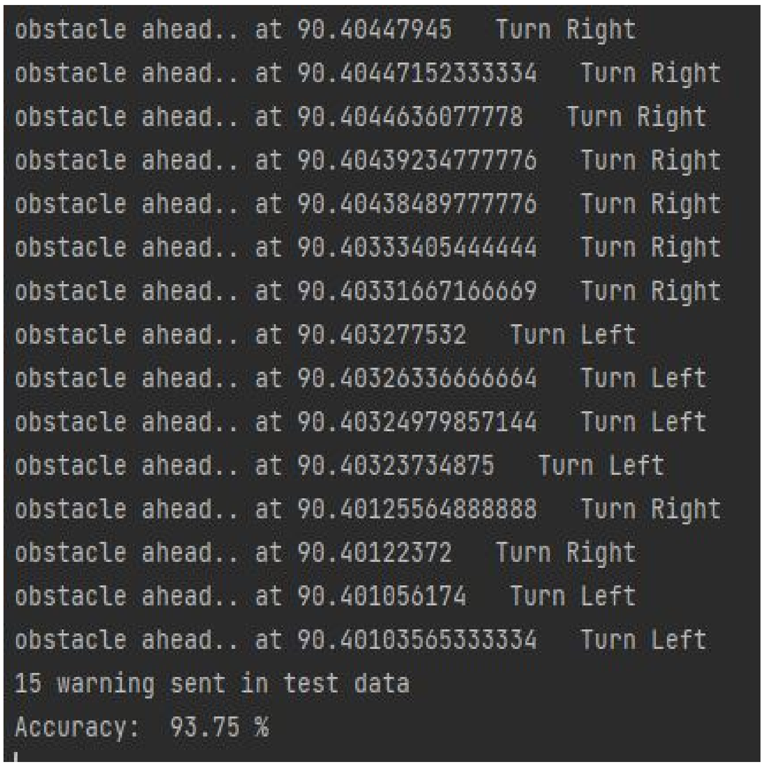
Figure 5.12: Gulshan 1 to Police Plaza Obstacle Detection Result

Finally, in Gulshan 1 to Police Plaza route, 38 obstacle points are detected.

Now, we can say the second part of algorithm 2 also worked nicely here. It detected the points where the deviation is greater than the threshold and marked those as obstacles. For BRACU to Amtoli, it detected 16 points, for Banani Signal to Banani Lake it detected 37 points, and for Gulshan 1 to Police Plaza, it detected 38 points.

5.2.3 Warning

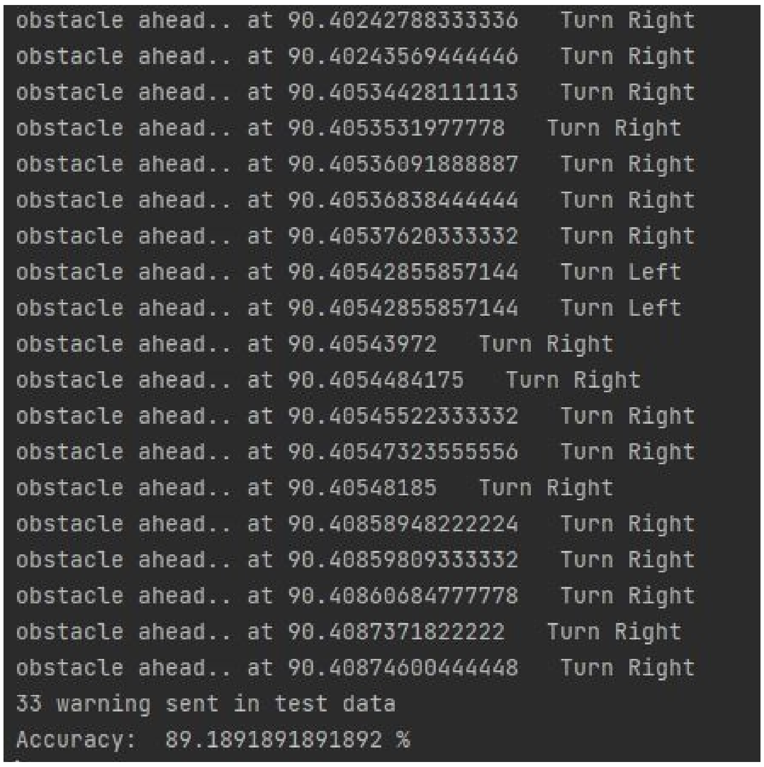
Now the algorithm 2 will give a warning by comparing the testing path. It will compare the data of each row of the testing path and if any of the row data is close enough to our detected obstacle points. It will give a warning that the obstacle is approaching from a few meter distance.



```
obstacle ahead.. at 90.40447945   Turn Right
obstacle ahead.. at 90.40447152333334   Turn Right
obstacle ahead.. at 90.4044636077778   Turn Right
obstacle ahead.. at 90.40439234777776   Turn Right
obstacle ahead.. at 90.40438489777776   Turn Right
obstacle ahead.. at 90.40333405444444   Turn Right
obstacle ahead.. at 90.40331667166669   Turn Right
obstacle ahead.. at 90.403277532   Turn Left
obstacle ahead.. at 90.40326336666664   Turn Left
obstacle ahead.. at 90.40324979857144   Turn Left
obstacle ahead.. at 90.40323734875   Turn Left
obstacle ahead.. at 90.40125564888888   Turn Right
obstacle ahead.. at 90.40122372   Turn Right
obstacle ahead.. at 90.401056174   Turn Left
obstacle ahead.. at 90.40103565333334   Turn Left
15 warning sent in test data
Accuracy:  93.75 %
```

Figure 5.13: BRACU to Amtoli Warning Result

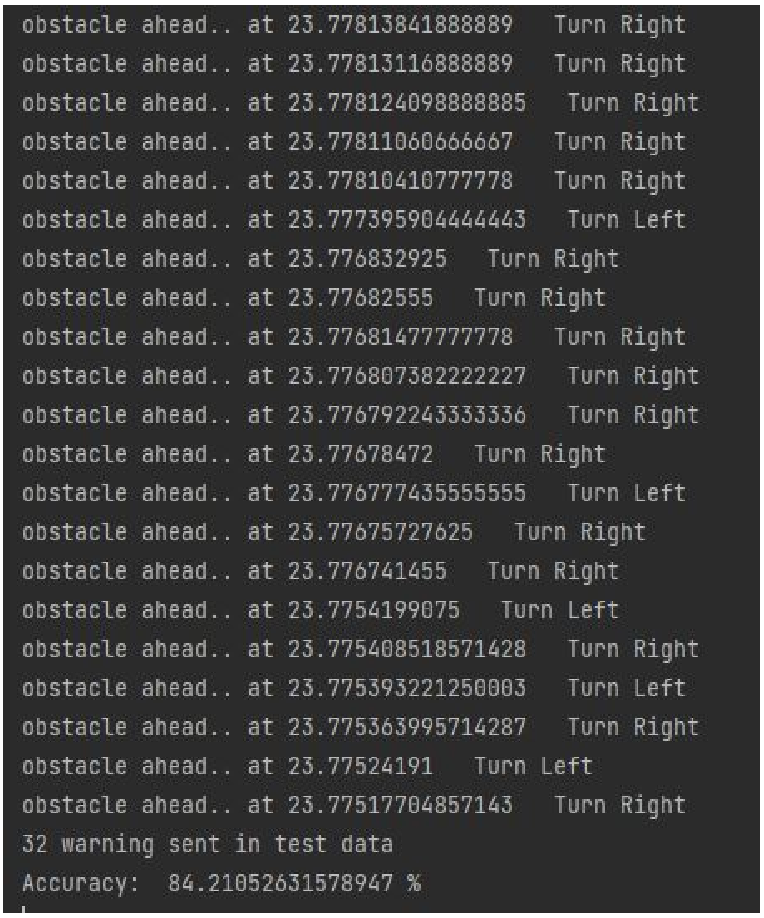
After applying Algorithm 2 to our BRACU to Amtoli routes smoothed path data, we can see in figure 5.13, it could successfully send 15 warnings



```
obstacle ahead.. at 90.40242788333336   Turn Right
obstacle ahead.. at 90.40243569444446   Turn Right
obstacle ahead.. at 90.40534428111113   Turn Right
obstacle ahead.. at 90.4053531977778   Turn Right
obstacle ahead.. at 90.40536091888887   Turn Right
obstacle ahead.. at 90.40536838444444   Turn Right
obstacle ahead.. at 90.40537620333332   Turn Right
obstacle ahead.. at 90.40542855857144   Turn Left
obstacle ahead.. at 90.40542855857144   Turn Left
obstacle ahead.. at 90.40543972   Turn Right
obstacle ahead.. at 90.4054484175   Turn Right
obstacle ahead.. at 90.40545522333332   Turn Right
obstacle ahead.. at 90.40547323555556   Turn Right
obstacle ahead.. at 90.40548185   Turn Right
obstacle ahead.. at 90.40858948222224   Turn Right
obstacle ahead.. at 90.40859809333332   Turn Right
obstacle ahead.. at 90.40860684777778   Turn Right
obstacle ahead.. at 90.4087371822222   Turn Right
obstacle ahead.. at 90.40874600444448   Turn Right
33 warning sent in test data
Accuracy:  89.1891891891892 %
```

Figure 5.14: Banani Signal to Banani Lake Warning Result

Next, we applied Algorithm 2 to Banani routes smoothed data. This time the algorithm could send 33 warnings that we can see from figure 5.14.



```
obstacle ahead.. at 23.77813841888889   Turn Right
obstacle ahead.. at 23.77813116888889   Turn Right
obstacle ahead.. at 23.778124098888885   Turn Right
obstacle ahead.. at 23.77811060666667   Turn Right
obstacle ahead.. at 23.77810410777778   Turn Right
obstacle ahead.. at 23.777395904444443   Turn Left
obstacle ahead.. at 23.776832925   Turn Right
obstacle ahead.. at 23.77682555   Turn Right
obstacle ahead.. at 23.77681477777778   Turn Right
obstacle ahead.. at 23.776807382222227   Turn Right
obstacle ahead.. at 23.776792243333336   Turn Right
obstacle ahead.. at 23.77678472   Turn Right
obstacle ahead.. at 23.776777435555555   Turn Left
obstacle ahead.. at 23.77675727625   Turn Right
obstacle ahead.. at 23.776741455   Turn Right
obstacle ahead.. at 23.7754199075   Turn Left
obstacle ahead.. at 23.775408518571428   Turn Right
obstacle ahead.. at 23.775393221250003   Turn Left
obstacle ahead.. at 23.775363995714287   Turn Right
obstacle ahead.. at 23.77524191   Turn Left
obstacle ahead.. at 23.77517704857143   Turn Right
32 warning sent in test data
Accuracy:  84.21052631578947 %
```

Figure 5.15: Gulshan 1 to Police Plaza Warning Result

For our last route, which is Gulshan 1 to Police Plaza, the algorithm could send 32 warnings.

All the figures in the section 5.2 shows that, the new two algorithms that we made have done a great job. Algorithm 1 allowed us to smooth a path very easily and it has done exactly what we were expecting it to do. Moreover, Algorithm 2 were able to detect obstacles. Also, it could send warning to a user from a little distance of the obstacles so that the person do not hit the obstacle.

5.3 Statistical Analysis

Route Name	**Obstacle Detected (No. of Coordinates)**	**Warning Sent (No. of Coordinates)**	**Accuracy (%)**	**Average Accuracy (%)**
BRACU to Amtoli	16	15	93.75	
Banani Signal to Banani Lake	37	33	89.18	89.05
Gulshan 1 to Police Plaza	38	32	84.21	

Table 5.1: Statistical Analysis

Here, in Table 5.1, we can see, our algorithm worked pretty fine on the route BRACU to Amtoli. It could send a warning to 15 obstacles point out of 16 with an accuracy of 93.75%. The Banani route and the Gulshan route have an accuracy of 89.18% and 84.21% consecutively. These routes are having less accuracy because of the threshold value. We used the same threshold for every route and it is giving us a good output for BRACU to Amtoli route. Moreover, we can not deny that the algorithm worked badly in other routes. They are just having less accuracy and as we told earlier this is because of the threshold value, we will work on it in future.

Chapter 6

Cloud

6.1 Google Cloud Platform

The idea of using cloud computing is rapidly gaining popularity in this modern world due to its flexibility. For this reason, we have planned this model in such a way that it can be uploaded to the cloud to store data and computational purposes centrally. Google Cloud Platform or GCP is a well known cloud platform all over the world. GCP is a collection of infrastructure and services for cloud storage, computation, analytics, and development. GCP, according to Google, is built on the same architecture as Google's own products, such as Google Search. This set of services and infrastructure goes much beyond basic cloud storage and computation resources, including some highly useful and economical machine learning and big data capabilities. Furthermore, Google Compute Engine (GCE) offers virtual machines (VMs) that are hosted on Google's worldwide data center network. A virtual machine (VM) is essentially a computer system that emulates the capabilities of a physical computer. We can utilize a virtual machine just like a regular computer without having to buy the necessary hardware [37]. Moreover, we may use GCE to immediately start an instance with predetermined CPU, RAM, and storage parameters, or we may build our own customised machine.

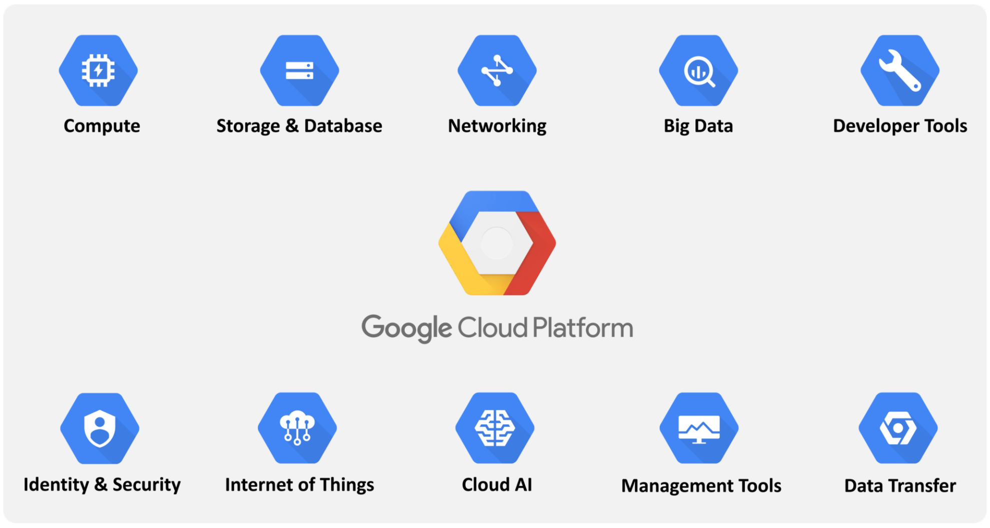


Figure 6.1: Services of Google Cloud Platform

6.2 Connecting Our Model With GCP

To connect and run our model in GCP, first we need to sign up for Google Cloud. Next, we need to create a new VM instance from the Compute Engine of GCP. There we need to specify some information like how much storage space we need, how much ram and CPU we will use etc. After creating a VM instance we need to set up access control to access the instance. Finally, we can connect to the VM using Google's remote desktop application from our desktop.

Now our model will be uploaded there. At first, the cleaned data sets of 3 routes in the cloud server will be stored in a central database of the cloud. Next, we will store our Path Smoothing algorithm and Obstacle Detection and Warning giving algorithm in the GCP, and also we will store the smoothed path data which we have achieved by running our two algorithms on raw data sets of 3 different routes. We have also planned to use a Text-To-Speech synthesizer as a communication medium between the user and the cloud server. By using a mobile phone and Text to speech synthesizer the user will send his destination to the cloud server. After that, the cloud will detect the user's current location by using the GPS of the user's mobile phone. Then, in the cloud server, will compare the user's path with the smoothed path that we have already stored in the cloud server and whenever the user steps in the wrong path other than smoothed path, the server will send a warning to the user using the Text-To-Speech synthesizer. The cloud server will also send a warning to the user whenever the user will be near an obstacle. Lastly, the server will also store the current path that the user is taking and feed this to the dataset for attaining better accuracy.

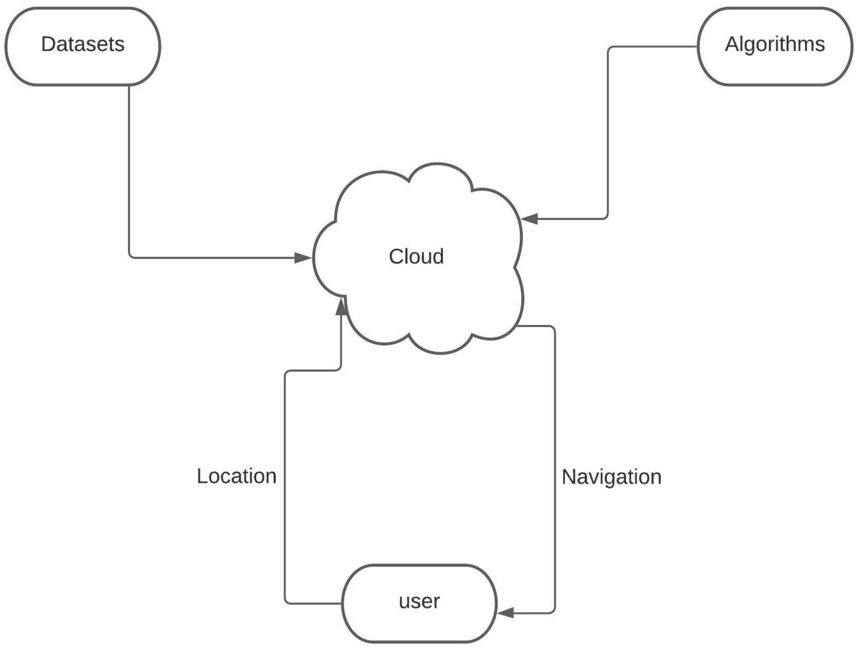


Figure 6.2: Sample Workflow of Our Model in Cloud

Though we were not able to upload our datasets, algorithms, and outputs to the cloud as it requires premium access. So, we implemented all these things on our computer and tested the model with sample walk path data. Here, we assumed our computer was a local server. We will implement this model to any of the cloud platforms in our future work.

Chapter 7

Conclusion

7.1 Research Overview

The introduced navigation system in this paper has been developed in order to make navigation safe and independent for blind people. Our system is designed based on cloud computing and smartphone. For our system, we have collected datasets of 3 different routes and for each route, we have collected 10 different data sets. We have utilized the smartphone's GPS for collecting data. At first, we thought we will use existing algorithms like A*, Dubin's curve, B spline interpolation then we have realized that these algorithms do not fit with our system and datasets. After that, we thought of using reinforcement algorithm Q learning but later on, we came to know that this algorithm also does not fit with our system. Finally, we decided to build our own algorithm and we have built two algorithms one is for path smoothing and another one is for obstacle detection and warning.

7.2 Contribution and Impact

Our model is mainly designed to help blind people. Their life is not as simple and normal as us and they face so many difficulties in their day-to-day life. Walking on the road is one of them. In our country, there are so many obstacles like path-holes, electric pillars, broken footpaths beside the road. A blind person needs to use an extra stick to detect those obstacles. So we thought to reduce the difficulties of blind people with our model. By using our model, they can easily get alert before approaching any obstacles. Moreover, they can be aware of the obstacles from a little distance without using any cane. Thus our model will bring a great impact on their life.

7.3 Future Work

Our model has so many scopes that we can improve in the future. We have built our model using three routes. Our future work will be to make it real-time so that blind people can use our model in any route. For that, image processing can be added. In addition, in the path data, there was 1-meter distance between each coordinate. We will try to reduce this distance in our future work as well. If we are able to reduce the distance, we will get more accurate data. Moreover, our algorithm worked pretty well on our selected routes but in the future, we are going to collect more data instead of 10 for a single route to make the path smoothing algorithm more efficient. However, The threshold we used in our algorithm can be improvised for better accuracy.

Bibliography

[1] Y. Yi and L. Dong, "A design of blind-guide crutch based on multi-sensors," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, IEEE, 2015, pp. 2288–2292.

[2] T. F. Express, *Visual impairment to increase dramatically: Study*, en. [Online]. Available: https://www.thefinancialexpress.com.bd/health/visual-impairment-to-increase-dramatically-study-1579444915 (visited on 05/31/2021).

[3] P. Strumillo, "Electronic interfaces aiding the visually impaired in environmental access, mobility and navigation," in *3rd International Conference on Human System Interaction*, IEEE, 2010, pp. 17–24.

[4] P. Angin, B. Bhargava, and S. Helal, "A mobile-cloud collaborative traffic lights detector for blind navigation," in *2010 Eleventh International Conference on Mobile Data Management*, IEEE, 2010, pp. 396–401.

[5] M. N. Abdullah and A. E. Korial, "Planning the path and avoidance obstacles for visually impaired/blind people," *IOSR Journal of Computer Engineering*, vol. 17, pp. 147–151, 2015.

[6] T. SP, V. Muthuswamy, and S. Kousik, "Cloud based mobile public transport assistance for visually impaired people,"

[7] M. Owayjan, A. Hayek, H. Nassrallah, and M. Eldor, "Smart assistive navigation system for blind and visually impaired individuals," in *2015 International Conference on Advances in Biomedical Engineering (ICABME)*, IEEE, 2015, pp. 162–165.

[8] S. Koley and R. Mishra, "Voice operated outdoor navigation system for visually impaired persons," *International journal of engineering trends and technology*, vol. 3, no. 2, pp. 153–157, 2012.

[9] J. Coughlan and R. Manduchi, "Functional assessment of a camera phone-based wayfinding system operated by blind and visually impaired users," *International Journal on Artificial Intelligence Tools*, vol. 18, no. 03, pp. 379–397, 2009.

[10] A. N. Lapyko, L.-P. Tung, and B.-S. P. Lin, "A cloud-based outdoor assistive navigation system for the blind and visually impaired," in *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, IEEE, 2014, pp. 1–8.

[11] J. Bai, D. Liu, G. Su, and Z. Fu, "A cloud and vision-based navigation system used for blind people," in *Proceedings of the 2017 International Conference on Artificial Intelligence, Automation and Control Technologies*, 2017, pp. 1–6.

[12] P. Angin, B. K. Bhargava, *et al.*, "Real-time mobile-cloud computing for context-aware blind navigation," *International Journal of Next-Generation Computing*, vol. 2, no. 2, pp. 405–414, 2011.

[13] R. Thomas, R. Ramanan, R. Roy, and S. Elisabath, "Read2me: A cloud- based reading aid for the visually impaired," 2017.

[14] B. Bhargava, P. Angin, and L. Duan, "A mobile-cloud pedestrian crossing guide for the blind," in *International Conference on Advances in Computing & Communication*, 2011.

[15] S. Ön and A. Yazici, "A comparative study of smooth path planning for a mobile robot considering kinematic constraints," in *2011 international symposium on innovations in intelligent systems and applications*, IEEE, 2011, pp. 565–569.

[16] L. Yu, Z. Wei, Z. Wang, Y. Hu, and H. Wang, "Path optimization of auv based on smooth-rrt algorithm," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, IEEE, 2017, pp. 1498–1502.

[17] M. Islam and M. Sadi, "Path hole detection to assist the visually impaired people in navigation," Sep. 2018. DOI: 10.1109/CEEICT.2018.8628134.

[18] E. Ko and E. Y. Kim, "A vision-based wayfinding system for visually impaired people using situation awareness and activity-based instructions," *Sensors*, vol. 17, no. 8, p. 1882, 2017.

[19] L. Yu, Z. Wei, Z. Wang, Y. Hu, and H. Wang, "Path optimization of auv based on smooth-rrt algorithm," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, IEEE, 2017, pp. 1498–1502.

[20] D. Nandini and K. Seeja, "A novel path planning algorithm for visually impaired people," *Journal of King Saud University-Computer and Information Sciences*, vol. 31, no. 3, pp. 385–391, 2019.

[21] M. A. Uddin and A. H. Suny, "Shortest path finding and obstacle detection for visually impaired people using smart phone," in *2015 international conference on electrical engineering and information communication technology (ICEEICT)*, IEEE, 2015, pp. 1–4.

[22] J. José, "Real-time path and obstacle detection for blind persons," Ph.D. dissertation, 2010.

[23] R. Zeineldin and N. El-Fishawy, "Fast and accurate ground plane detection for the visually impaired from 3d organized point clouds," Jul. 2016, pp. 373–379. DOI: 10.1109/SAI.2016.7556009.

[24] Y.-J. Ho and J.-S. Liu, "Collision-free curvature-bounded smooth path planning using composite bezier curve based on voronoi diagram," in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation-(CIRA)*, IEEE, 2009, pp. 463–468.

[25] J. Bai, S. Lian, Z. Liu, K. Wang, and D. Liu, "Virtual-blind-road following-based wearable navigation device for blind people," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 1, pp. 136–143, 2018.

[26] D. Priyasad, Y. Jayasanka, H. Udayanath, D. Jayawardhana, S. Sooriyaarachchi, C. Gamage, and N. Kottege, "Point cloud based autonomous area exploration algorithm," Jun. 2018. DOI: 10.1109/MERCon.2018.8421954.

[27] Y. J. Kanayama and B. I. Hartman, "Smooth local-path planning for autonomous vehicles1," *The International Journal of Robotics Research*, vol. 16, no. 3, pp. 263–284, 1997.

[28] E. Magid, D. Keren, E. Rivlin, and I. Yavneh, "Spline-based robot navigation," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 2296–2301.

[29] T. Yusof, S. Toha, and H. M. Yusof, "Path planning for visually impaired people in an unfamiliar environment using particle swarm optimization," *Procedia Computer Science*, vol. 76, pp. 80–86, 2015.

[30] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 1993.

[31] M. L. Mekhalfi, F. Melgani, Y. Bazi, and N. Alajlan, "A compressive sensing approach to describe indoor scenes for blind people," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 7, pp. 1246–1257, 2014.

[32] S. Kotyan, U. Venkanna, N. Kumar, and P. K. Sahu, "Drishtikon: An advanced navigational aid system for visually impaired people," in *2018 Conference on Information and Communication Technology (CICT)*, IEEE, 2018, pp. 1–6.

[33] P. Gao, Z. Liu, Z. Wu, and D. Wang, "A global path planning algorithm for robots using reinforcement learning," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2019, pp. 1693–1698.

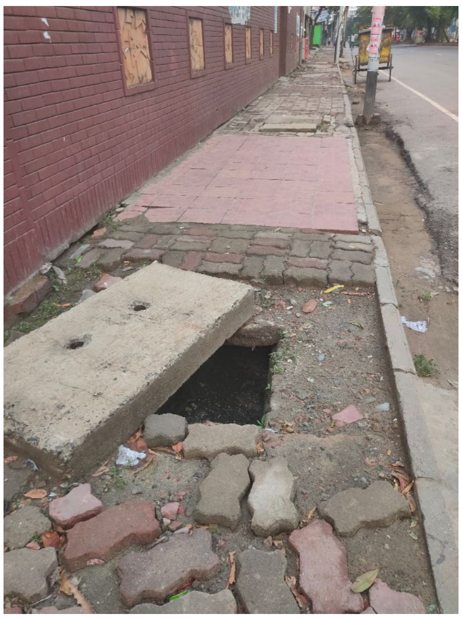
[34] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, p. 3170, 2018.

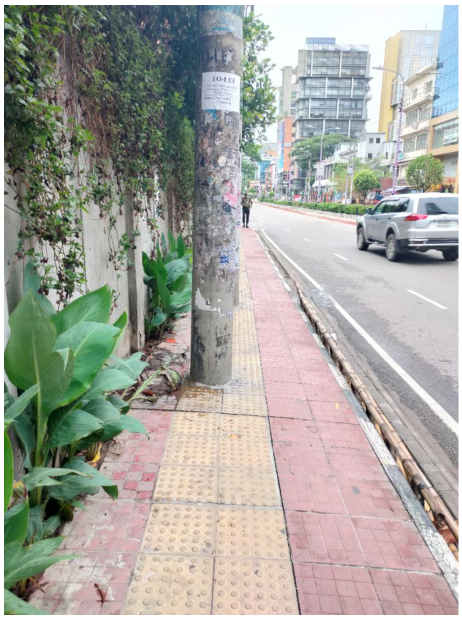
[35] *A* Algorithm — Introduction to the A* Seach Algorithm*, en-US, Section: Uncategorized, Dec. 2019. [Online]. Available: https://www.edureka.co/blog/a-search-algorithm/ (visited on 05/31/2021).

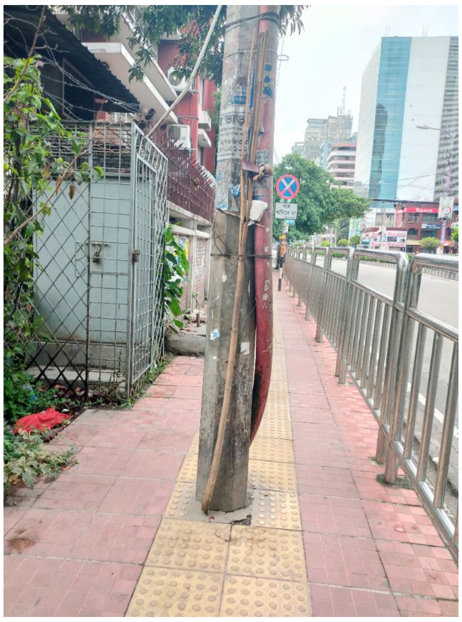
[36] *A* Search Algorithm*, en-us, Section: Algorithms, Jun. 2016. [Online]. Available: https://www.geeksforgeeks.org/a-search-algorithm/ (visited on 05/31/2021).

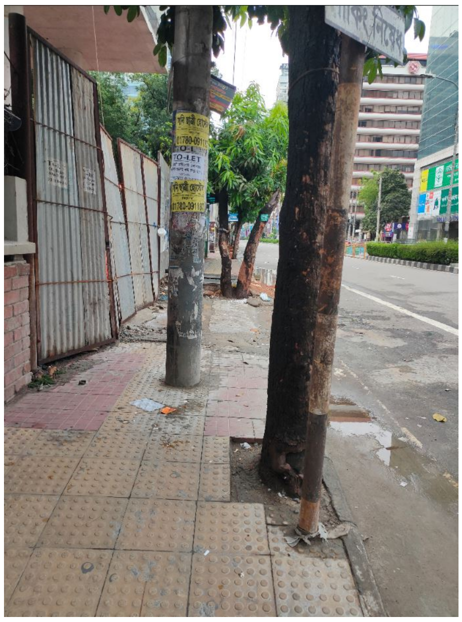
[37] *How to Run Trading Algorithms on Google Cloud Platform in 6 Easy Steps*, en, Jul. 2017. [Online]. Available: https://robotwealth.com/run-trading-algorithms-google-cloud-platform-6-easy-steps/ (visited on 05/31/2021).

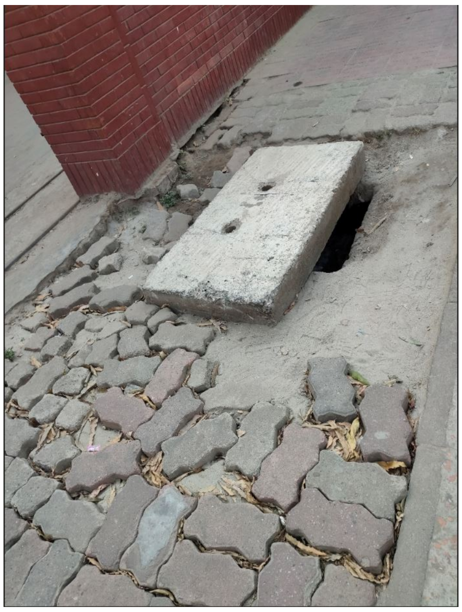
Appendix A: Further Figures

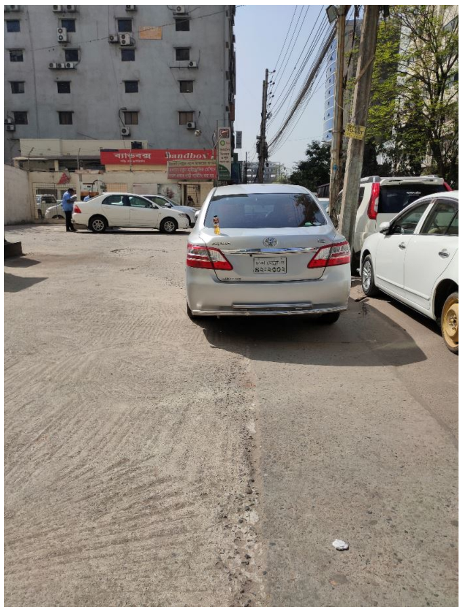

(a)

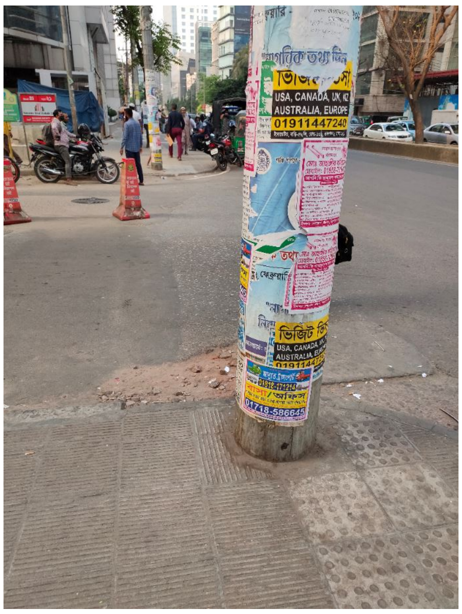

(b)

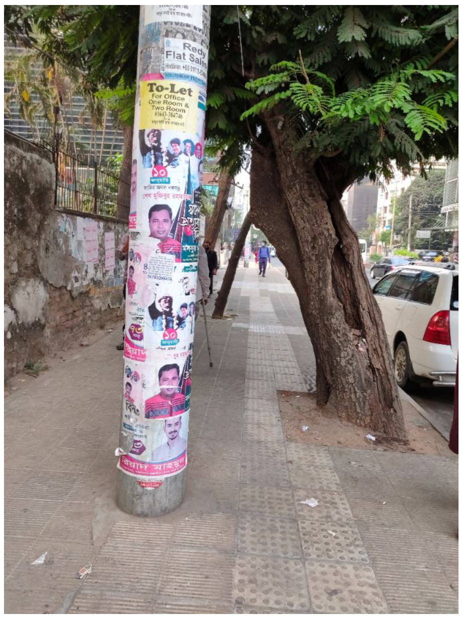

(c)

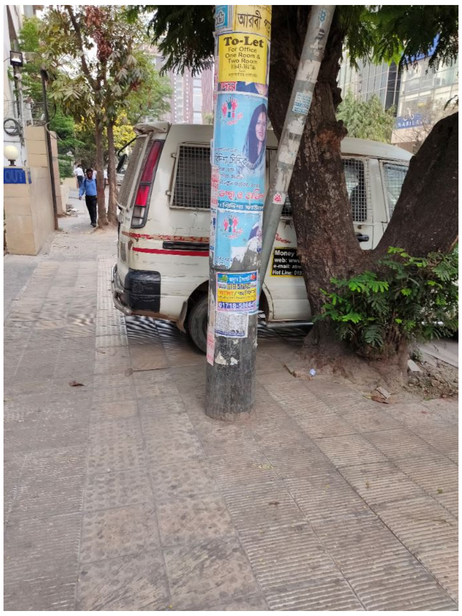

(d)


(e)

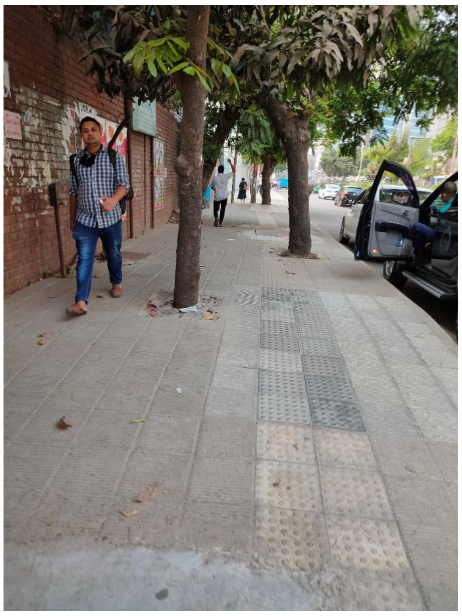

(f)


USA, CANADA, UK, NZ
AUSTRALIA, EUROPE
01911447240
(g)


To-Let
For Office
One Room &
Two Room
(h)



(i)



(j)

Figure 7.1: More Real Life Obstacles