Towards a Hiker Helper App: Defining and Labeling Traversable Space in a Forested Environment

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
James Nguyen
May 2024

Accepted by:
Dr. Adam Hoover, Committee Chair
Dr. Yongkai Wu
Dr. Xiaolong Ma

Abstract

This thesis investigates the problem of identifying traversable terrain in outdoor conditions. We are motivated by research in recent years toward identifying drivable space for the purpose of developing autonomous vehicles. Our motivating application is similar but also different. We envision a "Hiker Helper" that assists humans with dismounted navigation in forested terrain. A common challenge in this type of environment is identifying a viable path for moving through terrain that is congested with trees, bushes, other flora, and natural obstacles that would make navigation difficult. We envision training an artificial intelligence (AI) model to automatically analyze images of this type of terrain to identify potentially viable paths. The tool could highlight these areas in a camera view to help the user with navigation. Towards this goal, this thesis attempts to define traversable space based on single person dismounted navigation (e.g. hiking). This problem is more challenging than identifying drivable space for on-road navigation because the terrain appearance is much more variable. The terrain is not engineered to provide visible cues for navigation.

The work done in this thesis is intended to serve as a pilot project to aid in the study of this subject. One of the largest obstacles for this topic and for developing AI models is the lack of data and the quality of the data. The main results coming out of this project is a data annotation program that has been designed for users to quickly label data, definitions for labeling data, and a trained dataset. Use of optical flow was investigated thoroughly to determine its usefulness in helping semi-automate the process of labeling video frames by using a labeler's input and interpolating their annotations to other frames. The annotation program is designed to allow users to label data with greater speed than other methods that exist, so that training data for this highly variable problem can be created. The dataset contains about 2.4 hours worth of videos in 30 different locations with 260,169 labeled frames. The ground truth for the videos consists of line labels that detail a traversable path that a hiker can take. These line labels were then used to create polygons that highlights a trail

for pedestrians to walk through. Two independent annotators timed their labeling process for 80% of the data. The inter-rater reliability was measured by creating polygons that represented paths from the line labels created in the annotation program and then calculating the intersection-over-union, or IoU. The IoU was determined by dividing the overlapping area by the total area of the two paths. The inter-rater reliability between the two annotators was recorded to be 60.6% with an average label time being 0.40 and 0.31 seconds per frame for our two annotators.

Challenges for this project included determining exactly what kind of terrain was considered traversable and which direction to choose given multiple options. Issues such as lighting and motion blur in the video taking process disrupted the semi-automated labeling process by slowing or making the labeling process more complex. Future solutions that build on this thesis should strive to improve our definition for traversability which may make the labeling process more complex but may also increase inter-rater reliability. Future work could also try to speed up the labeling process by improving our video taking method for easier labeling or exploring more methods to alleviate the issues caused when optical flow is disrupted. The most important future work that could be done is to use the data set to train an AI that will automatically identify trails in an outdoor environment.

Acknowledgments

Firstly, I would like to thank Dr. Adam Hoover for the time and patience he has had in mentoring me. His teaching has motivated my graduate education and his guidance has been vital for my professional development. His research and his projects inspired me to pursue academic research and I would not have completed a thesis without him.

I would also like to thank Dr. Xiaolong Ma and Dr. Yongkai Wu for taking the time out of their busy schedules to serve on my defense committee. Additionally, I would like to thank all the professors that I have had the pleasure of learning from, and Clemson University for providing me an incredible environment to study and grow in.

I would also like to thank Pranava Swaroopa, who has taken time to volunteer to help with this project. He has contributed greatly as a part of our research group and his ideas and experience was invaluable to the completion of this thesis.

Lastly, I want to thank my friends and family for their unconditional support throughout my educational career. Their encouragement has been inspirational in keeping me motivated and determined to further my academic goals.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

This thesis considers the problem of identifying traversable terrain in outdoor off-road conditions in order to aid in developing tools for autonomous navigation outdoors. Outdoor conditions are much more complex mainly due to the fact that man-made markers are minimal or non-existent. Also, the term "outdoor environments" or "outdoor conditions" can mean many different things which can increase the variability making it difficult to create a model that can easily generalize with all of these factors in mind. For example, the ground can have different textures (leaves, sand, etc.) and elevation [1]. Or the obstacles can be bushes in one location, trees in another, tall grass, and even deep water [2]. Figure 1.1 shows an example of this type of terrain. In contrast, man-made paths created for high foot traffic or vehicles stand out greatly from the surrounding environment. Sidewalks and roads are generally a solid color and smooth in texture which is different from the forest or grass surrounding it which is usually highly textured and will often have variance in color [2]. Another major difference between on-road and off-road paths are that video/image data of roads and sidewalks will generally have a static amount of traversable area since the width of the path will rarely change while off-road paths can have great variance in the width of the path ahead.

Figure 1.1: Example of an outdoor environment.

The rest of this chapter reviews work related to our problem. First, we describe different types of navigation problems for which automated recognition of traversable space has been explored. There are many such problems, including on-road and underwater navigation. We describe each problem and its unique challenges, and compare these against our problem of hiking navigation. Second we describe existing datasets that have been collected for these problems. Due to the unique challenges in each domain, many use different types of ground truth. This helped guide our selection of ground truth primitives for the problem of hiking. Third, many previous works required the development of custom labeling software to support their unique challenges. This background helps justify our motivation to develop our own custom software. We first tried several available packages before finding that we needed unique functions that were not readily available in existing labeling software. Fourth, we describe some background in optical flow, which forms a core operation in our labeling software. Lastly we describe related work in metrics for quantifying agreement between two labelings of the same image. These are most commonly used to evaluate the performance of automated AI models by comparing their output to the human generated ground truth labels. In our case, we use similar metrics to compare the agreement between two human raters. We performed these measurements in order to quantify the objectivity of our definition of traversable space, and reveal where improvements could be made.

1.2 Types of Navigation

This section describes several different types of navigation problems. Variations include vehicle vs pedestrian, land vs water, and urban vs natural environments. Each problem has unique challenges based upon the varieties of traversable space and how that traversable space is typically marked or recognized. This section describes all these problems so that they can be contrasted against the unique challenges of our problem.

On-road autonomous driving has became a very popular research subject in recent years as demand and interest for driver-less cars continues to increase [3]. These autonomous driving systems are expected to be able to transport passengers on normal roads, highways, and in more cramped urban environments. Navigable space is typically defined by a lane that is approximately 3 meters wide so that most automobiles can comfortably fit within the lanes. The boundaries of the road lanes are commonly marked with dashed or solid lines that are yellow or white so that they heavily contrast with the dark asphalt road. These lines help humans clearly see the boundary of the lanes. For autonomous driving many sensors and types of data have been researched to develop the awareness and adaptability of these autonomous systems which include camera data, ultra sonic, LiDAR, radar, and more [4]. The desired output of the video data is to be able to identify objects such as signs and traffic signals while depth sensing sensors identify obstacles and their distances. Figure 1.2 shows an urban road environment with vehicle traffic. Autonomous systems are trained to correctly identify lane markers, traffic signals, and obstacles like people and other cars to minimize accidents. One of the main challenges for autonomous driving is obstacle and object recognition. Identifying and interpreting traffic signs, signals, and road markers is important for the awareness of the autonomous driving systems. Also, identifying objects, their distance, and their speed is a unique challenge for on-road AI systems. AI systems must be able to correctly identify the traversable space while following traffic rules and avoiding temporary moving obstacles. For on-road autonomous driving, the main application is to have vehicles transport passengers without a need for a driver controlling the vehicles on conditions that are "normal" for human drivers. Other applications include long-haul trucking and automated taxi services.

Figure 1.2: Example of a labeled urban on-road environment. Adapted from [5].

A second kind of navigation problem is autonomous underwater vehicle (AUV) navigation. In this area, navigation typically occurs in open water under the surface of the ocean. Research has been done to navigate underwater by analyzing the terrain of the seafloor so that an AUV can autonomously navigate without aid from the surface [6]. Navigable space is mostly trivial due to open water lacking the amount of obstacles or harsh terrain that a land vehicle would face. The seafloor is commonly sensed using bathymetry techniques to measure the depth of the seafloor. The layout of the seafloor is then recorded into a database. When an AUV using terrain based navigation techniques is deployed, it maps out the seafloor nearby and compares the seafloor to the information in its database to determine its own location. Terrain based navigation methods uses depth sensors like sonar to map the seafloor. Figure 1.3 shows an image of an AUV measuring the seafloor. Unlike road navigation, the ocean is mostly featureless so using object detection or other computer vision techniques are limited. Also, navigation in the ocean typically requires the use of support systems or inertia based navigation methods. However, the use of support systems may not be available in some cases and inertia based navigation has a tendency to acquire error over time. Using artificial intelligence to compare the nearby seafloor to a database can mitigate these issues, but the power requirement is also increased. AUVs are useful for mapping the seafloor for marine research and for military uses such as scanning for mines or surveillance.

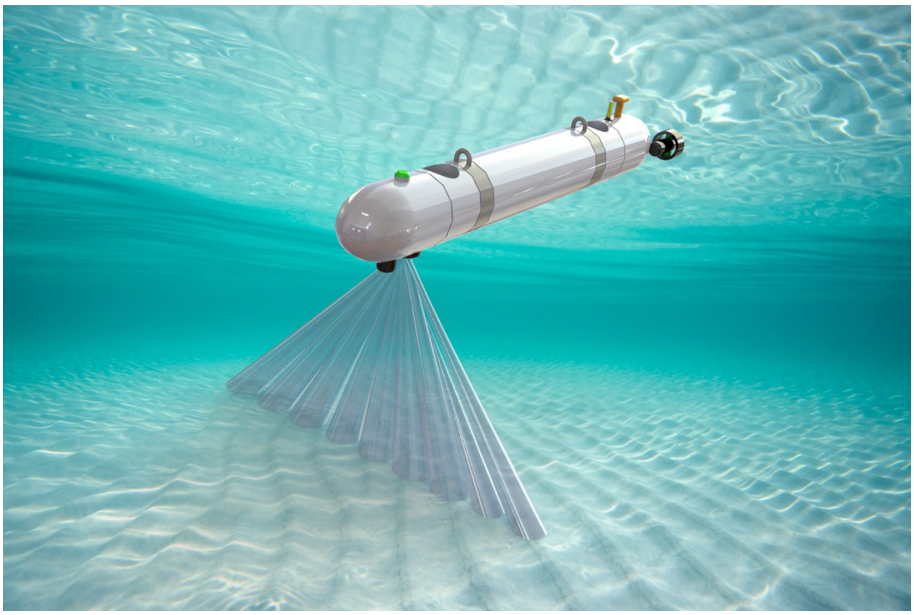


Figure 1.3: Diagram of AUV scanning seafloor. Adapted from [6].

Autonomous navigation for robots and vehicles has also been researched for outdoor off-road conditions where Borges et. al has defined off-road as any terrain that is not a part of a paved road network [2]. Navigable space is defined specifically for the designated vehicle or robot. Typically, navigable space for outdoor off-road vehicles is along dirt roads/trails, light fauna, and smaller rocks with a space wide enough to fit the vehicle intended. The roughness of the terrain, the type of terrain, and the density of the obstacles are integral factors that are examined before determining the boundaries of the navigable space. Off-road autonomous navigation research is heavily inspired by on-road autonomous navigation research and uses a lot of similar concepts such as using LiDAR and radar for depth sensing and cameras for object and obstacle recognition. Figure 1.4 contains two images from the project done by Sharma et al. and shows an off-road environment designed for typical off-road passenger vehicles. The image to the left is the original image before labeling and the image to the right is labeled based on how difficult the terrain is to traverse. The navigable space is mostly clearly defined where the dirt road contrasts with the forest and the middle area of the image contains smooth terrain. Around the edges of the road there is more variance in the roughness of the terrain and there is flora which hinders traversability [7]. AI navigation systems are trained to recognize traversability by interpreting the height and density of obstacles to determine whether they would damage the vehicle or expend too much energy too navigate through [7–9]. Unlike on-road

conditions, off-road conditions vary greatly and are not purposely made for uniformity. The width of the path can vary greatly as well and the path may not contrast heavily with the surroundings that would not be considered navigable. Classifying the terrain into different categories or quantifying the traversability by analyzing its roughness are unique challenges for off-road autonomous navigation. Off-road navigation has many use cases such as agriculture, search and rescue, planetary exploration, defense, mining, and environmental monitoring [2].

Figure 1.4: Example of an outdoor off-road path and the same path labeled. Adapted from [7].

Autonomous navigation has also been applied to pedestrian navigation for indoor and outdoor environments [10–12]. In these areas, navigation typically occurs in buildings or in outdoor walking areas made for pedestrians. Navigable space would typically be defined as areas that are wide enough to fit one or more people. Walking areas indoors are generally marked by characteristics that make them obvious at first glance like stairs, doorways, and space left intentionally empty such as hallways. Walking areas outdoors such as sidewalks are generally marked through color so that people can easily distinguish walking lanes from driving lanes. For tools that help those who are visually impaired, depth sensing sensors are used to determine obstacles, tripping hazards, and boundaries of structures. For tools that help with navigation indoors, using inertial measurement navigation methods has been useful when supplemented with camera data and depth sensors. Figure 1.5 shows a generated image of the goal for an indoor pedestrian navigation application. Indoor areas or covered outdoor areas like forests are unique to road conditions because barriers such as walls or tree canopies block Global Navigation Satelite Systems (GNSS). People generally rely on their phone's GPS applications for navigation, but this is unreliable indoors. For people who are visually impaired, they will also require assistance understanding the environment to avoid obstacles and danger such as roads. Applications for autonomous navigation systems for pedestrians are indoor location based services such as delivery,

indoor navigation through buildings, and tracking for emergency services [11] and helping with more general navigation for both indoor and outdoors for blind and visually impaired people [10, 12].

Figure 1.5: Example of a goal for indoor navigation. Adapted from [13].

In this thesis, we consider the problem of pedestrian navigation in forested terrain (e.g. off-path hiking). Compared to other navigation problems, the visibility of traversable space is extremely challenging. Navigable space can be defined as any space in which a human can reasonably walk. We assume that walk-able area is not marked by artificial boundaries. Figure 1.6 shows an example of an easily discernible path in an forested environment. However, codifying the set of rules by which this space is defined is in our opinion more difficult than codifying the same set of rules for on-road driving.

Figure 1.6: Example of an outdoor path for pedestrian navigation.

1.3 Datasets

This section briefly reviews some other datasets that have been created for research towards automated identification of traversable space and autonomous navigation. We surveyed these datasets to examine the methods used to capture data, the software used to label traversable space, and the amount of effort needed to complete labeling.

Dataset Name	Type of Navigation Problem	Locations / Environment	Label Software	Format of GT	Amount of Data
Cityscapes Dataset [14]	On road passenger vehicles	City streets	LabelMe	Pixel wise semantic segmentation of images	25,000 images
Mapillary Vistas Dataset [15]	On road passenger vehicles	Various locations and settings	Not described	Instance-specific semantic segmentation of images	25,000 images
Bathymetric Mapping Dataset [16]	Underwater robotics	Open Water	ROS and Robotic Localization	Video data, IMU, DVL and multibeam sonar data is synchronized with GPS data	4 surveys were taken
Caves Dataset [17]	Underwater vehicle	Underwater cave complex "Coves de Cala Viuda"	Not described	Contains depth, video, imaging sonar, DVL, and IMU recordings matched with markers	Data was gathered over 500m
RELLIS-3D [18]	Off road mobile robot	Off-road environment in Bryan, TX	Image Data: Appen Point Cloud: App by SemanticKITTI project.	Pixel wise semantic segmentation of video data. Point-wise annotation for 3D point clouds from LiDAR data.	13,556 LiDAR scans and 6,235 images
CAVS Traversability Dataset [7]	Off road passenger vehicles	Off-road environment in Starksville, MS	BasicAI	Pixel wise semantic segmentation of video data	1812 images
Oxford Inertial Odometry Dataset [19]	Indoor pedestrian	Indoor area	Vicon Motion Capture	IMU data matched with location, velocity, and orientation	158 sequences totalling more than 42 km distance
Orientation Network Dataset [20]	Outdoor pedestrian for visually impaired	Sidewalks in Siegen, Germany	LabelMe	Binary semantic segmentation	941 images

Table 1.1: Summary of some datasets for related works.

The Cityscapes Dataset was created to deepen the visual understanding of complex urban street scenes [14]. Due to the labeling being very detailed and containing a variety of locations, this dataset can be used for a wide range of applications, but the Cityscapes dataset is specifically tailored for autonomous driving of passenger vehicles in complex urban environments. The Cityscapes dataset was created by taking images of dense urban environments primarily in Germany, but also in many cities across the world. The Cityscapes dataset uses image data from cameras and goes beyond pixel level semantic labeling and includes instance-level semantic labeling. With instance-level semantic labeling, each pixel in the images were classified by what kind of terrain or object that pixel belonged to and each object is categorized as a separate object. Also, depth information using stereo cameras was recorded to help develop 3D scene understanding. The labeling was done using the LabelMe application over on 25,000 images. The labeling of the images was split into two groups, fine pixel-level annotations and course pixel-level annotations. The fine annotations on average took 1.5 h per image to label while the course annotations took around 7 minutes per image to label.

The Mapillary Vistas Dataset builds off of previous on road datasets like the Cityscapes dataset and is created for a similar purpose which is to help the development of road-scene understanding for autonomous navigation of passenger vehicles or mobile robots in urban areas [15]. This dataset was taken by using images from urban environments across the world and included a small amount of highways and rural roads as well. The Mapillary Vistas dataset uses image data that was labeled using instance-level segmentation like in the Cityscapes dataset where each pixel was given a classification for what type of terrain or object the pixel belonged to. Each object was also given an independent label to differentiate the same class of objects from each other. The labeling was done using a custom labeling tool and 25,000 images were labeled at a fine pixel-level. Image annotation was conducted by a team of 69 professional annotators with an average annotation time of 94 minutes per image.

The Bathymetric Mapping Dataset is one of very few public inertia and bathymetry-based datasets available [16]. This dataset is aimed at improving navigation for autonomous underwater vehicles, or AUVs, in navigating open water. Four surveys were taken in different locations around the Georgia and Rhode Island coasts to gather diverse data. This dataset was gathered using video data, IMU, DVL and multi-beam sonar data that was synchronized with GPS data. The sensors were attached to a marine vehicle on the surface of the water which allowed the team to accurately sync the sensor data to the GPS data for the ground truth. Navigation data was fused using ROS

and robot localization to create the structure for the ground truth.

The Caves Dataset is another dataset aimed at developing navigation of AUVs [17]. This dataset differs from the Bathymetric Mapping dataset by aiming for developing AUV navigation in a more closed environment in the caves on the Spanish coast instead of the open water. The location for the data collection was the underwater cave complex "Coves de Cala Viuda" in Costa Brava, Spain. This dataset was gathered using DVL, IMU, depth sensors, and a camera attached to an underwater robot. The ground truth was set by placing easily identifiable markers (in this case traffic cones). The position of the cones were compared to the sensor data to gather an accurate but not perfect ground truth. Navigation data was fused using ROS to create the structure for the ground truth. Images and sensor values were recorded for over 500 m in the cave system.

The RELLIS-3D Dataset is a dataset aimed at improving the robustness of autonomous navigation in off-road environments [18]. The dataset was collected on non-paved trails on the Rellis Campus of Texas A&M University. The data was collected by attaching a camera and LiDAR sensor to an off-road mobile robot. The ground truth consists of image annotations and point-wise annotations for the 3D point clouds. Each pixel in the images was classified by what kind of terrain or object that pixel belonged to. The image annotations was done by Appen, a company that crowdsources data annotation. The annotations for the LiDAR data were done using the application provided by the SemanticKITTI project. The dataset consists of 13,556 LiDAR scans and 6,235 images.

The CAVS Traversability Dataset was created to help make models that can effectively determine traversability for various off-road passenger vehicles in a variety of terrain [7]. This dataset was created by recording non-paved trails and roads around Mississippi State University. Video data was collected from cameras attached to various vehicles and images were selected from the videos for annotation. The images were labeled using semantic segmentation where each pixel was labeled with a class that represented three different levels of traversability. The classes were separated based on which vehicle would be needed to traverse that piece of terrain. Any terrain that was not considered traversable was not labeled and marked as not traversable space when training the machine learning model. The data collected was labeled using BasicAI and consists of 1,812 images.

The Oxford Inertial Odometry Dataset was created to help develop inertial navigation research for pedestrians with smart devices or mobile robots [19]. Inertial navigation can be used to track pedestrian trajectory and generate guides from IMU data. The data was captured in a lab

with a Vicon motion capture system and data was collected using IMU sensors and magnetometers in smart phones. The ground truth was created by syncing the IMU and magnetometer data with the motion capture data to precisely match the location of the testers with smart phones and the inertial data. The data was also split by the different types of devices and different ways that the devices were attached to a person such as in their hand, pocket, handbog, or trolley. The Oxford Inertial Odometry dataset consists of 158 sequences totalling more than 42 km of distance.

The Orientation Network Dataset was created to develop autonomous pedestrian navigation for visually impaired people [20]. This dataset was created for the Intelligent Assistive System for Visually Impaired People (IAS4VIP) project to provide independent mobility in indoor and outdoor environments and this data. The dataset was captured by taking images of sidewalks in Siegen, Germany. These images were labeled using binary semantic segmentation where the sidewalks or traversable area was given a label and all other background area was marked as not traversable. To label this data, they used the LabelMe software for annotation of the sidewalk images. This dataset consists of 941 images.

None of these datasets were acquired for the specific problem of pedestrian navigation in a forested environment. We therefore followed a different strategy. Of particular concern was the amount of time required to label every pixel in an image, when our goal was simply identifying a reasonable path, combined with the fact that the extent of traversable space may be very subjective depending on the person labeling the image. The following section therefore reviews several existing software packages we tried to use for our project.

1.4 Image Labeling Software

To label all of the frames of the videos, annotation software was needed. We tried 2 different annotation platforms which were MATLAB and V7. Each had pros and cons, but ultimately we decided to program our own annotation software to fit our needs. For V7, labeling at a very accurate level as shown in Figure 1.7 was made easier. We tried classifying the area by difficulty like in the project done for the CAVS Traversability Dataset [7]. Our project was heavily inspired by that work and is where our team started when trying to create a outdoor traversability dataset. When trying their method, the labeling took a relatively large amount of time (around 55-70 seconds) to label a single frame. Without concrete systems to quantify the difficulty of the terrain for a hiker, it was

challenging to find agreement on what would be easy, moderately difficult, and difficult to traverse through. Figure 1.7a shows a frame where a labeler has annotated an image with three different classifications for traversability. While we did not time how long on average labeling all pixels in an image would take, we can easily conclude that it would take more time than labeling just the traversable area and be just as complex of a problem. To address this, we chose to switch to a single classification where pixels are either traversable or obstacles as shown in Figure 1.7b. This method shortened the labeling process to around 35 seconds per frame. While these times aren't entirely unreasonable in creating a dataset, generating datasets that are highly adaptable and capable of generalizing across various circumstances would pose significant challenges. The main challenge when using the the polygon method in V7 as a labeling tool was that if the interpolation onto the next frames were not accurate enough, the vertices of the polygons had to be manually adjusted. In Figure 1.7b, there are 14 vertices that might need to be rearranged and in Figure 1.7a, there are many more vertices that would need to be manually modified at each frame which would require a significant amount of manual adjustment.

(a) Traversable area classified by difficulty in V7. (b) Single-class traversable area label in V7.

Figure 1.7: V7 Labeling Techniques for Traversability

MATLAB was another tool that we tried for annotating the videos. While MATLAB also had polygons, we primarily used it to try the superpixel method for labeling frames. With this method, it was made easy to "paint" on the labels instead of having rigid polygons. We found this method helpful as instead of having many vertices to adjust in a polygon, groups of pixels can be changed between the classifications rather easily. While this method was a little faster than using polygons, it still took a longer amount of time due to needing to adjust superpixel sizes, the classification of superpixels, and the MATLAB ground truth labeling tool lacks interpolation, so every frame will need manual adjustment. An interesting concept that was found in MATLAB was the

point tracking algorithm that uses the KLT feature tracking algorithm to track a set of points within a boundary. This point tracking algorithm is intended for use in video stabilization, camera motion estimation, and object tracking. This package in MATLAB was helpful in determining that we should use KLT tracking for interpolating labels since labels for one image could be propagated to the next image in the video sequence if features could be reliably matched between them. The MATLAB implementation of the KLT tracking algorithm required the use of bounding boxes to determine a group of points that could be tracked while our use case for the KLT tracking algorithm needed to only track isolated points on an image. Another issue with using the MATLAB environment was the speed of the platform. Running programs or applications in MATLAB took a much longer time than using V7 which is done over the internet or our C++ application that was developed for this project.

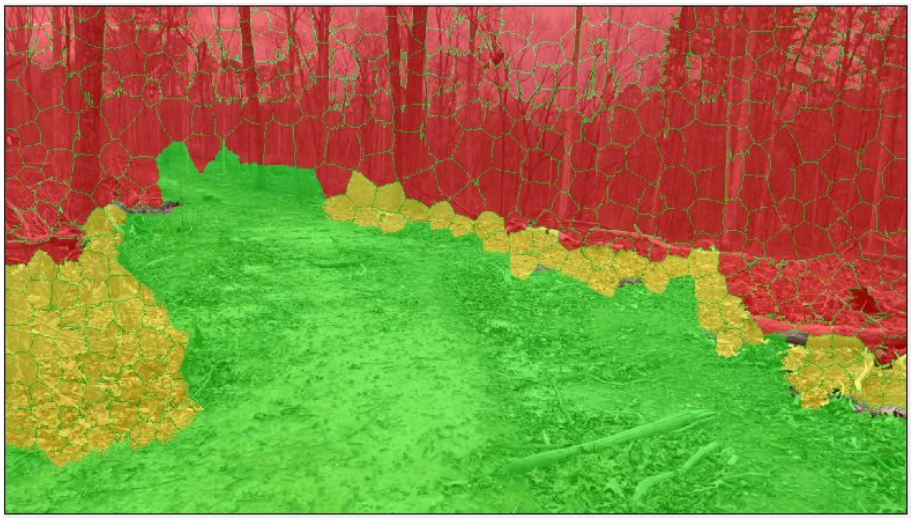


Figure 1.8: Superpixel labeling method in MATLAB.

In the end, we developed our own software to create a system that was much faster than the methods we tried which would include making the actual labeling process easier and include a automated process to speed up the labeling. We decided to test a different labeling method where we would label paths the user should take instead of classifying all of the traversable area. This method assumes that there is enough space (around 1 meter) for a user to traverse through and that there is a theoretical best path to take in a frame. Figure 1.9 shows an example of a draft that we wanted to be our end goal. In the figure, there is a line label that is created by placing a couple points onto the image. Around the line label, is a path with a varying width that is determined by how far away the pixels on the line are from the camera. For labels near the camera, the width will be larger and for

pixels in the upper part of the frame, the width will be smaller.

Figure 1.9: End goal for labeling process.

1.5 Optical Flow Basics

Optical flow is the motion of objects between consecutive frames of sequence, caused by the relative movement between the object and camera. There are two types of optical flow, sparse and dense. Sparse optical flow gives the flow vectors of certain interesting features. It can be automated to choose features such as edges and corners, but it can also be used to manually track points of interest. Dense optical flow gives the flow vectors of all the pixels in a frame [21]. In our case, we used sparse optical flow to track up to 10 points in a frame. Speed of the labeling process was one of our greater considerations so instead of tracking 1280x720 pixels, we instead tracked only 10. There are a couple types of implementations of sparse optical flow, but the most common and the one we used was the Lucas-Kanade method. The Lucas-Kanade method takes advantage of the fact that between consecutive frames of video, the two images are separated by a small time increment such that objects are not displaced significantly. This technique uses spatial intensity gradient information to direct the search of similar items to the position that would likely yield the best match. The technique is very useful for the translational image registration problem which is the task of aligning two or more images that have undergone a translational shift [22]. In our case, we use the Lucas-Kanade method to place points and then track those points across frames by having the program search for the textures on and around the points in the frames ahead or behind the manually adjusted frame.

Figure 1.10: Example of optical flow vectors created in a frame from a video. Pixels with movement like on the horses or the waves have longer vectors that indicate greater motion. Image taken from an article written by Lin [21].

1.6 Traversability Metrics

Determining traversability for off-road conditions poses a greater challenge compared to on-road scenarios due to the unpredictable and dynamic nature of off-road environments. Off-road terrains exhibit a wide range of irregularities, including uneven surfaces, unpredictable obstacles, and diverse textures, making it difficult to model and predict vehicle or robot behavior accurately. Unlike the structured and well-defined nature of roads, off-road paths introduce uncertainties such as varying soil types, changing slopes, and unexpected obstacles like rocks or vegetation. Additionally, factors such as weather conditions and environmental changes further contribute to the complexity of off-road traversability assessment. The absence of clear boundaries and standardized features exacerbates the difficulty of creating reliable models for off-road navigation, requiring more sophisticated sensing, mapping, and decision-making capabilities for autonomous systems to navigate safely and effectively in these challenging environments. In other research, the images were segmented and each pixel was labeled and given a classification to determine what kind of terrain or obstacles are present [18, 23, 24]. A research project done by Sharma et al. had each labeled pixel classified by how traversable the terrain was in the context of difficulty for automobiles [7]. Other research developed a cost map to provide a representation of the cost or quantifiable difficulty value for each location by measuring the roughness of the terrain and comparing it to the robots capabilities to determine if it can traverse

through the area [9, 25]. Terrain traversability analysis is commonly split into two distinct areas of research. The first is identifying the obstacles and/or the traversable area. A robot can then plan a path on the area that is deemed traversable. The second is using the cost map that includes elements such as terrain geometry, expected friction/traction, and the capabilities of the vehicle. Using the quantities created by the cost map, the robot or vehicle maps a path of least resistance [2].

For our project, we create a simple traversability map that contains a binary classification, traversable or not. The inter-rater reliability between two sets of annotations were measured to evaluate the robustness of our labeling method. The time-to-label for each video was recorded as well as problems that slowed down the labeling process or hindered accuracy of the labels. Unlike the methods mentioned above, the metrics collected for this project were used to measure the IRR and speed of the labeling process as opposed to evaluating the performance of a trained AI model.

1.7 Novelty

The main novelty of this work is to attempt to define traversable space for single person dismounted navigation in natural terrain (e.g. hiking). Unlike robots and vehicles, people are more adaptive and can reliably traverse through certain terrain where robots and vehicles would either fail or need to consider taking different paths. For this thesis, we have collected video data from smartphones and then we had two graduate students label a traversable path for each frame in each video. In the data labeling software, the labelers created line labels that represented the best path to take in the videos. The data labeling software was written so that the vast majority of frames are labeled automatically using optical flow to interpolate manual labels onto previous and next frames. Automatically labeling the next frames saves a substantial amount of time. Interpolating labels in the previous frames create a more seamless path which is also more accurate. In another program, the lines were given a width that was dependent on the position of the line in the video frame. These widths created corners of a polygon that contained the generated trail that a hiker was advised to take. Afterwards, the ground truths that were created were compared by finding the intersection over union (IoU) between the labels to determine inter-rater reliability (IRR). The IRR was used to determine how much agreement there was between two independent labelers and to determine the effectiveness of the approach. The total time to label was compared to methods that are already made freely available for academic use. We specifically try to address these questions:

1. How do we define the problem of pedestrian navigation in natural terrain such that navigable space can be objectively defined and labeled?
2. How long does it take to label this type of data?
3. With a definition of navigable space, what is the inter-rater reliability between two labelers using this annotation software?
4. How do we classify issues that can hinder the rate of labeling?

Chapter 2

Methods

2.1 Problem Definition

To identify traversable space in an outdoor environment, we must first define what is traversable. For this project, we are considering what is traversable for a person who is hiking. Naturally, what is considered traversable can vary greatly depending on who is hiking. People who are beginner hikers might consider a traversable area as only the accessible and walkable terrain. Experienced hikers would consider much more area to be traversable if they are willing to climb, walk through shallow water, crouch under obstacles, jump, etc. What we defined as traversable space is in between but much closer to the beginner level. We defined traversable paths as areas that could be walked on but may have smaller obstacles like sparse grass and sticks. We also included paths that had obstacles that required minimal climbing or may require crouching or for the hiker to walk over an obstacle such as logs or large rocks. Some of the paths that were considered traversable could be considered unstable such as paths that would require a person to lean on an obstacle in the environment or carefully walk through to avoid slipping. Areas that would require a person to swim, or walk through dense foliage, or navigate through dangerous terrain were not considered traversable.

(a) Image of an urban cityscape [26]. (b) Example of a cluttered off-road environment.

Figure 2.1: Urban vs. Off-road Conditions.

Creating a labeled dataset for outdoor environments is difficult due to the sheer number of variations that outdoor environments have as well as lacking the clear direction that a man-made path for vehicles or pedestrians would have. Road conditions almost always contain a clear path that obviously stands apart from the environment. For example, roads are easily distinguished using black asphalt with lane markers or gravel which is distinct from the grass, dirt, or pedestrian path around it. For pedestrians, sidewalks and paths are set up similarly where the walking path is set up to be obviously different from the environment and/or roads. Looking at Figure 2.1a, we can see that the urban cityscape has different colors for the driving lanes, for parking, and for the sidewalk. The driving lanes have painted lines that highly contrast with the road to make it easy to differentiate the lanes. Also, in an urban environment, the width of the spaces for cars, buses, and pedestrians remain relatively constant where as in the off-road environment, width of the path can quickly change. The off-road environment in Figure 2.1b portrayed is an extreme case where the trail is completely covered in leaves making it hard to differentiate the trail and the surroundings. In outdoor environments, the size of the expected traversable space will be nonlinear much more often and obstacles can obscure the traversable space often. The off-road terrain generally lacks the objects and cues that make navigating through the cities easier such as road markings and traffic lights. Off-road environments also contain much greater variability and complexity which can include water bodies, uneven terrain, and vegetation. In an urban environment, the traversable areas are kept as uniform as possible to make travel efficient and maximize safety. Outside of man-made trails, nature lacks this level of uniformity.

(a) Example of an easy to label frame.

(b) Example of a hard to label frame.

Figure 2.2: Easy and hard example of frames to label.

In Figure 2.2, we have two frames before labeling. On the left side or Figure 2.2a, is an off-road path that is meant for easy hikes and even passenger vehicles with off-road capabilities. The path is generally flat, constant width, requires no turning in sight, and has no obstacles in the way of the path. Figure 2.2b on the right, would be considered a harder label. Looking at the image, there can be two possible paths, one to the left and a straight path. The straight path, however, is being blocked by an obstacle. The obstacle is still navigable as a hiker and the straight path was taken in the video. While we tended to avoid obstacles, for our definition of traversability, we included that level of adaptability for dismounted navigation. During the filming and labeling process, there can be disagreement on what is traversable in situations like these. Some may consider a fallen tree traversable if they can climb over or crouch under it while others may consider the fallen tree a dead end.

(a) Path with shallow water.

(b) Path that requires climbing. Red line indicates possible path.

Figure 2.3: More examples of frames that are difficult to label.

Figure 2.4: A frame with completely open area.

In Figure 2.3, we are presented with more example of challenges for our definition of traversability. For Figure 2.3a, there is shallow water that would require a hiker to either step through water or make a short jump to avoid the water. For Figure 2.3b, the terrain would require a hiker to climb over rocks and a fallen tree. This situation might require a higher level of physicality and due to the instability of the ground (surface being not flat), a hiker may need to use their hands to safely navigate the terrain. Our definition of traversability includes this list of assumptions:

1. Camera height is kept around 4-5 feet off the ground
2. Camera angle remains straight unless there is an incline or decline
3. In the frame, there is a "horizon" where the amount of traversable space ends
4. There is one best path that is labeled
5. The path has enough space (around 1 meter) to traverse through

Since this project is a first attempt at trying this type of ground truth, there are limitations to our model. The first is that only one path is labeled. Since our labels are based on a "best" path, all other paths or terrain are not labeled as traversable. This can be a severe limitation in more open

areas since there will be multiple paths to take even when going the same direction. For example, Figure 2.2a is considered an easy frame to label, but since the path in this frame is wide, there maybe many ways to place a label on this frame like drawing a line label going straight from the middle or to the left or right. Wide paths or open areas can lower the amount of agreement between labelers. Trying to label a completely open area like in Figure 2.4 would be meaningless with our model as there would be no path to take and no guidance would be needed in an open area situation. In Figure 2.5, we can see the difference in opinion on what the best path to take in a frame is. Because the area is wide and traversable enough so that multiple ways can be taken to reach the same point, the agreement on an "optimal" path can be lower than a normal semantic segmentation label where all the pixels that can be considered traversable are labeled.

Figure 2.5: Example of two ground truths being applied to a single frame.

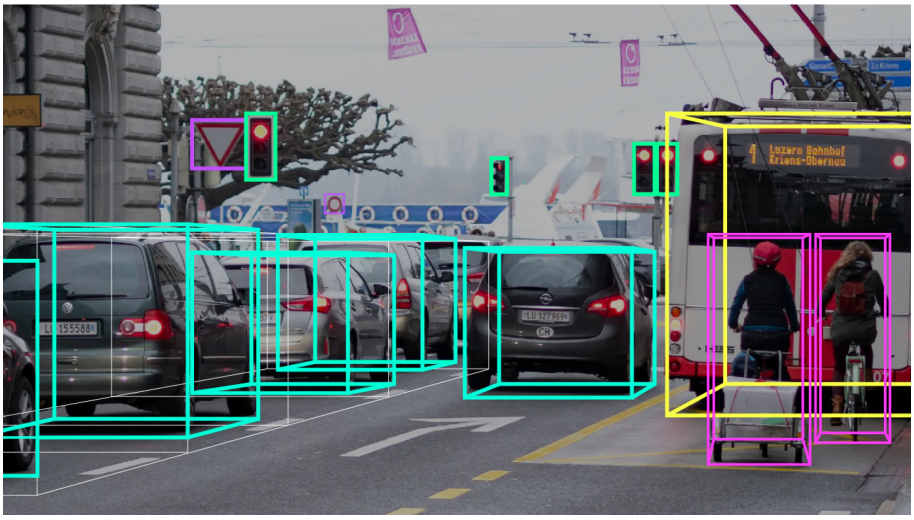


Figure 2.6: Example of a bounding box label [27].

There are several methods for labeling images for dataset creation. These methods can include bounding boxes, polygonal annotation, point annotation, and semantic segmentation. While many of these methods have been effective at handling certain tasks, semantic segmentation has been an especially valuable labelling method for navigation AI because it provides the most information about the surroundings. The other labeling methods such as bounding boxes as seen in Figure 2.6, polygonal annotation, and point annotation focus on key points within an image for object detection and classification. Semantic segmentation provides the most information for navigation as every pixel is given a classification. One issue that semantic segmentation can have is the time and effort needed to create large datasets. For navigation AI in particular, safety and accuracy is an extremely high priority so datasets for training navigation AI need to be highly varied which means a large amount of data needs to be labeled and the data needs to have high amounts of detail in the labeling process. We tried to find a method that could simplify the labeling process while making the labeling heavily automated so that large datasets could be made much faster.

(a) Example of an unlabeled frame.

(b) A frame with a line label.

(c) A frame with a generated path from the line label.

Figure 2.7: Frame labeling process.

For this thesis, we created a data labeling tool where annotators can create line labels that detail a path that a hiker should take. These line labels are created by placing a couple of points onto the frame. These line labels were then used to create polygons that extended from the line labels. The polygons are a path that dictates the traversable space within that frame. We labeled the optimal path in a given frame. This method of creating the ground truth does not ensure that all traversable areas in the image are labeled as traversable as that was not the goal. Also, the ground truth is set up so that each pixel has only two classifications, labeled (best/most traversable path) and unlabeled. To create a segmented image label, there are two main methods: 1) Use a paint tool and paint the pixels with their given class or 2) Create polygon shapes that encapsulate the pixels and organize them by their given class. Using these methods will create very accurate datasets if the annotators are trained properly but is very costly in terms of time. We implemented a method that is intended to speed up the labeling process greatly. For our project, instead of labeling pixels or creating polygons, we created line labels for a given frame that should indicate the best path for the hiker to take. Users of the program can place up to ten points onto a frame where the first point is the beginning of the path and the last point is the end of the path. The width of the path is

determined by the height of the pixels where the top of the image has height zero and the bottom of the image has height 720. The closer the path is to the bottom of the image, the wider the path should be since the bottom of the image contains the area closest to the person holding the camera. As the path goes up the image, the width decreases linearly until it reaches the "horizon" height. We set the horizon point to height 150 in the image where at this point we determined that the pixels located at this area of the image are too far away to create an accurate path. The labeling process is shown in Figure 2.7. When using this simplified and heavily automated process, we reduced our average label time from around 30 seconds per frame to around 0.3-0.4 seconds per frame for both of the labelers.

To summarize, for our data labeling method we wanted:

1. Line labels based on a couple of points.
2. An interpolation process to track those points across other frames.
3. A polygon that expands from the line label to represent the ground truth for the traversable area.
4. The polygon will be a "trail" that illustrates the most traversable path in a frame.
5. The width of the polygon or trail will decrease the further away it is from the camera or base of the image.

The annotation software was developed in C++ in Visual Studio IDE using OpenCV libraries. The GUI was developed using the WIN32 API and can run on Windows computers. While there were open source options available, we used this method due to familiarity with the WIN32 API and C++ programming. The annotation software was made specifically for this project and has several core features that were developed to make annotation as quick and smooth as possible. First, it is limited to line labels only. In our project, we use line labels to designate a path that the user should take through the area. Second, it utilizes optical flow using the Lucas-Kanade technique to automatically interpolate the labels into the next and previous frames. This feature allows the majority of frames in a video to be automatically labeled with little human intervention.

Figure 2.8: Screenshot of the UI of the data labeling tool developed for this project. The UI contains a top toolbar to customize options, the current frame on the left, and supplementary information and instructions on the right.

2.2 Data Collection

The data set that was gathered contains thirty different locations. A location was considered different if it was far away enough from other locations (around 1 mile). A location could also be close in proximity to another location if they were distinctly different trails or if the environment was different enough to consider the locations distinct. They might differ in ground cover or land form such as on a hill or near a lake or river. Some of the locations were somewhat close to each other but were marked as different trails and were treated as different locations when recording. Around five minutes of video were recorded at each location. The videos are around 30 seconds with some variance. Longer videos were clipped so that each location should have about ten thirty-second videos for labeling. During the filming process, we commonly walked and filmed for around 1-2 minutes and then looked for an area with more variety to avoid having the locations having the exact same structure in terms of foliage, sunlight, or traversable space. To record the data, the person with the camera:

1. Held the camera up around 5 feet from the ground.

2. When dealing with elevation, the camera was angled up on inclines and down on declines so that the path was more visible.

3. Walked slowly (no exact measurement was done but should be around 3 mph) so that the movement between frames did not cause too much displacement of features in images.

4. Held the camera as stable as possible in order to minimize motion blur.

5. Tried to record paths that were easily accessible and did not require too much ability or physicality to traverse through.

The filming process was done over two months and most of the video data was collected from locations around the Clemson area with some of the locations taking place in the lower North Carolina area. While the recorded video was around 2.4 hours long, the amount of hiking and video filming was longer so that there was enough variation between the videos. The vast majority of the videos contain areas that are not within other videos except for a few where some of the videos were taken when traversing a trail or path in the opposite direction from another video. Paths that also contained human markings such as signs or trash cans were avoided. Human-made trails are a common occurrence within the data set so the dataset is not based on a completely natural environment, but things like gravel paths, roads, and stairs were generally avoided. The majority of the videos were taken in the fall season which affected the ground cover for our dataset and every video was taken during the day. Very few or none of the videos contain low light conditions. None of the videos required the person filming to jump, walk through very dense foliage, or step in any body of water. Also, none of the videos required the person filming to traverse through terrain unsteady enough where both hands were needed to traverse safely, but there are videos where the person filming had to use one hand for stability. Areas where the user had to crouch or step over fallen trees were sometimes considered traversable if the effort needed was not demanding. The angle of the camera was kept forward generally, but in areas with an incline or decline, visibility of the paths were poor when maintaining a forward angle. The angle in some of the videos was adjusted to account for this where the camera was angled up for inclines and down for declines. Another issue that needed to be addressed was field of view in more cramped conditions. By default, the phone camera has a slight zoom in effect compared to normal vision when taking video. The zoom creates an effect where the default field of view usually omits the ground or path close to the camera. With the default settings on an iPhone 10, the recorded area is around 2.5 meters in front of the camera when the camera is

held five feet off the ground, held straight with no angle, and the zoom is left on 1x. Any ground area in front of the camera within 2.5 meters is not within the frame using this method. In some instances, where there were heavy obstructions from foliage, a steep incline, or a sharp turn, there was little to no space to place labels which made the path very short or impossible to label. To help alleviate this, some of the data was filmed using a 0.6x zoom so that more of the ground closer to the camera was included in the frame. At 1x zoom, the area in the frame that is visible starts around 2.5 meters away from the camera while at 0.6x zoom, the visible area starts around 2 meters away from the camera.

We attempted to create a generalized dataset but to do so we came up with a list of factors where each of its variations would need to be accounted for to achieve a truly generalized dataset:

- Location (ground type, clutter, canopy)
- Time of day/lighting from sun
- Angle/Height of the camera
- Weather
- Time of year (foliage/fauna/sun)
- Velocity of movement while camera is held
- Shakiness, horizontal movement of camera
- Camera lens (field of view)

To create a dataset that accounts for all of these factors and to do so independently would need an enormous amount of video recordings such that it was decided that creating a truly generalized dataset with our scope was not feasible. Instead, we opted to only focus on varying the locations of the videos for our dataset. Our dataset contains 30 unique locations with around 300 seconds of video at each location. Our recordings were taken at 30 frames per second so our dataset contains about 2.4 hours of video and 260,169 labeled frames. The labeling process took around a month to complete. If we were to create a dataset that was recorded to make sure that each factor was explored independently, we would need a dataset that is possibly hundreds of hours long with millions of labeled frames. In our dataset, there is variation that is not limited to just the location such as time of day, shakiness, and field of view that has noticeable effects on the dataset and labeling

process. Video taken from the smartphones were recorded in 1920x1080p. To make the labeling process easier, the videos were first resized to be 1280x720p before labeling using VLC Media Player software. This step was very helpful since it allowed the frame to easily fit on the displays that were used during the labeling process. Though resizing could have been done using OpenCV in the labeling software, we found it made the labeling software much slower to interpret the compressed data and resize it instead of using VLC Media Player which uncompressed the data first then resized it. The uncompressed video data took up much more space but ran much faster which was a higher priority for our project. The process to resize the photos was:

1. Download and open VLC Media Player software
2. Go to the top left and click the option "Media"
3. Add the video or videos that you plan on converting
4. Click "Convert/Save" at the bottom of the window
5. In the convert menu, change the profile by clicking the wrench icon
6. Under "Encapsuplation", set to "MP4/MOV"
7. Under "Video codec", set encoding parameters to have:
   (a) Codec to MPEG4
   (b) Frame rate to 30 fps
8. Under "Video codec", set resolution to have:
   (a) Scaling set to Auto
   (b) Frame size set to Width = 1280px and Height = 720px
9. Click "Save"
10. Set "Destination File" to the location where you want the converted videos to be saved
11. Click "Start" and then wait for the video to be converted

2.3 Labeling Software

The labeling application was built from scratch in Visual Studio IDE with C/C++ using Win32 API. It consists of two main components: a user interface (UI) with video playback controls, and an interpolation module. It is assumed that the user will label an image, use the interpolation module to fill in labels for the next set of frames, then skip ahead and continue labeling. Every time a manual label is done, interpolation is done backwards to smooth out the labels between manually adjusted frames. The following sections describe each in detail.

2.3.1 UI and Controls

The labeling process starts with loading in a video. Labeling sessions can be continued by loading in a .csv file that contains labels from a previous labeling session. The normal process for users was to place points onto a frame. Once more than one points was placed, a line is generated that connects the two points. This line represents the middle of the path that a hiker should take when traversing the terrain shown in the frame. These points can be freely modified to change the direction or length of the path. The points can also be easily deleted to reduce the amount of points in the line label when necessary. The lines between the points should automatically change each time a point is added or changed. To manually label a frame, the user can either place points or load in points from the last frame. This feature is particularly helpful when interpolation fails and the user want to continue labeling from points that were placed previously. The UI has controls to allow the user to move forwards and backwards through the video by single frame or multiple frames. The user may also view the video normally to observe the data or to review the labels. The UI contains a toolbar at the top for users to load in files, adjust how many frames get interpolated, toggle backwards interpolation, and to adjust how many frames get skipped when using fast forward/backward. Most importantly, the UI allows the labelers to interpolate frames forwards and backwards. Frame interpolation is done to track points placed onto the current frame and to track them in the next set of frames. The next section describes the interpolation process in detail.

2.3.2 Interpolation

Algorithm 1 Forward Interpolation Loop

while $iterations < nFrames$ **do** ▷ nFrames is the number of frames that will be interpolated

1) Update GUI on progress

2) Read in next frame and set to grayscale

3) Implement function for Optical Flow

4) Check the status of each output point:

if All points are good **then**

Use points on next frame

else if One or more points were lost **then**

Exit loop and give the user a warning that interpolation has stopped

5) Check if first point is too low in the image:

if $y \geq YMax$ **then**

Set y-value to Max Size - Optical Flow Window

Calculate new x-value

6) Check if last point is too close to edge of image:

if $x \leq XMin$ OR $x \geq XMax$ **then**

Set x-value to boundary

Calculate new y-value

6) Record points into ground truth

7) Increment *iterations*

The interpolation process is coded to fill in labels for all frames between any two manually labeled frames. The optical flow is first calculated from the oldest frame forwards to the newest frame. The user then skips to the newest frame and adjusts the labels to match the traversable path. When the newest frame is adjusted, the edits made are interpolated backwards to the last manually adjusted frame. Each direction of optical flow yields different results. The final result is taken as a weighted average of the two optical flow computations where the weight of each is highest for the frame that is closest to the starting point and the weight drops off as the optical flow calculations continue further from the starting point. The goal is to weight the impact of the manual points based

on their distance over time from the last manual points used to make predictions.

The weighting and interpolation steps are complicated by the fact that path points can vertically move up and down the image plane as an image sequence is analyzed due to the walking motion of the camera holder. This can result in path points disappearing (falling off the bottom of the image) and new points appearing (a farther point appearing in the second image that was not labeled in the first image). We used several equations and heuristics to solve these problems. Algorithm 1 details the main optical flow code, and the following text describes the extra conditions tested for special cases.

During forward interpolation, the first label point near the bottom will drop off quickly because the optical flow algorithm will continue to track points the user has defined until the video has moved past where the point should be and the bottom point is no longer in the frame. When the first point is is low enough in the image where the optical flow window would extend outside the boundaries of the image, the y-value of the first label point in the image is adjusted. To do this, Equations 2.1, 2.2, 2.3, 2.4 are used to adjust the location of the first point.

$$\text{slope} = \frac{\text{nextY} - \text{currentY}}{\text{nextX} - \text{currentX}} \tag{2.1}$$

$$\text{y-intercept} = \text{nextY} - \text{slope} \cdot \text{nextX} \tag{2.2}$$

$$\text{newX} = \frac{\text{Ymax} - \text{y-intercept}}{\text{slope}} \tag{2.3}$$

$$\text{newY} = \text{Ymax} \tag{2.4}$$

Forward interpolation will cancel when there is an issue like motion blur or glare, but the most typical disruption to forward interpolation will be when the camera has moved far enough so that the first and second point will have the same y-value in the image. Since the first point is kept along the bottom of the image, the second point will often drop to the same value of the first point. When this happens, the interpolation process will cancel and the user will have be given a text message to readjust the points. Once the first and second points are separated, the interpolation process can continue. Also, there are times when the video has turning which can cause interpolation

to cancel due to points being dropped off in the x-direction. To fix this, a similar solution is applied as above but with the last point only as it is the point to usually have issues. In this situation, we set a fixed boundary for the x-value and found a new y-value using Equations 2.1, 2.2, 2.5, and 2.6.

$$\text{newY} = \text{slope} \cdot \text{XBound} + \text{y-intercept} \tag{2.5}$$

$$\text{newX} = \text{XBound} \tag{2.6}$$

During backwards interpolation, the location of the points of the ground truth are changed using a linear equation with weights. In Equation 2.7, the point data, PD, contains the x and y values of a point recorded into the ground truth after backwards interpolation has been applied. The point data is calculated by adding the forwards interpolated data (FID) multiplied by the forwards weight (FW) and the backwards interpolated data (BID) multiplied by the backwards weight (BW). The weights are determined by first finding a distance through taking the number of frames between the current frame being manually adjusted and the last frame that was manually adjusted. The increment used to adjust the weight is simply a fraction using the distance as described in Equation 2.8. At the beginning of the backwards interpolation process the backwards weight (BW) is set to 1 minus the increment while the forwards weight (FW) is set to 0 plus the increment. The interpolation process starts from the current frame and works backwards to reach the last manually adjusted frame. For each frame, the BW is decremented and the FW is incremented by the value found in Equation 2.8. The points in the frames closer to the frame that was just adjusted are weighted more towards the newest adjustments where the points in the frame closer to the last manually adjusted frame are weighted towards the forward interpolation done for the last manually adjusted frame.

$$\text{PD} = \text{FID} \cdot \text{FW} + \text{BID} \cdot \text{BW} \tag{2.7}$$

$$\text{increment} = \frac{1}{\text{number of frames in between}} \tag{2.8}$$

Another issue that needed to be solved is when points were added or removed before backwards interpolation. To solve the first situation where a point is removed before backwards interpolation, the labeling software tracks the index of the point that was deleted. Then, while

backwards interpolation is being applied, the index of the deleted point is not adjusted. For example, a user has four points at frame 0 and then forwards interpolates those four points 167 frames ahead. At frame 167, the user deletes the second point and then adjusts all the other points. When backwards interpolation is applied with the current frame that only has three points, the program will interpolate the values of the points onto points one, three, and four while not moving the point two for the previous frames that had four points placed. For the second situation, where a point is added before backwards interpolation, the point is simply added onto the previous frames. However, this solution can cause issues where the added point gradually moves up in the image as the interpolation process moves backwards through the frames. For this project, the points were only tracked to a certain height on the image so if a point had a y-value that was too low, it was removed from the interpolation process. To clarify the top of the image has a y-value of zero and the bottom of the image had a value of 720 in our case. When the point that was added would reach what we called the "horizon" which in our case was set to 150, the point was removed from the interpolation process. The main purpose of the backwards interpolation process that was implemented was to help smooth the path created for the ground truth. Without backwards interpolation, the path in the ground truth will often change abruptly which could only be reduced by more manual labeling.

2.4 Demonstration, Analysis, and Dataset Software

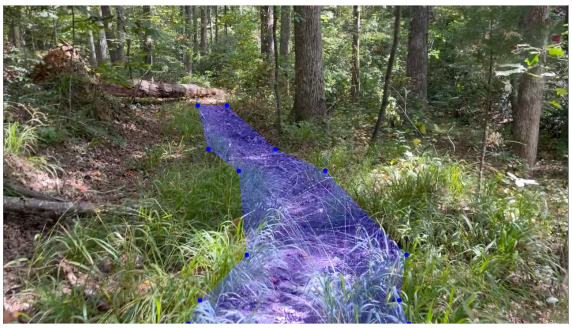
Four programs were developed in Python to create results from the labels created with the C++ labeling tool. The first program was created in Python to demonstrate the results of the labels created with the labeling software by displaying the paths created while the video is playing. The Python software first reads in the frame of the video then loads in the points from the .csv file. Using the points from the ground truth .csv file, a polygon of the path is generated where the height of the points determines the width of that section of the polygon as described in the Section 2.1. Our project utilizes a max width of 300 pixels to determine the traversable area for our data. Using the generated polygon, a colored mask is applied to the frame that gives a transparent color where the path should be. The main feature of the program is that it can accept two different ground truth .csv files. It will generate two different polygons, assign a color to each and then display them in the video. The overlap is a third color that is a mix of both of the colors. As seen in Figure 2.5, two different annotators have labeled the image with red and blue paths. The overlap of the two paths

has a more pinkish color. The overlap of the two paths is calculated to find the intersection of the two labels. The total colored area is the union of the labels. This program is also used to calculate the IoU of the two different ground truths by dividing the intersection area with the union area. For each frame, the area of the overlap (intersection) of the polygons is compared to the total area of the polygons added together (union). The mean IoU is calculated for a video by adding the IoU of each frame and then dividing by the total number of frames. The Polygon and OpenCV libraries were necessary to create the polygon paths and the video demonstration.

To find the optimal max width, a range of max widths were tested on the dataset to determine which yields the best IoU. With a width of 0, the IoU is guaranteed to be 0. With a width of 1280 which was the max size of the video recorded, the IoU will likely be close to 1. A lower width generally has higher accuracy of the traversable space but will lead to a decrease in the agreement or IoU between two labelers. A higher width can lead to higher IoU, but the generated paths will likely contain more terrain that is not traversable. The max widths that were tested ranged from 100 pixels wide to 625 pixels wide. Figure 2.9 contains three figures that show different widths being applied to a line label. Figure 2.9a has the smallest width at 100 pixels wide. When the width of the path is narrow, it is likely for the path to accurately map onto traversable area, but it is less likely for two labelers to have high agreement due to the overlap in less narrow spaces being limited. Figure 2.9b shows the same frame but with a max width of 300 pixels. Figure 2.9c shows the same frame again but with a max width of 500 pixels. The path generated in this image is larger than the previous two and completely covers the traversable terrain, however, it also contains a lot of the tall grass in the surrounding area. While IoU is high for generated paths for wide widths, it runs a higher risk of labeling terrain that is not traversable.

(a) Path with max width = 100.

(b) Path with max width = 300.



(c) Path with max width = 500.

Figure 2.9: Example of a labeled frame with different max widths for the generated path.

To relate the max width to a real-world length, a photo of a meter long measuring tape was taken for Figure 2.10. At default settings for an iPhone 10, the meter long measuring tape was measured to be 351 pixels long in the image. When using a max width of 300 for the base of the image, a generated path would be 282 pixels long at the height in the image where the measuring tape was placed. Comparing the length of pixels in Figure 2.10 to the generated path means that the generated paths should be around 0.803 meters long.

Figure 2.10: Image with measuring tape to compare to max width.

Another software was developed in Python to analyze the schema that were developed as well as the time to label for each annotator. This software calculated time statistics such as mean time needed to label a video. A standard deviation, min, and max for label times were also recorded. Most importantly, this software calculated the occurrences of issues that were found. Issues that slowed down the labeling process were recorded into the schema. Lastly the software generated histograms to display the distribution of time to label for a video. This software was designed to find the information of multiple labelers and to be able to easily compare the values that were generated from the schema.

The last Python software was responsible for creating the dataset. It uses the code from the first Python software used for demonstration purposes and packages the frames and the annotation into a usable dataset ready for training a machine learning model as shown in Figure 2.12. First, it generates a polygon like in the earlier Python software, but it uses the polygon to apply a mask to the original frame so that all pixels within the polygon are white and all pixels outside the polygon are black. This masked frame is saved as the ground truth for the dataset. Figure 2.11 shows an example of what is generated for our ground truth. Figure 2.11a is a frame taken from one of the videos and Figure 2.11 is the ground truth created from the Python software given the line labels from the annotation tool we developed.

(a) Original Frame (b) Ground Truth from Path Label

Figure 2.11: Example of one frame and its annotation for the dataset.

```
dataset/
└── location1/
    ├── video1/
    │   ├── frames/
    │   │   ├── frame1.jpg
    │   │   ├── frame2.jpg
    │   │   └── ...
    │   └── annotations/
    │       ├── frame1_mask.png
    │       ├── frame2_mask.png
    │       └── ...
    ├── video2/
    │   ├── frames/
    │   │   ├── frame1.jpg
    │   │   ├── frame2.jpg
    │   │   └── ...
    │   └── annotations/
    │       ├── frame1_mask.png
    │       ├── frame2_mask.png
    │       └── ...
    └── ...
```

Figure 2.12: Markdown of dataset structure.

2.5 Metrics

Our main goal was to create a data annotation tool that could be used to label video data faster and with high inter-rater reliability (IRR). Inter-rater reliability is the degree of agreement among independent observers who rate, code, or assess the same phenomenon. IRR between labelers was measured to test the consistency and validity of our method. Consistency among raters indicates

that the data collection is reliable and reproducible. Methods with high IRR minimize the potential for biases introduced by individual raters which means that the method can be generalized to a broader sample or group of labelers. Consistently reliable data is crucial in developing machine learning models. To determine inter-rater reliability, we use IoU which was gathered using the Python program that was developed to analyze the label data made in the C++ annotation program. First, the points from the labels created in the annotation tool are used to create a polygon. The polygon is made by extending two points horizontally from the label points as corners for the polygon. This polygon is the path that is created from the labels. To find IoU the intersection of the polygons between the two sets of labels is calculated. Then the union of the labels is calculated. The IoU for a frame is the intersection area divided by the union area. If there is no union then the IoU is zero.

Another set of metrics was recorded to try and find the average time to label. The distribution of labeling time was analyzed to explore potential correlations with a predictive model, aiming to accurately forecast labeling time. This analysis encompassed identifying minimum and maximum labeling times, as well as the standard deviation for each labeler. The labeling process was done almost completely independently for this project. Since we implemented a novel methodology that was meant to serve as a base model for a labeling process that can be used on a much larger scale, we intentionally minimized communication regarding labeling instructions. Instead, the labelers individually documented any obstacles encountered that impeded efficiency or accuracy in the labeling process. While both labelers adhered to the guidelines outlined in Section 2.1, a structured analysis highlighting issues encountered during labeling, excluding bugs and errors, was undertaken separately, without collaboration, to identify potential challenges that may arise. Each labeler also tracked the frequency of each of the problems that occurred across the whole dataset.

Chapter 3

Results

3.1 Time to Label

In total, 284 videos were recorded and labeled. Outside of a very few frames that contained no traversable area, the vast majority (well over 99%) of the 260,169 frames were labeled. The average number of frames in a video was 916 frames and since the videos were recorded at 30 frames per second, the average video length was 30.54 seconds long. The dataset statistics are summarized in Figure 3.1. During the labeling process, the labelers recorded the time needed to label each video for 24 of the locations which encompasses 80% of the data. On average, Labeler 1 took 6.14 minutes to label and Labeler 2 took 4.69 minutes to label. With the average video length being 30.54 seconds we can estimate that the mean labeling rate for the labelers was 0.40 and 0.31 seconds per frame respectively. If we were to generalize our results to other labelers, they can estimate that labeling a dataset of this size (2.4 hours of video) should take 22.4-28.9 hours total.

Video Statistics	Count
Total Frames	260,169 frames
Total Number of Videos	284 videos
Average Frame Count	916 frames
Average Video Length	30.54 seconds

Table 3.1: Video statistics from data collection.

Labeling Time Statistics	Time for Labeler 1 (min)	Time for Labeler 2 (min)
Mean Time	6.14	4.69
Median Time	5.56	5.00
Standard Deviation	2.62	1.73
Minimum Time	1.87	1.00
Maximum Time	15.58	15.00

Table 3.2: Time statistics from labeling.

Although the mean time is different between the two labelers, both of the standard deviations are similar in size proportionally to the mean suggesting a similar distribution of results. The histograms of labeling times can be seen in Figures 3.1 and 3.2. When looking at the histograms, the labeling times fit a normal distribution somewhat closely as the histogram for labeler 1 is mostly bell-shaped, but the histogram suggests a slight right skew as there are more data points to the left of the mean than there are to the right. Also, the mean is 10.4% greater than the median which indicates a slight right skew. Looking at the histogram most of the label times fit within a small range minus some exceptions where labeling took over 12 minutes. Figure 3.2 shows the distribution of label times for Labeler 2. This histogram also resembles a normal distribution as there is high symmetry around the mean and a distinct bell shape. The mean and median are similar as they are 4.69 and 5.00 minutes respectively but the median is 6.6% greater than the mean suggesting a very slight left skew.

Calculations were also done to evaluate the skewness of the label times. To evaluate skewness, two different values were calculated, Pearson's median skewness coefficient also known as Pearson 2 skewness coefficient and Fisher Pearson skewness. The equation for Pearson 2 skewness coefficient is:

$$\text{Pearson 2 Skewness} = 3 \cdot \frac{\text{Mean - Median}}{\text{Standard Deviation}} \tag{3.1}$$

The Pearson 2 skewness statistic is simple and has appeal due to how intuitive it is to understand. It is easy to interpret and generally also matches a visual interpretation of the skewness of the data. With Equation 3.1, a distribution will have a right skew if skewness coefficient is positive and a left skew if the skewness coefficient is negative. Looking at Table 3.3, the Pearson 2 skewness coefficients match our quick overview of the skew from the mean and median statistics as well as the visual

interpretation of the skew. Labeler 1 had a slight right skew coefficient of 0.658 and Labeler 2 had a slight left skew coefficient of -0.535. However, under testing, this metric lacks power which in statistics is the probability that a statistical test will correctly identify a real effect. Statistical applications no longer use Pearson 2 skewness calculations and instead opt for Fisher Pearson Skewness instead [28]. The Fisher Pearson skewness coefficient formula is:

$$\text{Fisher Pearson Skewness} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3}^1. \tag{3.2}$$

The Fisher Pearson skewness coefficient formula calculates the skewness by dividing the third moment of the distribution by the cube of the standard deviation. Essentially, it measures how much the distribution deviates from symmetry. Although it is harder to interpret intuitively, it is less prone to error and demonstrates higher power compared to the Pearson 2 skewness statistic and is often used in applications to measure skewness. When using Equation 3.2, we calculated that Labeler 1 had a moderate right skew coefficient of 1.202 and Labeler 2 had a moderate right skew coefficient of 1.102. Using this metric, Labeler 1 had a stronger left skew than expected, but Labeler 2 was calculated to have a moderate right skew instead of a slight left skew. One key takeway from the Fisher Pearson skewness coefficients is that both label time distributions have a moderate right skew. It also suggests that the distributions are very similar as the magnitudes of the two distributions only differ by 0.1.

[1]In the formula for Fisher Pearson Skewness, n represents the number of observations, x_i denotes each individual observation, $\bar{x}$ is the sample mean, and s is the sample standard deviation.

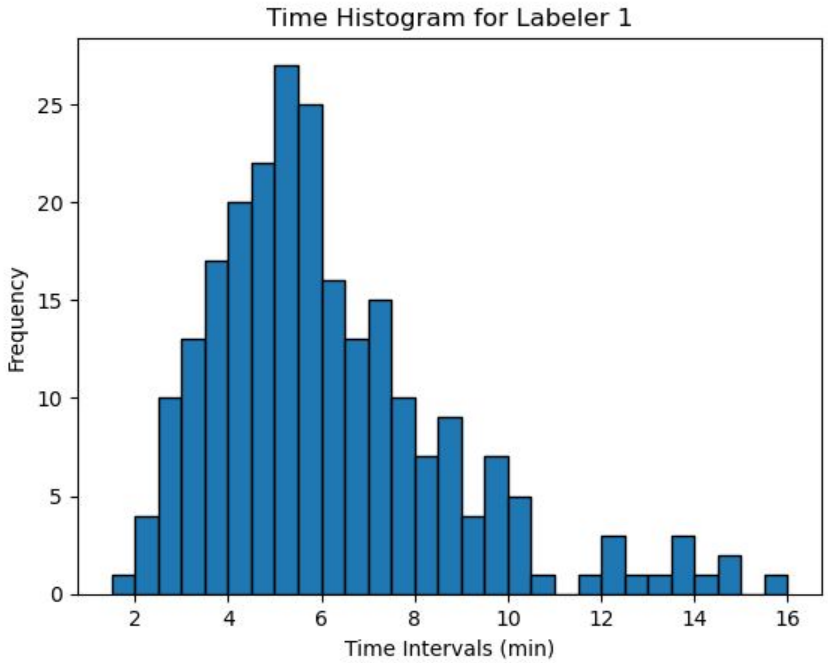


Figure 3.1: Labeling time distribution for Labeler 1.

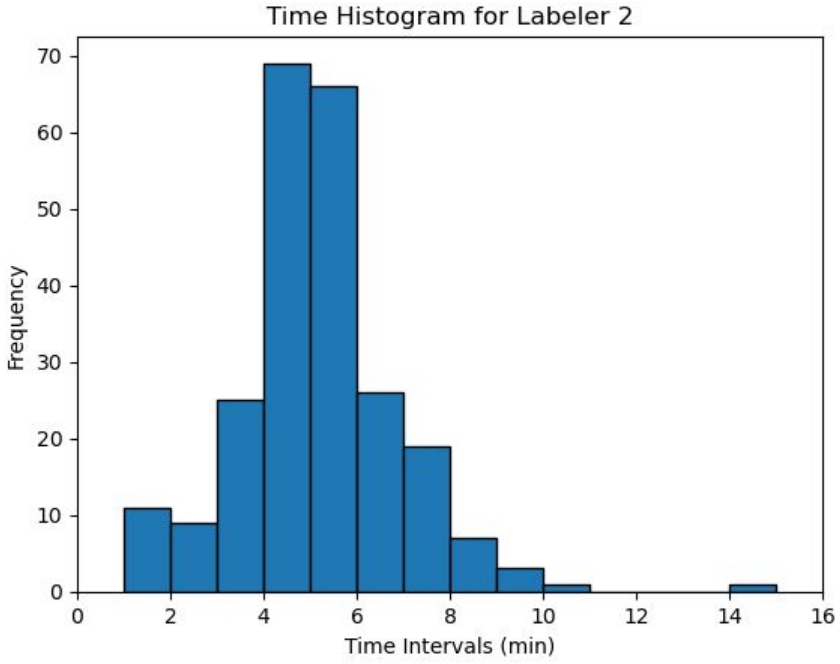


Figure 3.2: Labeling time distribution for Labeler 2.

Labeler	Pearson 2 Skewness Coeff.	Fisher Pearson Skewness Coeff.
Labeler 1	0.658	1.202
Labeler 2	-0.535	1.102

Table 3.3: Skewness calculations results from label times.

3.2 Labeling Issues

As stated in the Section 2.5, the two labelers independently recorded issues that affected the accuracy and speed of the labeling process. The limited amount of collaboration was intentional to observe the issues highlighted during labeling and to simulate the experience of a new user utilizing this method and software for the first time. Both of the labelers recorded issues they ran into and then categorized their notes into different groups. These groups can be seen in Table 3.2 where the effect and solution during the labeling process is given to each problem as well as which labeler reported the problem as an issue during labeling. For example, both labelers recorded curves, glare, and motion blur as an issue when they were labeling. However, the labelers recorded multiple issues that the other labeler did not. For example, Labeler 1 reported issues with the field of view (or FoV) of the camera and obstructions while Labeler 2 reported short and narrow paths as issues. Some of the issues were similar but were named differently and had slightly different properties like open area issue for Labeler 1 and two paths issue for Labeler 2.

Problem	Effect	Solution	Labeler with Issue
Curves	Curves cause interpolation to drop off quickly and may obscure the path ahead.	Extra manual labeling is needed to make sure that the path correctly moves around the curve.	1,2
Glare	Glare disrupts optical flow.	Frames with high amounts of glare will need to be manually labeled frame by frame. To do so, the labels from the frame prior are copied into the current frame and then adjusted.	1,2

Motion Blur	Motion blur disrupts optical flow.	Frames with high amounts of motion blur will need to be manually labeled frame by frame. To do so, the labels from the frame prior are copied into the current frame and then adjusted.	1,2
Open Area	Open areas makes it difficult to choose a "best" path.	The video will need to be viewed carefully before labeling is done and the path that was taken in the video is the path that will be labeled.	1
Obstructions	Obstructions make the visible path smaller and can disrupt optical flow.	When obstructions disrupt optical flow, the frames will need to be manually adjusted to make sure that the path is accurate.	1
FoV Issues	Higher FoV causes the ground close to the camera to be out of frame which makes labeling the path in cramped conditions difficult or impossible.	When labeling, some frames had no labels and were not counted if there was no traversable space. After this issue was found, the focal length was shortened so that more of the ground was captured.	1
Dead Ends	Dead ends have little to no traversable area once reached.	Leave frames empty or label the little traversable area left.	2
Bright Spots	Bright spots disrupt optical flow.	Issue and solution is similar to glare.	2
Short Path	Short paths have very little change and require very little manual labeling.	Label normally.	2
Narrow Path	Narrow paths have very little visible traversable area due to high density of obstructions.	Consider the width of the path carefully and add additional points to accurately trace the trajectory.	2

Two Paths	Two available paths makes is so that a single path must be determined to be "optimal".	Observe the video further to determine the path it followed and only label the corresponding route while disregarding others.	2
No Issues	The video contains no content that would make the labeling process more complex or less accurate.	Label normally.	1,2

Table 3.4: Common issues and their effect on the labeling process.

The frequency of the problems found in Table 3.2 was also recorded and can be seen in Table 3.5 and Table 3.6. Both labelers agreed that there were issues that slowed down the labeling process on a majority of the videos but Labeler 2 reported issues at a higher frequency than Labeler 1. Both labelers also agreed that motion blur was the most common issue for labeling. This outcome was expected as the off-road paths are not paved like man-made paths such as roads and sidewalks. Off-road terrain can have roughness that makes traversing through the terrain to be less stable compared to urban pathways made for pedestrians or roads for vehicles. While some of the data was taken on smoother trails, the variance in elevation, ground cover, and obstacles cause extra movement that leads to motion blur during the filming process. Motion blur breaks the interpolation process which can slow down the labeling process greatly. Labeler 2 reported motion blur as an issue for about half of the videos taken while Labeler 1 reported motion blur on 35.15% of the videos. Glare was also a common issue for both labelers as both recorded it as an issue and at a somewhat common frequency. From the Table 3.5 and 3.6, Labeler 1 recorded slightly more lighting issues compared to Labeler 2. Curves was another issue that both labelers recorded as a problem but there were distinct differences on the frequency of the problem. Labeler 1 reported the issue at double the rate of Labeler 2 for curving issue. The most distinct difference between the two tables was that Labeler 2 heavily focused on recording issues that were about the path taken in the video while Labeler 1 mostly focused on how the environment affected the video. Labeler 2 recorded dead ends, short paths, narrow paths, and two paths as issues while Labeler 1 only recorded open areas as an issue with the paths recorded in the video and at a very low rate of 5.02% while Labeler 1 recorded

obstructions and obstacles as a major issue in the labeling process while Labeler 2 did not.

Problem	Frequency for Labeler 1
Curves	22.18%
Glare	17.57%
Motion Blur	35.15%
Open Area	5.02%
Obstructions	19.25%
FoV Issues	6.69%
No Issues	33.05%

Table 3.5: Common issues and frequency of issues for Labeler 1.

Problem	Frequency for Labeler 2
Curves	10.97%
Glare	12.24%
Motion Blur	49.79%
Dead Ends	7.59%
Bright Spots	2.12%
Short Path	11.39%
Narrow Path	8.02%
Two Paths	5.91%
No Issues	24.05%

Table 3.6: Common issues and frequency of issues for Labeler 2.

Table 3.7 gives the average time to label for both labelers depending on which problem was reported. If the labeler did not record a specific issue, the average time is left as "NA". For Labeler 1, the No Issues group had an average time of 3.99 minutes while the average time of the No Issues group for Labeler 2 was 4.58 minutes. The average time for the videos that presented no issues was somewhat similar for both labelers. However, the time for both of the labelers was very different once issues were reported. For any videos that recorded issues, Labeler 1 had an average label time of 7.21 minutes while Labeler 2 had an average label time of 4.73 minutes. When Labeler 2 recorded an issue, there was not a significant increase in time because Labeler 2 reported on issues that they felt impacted label accuracy but not label time such as short path, or narrow path issue. Labeler 2's average time to label also seemed to be less impacted overall from issues compared to Labeler 1. When Labeler 1 had an issue, the time to label drastically increased. A major similarity that exists between both average times is that glare both drastically affected both label times where it is reported as Labeler 1's second most impactful problem and Labeler 2's most impactful problem.

Video Group	Avg. Time for Labeler 1 (min)	Avg. Time for Labeler 2 (min)
No Issues	3.99	4.58
With Any Issues	7.21	4.73
Has Curves	8.02	5.85
Has Glare	8.61	6.07
Has Motion Blur	7.82	4.97
Has Open Area	7.14	NA
Has Obstructions	7.96	NA
Has FoV Issues	9.41	NA
Has Dead End Issue	NA	5.11
Has Bright Spots	NA	5.40
Has Short Path Issue	NA	2.44
Has Two Path Issue	NA	5.57
Has Narrow Path Issue	NA	4.47

Table 3.7: Effect of problems on labeling time.

The following figures contains examples of frames that contained issues for Labeler 1. Labeler 1 recorded issues with curves, glare, motion blur, open area, obstructions, and FoV. As stated in Table 3.2, curves cause interpolation to drop off quickly compared to more straight paths. When looking at Figure 3.3, this frame shows a hiking trail that is curving towards the left. As the person filming moves along the trail, the camera will rotate to the left. When the camera is rotated, the area in the frames quickly change which means that interpolating points becomes less effective as the newer areas must be labeled manually.

Figure 3.3: Frame with curving issue.

When looking at Figure 3.4, this frame contains high glare which disrupts interpolation. Sometimes data was recorded while facing the sun. When the sun shines directly on the camera lens, distortion from glare occurred. The optical flow algorithm for interpolation relies on comparing the color, changes in color, and edges between two frames. Glare not only very quickly changes the colors in the image, but the rays also adds edges to the images. Frames affected by glare had a very high chance of requiring manual labeling.

Figure 3.4: Frame with high glare.

Figure 3.5, shows an image that contains a high level of motion blur. Motion blur also tends to disrupt optical flow similar to glare. The source of the motion blur came from the movement

of the camera during the filming process. Off-road terrain often times is not stable like indoor or urban pathways so the camera experienced motion often during filming. Our optical flow algorithm is dependent on analyzing the edges near the label points and motion blur causes the edges to disappear which disrupts interpolation. Frames with motion blur also had a high chance of requiring manual labeling.

Figure 3.5: Frame with motion blur.

The open area problem was recorded when it was difficult to tell at a glance which direction to label due to having a large amount of traversable area or many occurrences of multiple pathways. Looking at Figure 3.6, there are two paths forward, one to the right and one to the left. In this instance, the labeler had to re-watch that section of the video before labeling so that the labeled path matched the path taken in the video. When multiple occurrences happened, it slowed the labeling process. When an area was very open with high amounts of traversable space, it took time for the labeler to re-watch that section of the video to find which direction the person filming had walked. It also took extra time to think of an "optimal" path to take in the video since there were many options compared to a smaller more restrictive area.

Figure 3.6: Frame that contains open area.

Figure 3.7 shows a common problem where obstacles obstruct the view of the path. Obstructions mainly cause two issues. The first is that the path must be manually labeled once the obstruction has been passed to label the new traversable area in the frame. The second and main issue is that it can confuse optical flow. The edges of the terrain are used to track the location of a label point and overhangs from fauna or obstacles such as trees or rocks have distinct edges that are tracked when the label point is close enough. Instead of tracking the terrain, it will instead stick to the edge of the obstruction which will need manual fixing.

Figure 3.7: Frame with obstructions.

Figure 3.8 shows an example of a frame that is hard to label due to field of vision or FoV

issues. The path taken in the video is to the right but the tree and the FoV makes it so that the path is barely visible. As mentioned in Section 2.2, holding the camera straight at 5 ft and using default or 1x zoom on an iPhone 10 causes the frame to start showing the ground at around 2.5 meters away. In cramped conditions, this leads to the traversable space being hard to see. A portion of the videos taken later were filmed using 0.6x zoom so that the ground appeared in frame around 2 meters away which helped alleviate some of the FoV issues.

Figure 3.8: Frame with FoV issue.

3.3 IRR Measurements

As mentioned in Section 2.4, a range of different max widths was tested. The paths that were created for the ground truth are widest at the bottom of the frame and the width gradually decreases with the height in the frame. The max width indicates the width at the bottom of the frame. With a very high max width, IRR will be high but the labels will be less accurate since it will include a lot of the surrounding area around the traversable area that may contain obstructions. With a very low max width, IRR will be very low but the labels will by more accurate in mapping the traversable space. We tested a range of max widths from 100 pixels wide to 600 pixels wide by measuring IoU of the entire dataset across the entire range and incremented the max width by 25 pixels each time. As seen in Figure 3.9, max width has a positive relationship with IoU but the relationship is not linear. Since the relationship is not linear we chose our max width to be a value that represents the knee of the curve. Using the Kneed library in Python, the knee of the curve was

determined to be where the max width was 300 pixels wide with an IoU of 0.6060. All of the other IoU values can be seen in Table 3.8.

Max Width (pixels)	Total IoU
100	0.3183
125	0.3745
150	0.4246
175	0.4652
200	0.5020
225	0.5324
250	0.5604
275	0.5840
300	0.6060
325	0.6246
350	0.6422

Max Width (pixels)	Total IoU
375	0.6574
400	0.6718
425	0.6843
450	0.6964
475	0.7069
500	0.7172
525	0.7262
550	0.7350
575	0.7428
600	0.7505

Table 3.8: Effect of changing max width on IoU between labelers.

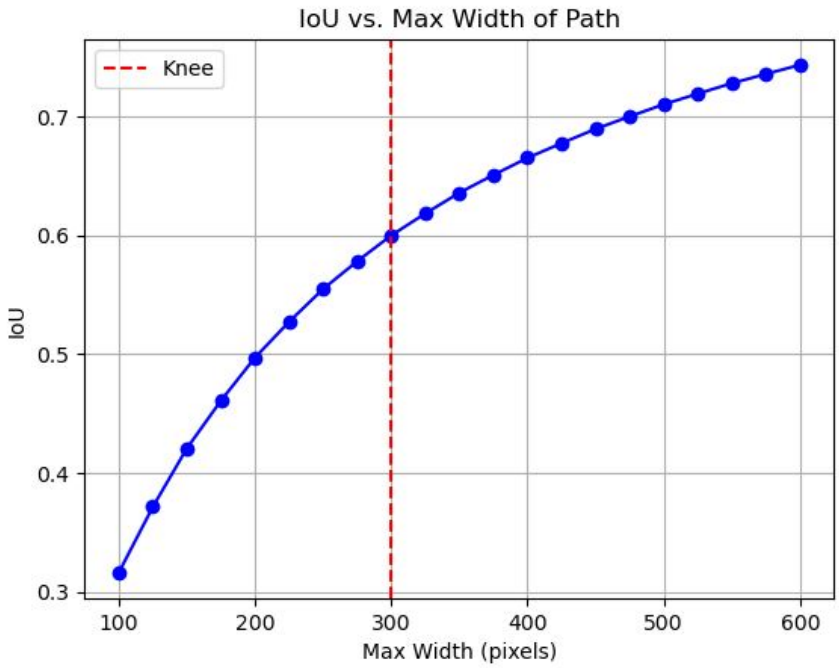


Figure 3.9: Graph of the relationship between max width and IoU.

Chapter 4

Conclusion

4.1 General Discussion

In this thesis, we developed a method for identifying traversable terrain for dismounted navigation in order to help develop an autonomous navigation tool that can help hikers navigate forested off-road terrain. Over 2.4 hours of video data was collected to help make a dataset for this purpose. We also created a tool to help label the video data that was collected. Our goal was to create a data labeling tool that could be used to quickly label large datasets by simplifying the labeling process to line labels and also semi-automating the labeling. Table 4.1 contains all of the datasets mentioned in Section 1.3, but with the new Clemson Hiker Helper dataset. Looking at Table 4.1, our dataset contains an extremely large amount of data compared to other datasets that contain labeled image data. The trade-off is that the ground truth format for our dataset is much simpler being binary segmentation where the area around the line labels is considered traversable space and everything else is considered not traversable. Our research team had two people collect and label the data. Metrics for time to label and IRR were collected to evaluate our definition of traversability and the effectiveness of our annotation tool that we developed. The experiments in this work answer four questions:

> How do we define the problem of pedestrian navigation in natural terrain such that navigable space can be objectively defined and labeled?

We defined traversable area as terrain that could be walked on but may have smaller obstacles

like grass and sticks. Since a human is more adaptable then a mobile robot, we also included areas that had obstacles that required crouching under or walking over logs and large rocks. Areas that would require a person to swim, walk through dense foliage, or navigate through dangerous or unstable terrain were not considered traversable. We also assume traversable area to have enough space (about 1 meter) for a human to walk through.

How long does it take to label this type of data?

We had two annotators label 284 videos which include 260,169 frames. The times to label for each video was recorded on 80% of the data and on average the mean label rate for the two annotators was 0.40 and 0.31 seconds per frame. The mean label rate for our method was much faster compared to semantic segmentation labeling methods we tried with MATLAB and V7 which took at least 35 seconds per frame and possibly over 1 minute depending on the level of detail required.

With a definition of navigable space, what is the inter-rater reliability between two labelers using this annotation software?

We had our annotators create line labels that outlined an optimal path that a hiker should take for our ground truth. These line labels were then extended into polygons where the width of the polygon was dependent on where the point was placed. To determine the inter-rater reliability between the two labelers, we used intersection-over-union or IoU of the polygons that were generated. IoU was found by taking the overlapping area of the polygons and dividing that by the total area of the polygons when put together. The IoU is used as our metric to determine the agreement between both of the annotators. At a max width of 300 pixels for our path, the IoU was 60.60%.

How do we classify issues that can hinder the rate of labeling?

Both of the annotators recorded the time to label and issues that they felt hindered the speed and/or accuracy of the labeling process. The annotators labeled and recorded issues mostly independently. The amount of collaboration was intentionally limited to simulate the experience of a new user using our labeling software. When an annotator ran into a problem that increased the label time or had a negative effect on labeling the traversable space accurately, they noted the issue down. The issues were grouped into distinct categories once the labeling process was complete.

4.2 Limitations

Our definition of traversable space had limitations that affected the IRR between our two labelers. First, the path generated for our ground truth assumes that the traversable space in an image is 0.8 meters according to our max width. If an area contains more open space, then there will be space that is traversable but not marked as such. This limitation decreases IRR since multiple users may choose different ways to go forward on a path if there is enough space for variation. Second, our definition of traversable space also was not concrete enough to eliminate bias between labelers. Since human movement is much more adaptable compared to a robot's, our definition of traversability included most terrain that would be considered walkable but not challenging enough to put a hiker in a dangerous or uncomfortable position. Different labelers may consider different situation as traversable or not traversable. Our broad definition of traversable space may have had a negative effect on agreement between labelers which limited IRR. Third, our definition of traversable space was limited to a single path. Our goal was to train data based on the best path to take in a video, but multiple scenarios we recorded had branching paths. If two paths are equally viable, there is no way in our current method to address taking multiple paths in one video. For labeling, both of our labelers decided to label the path that was taken in the video so this limitation may not have affected IRR, but can limit the accuracy and ability of a machine learning model that has been trained on this data.

Our method of labeling also contains limitations that negatively affected label time and accuracy. First, our method assumes that the path starts near the center of the image and is connected to the bottom of the image. Our current method does not allow paths in the distance to be considered a viable path. This limitation makes it so that areas in the frame may not be considered traversable even if the terrain is perfectly traversable but not directly connected to the base of the frame. Second, our method also assumes that the width of the path consistently resembles the traversable space in a frame well across all locations. When labeling, the width of the traversable space varied significantly as some paths were narrow and meant only for human traversal while other paths were created wide enough to accommodate off-road passenger vehicles. The width that we used to generate our dataset was determined from analyzing how the max width of the path affected our average IoU. This limitation of using a single max width for all locations leads to untraversable area being labeled as traversable in some locations while traversable area goes mostly unlabeled

in other locations. In Section 3.2, issues were recorded during labeling. Motion blur and glare in particular were large issues that were reported by both labelers. Reducing the frequency of these issues in the filming process or improving the annotation tool could lead to lower label times.

4.3 Future Work

There are a variety of paths this project may take in the future. The work done for this project was meant to serve as a pilot project to aid in the development of a "Hiker Helper" tool that would autonomously navigate for a hiker in an off-road environment. Our main objective was to create a method in gathering and labeling data. To create good data for training a model, it is important to limit the disagreement between labelers to create consistent data that is labeled properly. Our definition of traversability could be further improved to limit individual bias between labelers. Currently, our definition is too vague and the addition of a more strict rule set could be made to provide clear guidelines for labelers on how to identify and label traversable space. This rule set could be made by performing a survey and asking people what kind of terrain they would be willing to traverse through and what difficulty they expect. Using this data, we could specify the size of obstacles that would make a path not traversable. We would also could find examples of what kind of inclines/declines or foliage that would make a path not traversable.

Improvements on the annotation tool developed for this project could be made to so that the labels for traversable space is more accurate and can be made faster. Currently, our label tool only allows for one path to be labeled and the path must be connected to the base of the frame in the video. Future work could be done to allow labeling of multiple paths or paths in the distance. The annotation tool could also be improved by allowing the labelers see the paths generated from the line label. Adding this feature would also enable a feature that could allow labelers to vary the width of the paths during the labeling process. Adding the ability to change the widths during the labeling process would let the labelers more accurately label the traversable space in the frames for paths that are more open or narrow. To help alleviate the issues with labeling, improvements to the video taking process can be done. Steps could be taken to minimize motion blur in the video like increasing shutter speed or using stabilization equipment. Glare could be minimized by avoiding taking videos in direct sunlight.

The most crucial future work would be to train an machine learning model using the dataset

generated. The dataset created in this project could be used for supervised learning of an artificial intelligence since the entirety of the dataset is labeled. A model could be trained to test whether or not our method for creating and labeling data can be used to make a navigation tool that can autonomously find traversable terrain and provide guidance in navigation.

Dataset Name	Type of Navigation Problem	Locations / Environment	Label Software	Format of GT	Amount of Data
Cityscapes Dataset [14]	On road passenger vehicles	City streets	LabelMe	Pixel wise semantic segmentation of images	25,000 images
Mapillary Vistas Dataset [15]	On road passenger vehicles	Various locations and settings	Not described	Instance-specific semantic segmentation of images	25,000 images
Bathymetric Mapping Dataset [16]	Underwater robotics	Open Water	ROS and Robotic Localization	Video data, IMU, DVL and multibeam sonar data is synchronized with GPS data	4 surveys were taken
Caves Dataset [17]	Underwater vehicle	Underwater cave complex "Coves de Cala Viuda"	Not described	Contains depth, video, imaging sonar, DVL, and IMU recordings matched with markers	Data was gathered over 500m
RELLIS-3D [18]	Off road mobile robot	Off-road environment in Bryan, TX	Image Data: Appen Point Cloud: App by SemanticKITTI project.	Pixel wise semantic segmentation of video data. Point-wise annotation for 3D point clouds from LiDAR data.	13,556 LiDAR scans and 6,235 images
CAVS Traversability Dataset [7]	Off road passenger vehicles	Off-road environment in Starksville, MS	BasicAI	Pixel wise semantic segmentation of video data	1812 images
Oxford Inertial Odometry Dataset [19]	Indoor pedestrian	Indoor area	Vicon Motion Capture	IMU data matched with location, velocity, and orientation	158 sequences totalling more than 42 km distance
Orientation Network Dataset [20]	Outdoor pedestrian for visually impaired	Sidewalks in Siegen, Germany	LabelMe	Binary semantic segmentation	941 images
Clemson Hiker Helper Dataset	Outdoor pedestrian navigation	Off-road environments near Clemson, SC and Charlotte, NC	Custom annotation tool	Binary semantic segmentation of traversable path	260,169 frames

Table 4.1: Comparison of our new dataset to selected other datasets.

Bibliography

[1] C. Sevastopoulos and S. Konstantopoulos, "A survey of traversability estimation for mobile robots," *IEEE Access*, vol. 10, pp. 96331–96347, 2022.

[2] P. Borges, T. Peynot, S. Liang, B. Arain, M. Wildie, M. Minareci, S. Lichman, G. Samvedi, I. Sa, N. Hudson, *et al.*, "A survey on terrain traversability analysis for autonomous ground vehicles: Methods, sensors, and challenges," *Field Robot*, vol. 2, no. 1, pp. 1567–1627, 2022.

[3] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," *Journal of modern transportation*, vol. 24, pp. 284–303, 2016.

[4] D. Parekh, N. Poddar, A. Rajpurkar, M. Chahal, N. Kumar, G. P. Joshi, and W. Cho, "A review on autonomous vehicles: Progress, methods and challenges," *Electronics*, vol. 11, no. 14, p. 2162, 2022.

[5] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, *et al.*, "A2d2: Audi autonomous driving dataset," *arXiv preprint arXiv:2004.06320*, 2020.

[6] J. Melo and A. Matos, "Survey on advances on terrain based navigation for autonomous underwater vehicles," *Ocean Engineering*, vol. 139, pp. 250–264, 2017.

[7] S. Sharma, L. Dabbiru, T. Hannis, G. Mason, D. W. Carruth, M. Doude, C. Goodin, C. Hudson, S. Ozier, J. E. Ball, *et al.*, "Cat: Cavs traversability dataset for off-road autonomous driving," *IEEE Access*, vol. 10, pp. 24759–24768, 2022.

[8] Y. Bai, B. Zhang, N. Xu, J. Zhou, J. Shi, and Z. Diao, "Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review," *Computers and Electronics in Agriculture*, vol. 205, p. 107584, 2023.

[9] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning ground traversability from simulations," *IEEE Robotics and Automation letters*, vol. 3, no. 3, pp. 1695–1702, 2018.

[10] S. Khan, S. Nazir, and H. U. Khan, "Analysis of navigation assistants for blind and visually impaired people: A systematic review," *IEEE access*, vol. 9, pp. 26712–26734, 2021.

[11] Q. Wang, M. Fu, J. Wang, H. Luo, L. Sun, Z. Ma, W. Li, C. Zhang, R. Huang, X. Li, *et al.*, "Recent advances in pedestrian inertial navigation based on smartphone: a review," *IEEE Sensors Journal*, 2022.

[12] F. E.-Z. El-Taher, A. Taha, J. Courtney, and S. Mckeever, "A systematic review of urban navigation systems for visually impaired people," *Sensors*, vol. 21, no. 9, p. 3103, 2021.

[13] M. Hughes, "Gatwick airport launches indoor navigation system to help passengers find their way." `https://thenextweb.com/news/gatwick-airport-launches-indoor-navigation-system-hellscape`. [Accessed 23-03-2024].

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[15] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proceedings of the IEEE international conference on computer vision*, pp. 4990–4999, 2017.

[16] K. Krasnosky, C. Roman, and D. Casagrande, "A bathymetric mapping and slam dataset with high-precision ground truth for marine robotics," *The International Journal of Robotics Research*, vol. 41, no. 1, pp. 12–19, 2022.

[17] A. Mallios, E. Vidal, R. Campos, and M. Carreras, "Underwater caves sonar data set," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1247–1251, 2017.

[18] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis," in *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 1110–1116, IEEE, 2021.

[19] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni, "Deep-learning-based pedestrian inertial navigation: Methods, data set, and on-device inference," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4431–4441, 2020.

[20] O. Gamal, S. Thakkar, and H. Roth, "Towards intelligent assistive system for visually impaired people: Outdoor navigation system," in *2020 24th international conference on system theory, control and computing (ICSTCC)*, pp. 390–397, IEEE, 2020.

[21] C.-e. Lin, "Introduction to motion estimation with optical flow," Apr 2023.

[22] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, pp. 674–679, 1981.

[23] F. Vulpi, A. Milella, R. Marani, and G. Reina, "Recurrent and convolutional neural networks for deep terrain classification by autonomous robots," *Journal of Terramechanics*, vol. 96, pp. 119–131, 2021.

[24] H. Wu, W. Zhang, B. Li, Y. Sun, D. Duan, and P. Chen, "Visual terrain classification methods for mobile robots using hybrid coding architecture," in *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*, pp. 17–22, IEEE, 2019.

[25] S. Chhaniyara, C. Brunskill, B. Yeomans, M. Matthews, C. Saaj, S. Ransom, and L. Richter, "Terrain trafficability analysis and soil mechanical property identification for planetary rovers: A survey," *Journal of Terramechanics*, vol. 49, no. 2, pp. 115–128, 2012.

[26] D. de Geus, P. Meletis, C. Lu, X. Wen, and G. Dubbelman, "Part-aware panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5485–5494, 2021.

[27] C. Lasorsa, "An introduction to bounding boxes [+ best practices]."

[28] D. P. Doane and L. E. Seward, "Measuring skewness: a forgotten statistic?," *Journal of statistics education*, vol. 19, no. 2, 2011.

[29] F. E.-Z. El-Taher, L. Miralles-Pechuán, J. Courtney, K. Millar, C. Smith, and S. Mckeever, "A survey on outdoor navigation applications for people with visual impairments," *IEEE Access*, vol. 11, pp. 14647–14666, 2023.

[30] C. S. Johnson, S. Mahajan, M. Ordu, S. Sherugar, and B. N. Walker, "Will o'the wisp: Augmented reality navigation for hikers," in *HCI International 2016–Posters' Extended Abstracts: 18th International Conference, HCI International 2016 Toronto, Canada, July 17–22, 2016 Proceedings, Part II 18*, pp. 365–371, Springer, 2016.

[31] J. L. Martínez, J. Morales, M. Sánchez, M. Morán, A. J. Reina, and J. J. Fernández-Lozano, "Reactive navigation on natural environments by continuous classification of ground traversability," *Sensors*, vol. 20, no. 22, p. 6423, 2020.

[32] C. Niu, C. Newlands, K.-P. Zauner, and D. Tarapore, "An embarrassingly simple approach for visual navigation of forest environments," *Frontiers in Robotics and AI*, vol. 10, 2023.

[33] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS research*, vol. 43, no. 4, pp. 244–252, 2019.

[34] L. F. Oliveira, A. P. Moreira, and M. F. Silva, "Advances in forest robotics: A state-of-the-art survey," *Robotics*, vol. 10, no. 2, p. 53, 2021.

[35] M. W. McDaniel, T. Nishihata, C. A. Brooks, P. Salesses, and K. Iagnemma, "Terrain classification and identification of tree stems using ground-based lidar," *Journal of Field Robotics*, vol. 29, no. 6, pp. 891–910, 2012.

[36] V. WHO, "Global status report on road safety 2018," *World Health Organization*, 2018.

[37] J. Zhao, B. Liang, and Q. Chen, "The key technology toward the self-driving car," *International Journal of Intelligent Unmanned Systems*, vol. 6, no. 1, pp. 2–20, 2018.

[38] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[39] G. Chen, Z. Mao, H. Yi, X. Li, B. Bai, M. Liu, and H. Zhou, "Pedestrian detection based on panoramic depth map transformed from 3d-lidar data," *Periodica Polytechnica Electrical Engineering and Computer Science*, vol. 64, no. 3, pp. 274–285, 2020.

[40] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Autonomous robots*, vol. 18, pp. 81–102, 2005.

[41] C. Thorpe and H. Durrant-Whyte, "Field robots," in *Robotics Research: The Tenth International Symposium*, pp. 329–340, Springer, 2003.

[42] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE expert*, vol. 11, no. 2, pp. 19–27, 1996.

[43] M. D. Adams, "Lidar design, use, and calibration concepts for correct environmental detection," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 753–761, 2000.

[44] S. Badal, S. Ravela, B. Draper, and A. Hanson, "A practical obstacle detection and avoidance system," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pp. 97–104, IEEE, 1994.

[45] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, *et al.*, "Toward reliable off road autonomous vehicles operating in challenging environments," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 449–483, 2006.

[46] P. Papadakis, "Terrain traversability analysis methods for unmanned ground vehicles: A survey," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1373–1385, 2013.

[47] E. Uğur and E. Şahin, "Traversability: A case study for learning and perceiving affordances in robots," *Adaptive Behavior*, vol. 18, no. 3-4, pp. 258–284, 2010.

[48] B. Bayat, N. Crasta, A. Crespi, A. M. Pascoal, and A. Ijspeert, "Environmental monitoring using autonomous vehicles: a survey of recent searching techniques," *Current opinion in biotechnology*, vol. 45, pp. 76–84, 2017.

[49] V. A. Jorge, R. Granada, R. G. Maidana, D. A. Jurak, G. Heck, A. P. Negreiros, D. H. Dos Santos, L. M. Gonçalves, and A. M. Amory, "A survey on unmanned surface vehicles for disaster robotics: Main challenges and directions," *Sensors*, vol. 19, no. 3, p. 702, 2019.

[50] R. Lösch, S. Grehl, M. Donner, C. Buhl, and B. Jung, "Design of an autonomous robot for mapping, navigation, and manipulation in underground mines," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1407–1412, IEEE, 2018.

[51] X. Xing, R. Zhou, and L. Yang, "The current status of development of pedestrian autonomous navigation technology," in *2019 26th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, pp. 1–3, IEEE, 2019.

[52] D. Pascolini and S. P. Mariotti, "Global estimates of visual impairment: 2010," *British Journal of Ophthalmology*, vol. 96, no. 5, pp. 614–618, 2012.

[53] R. Staff, "Velodyne Touts Latest Lidar Advances, Autonomous Vehicle Partnerships — roboticsbusinessreview.com." `https://www.roboticsbusinessreview.com/unmanned/unmanned-ground/velodyne-touts-latest-lidar-advances-autonomous-vehicle-partnerships/`. [Accessed 28-12-2023].

[54] W. Sharon Rothacker, W. Sheryl Vanderwalker, W. Tim Beckham, W. Cheri Kimball, W. Travis Gordon, W. Doug Reed, W. Andrew Crites, W. Todd Sholty, W. Bob Jones, R. Roehm, and et al., "Lesson five: Inclement weather conditions (7.5)."

[55] G. Capi and H. Toda, "Development of a new robotic system for assisting visually impaired people," *International Journal of Social Robotics*, vol. 4, pp. 33–38, 2012.

[56] J. J. Leonard and A. Bahr, "Autonomous underwater vehicle navigation," *Springer handbook of ocean engineering*, pp. 341–358, 2016.

[57] Ben, "Crosswalk safety and driving in nyc - parking tickets." `https://parkingtickets.org/ny-new-york/crosswalk-safety-and-driving-in-nyc/`. [Accessed 23-03-2024].

[58] C. Wang, C. Cheng, D. Yang, G. Pan, and F. Zhang, "Underwater auv navigation dataset in natural scenarios," *Electronics*, vol. 12, no. 18, p. 3788, 2023.