SELF-SUPERVISED NEAR-TO-FAR LEARNING FOR TERRAIN-ADAPTIVE OFF-ROAD AUTONOMOUS DRIVING

by

ORIGHOMISAN MAYUKU

A thesis submitted to the Graduate Program in the Department of Mechanical and Materials Engineering in conformity with the requirements for the degree of Doctor of Philosophy

Queen’s University
Kingston, Ontario, Canada
February 2022

Copyright © Orighomisan Mayuku, 2022
Autonomous vehicles are an increasingly important tool for exploring unknown off-road environments where navigation decisions must include considerations of terrain traversability, unlike in urban environments that have engineered roads. In this thesis, a new self-supervised method is proposed for prioritizing traversable terrain while autonomously navigating a vehicle to a goal position in an unknown off-road environment. Leveraging the colour discriminant bias of off-road terrain types, and using images from a vehicle-mounted camera, a homography that maintains the real-world planar layout of the terrain is employed to cluster terrain types by colour and autonomously register corresponding traversability characteristics such as roughness and slip for terrain-adaptive navigation. As it navigates the vehicle, the algorithm also generates training images for use in contemporary end-to-end navigation schemes. Compared to the existing expert-guided near-to-far approaches, the test results demonstrate the higher autonomy introduced by the proposed approach for navigating off-road environments with unknown traversability characteristics, and highlight its fit to contemporary supervised semantic segmentation schemes that require foreknowledge of traversability characteristics, are limited by insufficient data, and suffer significant class imbalance and poor cross-domain performance. Finally, the effectiveness of non-discretionary self-supervised image labelling is discussed.
Acknowledgments

Thanks to my supervisors, Profs. Brian W. Surgenor and Joshua A. Marshall, for accepting me in their labs and giving me the freedom to explore my research interests while generously providing the support and guidance that I needed—I gained important skills in initiating, developing and managing research projects. And thanks to the Niger Delta Development Commission (NDDC), Queen’s, and the Natural Sciences and Engineering Research Council of Canada (NSERC, Grant DNDPJ 533392-18) for the opportunity and financial support.

Thanks to Jeremy Roy and Michael Fader for keeping Offroad Robotics’ documentation updated; their support facilitated my research. My appreciation to Dr. Keyur Joshi for showing me the ropes around Queen’s. I’m grateful for the research space and extra resources provided by Ingenuity Labs.

Thanks Mum for your perennial sacrifice, and thanks Maye for your selflessness. Anino, I’m very grateful for your loving patience and understanding.
# Contents

Abstract  

Acknowledgments  

Contents  

List of Tables  

List of Figures  

List of Abbreviations and Symbols  

Chapter 1: Introduction  

1.1 Background and Motivation  

1.2 Problem Definition  

1.3 Objective  

1.4 Organization of Thesis  

Chapter 2: Literature Review  

2.1 Terrain Traversability Estimation in Off-Road Environments  

2.2 CNN-Based Supervised Methods for Off-Road Autonomous Navigation  

2.2.1 Off-Road Datasets and Challenges  

2.2.2 Semantic Segmentation with RGB-Only Inputs vs Multi-Modal Fusion  

2.2.3 Proposed RGB-Texture Fusion for Improved Semantic Segmentation  

2.2.4 Results from RGB-Only vs RGB-Texture Fusion Inputs  

2.2.5 Challenge of Domain Adaptation in Off-road Environments  

2.3 Semi-Supervised Methods for Navigation in Off-Road Environments  

2.4 The Proposed Self-Supervised Near-to-Far Approach  

2.5 Chapter Summary  

Chapter 3: The Image Processing Pipeline 38
3.1 Single-Shot Image Processing 38
3.1.1 Perspective Transformation 38
3.1.2 Visual Features and Clustering 43
3.2 Analysis of Image Processing Parameters 46
3.2.1 Colour Difference 47
3.2.2 Colour Difference in Illumination Changes and $E_T$ 51
3.2.3 SLIC Parameter $p_s$ 51
3.2.4 DBSCAN Supercluster Parameter $E_I$ 56
3.3 Chapter Summary 59

Chapter 4: Traversability Characterization, Feature Registration, Path Planning and Cost Design 63
4.1 Traversability Characteristics and Estimation 63
4.2 Self-Supervised Feature Registration and the Specific Path Planning Requirements 70
4.3 Path Planning for Non-Binary Terrains 72
4.3.1 A Naive A* Path Planner and Traversability Cost Metric Design 76
4.3.2 Sampling Based Motion Planning (SBMP) for Traversability- and Workspace-Aware Path Planning 86
4.3.3 SBMP Evaluation Function 97
4.4 Terrain Undulations, Obstacle Avoidance and Speed Adaptation 102
4.5 Summary of the Motion Planning Strategy 118

Chapter 5: Experimental Methods and Results 120
5.1 Localization, Navigation and Control 120
5.2 Experimental System Overview, Test Facilities and Locations 123
5.3 Experimental Procedure and Results 124
5.3.1 Results from Indoor Tests with A* Path Planner 127
5.3.2 Results from Field Tests with A* Path Planner 135
5.3.3 Results from Indoor Tests SBMP 142
5.4 Training a CNN with the Labelled Images 142
5.5 Some Limitations of the Proposed Approach 149

Chapter 6: Conclusions 153
6.1 Contributions 154
6.2 Future Work 155

References 157

Appendix A: System Flow Chart and Pseudocodes 179
<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>System Flow Chart</td>
<td>179</td>
</tr>
<tr>
<td>A.2</td>
<td>Pseudocodes</td>
<td>181</td>
</tr>
<tr>
<td>B.1</td>
<td>Nonlinear Least Squares for Homography</td>
<td>182</td>
</tr>
<tr>
<td>B.2</td>
<td>The Feedback Linearized Controller</td>
<td>184</td>
</tr>
<tr>
<td>C.1</td>
<td>Hardware Specifications</td>
<td>188</td>
</tr>
<tr>
<td>D.1</td>
<td>Applicable Environments</td>
<td>192</td>
</tr>
</tbody>
</table>
List of Tables

2.1 FCN8 on SOOR dataset (IOU). .............................................. 24
2.2 FCN8 on Freiburg Forest dataset (IOU). ............................. 27
2.3 Multi-scale networks on SOOR dataset (IOU). ........................ 27
2.4 Dataset classes for multi-domain test. ................................. 29
2.5 Multi-domain testing results (IOU). ....................................... 30

3.1 List of tuned image processing parameters. .......................... 46
3.2 Colour difference for the terrain types in Figure 3.4 with respect to the tilled soil terrain type. ............................... 49
3.3 Colour difference for the terrain types in Figure 3.5 with respect to the tilled soil terrain type in Figure 3.4. ......................... 50
3.4 Colour difference for the grass patches in Figure 3.6 with respect to the dark patch captured with the camera facing away from the sun. 52
3.5 Colour difference for the sand samples in Figure 3.7 with respect to the dark patch. .................................................. 53
3.6 Colour difference for the road samples in Figure 3.8. ............... 54

4.1 Traversal data for the indoor terrain types in Figure 4.1 at 0.5 m/s using Test Configuration 1. ........................................... 80
4.2 Traversal data \( \bar{a} \) from different tests for the indoor terrain types in Figure 4.1 at 0.5 m/s using Test Configuration 2. 81

4.3 Traversal costs for outdoor Sample Terrain C. 102

C.1 Key specifications of Husky. 188

C.2 Sensor specifications. 189

C.3 Vicon motion camera system specifications. 190

C.4 Deep learning test rig. 191
List of Figures

1.1 Sample urban vs off-road environments. .............................. 2

2.1 Imbalance in the Freiburg Forest dataset. ............................. 14

2.2 Image sizes in the Freiburg Forest Dataset. Due to the disparate sizes, the aspect ratios change after resizing to the uniform CNN input size. 16

2.3 Imbalance in the SOOR dataset. ........................................ 17

2.4 Sample images from the off-road datasets. .............................. 18

2.5 Sample LBP texture representations for off-road environments at different resolutions. ............................................. 22

2.6 Predictions from the SOOR dataset. ...................................... 25

2.7 An illustration of the proposed self-supervised near-to-far approach. 35

2.8 Architecture of the proposed method. ..................................... 36

2.9 Technical steps in the proposed method. ................................. 37

3.1 Illustration of the perspective transformation between the ROI in the source image and the transformed view. ............................... 39

3.2 The image processing pipeline. ............................................. 45

3.3 Collage of terrain types showing sample sections for analysis. ....... 47

3.4 Box plot comparison of terrain samples with mainly achromatic colour components. ............................................................... 49
3.5 Box plot comparison of terrain samples with mainly chromatic colour components. ............................... 50
3.6 Box plot comparison of grass samples at different illumination levels. ................................. 52
3.7 Box plot comparison of sand samples at different illumination levels. ...................................... 53
3.8 Box plot comparison of road samples at different illumination levels. ....................................... 54
3.9 SLIC superpixels and resultant DBSCAN clusters for varying $p_s$ ............................................. 55
3.10 SLIC superpixels and resultant DBSCAN clusters for varying $p_s$ (cont’d). ............................ 56
3.11 SLIC superpixels and resultant DBSCAN clusters for varying $p_s$ (cont’d). ............................ 57
3.12 Clustering results for Figure 3.2 with $E_I = 17$. ................................................................. 57
3.13 Detailed clustering results for Figure 3.11(a) with $E_I = 8.5$ .............................................. 58
3.14 Detailed clustering results for Figure 3.11(a) with $E_I = 17$. .............................................. 58
3.15 Detailed clustering results for Figure 3.11(a) with $E_I = 12$. .............................................. 59
3.16 Clustering results from including the achromatic component in the SLIC and DBSCAN clustering steps only. ................................................................. 60
3.17 Clustering results from including the achromatic component in the SLIC clustering step only. ................................................................. 61
3.18 Results from the vision pipeline for some representative outdoor terrains. ................................ 62
4.1 Terrain types for terrain characterization tests: black tiles, granite, rough wood and faux grass. ................................................................. 67
4.2 Sample acceleration data for robot traversal over the indoor terrain types in Figure 4.1. ................................................................. 68
4.3 Corresponding angular rates for Figure 4.2. ................................................................. 69
4.4 The self-supervised feature registration process. ................................................................. 71
4.5 Test configurations for traversability data collection. ................................................................. 80
4.6 Sample cost image showing identified classes (from Chapter 3) for the path planner. .......................................................... 82
4.7 Effect of cost function on generated path for the cost image in Figure 4.6 using A*. .......................................................... 83
4.8 Effect of cost function on generated path for the cost image in Figure 4.6 using the Dijkstra path planner without distance considerations. .................................................. 84
4.9 Cubic splines interpolating prescribed points for generation of sample paths. The larger circles represent the start and end positions and the smaller circles represent intermediate points. ........................................ 88
4.10 Change of coordinates from image to Cartesian. .......................... 89
4.11 Some inapplicable situations when using spline interpolation. ...... 93
4.12 Generated splines when \( |x_{\text{start}} - x_{\text{goal}}| \gg 0 \). .......................... 94
4.13 Generating sample paths using five control points for quartic Bezier curves (see Figure 4.9). ......................................................... 95
4.14 Sample paths generated by using quartic Bezier curves for the situations in Figure 4.11. ......................................................... 97
4.15 Sample paths generated by using quartic Bezier curves for the situations in Figure 4.12. ......................................................... 98
4.16 Spline with simulated robot workspace at a path point for evaluating sample paths. ......................................................... 99
4.17 Clustering details for Sample Image B. ......................................... 101
4.18 Chosen paths for Sample Terrain A when \( \lambda = 0.2, \mu = 0.3 \) and \( v = 0.5 \). 103
4.19 Chosen paths for Sample Terrain A when \( \lambda = 0.15, \mu = 0.25 \) and \( v = 0.6 \). 104
4.20 Chosen paths for Sample Terrain A when \( \lambda = 0, \mu = 0 \) and \( v = 1 \). 105
4.21 Chosen paths for Sample Terrain B when $\lambda = 0.2$, $\mu = 0.3$ and $\nu = 0.5$.
4.22 Chosen paths for Sample Terrain B when $\lambda = 0.15$, $\mu = 0.25$ and $\nu = 0.6$.
4.23 Chosen paths for Sample Terrain B when $\lambda = 0$, $\mu = 0$ and $\nu = 1$.
4.24 Chosen paths for Sample Terrain C when $\lambda = 0.2$, $\mu = 0.3$ and $\nu = 0.5$.
4.25 Chosen paths for Sample Terrain C when $\lambda = 0.15$, $\mu = 0.25$ and $\nu = 0.6$.
4.26 Chosen paths for Sample Terrain C when $\lambda = 0$, $\mu = 0$ and $\nu = 1$.
4.27 Chosen paths for Figure 4.24 with small unlabeled clusters given high costs.
4.28 Chosen paths for Figure 4.25 with small unlabeled clusters given high costs.
4.29 Chosen paths for Figure 4.26 with small unlabeled clusters given high costs.
4.30 An example of the process for incorporating obstacle avoidance in the proposed approach by using point clouds from a 2D LiDAR sensor.

5.1 Right: The localization and navigation strategy; Left: Definition of path following errors to the closest path point.
5.2 An architecture of the localization filter.
5.3 Indoor test terrain.
5.4 The test platform, Husky.
5.5 The outdoor terrain for preliminary testing.
5.6 The three test scenarios (L-R): Top: Source image with ROI. Bottom: Transformed ROI with paths from path planner.
5.7 Scenario 1: Class generation.
5.8 Scenario 2: Class identification.
5.9 Scenario 3: Novelty detection.
5.10 Scenario 1: All terrain types unknown; Test Configuration 2.
5.11 Scenario 2: All significant terrain types known; Test Configuration 2.
5.12 Scenario 3: At least one unknown terrain type; Test Configuration 2.
5.13 Scenario 2: All significant terrain types known; Test Configuration 2.
5.14 Scenario 1: All terrain types unknown; GNSS + odometry + IMU.
5.15 Scenario 3: At least one unknown terrain type; GNSS + odometry + IMU.
5.16 Scenario 1: All terrain types unknown; odometry + IMU.
5.17 Scenario 2: All significant terrain types known; odometry + IMU.
5.18 Scenario 1: All terrain types unknown; odometry + IMU.
5.19 Scenario 1: All terrain types unknown; GNSS + IMU.
5.20 Path following tests for proposed paths.
5.21 Scenario 1: All terrain types unknown.
5.22 Scenario 2: All significant terrain types known.
5.23 Scenario 3: At least one unknown terrain type.
5.24 Scenario 2: All significant terrain types known.
5.25 Classification of terrain types by the vision pipeline (see also Figure 4.4(a)).
5.26 Some challenging cases because of significant intermixing between terrain types, and partial snow and dust cover.

A.1 A representation of the proposed method.
B.1 The unicycle in a reference frame.
D.1 Sample Terrain 1 from Queen’s mining test site.
D.2 Sample Terrain 2 from Queen’s mining test site.  . . . . . . . . . . . . . . . . . . 194
D.3 Sample Terrain 3 from Queen’s mining test site.  . . . . . . . . . . . . . . . . . . 195
D.4 Sample Terrain 1 from Mars Curiosity Image Gallery.  . . . . . . . . . . . . . . 196
D.5 Sample Terrain 2 from Mars Curiosity Image Gallery.  . . . . . . . . . . . . . . 196
D.6 Sample Terrain 3 from Mars Curiosity Image Gallery.  . . . . . . . . . . . . . . 197
# List of Abbreviations and Symbols

## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>A* Search Algorithm</td>
</tr>
<tr>
<td>ANOVA</td>
<td>Analysis of Variance</td>
</tr>
<tr>
<td>CAM</td>
<td>Colour Appearance Model</td>
</tr>
<tr>
<td>CIE</td>
<td><em>Commission Internationale de l’éclairage</em> (International Commission on Illumination)</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defense Advanced Projects Agency</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-based Spatial Clustering of Applications with Noise</td>
</tr>
<tr>
<td>DWA</td>
<td>Dynamic Window Approach</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>FBL</td>
<td>Feedback Linearized</td>
</tr>
<tr>
<td>FCN</td>
<td>Fully Convolutional Network</td>
</tr>
<tr>
<td>FOV</td>
<td>Field of View</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>IRI</td>
<td>International Roughness Index</td>
</tr>
<tr>
<td>J’a'b'</td>
<td>CAM16 Uniform Colour Space</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>CIELAB Colour Space</td>
</tr>
<tr>
<td>LAGR</td>
<td>Learning Applied to Ground Robots</td>
</tr>
<tr>
<td>LBP</td>
<td>Local Binary Pattern</td>
</tr>
<tr>
<td>LiDAR</td>
<td>Light Detection and Ranging</td>
</tr>
<tr>
<td>LMA</td>
<td>Levenberg-Marquardt Algorithm</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>NIR</td>
<td>Near-Infrared</td>
</tr>
<tr>
<td>PSPNet</td>
<td>Pyramid Scene Parsing Network</td>
</tr>
<tr>
<td>PSR</td>
<td>Present Serviceability Rating</td>
</tr>
<tr>
<td>RANSAC</td>
<td>Random Sample Consensus</td>
</tr>
<tr>
<td>ResNet</td>
<td>Residual Neural Network</td>
</tr>
<tr>
<td>RGB</td>
<td>RGB Color Space</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of Interest</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>RRT</td>
<td>Rapidly Exploring Random Tree</td>
</tr>
<tr>
<td>RTK</td>
<td>Real-Time Kinematic</td>
</tr>
<tr>
<td>SBMP</td>
<td>Sampling Based Motion Planning</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SLIC</td>
<td>Simple Linear Iterative Clustering</td>
</tr>
<tr>
<td>SOOR</td>
<td>Southern Ontario Off-Road Dataset</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>Tr</td>
<td>Transformed Image</td>
</tr>
<tr>
<td>UCS</td>
<td>Uniform Colour Space</td>
</tr>
<tr>
<td>UTM</td>
<td>Universal Transverse Mercator</td>
</tr>
</tbody>
</table>
## Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta E$</td>
<td>Colour Difference</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Path curvature at a point</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>SBMP path evaluation function</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Curvature weighting parameter in $\Lambda$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Path length weighting parameter in $\Lambda$</td>
</tr>
<tr>
<td>$a^*$</td>
<td>Green–red component of the CIELAB colour space</td>
</tr>
<tr>
<td>$a'$</td>
<td>Green–red component of the CAM16 Uniform Colour Space</td>
</tr>
<tr>
<td>$b^*$</td>
<td>Blue–yellow component of the CIELAB colour space</td>
</tr>
<tr>
<td>$b'$</td>
<td>Blue–yellow component of the CAM16 Uniform Colour Space</td>
</tr>
<tr>
<td>$J'$</td>
<td>Lightness component of the CAM16 Uniform Colour Space</td>
</tr>
<tr>
<td>$L^*$</td>
<td>Lightness component of the CIELAB colour space</td>
</tr>
<tr>
<td>$a$</td>
<td>$x$ and $y$ linear acceleration readings from IMU</td>
</tr>
<tr>
<td>$H$</td>
<td>Homography matrix</td>
</tr>
<tr>
<td>$\tilde{a}$</td>
<td>Linear acceleration-based cost metric</td>
</tr>
<tr>
<td>$v$</td>
<td>Traversability weighting parameter in $\Lambda$</td>
</tr>
<tr>
<td>$B$</td>
<td>Bezier curve</td>
</tr>
<tr>
<td>$c$</td>
<td>Traversability cost</td>
</tr>
<tr>
<td>$e$</td>
<td>Epochs</td>
</tr>
<tr>
<td>$E_{DB}$</td>
<td>DBSCAN colour difference clustering threshold parameter</td>
</tr>
<tr>
<td>$E_I$</td>
<td>Colour difference parameter to compare DBSCAN clusters in an image</td>
</tr>
<tr>
<td>$E_T$</td>
<td>Colour difference parameter to compare DBSCAN clusters in different images</td>
</tr>
<tr>
<td>$F$</td>
<td>List of colour features and associated traversability costs for identified terrain types</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>( f )</td>
<td>A* cost function</td>
</tr>
<tr>
<td>( F_T )</td>
<td>List of traversability costs for identified terrain types</td>
</tr>
<tr>
<td>( F_V )</td>
<td>List of colour features for identified terrain types</td>
</tr>
<tr>
<td>( g )</td>
<td>Cost of traversal from the start position in the A* grid</td>
</tr>
<tr>
<td>( h )</td>
<td>A* path planning heuristic</td>
</tr>
<tr>
<td>( k )</td>
<td>Prescribed number of superpixels for SLIC</td>
</tr>
<tr>
<td>( p_s )</td>
<td>SLIC weighting parameter between colour difference and proximity</td>
</tr>
<tr>
<td>( q )</td>
<td>Cost function that maps ( a ) to traversability costs</td>
</tr>
<tr>
<td>( S_x )</td>
<td>Interpolated piecewise spline</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background and Motivation

In structured environments, well-paved and delineated roads mean that vehicle navigation decisions can be mainly guided by collision avoidance, traffic signs and regulations. However, in unstructured off-road environments containing combinations of mostly natural terrain types, navigation decisions are necessarily influenced by the choice of terrain type on which to drive, as measured by traversability characteristics such as roughness and slip (Figure 1.1).

Like in urban driving, contemporary navigation techniques based on semantic segmentation that employ shape-discriminating convolutional neural networks (CNNs) have been applied to off-road environments, but with limited success [2], [3]. This reduced performance is partly due to limited datasets. While semantic segmentation datasets are generally limited compared to per-class datasets because of the pixel-level labelling constraints, off-road datasets are particularly limited because of the remoteness of the environments for data collection. For example, the Freiburg Forest (DeepScene) off-road dataset [2] contains only hundreds of images compared to tens of
1.1. BACKGROUND AND MOTIVATION

Figure 1.1: Sample urban [1] vs off-road environments.

thousands in the Cityscapes urban dataset [4]. For context, the benchmark per-class dataset, ImageNet [5], comprises tens of millions of images. Furthermore, pixel-based classification exacerbates dataset class imbalance. For example, vegetation pixels typically dominate water puddles, representing a form of sample bias.

To improve segmentation performance, other modalities, such as depth, have been fused with RGB inputs with modest performance improvements [2], [3]. A further complication is the reduced shape-based class distinctness that CNNs rely on. For example, vegetation is subjectively subdivided into “Grass” and “Traversable grass”, and individual clusters of one class are not unique in shape. This lack of class distinctness is less of a problem with uniform terrain samples, such as sand [6]. Moreover, utilizing the segmented images for navigation presumes foreknowledge of traversability characteristics for each class, which is not guaranteed.

In terms of form, an image of a field of loose sand is not very different from one of
1.2. PROBLEM DEFINITION

red clay, but there is marked difference in terms of colour and texture. Vision-based near-to-far methods take advantage of this colour dependence to register colour features and traversability characteristics for off-road navigation [7]. In these schemes, the vehicle is equipped with a sensor suite comprising exteroceptive sensors for colour and texture features, and proprioceptive sensors for traversability characteristics. Colour features acquired at one time from an image of terrain far from the vehicle are matched to traversability characteristics acquired at another time after subsequently driving through the corresponding terrain. This time lag in the corresponding data for the same terrain patch presents a problem for data matching or feature registration. Current methods require expert supervision principally to address the time lag for feature registration, and for feature extraction [7], [8], [9], [10], [11]. A higher level of autonomy requires that the expert be eliminated from the overall pipeline. Such a self-supervised near-to-far approach is proposed in this thesis.

1.2 Problem Definition

The definitions of the types of learning, i.e., supervised, unsupervised, semi-supervised, self-supervised, are not consistent in the literature even among deep learning pioneers [12]; the particular definition used depends on the context and usage domain. But supervised learning generally uses expert-labelled data to train a network while self-supervised learning uses unlabelled data for training. For example in the area of natural language processing for text completion, the words of a sentence can be predicted using latent patterns learned from inherent rules encoded in the language structure of unlabelled input training data. An analogous but much more difficult problem in computer vision is predicting future video frames from current input frames. The
1.2. PROBLEM DEFINITION

input data, i.e., video frames, has limited inherent structure and exhibits more complex variations, thus input frames still require significant curation before training [12], [13].

It is important to note that the self-supervised approach proposed in this thesis pertains to labelling and not the learning or training approach. Self-supervised labelling autonomously labels terrain data to be used in supervised learning, which typically requires significant effort and resources for expert-labelling. Automating the labelling process removes the expert from the pipeline and allows the comparatively smaller training data requirements of supervised learning to be exploited.

In structured environments, objects are shape distinct, thus, shape drives data labelling in the image space; however, colour and texture are the key discriminating features for off-road terrains. To functionally label off-road terrain, colour or texture features of a terrain type are matched to sensor readings of traversability through a process of feature registration. The sensor readings are interpreted in terms of traversability by an expert with respect to characteristics such as roughness or slip. If the expert has prior knowledge of the characteristics of the particular terrain type, the traversability-estimating sensor is eliminated from the pipeline.

In unknown environments, the expert uses proprioceptive sensor data to estimate the traversability characteristics of terrain types distinguished by colour or texture features that are acquired from exteroceptive sensors. The labelling or feature registration process involves associating the data from both sensor types. But the proprioceptive sensor data must be interpreted in terms of relative levels of physical traversability characteristics, such as roughness or slip, in a calibration process. The result of this calibration process can be used for self-supervised labelling thereby
eliminating the expert. However, due to the time lag in the corresponding data to be registered, the self-supervised system must also autonomously define the regions of terrain from which data will be acquired, and ensure correct matching of the respective sensor data.

Feature registration is key to near-to-far approaches. Previous approaches have retained the expert in the pipeline to teleoperate the robot to regions of interest for feature registration, or to define paths to such expert-identified regions [7], [8], [9], [10]. The learning models used also often require expert-guided feature-extraction, consequently, these approaches cannot be conveniently used end-to-end.

The problem addressed in this thesis is the development of a self-supervised near-to-far method that autonomously performs feature registration thereby removing the expert from the pipeline. The end-to-end system will eliminate the need for expert-guided feature extraction thereby enabling adaptive navigation from raw input images while generating image-label pairs for training and deploying a robust CNN online.

The key contribution is a new self-supervised autonomous near-to-far approach that engenders a higher level of autonomy than previous expert-guided approaches.

1.3 Objective

The objective of this thesis is to formulate, develop and implement a self-supervised method for adaptively prioritizing smooth terrain while navigating an exploratory robot to a goal position in an unknown off-road environment with reasonably flat terrain by acquiring and autonomously registering terrain traversability characteristics and corresponding colour features online. The system should be reasonably robust to typical variations in chroma and illumination for a terrain type, and fit into
1.3. OBJECTIVE

contemporary end-to-end schemes that use robust supervised semantic segmentation.

The main tasks carried out to achieve this objective were as follows:

1. Identify and demonstrate the limitations of contemporary supervised navigation schemes in off-road terrains.

2. Propose and demonstrate mitigating methods for the identified limitations of contemporary supervised navigation schemes in off-road terrains.

3. Design a vision pipeline that identifies terrain types by colour in the image space.

4. Choose a robust and consistent sensor modality for estimating traversability characteristics.

5. Develop a self-supervised feature registration method that autonomously identifies regions of interest in the image space for feature registration.

6. Design a dictionary management system to store and update registered features.

7. Design a cost function that maps proprioceptive sensor readings to traversability costs in the image space for path planning.

8. Develop a practical path planning approach that uses the designed cost function to plan smooth traversable paths through the environment in the image space.


10. Test the overall autonomous approach in representative environments by interfacing the vision, feature registration, path planning, navigation and control pipelines.
1.4 Organization of Thesis

The chapters in this thesis group related tasks.

Chapter 2 reviews contemporary supervised methods for vision-based navigation in off-road environments; and proposes and demonstrates a method for mitigating identified limitations by multi-modal fusion. This is followed by a review of existing near-to-far methods before the overall architecture of the proposed method is introduced. The chapter starts with a review of traversability estimation methods.

Chapter 3 presents the vision pipeline, and introduces the dictionary management system for storing and updating terrain features. The chapter details the steps for identifying terrain types by colour, and demonstrates the introduced vision pipeline. Considerations for including robustness in the pipeline to handle reasonable variations in chroma and illumination in a terrain type are explained and demonstrated.

Chapter 4 explains the considerations behind the choice of sensor modality for traversability estimation and studies the applicability, consistency and robustness of the chosen sensor modality. The chapter formalizes the proposed self-supervised feature registration approach and its link to path planning. Relevant path planning approaches are reviewed and some limitations demonstrated, leading to the proposal and demonstration of a robust and practical path planning approach and a method for including collision avoidance for static obstacles.

Chapter 5 details the navigation and localization strategy and describes the experimental setup and facilities. Results from indoor and outdoor tests are discussed. The chapter concludes with a discussion of the limitations of the proposed approach.

Chapter 6 summarizes the thesis, highlights the key original contributions of the thesis, and outlines considerations for future work.
Chapter 2

Literature Review

In this chapter, some inherent challenges of supervised navigation methods in off-road environments are reviewed and demonstrated and a method to mitigate some of these challenges by using multimodal fusion is proposed and tested. Also, existing near-to-far methods are reviewed. Finally, the proposed method is introduced.

First, a review of traversability estimation methods in off-road environments is presented.

2.1 Terrain Traversability Estimation in Off-Road Environments

Traversability, the ease of moving between positions on a terrain, is a key consideration in navigating off-road environments; thus, it is well-studied [14]. Although terrain traversability has a wide range of definitions in the literature [15], in terms of practical terramechanics, it has been defined with respect to physical characteristics such as slip, roughness or compliance. These characteristics are defined in Section 4.1.

Terrains can be grouped into nominal classes such as grass or sand, or into functional classes such as rough or smooth terrain. Nominally different terrains can fall into the same functional class. Since nominal classification does not require knowledge
of traversability characteristics, it can be done remotely based on visual and textural features obtained from exteroceptive sensors, e.g., cameras [16]. But the nominal labels will only have practical meaning for vehicle navigation if their traversability characteristics are previously known.

Thus, of key interest in autonomous vehicle navigation, particularly in environments with unknown terrain, is functional classification, which is based on the traversability characteristics of the terrains and requires the use of proprioceptive sensors to directly interact with the terrain to estimate its characteristics [17], [18]. The process of functional classification is terrain characterization. Due to resolution limitations of exteroceptive sensors such as stereocameras, traversability characteristics cannot be accurately or precisely defined remotely [19]. And when traversability characteristics, e.g., roughness, are defined remotely using contemporary high-resolution 3D sensors [20], the labels are still nominal and require characterization [21] — because terrains with the same roughness characteristics in image space can have distinct traversability characteristics due to physical effects such as friction, traction and compliance.

Finally, predictive, terrain-adaptive autonomous navigation requires that the remote and local features from both types of sensors be matched in order to attach physical meaning to the remotely-obtained vision-based nominal labels. This is the feature registration process, which is a principal step in the near-to-far navigation approach introduced in Section 2.4. Hence, in the future, the characteristics of terrain types identified remotely can be predicted, and this information will be used to guide navigation or motion planning.
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

Traversability characteristics, such as roughness and slip, have long been characterized empirically based on terramechanics [22]. Classical terramechanics methods strived to model wheel-ground interaction empirically [23] but these methods often require the identification and estimation of an impractical number of parameters. Hence, more recent characterization have indirectly used sensors such as inertial measurement units (IMUs) and encoders to classify terrain [18], [24], [25], [26], [27], [28]. And in [29], a procedure for velocity independent classification using vibration data was introduced; similarly, neural networks have been used with motor current data for characterization [15].

In terms of vision-based classification, in Mars rovers, terrain traversability has been assessed using geometric depth information from stereo cameras [30], and this method has been further developed for ease of navigation [16], [31]. Also, slip has been estimated remotely [32], and laser data used for classification in [33]. To benefit from the comparative advantages of LiDAR and camera, visual data from both sensors have been combined using 2D and 3D LiDAR data in [34] and [35].

In [14], off-road terrain traversability analysis methods are reviewed.

2.2 CNN-Based Supervised Methods for Off-Road Autonomous Navigation

Contemporary supervised methods are CNN-based because CNNs afford end-to-end navigation and can be robust to changes in environmental conditions. Specifically related to off-road terrains, in [36], trail direction was inferred from aerial images, and, in [6], labelled images of terrain were obtained from a walking robot with small
footprint. Although the robot’s limited footprint meant that pixel labels were signif-
icantly sparse, the uniformity of the terrain facilitated segmentation.

One recent, interesting approach [37] is self-supervised training for adaptive driv-
ing through rough off-road terrain by CNN-based reinforcement learning. Given a
terrain image, their network attempts to choose a set of steering commands that
would guide the vehicle through smooth terrain. Training data comprises a set of
terrain images, a set of steering commands over the planning horizon, and IMU gen-
erated functional labels, i.e., smooth or rough terrain, corresponding to each steering
command. Unlike the proposed approach and previous approaches [6], [8], [11] that
also attempted self-supervised labelling for network training, their network does not
produce pixel-wise labels to generate a cost map for path planning and path tracking;
rather, it attempts to directly label the consequences of each steering action over the
input image as leading to smooth or rough terrain.

Also, unlike these previous works [6], [8], [11] that used pre-defined paths or tele-
operation to collect and label training data, they used a pre-defined policy that gen-
erates pseudo-random paths through the terrain. But, unlike these previous works,
the practical issue of the time delay in feature registration between when the terrain
image is taken and when the robot actually drives over it does not seem to have been
addressed; in the previous works, this time delay was handled by expert curation.
It is important to note here that, in the proposed approach, feature-registration is
autonomous; the system autonomously handles this time delay and autonomously
identifies unique terrain segments for feature registration without the need for tele-
operation or pre-defined paths or policies for path generation.

Overall, this scheme requires multiple learning runs over the terrain to generate the
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

significant data required for self-supervised training: in their preliminary tests on a 0.25 km² test area, data collection entailed more than 5 km of driving. Interestingly, in addition to the current image of the terrain in front of the robot, the network also requires a number of images from previous steps. Essentially, the consequences of future steering commands through the current image seem to depend on previous terrain images; this assumes a certain underlying structure to the unstructured terrain that is not very intuitive. All these concerns warrant the question of generalizability to unseen terrain, even in the same environment; there is a high likelihood that the trained network will only work for the learned terrain. Furthermore, their approach is not goal-oriented: the network reactively chooses a control action that moves the robot towards smooth terrain; however, the required approach for navigation and active exploration should not only guide the robot through smooth terrain but also progressively towards a goal.

Finally, since their system does not require the identification and characterization of constituent terrain types, the rough terrain in their sample environment can be conveniently detected by using 3D exteroceptive sensors, e.g., a stereocamera, which can be used for goal-oriented navigation through smooth terrain without any need for data collection and training as was previously done in [20], [30], [38]. Importantly, the need for identification and characterization of terrain types to augment 3D sensors for navigation was discussed in [38] and [39], and is further discussed in Section 4.1. It must be noted, however, that their method hinges on the assumption that the black box network learns encoded abstract states, which could possibly include the characteristics of the terrain types in addition to the overall roughness of the terrain; but this assumption can only be validated by quantitative analysis of real-world test
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

results on terrain from the sample environment but unseen at training time, and comparison with 3D sensor-only autonomous navigation.

An overwhelming challenge with these CNN-based methods is the significant time or resources required for data collection, curation, and labelling. Even with the existence of promising techniques such as meta learning [40], this challenge remains. The following sections detail some challenges with CNN-based supervised schemes, and the use of multimodal fusion to improve performance.

2.2.1 Off-Road Datasets and Challenges

Significant amounts of labelled image data is required to train convolutional neural networks (CNNs) that are deployed in an end-to-end vision stack. Because CNNs learn by complex representations of shapes, they are well suited to urban environments that have distinct semantic classes, such as cars, person, street light etc. But, in off-road environments, classes are less distinct in shape because of the labelling strategy, which is influenced by the unique challenges encountered in off-road driving. One of these challenges is that, unlike in urban driving, the terrain cannot be taken for granted in off-road driving. Hence, terrain traversability is a key consideration [41], [8], [6], [37].

The traversability consideration means that, in off-road datasets, vegetation is often split into non-unique sub-classes, such as grass and traversable grass, based on perceived ease of traversability. Also, in urban driving there is a strict requirement to distinctly identify similar classes—for example, street lights from traffic lights—because the vehicle control strategy will be different in each case. Yet, this is not the case in remote off-road environments. Consequently, in off-road datasets, an obstacle
class is often defined comprising an eclectic range of non-unique natural and man-
made objects such as rocks of varying sizes and shapes, fences, power pylons, etc. A further complication is the natural class and spatial imbalance in the datasets; e.g., as shown in Figure 2.1 for the Freiburg Forest dataset [2] in which obstacles are represented in more images than trees, but occupy fewer pixels showing spatial imbalance.

Whereas classification datasets such as ImageNet can contain millions of images, segmentation datasets contain fewer images because pixel-based labelling entails significant manual effort. Furthermore, off-road segmentation datasets contain even fewer images because remote off-road environments make data collection less convenient. For example, the Freiburg Forest dataset comprises only 366 images, compared to thousands of images in Cityscapes [4]. The limited size of off-road datasets, combined with the non-unique classes, makes semantic segmentation using CNNs with RGB-only inputs insufficient.
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

Terrestrial off-road terrains can be forests with trails, or the more diverse savannah- or prairie-like terrains. Partially-labelled real-world [43] and synthetic datasets [44] have been used to demonstrate the effectiveness of transfer learning to leverage large classification datasets; but it was also shown that transfer learning is no substitute for a large dataset, and that synthetic datasets must be carefully developed to avoid negative effects in transfer learning. With regards to niche off-road terrain-only datasets, in the dataset in [45] timestamped sensor data was used to label corresponding traversed images, but it was acknowledged that the timestamped data to image correspondence can be very inaccurate.

Although other off-road datasets exist in the literature [3], [46], one predominant publicly available off-road dataset is the Freiburg Forest (DeepScene) dataset [2]. However, a critical problem with this dataset is that the images are not of the same size, suggesting inconsistencies in its pre-processing. Aspect ratio is important in CNNs, particularly for smaller features such as those in the Object class. For example, images in benchmark datasets such as Cityscapes are of uniform size. Zero-padding the borders of the images to the largest image size would create a new unknown class, and does not solve the aspect ratio problem. It must be noted, however, that disparate aspect ratios may be considered to be a form of regularization during training, similar to skewing. From Figure 2.2, the important question of what input size to use in the CNN arises. In smaller networks [47], a smaller input size not only improves resolution in the encoder, it also mitigates the effects of differing aspect ratios. But deeper networks such as DeepLab [48] require larger and consistent input sizes. In addition, it is unclear why, unlike in the train set, there is no tree sample in the test set. In summary, the dataset is not reasonably uniform enough to allow for full and
consistent evaluation. This is one significant motivation for creating the Southern Ontario Off-Road (SOOR) dataset.

It is argued that the SOOR dataset (Figure 2.3) is more challenging due to its diversity. Unlike SOOR, there are multiple images with repeated scenes in the DeepScene dataset albeit at different illumination levels. In addition to classes in the Freiburg Forest dataset, the SOOR dataset comprises three vegetation subdivisions: Traversable grass, Grass and Vegetation; representative of three levels of difficulty of traversal. It also includes a Water class. The SOOR dataset consists of 257 densely labelled images of size $768 \times 384$ pixels taken at different times on three separate days in July in southern Ontario, Canada. Both datasets exhibit natural class and sample imbalance as shown in Figures 2.1 and 2.3. Sample images from each dataset
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

![Bar chart showing imbalanced classes in the SOOR dataset](https://example.com/chart.png)

Figure 2.3: Imbalance in the SOOR dataset [42]. (© 2021 IEEE)

are shown in Figure 2.4.

2.2.2 Semantic Segmentation with RGB-Only Inputs vs Multi-Modal Fusion

To mitigate the spatial imbalance problem, in addition to augmentation techniques such as zooming, contemporary approaches use deeper networks with flexible convolutions [49], spatial pyramid pooling [50], and dilated convolutions [48] to increase the receptive field of the network in its deeper layers to ameliorate class imbalance effects. But the limited size of the off-road datasets reduces the benefits of these techniques.

A general approach to improving segmentation performance is multi-modal fusion, whereby the RGB input is supplemented by other data sources such as infrared and depth data [2]. Apart from the fact that not all of these input modalities are intrinsic properties of the feature classes, such as depth, geometric matching of the different sensor inputs is a key issue. Similar pre-processing requirements, such as the need to
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

Figure 2.4: Sample images from the off-road datasets [42]. (© 2021 IEEE)
de-noise stereo disparity maps, limit the scope for online application.

In [2], multimodal fusion of RGB, depth and near-infrared (NIR) data was studied, and in [3] RGB and LiDAR data fusion was studied. Although fusion with 3D data can improve segmentation performance, the authors reported that the size of the fused network can limit capacity for deployment online. Also, because of the characteristic noise, stereo disparity maps required refining, and predicting depth with a separate network [51] involves an extra step. It must be emphasized that, unlike texture, some of these modalities, such as depth and vegetation indices, are not intrinsic properties of all the features in the off-road environment.

The later the stage of fusion the more flexible the network is to changes in input modalities. Early or mid-stage fusion is achieved by adding or concatenating the weights or inputs. Late-stage fusion can also be achieved by ensembling or by mixture of experts, which is also applicable to mid-stage fusion. Practically, the most appropriate approach is ascertained by experimentation. A review of multi-modal fusion is presented in [52]. Although late-stage fusion gave the best results in [2], in this work, channel concatenation is used because it only marginally increases the size of the network, which makes for faster inference.

2.2.3 Proposed RGB-Texture Fusion for Improved Semantic Segmentation

One option for fusion with RGB data that has not yet been explored in this end-to-end context, to the extent of this review, is texture. Although CNNs are typically believed to classify objects by shape, they have been shown to have a strong texture bias [53]. While texture has been used in classification tasks in remote sensing [54], [55], image
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

matching [56] and supervised and unsupervised clustering on texture classification benchmarks [57], the texture descriptors have primarily been hand-crafted. Importantly, texture features have shown good performance in classifying monochrome planetary images [16], [31], [58].

Texture features include Gabor, Haralick, wavelet, and local binary pattern (LBP) [59] descriptors; of these, LBP is often preferred due to its ease of computation and robustness [60]. The LBP descriptor is also relatively invariant to illumination and rotation [60], and its flexibility has been leveraged for multi-resolution analysis [61]. The LBP patterns can be formed in the image plane, thus they fit contemporary end-to-end CNN-based navigation schemes. RGB-LBP fusion eliminates the need for transformations that ensure geometric matching of separate sensor inputs. In this work, RGB-LBP fusion for improved segmentation is proposed and studied.

As shown in Figure 2.5, the LBP descriptor uses a binary representation of pixel neighbourhoods to define the texture of image features. For pixel $n$ with intensity $I$ in the grayscale image with a circularly symmetric number of neighbor pixels $P$ within radius $R$, the LBP is defined as [59]

$$LBP_{P,R} = \sum_{p=0}^{P-1} 2^p s(x),$$

(2.1)

where

$$s(x) = \begin{cases} 
1, & x \geq 0, \\
0, & x < 0, 
\end{cases}$$

(2.2)

$$x = I_p - I_n.$$ 

(2.3)
And the rotation invariant form

\[
LBP^{ri}_{P,R} = \min (ROR(LBP_{P,R}, i)) \quad | \quad i = 0, 1, \ldots, P - 1),
\]

(2.4)

where circular bit-wise operator \(ROR(x, i)\) does a right shift on the \(P\)-bit number \(x\) \(i\) times.

\(P\) is set as eight to correspond to the 8-bit images for uniformity during normalization. For multi-resolution analysis, multiple \(R\)-scale features are combined [61], but single \(R\) descriptors are used per fusion in this study for ease of analysis. While the CNN will still learn latent features from multiple \(R\)-scale LBP inputs, the results will become less intuitive. Multiple \(R\)-scale LBP inputs can be used in future work.

In Figure 2.5, the upper two rows are for a sample image from the Freiburg Forest (DeepScene) dataset, and the lower two rows for a sample image from the SOOR dataset. At \(R = 1\), the per-class texture representations are not visibly distinct because information from the neighbourhood is inadequate. As \(R\) increases to 10, the texture patterns become more distinct but the artefacts at the boundaries between classes increase. By \(R = 20\) the pattern resolution visibly decreases and artefacts increase at the image edges. In [53] it was shown that, in addition to shape, CNNs have a strong texture bias. While the outer shape for each class is defined by its boundaries in the labelled images, the internal texture contours provide additional discriminating features for the CNN. To reasonably capture local information, \(R\) is limited to 3, 5 and 7.
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

2.2.4 Results from RGB-Only vs RGB-Texture Fusion Inputs

In training the networks, augmentation was done while training using a data generator. Positional augmentation was limited to random vertical and horizontal flips because rotations through acute angles alter labels. Vertical flips improved performance particularly for the irregularly shaped water puddles. Colour augmentation for the images was limited to only brightening in the range \([0.5, \ldots, 1.5]\). Although it is possible to include more augmentation methods, a comparison between a comprehensive augmentation regime and its subsets in [2] did not show significant differences in segmentation performance for the Freiburg Forest dataset.
For the Freiburg Forest dataset, their validation set was used for testing. But, during training, their training set was split into the 75−25 training-validation ratio for early stopping to limit overfitting by monitoring the validation loss profile relative to the training loss profile. And, because the dataset is small, three separate training-validation splits were used to eliminate biases in each validation set. The same strategy was employed for our dataset, after keeping 10% of the dataset for testing. The common learning rate policy was $1 - \frac{e}{\max e}$ of the specified initial learning rate, where $e$ is the epoch; with a batch size of four. The learning rate was extensively tuned in each case. To evaluate the techniques on their own merits, the basic stochastic gradient descent (SGD) was used without regularization methods such as patch extraction or cropping. The platform was Tensorflow with an NVIDIA Quadro RTX 6000 GPU.

The FCN8 [47] network architecture was used as a benchmark with 448 × 224 pixels input size, and two notable multi-scale networks, PSPNet and DeepLabv3, that use different multi-scale approaches were selected from the Cityscapes leaderboard. PSPNet uses spatial pyramid pooling while DeepLabv3 uses dilated convolutions. The implementations were based on the authors’ papers and their specified repositories. The PSPNet encoder was the same as the FCN8 encoder, but a ResNet-50 encoder was used for DeepLabv3.

Results from the SOOR dataset test set are shown in Table 2.1 for two different 75-25 training-validation splits. The metric is the practical intersection over union (IOU) [62] that measures spatial matching between labels and predictions. The difficult classes are the Object, Water and Traversable grass class.

While water puddles in potholes can have similar shapes, they come in different
### Table 2.1: FCN8 on SOOR dataset (IOU) [42]. (© 2021 IEEE)

#### First split.

<table>
<thead>
<tr>
<th>Modality</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP&lt;sub&gt;8,1&lt;/sub&gt;</td>
<td>0.818</td>
<td>0.676</td>
<td>0.674</td>
<td>0.789</td>
<td>0.386</td>
<td>0.238</td>
<td>0.156</td>
<td>0.534</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,3&lt;/sub&gt;</td>
<td>0.854</td>
<td>0.740</td>
<td>0.740</td>
<td><strong>0.854</strong></td>
<td><strong>0.493</strong></td>
<td>0.609</td>
<td>0.155</td>
<td>0.635</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,5&lt;/sub&gt;</td>
<td>0.866</td>
<td>0.752</td>
<td><strong>0.747</strong></td>
<td>0.852</td>
<td>0.487</td>
<td><strong>0.662</strong></td>
<td>0.171</td>
<td><strong>0.648</strong></td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,7&lt;/sub&gt;</td>
<td><strong>0.870</strong></td>
<td><strong>0.763</strong></td>
<td>0.721</td>
<td>0.826</td>
<td>0.475</td>
<td>0.521</td>
<td><strong>0.198</strong></td>
<td>0.625</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,10&lt;/sub&gt;</td>
<td><strong>0.870</strong></td>
<td>0.761</td>
<td>0.694</td>
<td>0.814</td>
<td>0.425</td>
<td>0.644</td>
<td>0.191</td>
<td>0.628</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,200&lt;/sub&gt;</td>
<td>0.797</td>
<td>0.638</td>
<td>0.604</td>
<td>0.755</td>
<td>0.308</td>
<td>0</td>
<td>0.016</td>
<td>0.445</td>
</tr>
<tr>
<td>RGB</td>
<td>0.908</td>
<td>0.817</td>
<td>0.806</td>
<td>0.914</td>
<td>0.609</td>
<td>0.617</td>
<td>0.350</td>
<td>0.717</td>
</tr>
<tr>
<td>RGB-LBP&lt;sub&gt;8,3&lt;/sub&gt;</td>
<td>0.910</td>
<td>0.827</td>
<td>0.785</td>
<td>0.907</td>
<td>0.603</td>
<td>0.682</td>
<td>0.353</td>
<td>0.724</td>
</tr>
<tr>
<td>RGB-LBP&lt;sub&gt;8,5&lt;/sub&gt;</td>
<td><strong>0.912</strong></td>
<td>0.825</td>
<td>0.801</td>
<td><strong>0.918</strong></td>
<td>0.615</td>
<td>0.751</td>
<td>0.348</td>
<td><strong>0.739</strong></td>
</tr>
<tr>
<td>RGB-LBP&lt;sub&gt;8,7&lt;/sub&gt;</td>
<td><strong>0.912</strong></td>
<td><strong>0.830</strong></td>
<td><strong>0.802</strong></td>
<td>0.909</td>
<td><strong>0.619</strong></td>
<td><strong>0.754</strong></td>
<td><strong>0.357</strong></td>
<td><strong>0.741</strong></td>
</tr>
</tbody>
</table>

#### Second split.

<table>
<thead>
<tr>
<th>Modality</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP&lt;sub&gt;8,1&lt;/sub&gt;</td>
<td>0.809</td>
<td>0.647</td>
<td>0.680</td>
<td>0.814</td>
<td>0.330</td>
<td>0.016</td>
<td>0.130</td>
<td>0.489</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,3&lt;/sub&gt;</td>
<td>0.854</td>
<td>0.739</td>
<td>0.728</td>
<td>0.835</td>
<td>0.438</td>
<td>0.543</td>
<td>0.190</td>
<td>0.618</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,5&lt;/sub&gt;</td>
<td>0.869</td>
<td>0.753</td>
<td>0.725</td>
<td>0.835</td>
<td>0.451</td>
<td><strong>0.653</strong></td>
<td>0.247</td>
<td>0.648</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,7&lt;/sub&gt;</td>
<td>0.865</td>
<td><strong>0.758</strong></td>
<td><strong>0.741</strong></td>
<td>0.849</td>
<td><strong>0.481</strong></td>
<td>0.600</td>
<td>0.195</td>
<td>0.641</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,10&lt;/sub&gt;</td>
<td><strong>0.875</strong></td>
<td><strong>0.758</strong></td>
<td>0.732</td>
<td><strong>0.851</strong></td>
<td>0.436</td>
<td>0.595</td>
<td>0.185</td>
<td>0.633</td>
</tr>
<tr>
<td>LBP&lt;sub&gt;8,200&lt;/sub&gt;</td>
<td>0.836</td>
<td>0.647</td>
<td>0.612</td>
<td>0.827</td>
<td>0.356</td>
<td>0.093</td>
<td>0.069</td>
<td>0.492</td>
</tr>
<tr>
<td>RGB</td>
<td>0.903</td>
<td>0.806</td>
<td>0.786</td>
<td>0.904</td>
<td>0.562</td>
<td>0.609</td>
<td>0.296</td>
<td>0.695</td>
</tr>
<tr>
<td>RGB-LBP&lt;sub&gt;8,3&lt;/sub&gt;</td>
<td>0.909</td>
<td>0.804</td>
<td><strong>0.797</strong></td>
<td><strong>0.914</strong></td>
<td>0.614</td>
<td>0.681</td>
<td>0.316</td>
<td>0.719</td>
</tr>
<tr>
<td>RGB-LBP&lt;sub&gt;8,5&lt;/sub&gt;</td>
<td><strong>0.915</strong></td>
<td><strong>0.830</strong></td>
<td>0.794</td>
<td>0.902</td>
<td>0.582</td>
<td><strong>0.740</strong></td>
<td><strong>0.350</strong></td>
<td><strong>0.731</strong></td>
</tr>
<tr>
<td>RGB-LBP&lt;sub&gt;8,7&lt;/sub&gt;</td>
<td>0.910</td>
<td>0.811</td>
<td>0.796</td>
<td>0.905</td>
<td><strong>0.615</strong></td>
<td>0.703</td>
<td>0.326</td>
<td>0.724</td>
</tr>
</tbody>
</table>

1 Class: (1) Trail (2) Grass (3) Vege. (4) Sky (5) Obstacle (6) Water (7) Traversable grass.
sizes and forms. And, as shown in Figure 2.3, the Water class is the most imbalanced of these, thus it is highlighted in Table 2.1. Water can be uniquely difficult because it reflects other classes in the environment, but this reflection serves to highlight the role of texture in CNNs. As seen in the third figure in Figure 2.6, in which the water puddle reflects grass, sky, etc., texture provides added discriminating information to the CNN. The texture network labels a section of the water puddle Sky according to the sky reflection on the water, but the RGB-only network struggles with this distinction. However, the multi-modal RGB-LBP network combines both information for improved segmentation.

Importantly, the significance of texture and shape to CNNs is evident in the texture-only networks in Table 2.1. Two important cases are when $R = 1$ and $R = 200$. In the former case, the texture features are derived from pixels in the immediate neighbourhood, thus the patterns are not uniquely defined for each class as when $R$ is 3, 5 or 7 (Figure 2.5). In the latter case, the texture features are scrambled as they are derived from remote pixels. However, the overall shape of the classes as defined
by their respective boundaries in the labelled images provides enough information for reasonable segmentation in the larger classes.

The interactions between Traversable grass and the morphologically similar Grass are seen in the wrongly segmented sections in Figure 2.6; the discretionary basis of distinguishing Traversable grass (yellow) from Grass (green) during labelling means that the CNN often mislabels them. And the added texture information does not significantly ameliorate the eclectic nature of the Object class. Hence the relatively poorer segmentation performance in these classes relative to Sky and Trail. Critically, RGB-texture fusion does not worsen overall segmentation performance.

Results from the Freiburg Forest dataset are shown in Table 2.2. The most difficult class in this dataset is the Objects class (Figure 2.1) because of the reasons discussed in Section 2.2.1. Merging the Tree and Vegetation classes, improves results particularly in the Object class [2]. But, here, the more challenging case is considered by training the CNN with separate Tree and Vegetation classes. However, the Tree class is not represented in their test set, hence the overall performance is not fully evaluated. Although the improvement offered by the multi-modal RGB-LBP input over RGB-only input for the Object class is not marked, it is not insignificant.

The results from the multi-scale networks, shown in Table 2.3, follow the same trend as FCN8. Although, the FCN8 encoder used for PSPNet has limited capacity, and the deeper DeepLabv3 network improves overall performance in the Water class, the results follow the same trend for the Water class in Table 2.2. The LBP texture features do not interfere with the separate multi-scale techniques used in each network.
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

Table 2.2: FCN8 on Freiburg Forest dataset (IOU) [42].

<table>
<thead>
<tr>
<th>Modality</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP8,3</td>
<td>0.692</td>
<td>0.766</td>
<td>0.841</td>
<td><strong>0.879</strong></td>
<td>0.157</td>
<td>0.556</td>
</tr>
<tr>
<td>LBP8,5</td>
<td>0.709</td>
<td>0.703</td>
<td><strong>0.848</strong></td>
<td>0.875</td>
<td>0.148</td>
<td>0.559</td>
</tr>
<tr>
<td>LBP8,7</td>
<td><strong>0.720</strong></td>
<td><strong>0.782</strong></td>
<td>0.840</td>
<td>0.873</td>
<td><strong>0.166</strong></td>
<td><strong>0.564</strong></td>
</tr>
<tr>
<td>LBP8,10</td>
<td>0.701</td>
<td>0.767</td>
<td>0.840</td>
<td>0.866</td>
<td>0.099</td>
<td>0.545</td>
</tr>
<tr>
<td>RGB</td>
<td>0.846</td>
<td>0.857</td>
<td>0.881</td>
<td>0.910</td>
<td>0.200</td>
<td>0.616</td>
</tr>
<tr>
<td>RGB-LBP8,3</td>
<td>0.849</td>
<td><strong>0.864</strong></td>
<td>0.868</td>
<td>0.910</td>
<td><strong>0.196</strong></td>
<td>0.615</td>
</tr>
<tr>
<td>RGB-LBP8,5</td>
<td>0.838</td>
<td>0.858</td>
<td>0.865</td>
<td><strong>0.912</strong></td>
<td><strong>0.282</strong></td>
<td><strong>0.626</strong></td>
</tr>
<tr>
<td>RGB-LBP8,7</td>
<td><strong>0.850</strong></td>
<td>0.862</td>
<td><strong>0.874</strong></td>
<td>0.907</td>
<td>0.274</td>
<td>0.623</td>
</tr>
</tbody>
</table>

1 Class: (1) Trail (2) Grass (3) Vegetation (4) Sky (5) Obstacle.

Table 2.3: Multi-scale networks on SOOR dataset (IOU) [42].

<table>
<thead>
<tr>
<th>Class</th>
<th>FCN8 In1(^1)</th>
<th>FCN8 In2(^1)</th>
<th>PSPNet In1</th>
<th>PSPNet In2</th>
<th>DeepLabv3 In1</th>
<th>DeepLabv3 In2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trail</td>
<td><strong>0.907</strong></td>
<td><strong>0.907</strong></td>
<td>0.892</td>
<td>0.906</td>
<td>0.896</td>
<td>0.864</td>
</tr>
<tr>
<td>Grass</td>
<td>0.814</td>
<td>0.819</td>
<td>0.797</td>
<td>0.814</td>
<td><strong>0.841</strong></td>
<td>0.798</td>
</tr>
<tr>
<td>Vegetation</td>
<td>0.775</td>
<td>0.794</td>
<td>0.732</td>
<td>0.749</td>
<td>0.799</td>
<td><strong>0.804</strong></td>
</tr>
<tr>
<td>Sky</td>
<td><strong>0.920</strong></td>
<td>0.910</td>
<td>0.840</td>
<td>0.892</td>
<td>0.904</td>
<td>0.914</td>
</tr>
<tr>
<td>Obstacle</td>
<td>0.573</td>
<td><strong>0.601</strong></td>
<td>0.541</td>
<td>0.536</td>
<td>0.591</td>
<td>0.599</td>
</tr>
<tr>
<td>Water</td>
<td>0.622</td>
<td>0.717</td>
<td>0.455</td>
<td>0.738</td>
<td>0.722</td>
<td><strong>0.776</strong></td>
</tr>
<tr>
<td>Trav. grass</td>
<td><strong>0.336</strong></td>
<td>0.330</td>
<td>0.186</td>
<td>0.231</td>
<td>0.307</td>
<td>0.318</td>
</tr>
<tr>
<td>Mean</td>
<td>0.707</td>
<td><strong>0.726</strong></td>
<td>0.635</td>
<td>0.695</td>
<td>0.723</td>
<td><strong>0.725</strong></td>
</tr>
</tbody>
</table>

1 Inputs: (1) RGB (2) RGB-LBP8,5.
2.2.5 Challenge of Domain Adaptation in Off-road Environments

Off-road environments are less-structured than urban environments due to the natural randomness in the distribution of off-road features; and the considerable effort required to collect, curate and label off-road datasets makes domain adaptability a key concern. There is considerable benefits to be gained if a CNN trained with one dataset can be deployed in varied off-road environments with similar performance and minor modifications.

In this work, domain adaptation refers to the flexibility to use a CNN, trained on one off-road environment, with images from another off-road environment without labelling, and without a drop in performance. Domain adaptation is commonly treated as an obstacle avoidance problem [63], [64] but, practically, in vehicle navigation obstacle avoidance has long been implemented using 3D sensors [41]. With contemporary semantic segmentation schemes, domain adaptation is the more challenging task of finding correspondences between images at pixel level [65], [66], [67], or as defined in [68]. In this section, the domain adaptation challenge is experimentally demonstrated in two off-road environments by comparing CNNs trained on the SOOR and Freiburg Forest datasets.

Domain adaptation in structured environments, with focus on the domain shift between synthetic and real images, is well-studied using adversarial learning techniques [69], [70], [71], [72]. Similarly, unsupervised domain adaptation, or the domain gap between real images of different cities was studied in [73], [74]. Here, it is contended that the domain gap in off-road environments are comparatively larger because of the natural random distribution and variation of environmental features. Also, because the textural patterns of off-road classes are relatively consistent, textural features
2.2. CNN-BASED SUPERVISED METHODS FOR OFF-ROAD AUTONOMOUS NAVIGATION

Table 2.4: Dataset classes for multi-domain test.

<table>
<thead>
<tr>
<th>Freiburg Forest Dataset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trail</td>
</tr>
<tr>
<td>SOOR Dataset.</td>
</tr>
<tr>
<td>Trail</td>
</tr>
</tbody>
</table>

are relevant [54], [55]. This section explores off-road domain adaptation using the Freiburg Forest and SOOR datasets as source and target domains separately, and highlights the effect of texture.

For uniformity, some common classes in each dataset were merged according to Table 2.4. For the SOOR dataset the Grass and Traversable grass classes were merged; and for the Freiburg Forest dataset, the Tree and Vegetation classes were merged. Images with water were not used in the test. From the sample images from each dataset shown in Figure 2.4, it is seen that the images in the SOOR dataset are of higher resolution, which aids finer LBP texture representations. Both datasets have grass and dense foliage (Vegetation), but the grass in the Freiburg Forest dataset is uniform compared to the grass in the SOOR dataset. Similarly, while both Vegetation classes consists of trees, the branches and leaves are denser in the SOOR dataset. In addition, the Trail class in the Freiburg Forest dataset primarily consists of sand, but the Trail class in the SOOR dataset comprises sand and gravel. Both datasets contain natural and man-made obstacles.

In the tests, four DeepLabv3 networks were trained on each dataset with RGB-only and RGB-LBP input modalities, respectively.

For the networks trained and evaluated on the same dataset, merging similar classes made each class more distinct and reduced the role of texture as seen in the
comparative segmentation performance of both input modalities in Table 2.5. A different fusion strategy [2], [52] or choice of LBP resolution can give different results.

However, the results are different for the networks evaluated on a different dataset. As shown in Figure 2.4, the Trail, Vegetation and Obstacle classes are the most different across both environments. While the networks trained on the higher resolution SOOR dataset gave better results, overall, the best results are for the Sky class that is the most consistent amongst the classes across both environments. The texture-fused networks reflect this consistency, while emphasizing the innate differences in the Grass class, which is less uniform in the SOOR dataset (Figure 2.4).
The overall cross-domain performance is not good enough for safe deployment, implying that each new off-road environment will require a labelled dataset. While conventional adversarial techniques offer potential for improving this poor cross-domain performance, the improvements afforded by these adversarial techniques across urban environments, which exhibit smaller domain gaps compared to off-road environments, are very modest [73], [74].

Also, although differences in the image sizes in both datasets may play a role, the disparate image sizes in the Freiburg Forest dataset did not seem to have a significant effect on performance in same-domain training. Finally, while it is possible to define the same classes when creating synthetic dataset analogues to study domain shift vs. real datasets, it is unrealistic to expect different environments to have the exact same structural or semantic classes in the real world. The differences between real-world environments have been referred to as global and class-wise domain shifts [74].

Thus, self-supervised labelling is important and should be taken advantage of, where possible, to autonomously create datasets without the significant effort required for human labelling.

2.3 Semi-Supervised Methods for Navigation in Off-Road Environments

Due to the absence of pedestrians, traffic rules and dynamic obstacles, a binary representation can be used for remote off-road environments for the purposes of autonomous driving. Two high-level classes are used: Terrain and Obstacle; and the focus is on choosing traversable terrain to navigate the vehicle through by identifying constituent terrain types.

As discussed in [75], traversability, including obstacles, should be treated as an
affordance because it depends on the operating vehicle specifications, e.g., a truck will better handle rough rocky terrain than a smaller vehicle. An affordance, in this context, was defined as an actionable property that exists between the environment and the actor. For the purposes of this work, an obstacle is a terrain feature or object that can hinder the safe operation of the robot. In this sense, rocks, water bodies, and quicksand are all examples of obstacles in remote off-road environments. The safe operation of the robot can be defined \textit{a priori} in terms of traversability characteristics such as roughness and slip by calibration. In terms of roughness, not all obstacles can be identified remotely using 3D sensors such as stereocameras and LiDAR sensors because of resolution constraints; for example, after calibration, a vehicle may be known to safely traverse bumps up to a specified size, but the 3D sensor might be unable to accurately sense objects of the specified size. In general, not all obstacles can be safely identified remotely without supervision, e.g., quicksand and water bodies.

In Chapter 3, a process for identifying terrain types by colour is introduced. The process of choosing traversable terrain to guide navigation decisions necessitates that the identified terrain types be characterized. One broad class of semi-supervised methods that facilitates navigation through traversable terrain are the near-to-far methods.

Proposed during the DARPA LAGR program [41], [76], the near-to-far methods associate proprioceptive sensor data to exteroceptive sensor data to extend the predictive range of the robot [9], [75] [77], [78], [79], [80] with the goal of prioritizing traversable terrain during autonomous navigation. In one scheme, a supervisory vibration classifier was used to label the training data for a color classifier to predict the
traversability of future terrains from images [7], [81]. However, the online data collection scheme did not have the capability to autonomously identify and deliberately traverse uniform terrain sections for data collection, thus, feature registration still entails some supervision. Similarly, a probabilistic approach was presented in [82]; however, the data collection trajectories were specified, and variations in terrain type and structure were limited. Due to their noisy nature, disparity maps cannot be used to predict roughness at a practical resolution, and slip cannot be reliably estimated from vision, either; so, acquiring traversal data requires driving over the terrain. Thus, in another scheme, separate models were trained with provided semantic labels for color and traversability features respectively, then co-trained bi-directionally in a classification scheme [29].

Variants of the basic near-to-far technique have been used in traversability analysis: In [83], terrain deformation was correlated with vehicle attitude and configuration to estimate traversability in deformable terrain to avoid sinking in soft ground; similarly, slip information was predicted from camera data [32], and driving energy requirements predicted from vision sensors [84]. Also, aerial and ground vehicle teams in which the exteroceptive data comes from aerial vehicles have been used [85], [86]. A subset of the near-to-far approach uses only the vision space, whereby pixel rich stereo data in the near field is used to classify less dense far field images to increase look ahead distance [19]. However, prior training with the proprioceptive space is still necessary to provide dynamic information about the terrain for traversal decisions.

These classical classification approaches do not suit contemporary end-to-end navigation schemes that are based on semantic segmentation using CNNs. And
2.4. THE PROPOSED SELF-SUPERVISED NEAR-TO-FAR APPROACH

unlike the most similar works [6], [7], [8], [10], [11], [84] the proposed method autonomously chooses uniform terrain patches for self-supervised feature registration while autonomously navigating the robot to a goal in an end-to-end scheme. As it progresses towards the goal, the proposed method collects images and labels terrain types by traversal cost in the image plane to train a CNN for robust semantic segmentation. The dense, pixel-wise labels cover all known terrain types in the image compared to the weak labels in [6], [8], [11] that only cover the limited pixels traversed by the vehicle’s legs or wheels.

It must be noted that, for exploratory navigation in unmapped off-road environments with unknown terrain characteristics, there is an inherent risk that the vehicle will get stuck in cul-de-sacs or in latent hazards, e.g., quicksand.

2.4 The Proposed Self-Supervised Near-to-Far Approach

A key aspect of the near-to-far approach is feature registration, which is the process of associating the visual and traversability characteristics of each terrain type. The reviewed near-to-far approaches require expert guidance during feature registration; but in the approach proposed in this work, the feature registration process is self-supervised. As shown in Figure 2.7, the proposed approach entails three steps. In Step 1, a terrain image is acquired and similar regions clustered by color; in Step 2, representative regions of each terrain type are driven over to acquire traversability characteristics and associate with the colour features in the self-supervised feature registration process; and in Step 3, known traversal costs, based on the registered traversability characteristics, are used to plan a path through the next image, thereby prioritizing traversable terrain while navigating to the goal. In this step, previously
unknown terrain types are also identified and their features registered as the vehicle progresses towards the goal. A flow diagram of the proposed approach is presented in Figure A.1 in Appendix A.1.

A basic assumption of colour-based near-to-far schemes is that traversability characteristics of different samples of a terrain type show reasonably narrow variations.

An overall architecture of the proposed approach is shown in Figure 2.8, and the technical steps required to achieve the proposed approach are outlined in Figure 2.9. Since the environment is unmapped, the global map is simply a store of the specified goal position and serves to localize the robot in terms of global coordinates, as discussed in Section 4.3. The constituent blocks in Figure 2.8 are discussed in the subsequent chapters in the order shown in Figure 2.9.
2.5 Chapter Summary

In this chapter, contemporary CNN-based supervised navigation approaches for autonomous vehicles in off-road environments have been reviewed. It has been shown that the difficulty of off-road environments combined with the significant data requirements and the reduced shape-uniqueness among off-road feature classes make these supervised approaches less effective in off-road environments compared to urban environments. A method for improving segmentation performance by RGB-LBP multimodal fusion was proposed and was shown to improve segmentation performance in some difficult classes. In addition, the poor cross-domain performance of trained CNNs has been demonstrated for off-road environments, implying that a dataset has
to be built for each off-road environment for reasonable performance; this is imprac-
tical. Hence, a self-supervised labelling method was proposed in a near-to-far scheme
that also navigates the robot to a specified goal by using progressively acquired and
continually updated terrain information. The scheme takes advantage of the peculiar
nature of off-road environments to define two high level classes: Obstacle and Ter-
rain; and then characterizes the Terrain class into sub-classes for adaptive navigation.
The method differs from previous near-to-far approaches because it does not require
expert-guidance or teleoperation, thus, it introduces a higher level of autonomy.
Chapter 3

The Image Processing Pipeline

3.1 Single-Shot Image Processing

First, a perspective transformation is used to maintain the 2D spatial distribution of terrain constituents. Then the terrain types are identified by colour-based clustering for subsequent self-supervised feature registration. The perspective transformation approach assumes a robot-mounted camera but, in the case of aerial images, a similar approach can be used to scale and ortho-project the image [37]. The goal of the vision pipeline is to auto-label similar terrain types in a terrain image by colour in the 2D world plane for end-to-end navigation, and for training a segmentation CNN—possibly augmented by texture features as discussed in Section 2.2.3. The labels follow the introductory discussion in Section 2.3. Unlike classical methods, e.g., [8], [86], [88], that require supervised feature extraction, this pipeline is end-to-end.

3.1.1 Perspective Transformation

While previous approaches have clustered superpixels in the 2D image plane [8], this plane does not preserve the geometric layout of the terrain for self-supervised
feature registration and requires external depth correlation. In this work, an empirical approach to perspective transformation is employed to maintain the planar layout of the terrain. Accounting for the keystoning effect due to camera field of view (FOV), a fixed region of interest (ROI) of real world dimensions \((w \times h \text{ m})\) in the image is transformed to the “top-down” view by using a homography. The ROI in image coordinates represents a rectangle in the ortho-projected view as shown in Figure 3.1.

In Figure 3.1, if \((u_i, v_i)\) represents points in the transformed image \((\text{Tr})\), and \((U_i, V_i)\) represents corresponding points on the ROI respectively, where \(i = 1, \ldots, n\) and \(n \geq 4\) is the number of corresponding point pairs; and \(p_i = (u_i, v_i, 1)^T\), \(P_i = (U_i, V_i, 1)^T\) are representations of these points in homogeneous coordinates, then the planar homography \(H\) maps \(p_i \sim HP_i\). Note that \(p\) and \(P\) are to scale and thus represent actual pixel measurements in image coordinates. And since \(p_i\) and \(HP_i\) are in the same direction, \(p_i \times HP_i = 0\). Detailed derivation of the homography and the
solution procedure are given in [89]. Defining

\[ HP_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} P_i = \begin{bmatrix} h^{1T} P_i \\ h^{2T} P_i \\ h^{3T} P_i \end{bmatrix}, \] (3.1)

where \( h^{1T}, h^{2T} \) and \( h^{3T} \) are the rows in \( H \), the cross product becomes

\[ p_i \times HP_i = \begin{bmatrix} v_i h^{3T} P_i - h^{2T} P_i \\ h^{1T} P_i - u_i h^{3T} P_i \\ u_i h^{2T} P_i - v_i h^{1T} P_i \end{bmatrix} = 0. \] (3.2)

Eq. 3.2 can be expressed as a vector matrix product as

\[ \begin{bmatrix} 0^T & -P_i^T & v_i P_i^T \\ P_i^T & 0^T & -u_i P_i^T \\ -v_i P_i^T & u_i P_i^T & 0^T \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0. \] (3.3)

Because the third equation in (3.3) can be obtained from a sum of the products of the components of \( p_i \) and the first two equations, it can be omitted to give two linearly independent equations

\[ \begin{bmatrix} 0^T & -P_i^T & v_i P_i^T \\ P_i^T & 0^T & -u_i P_i^T \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \end{bmatrix} = 0, \] (3.4)
3.1. SINGLE-SHOT IMAGE PROCESSING

\[ A_i h = 0, \quad h = \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix}. \] (3.5)

Alternatively, in explicit terms, \( p_i \sim HP_i \) is

\[ v_i = \frac{h_{21}U_i + h_{22}V_i + h_{23}}{h_{31}U_i + h_{32}V_i + h_{33}}, \] (3.6)
\[ u_i = \frac{h_{11}U_i + h_{12}V_i + h_{13}}{h_{31}U_i + h_{32}V_i + h_{33}}. \] (3.7)

Rearranging (3.6) and (3.7) gives

\[ h_{21}U_i + h_{22}V_i + h_{23} - h_{31}U_i v_i - h_{32}V_i v_i - h_{33}v_i = 0, \] (3.8)
\[ h_{11}U_i + h_{12}V_i + h_{13} - h_{31}U_i u_i - h_{32}V_i u_i - h_{33}u_i = 0. \] (3.9)

and for \( n \) corresponding point pairs

\[
\begin{bmatrix}
0 & 0 & 0 & -U_i & -V_i & -1 & U_i v_i & V_i v_i & v_i \\
U_i & V_i & 1 & 0 & 0 & 0 & -U_i u_i & -V_i u_i & -u_i \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & -U_n & -V_n & -1 & U_n v_n & V_n v_n & v_n \\
U_n & V_n & 1 & 0 & 0 & 0 & -U_n u_n & -V_n u_n & -u_n \\
\end{bmatrix}
\begin{pmatrix}
\begin{pmatrix}
h_{11} \\
\vdots \\
h_{33}
\end{pmatrix}
\end{pmatrix} = 0, \tag{3.10}
\]

which is the same as (3.4).

Because the transformed image is a calibration pattern, its defined pixel coordinates are considered exact, but the corresponding measurements in the ROI are noisy. When \( n > 4 \), the solution procedure entails minimizing \( \|Ah\| \) subject to \( \|h\| = 1 \). An
approximate solution is the unit singular vector associated with the smallest singular value of $A$ as determined from singular value decomposition (SVD). Thus, $Ah$ is actually a vector of residuals. The solution is further refined by using a nonlinear least squares scheme such as the Levenberg-Marquardt Algorithm (LMA) (Appendix B.1) to minimize the transfer error expressed as a Euclidean distance in inhomogeneous terms

$$\sum_{i=1}^{n} d(p_i, HP_i)^2$$

or

$$\sum_{i} \left( u_i - \frac{h_{11}U_i + h_{12}V_i + h_{13}}{h_{31}U_i + h_{32}V_i + h_{33}} \right)^2 + \left( v_i - \frac{h_{21}U_i + h_{22}V_i + h_{23}}{h_{31}U_i + h_{32}V_i + h_{33}} \right)^2. \quad (3.12)$$

In addition, RANSAC is used to iteratively refine the solution on non-colinear subsets of point pairs.

Using this homography, for any pixel at $(u, v)$ in the transformed image (Tr), the intensity can be approximated from the ROI as

$$\text{Tr}(u, v) = \text{ROI}(f_x(u, v), f_y(u, v)). \quad (3.13)$$

$f_x$ and $f_y$ are in $H^{-1}$ in the same form as (3.6) and (3.7). Discrete values in image coordinates are obtained from the continuous values in (3.13) by interpolation. The position of the fixed ROI depends on the camera height and angle. For a fixed camera position, while the ROI is invariant to yaw angle, it is affected by pitch and roll of the robot frame that holds the camera; but this variance is not unique to this scheme,
and similar empirical mitigating methods used in [90] apply.

For the transformed image, smaller real-world sizes approximate terrain undulations better, and interpolation effects become more pronounced with bigger ROI sizes. The transformed image is scaled to real-world size in (3.13). This scaling is important because it determines the resolution of waypoints from the path planner. Smaller sizes mean faster processing, with less noise but increasingly widely-spaced waypoints with higher possibility of less-smooth path following. Terrain undulations and obstacles are discussed in Section 4.4 with respect to the transformed image.

### 3.1.2 Visual Features and Clustering

Clustering, the grouping of similar features, is fundamental in unsupervised colour-texture approaches [41]; DBSCAN [91] is used in this work because of its robustness to noise [92]. The required adjacency map was built by using the SLIC superpixel algorithm [93], chosen because of its ease of tuning. To compute the colour difference ($\Delta E$), these algorithms require a perceptually uniform colour space. Traditionally the L*a*b* space was used, where L* represents lightness, and the chromatic features represent the green-red and blue-yellow components; and $\Delta E$ was based on a Euclidean distance metric (CIE76 colour difference formula). For a cluster of pixels, the mean or median intensities are used. However, colour models are continually improved: in this work, the J’a’b’ space is chosen from the CAM16-UCS model [94], based on CIECAM02. While clustering similar classes, neglecting the achromatic J’ helps reduce the effects of illumination by correctly giving a smaller $\Delta E$, as studied in Section 3.2. Thus, $\Delta E$ is defined as

\[
\Delta E = \sqrt{(a'_1 - a'_2)^2 + (b'_1 - b'_2)^2}.
\]  
(3.14)
3.1. SINGLE-SHOT IMAGE PROCESSING

In clustering the ROI into superpixels, SLIC considers proximity \( (d_s) \) and \( \Delta E \). Consider \( N \) number of pixels, and \( k \) number of superpixels, for uniform-sized superpixels, the grid interval is: \( S = \sqrt{N/k} \). Considering a candidate superpixel with center at pixel \( j \), the combined SLIC distance metric \( D \) for a constituent pixel \( i \) is

\[
d_s = \sqrt{(u_i - u_j)^2 + (v_i - v_j)^2}, \quad (3.15)
\]

\[
D = \sqrt{(\Delta E)^2 + (d_s/S)^2 p_s^2}, \quad (3.16)
\]

where \( p_s \) specifies the relative priority of colour or distance. Pixel \( i \) is added to the candidate superpixel for which its distance is lowest.

The resulting superpixels are clustered by using DBSCAN with an adjacency matrix and an empirical \( \Delta E \) threshold \( (E_{DB}) \). The image processing pipeline is shown for a sample image in Figure 3.2. The resulting labeled clusters from DBSCAN are not uniquely numbered. Thus, as a final step, clusters are compared and renumbered uniquely if they meet a less-strict \( \Delta E \) threshold \( (E_I) \)–without regards for proximity (Figure 3.2(d)). Noisy classes (e.g., Figure 3.2(e) (right)) can be mitigated by using morphological processing. To collect traversability data for self-supervised feature registration, the robot drives over the largest cluster per class. Note how the darker tyre marks on the wood are separately clustered. After the first run, a feature list of per-class mean (or median) colour features \( F_V \) is stored as

\[
f_i = [J', a', b']^T, \quad (3.17)
\]

\[
F_V = \begin{bmatrix} f_1, \ldots, f_n \end{bmatrix}. \quad (3.18)
\]

where \( n \) is the number of known unique clusters. Subsequently, given a new clustered
3.1. SINGLE-SHOT IMAGE PROCESSING

(a) Source image with ROI. (b) Transformed image. (c) SLIC superpixels. (d) DBSCAN clusters. (e) DBSCAN clusters identified as the same type.

Figure 3.2: The image processing pipeline [87]. (© 2021 IEEE)

Image, to identify previously-seen and unseen classes, the $\Delta E$ is calculated for class $j$ in the new clustered image with respect to $F_V$ based on an empirical colour threshold $E_T$, and stored in list $U$ as

$$\Delta E_{f_i,f_j} = \begin{cases} \Delta E_{f_i,f_j}, & \Delta E_{f_i,f_j} < E_T, \\ \infty, & \Delta E_{f_i,f_j} \geq E_T, \end{cases} \quad (3.19)$$

$$U = [\Delta E_{f_1,f_j}, \ldots, \Delta E_{f_n,f_j}], \quad (3.20)$$
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Table 3.1: List of tuned image processing parameters [87].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>$p_s$</th>
<th>$E_{DB}$</th>
<th>$E_I$</th>
<th>$E_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k^1$</td>
<td>10 to 25</td>
<td>&lt; 35</td>
<td>&lt; 4</td>
<td>&lt; 8</td>
<td>&lt; 13</td>
</tr>
</tbody>
</table>

$^1$ Average pixels per superpixel.

and, in $F_V$, the new class number $m$ for this class $j$ is

$$U = 0 \quad \text{if } \Delta E_{f_i,j_j} \geq E_T \quad \forall i = 1, \ldots, n,$$

$$m = \begin{cases} 
\arg \min_i(U), & U \neq 0, \\
n + 1, & U = 0. 
\end{cases}$$

Note that $F_V$ is updated accordingly.

The colour difference thresholds were determined empirically by comparing similar colour pairs. Outdoor grass was used as a benchmark because of its wide variations in colour and illumination. The thresholds are slackened as cluster size increases due to increased noise. A stricter threshold increases the likelihood of the preferred false negatives. Table 3.1 shows a list of tuned parameter ranges. The process of setting the thresholds is described in Section 3.2.

3.2 Analysis of Image Processing Parameters

Since the techniques used in the image processing pipeline have key dependence on colour properties, it is pertinent to study the colour properties of off-road terrain types under varying conditions with respect to the clustering parameters. Understanding how the results change with varying parameters provides background to how the parameter choices in Table 3.1 were determined. The samples of terrain types used in
this calibration process were extracted from terrain samples as shown in Figure 3.3. For ease of reference, results are shown for the commonly used CIELAB colour model implemented in the open source computer vision software package scikit-image [95]. Although the chromatic components of uniform colour spaces are unbounded, the components are often normalized and clamped in software packages due to practical considerations such as bit resolution; note that bits are not proper colour units.

### 3.2.1 Colour Difference

Because of the critical colour dependence of this scheme, it is important to limit the analysis to the chromatic components of the uniform colour space when calculating
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

colour difference \( \Delta E \), hence the achromatic or lightness component was not used in (3.14). However, it is also important to study the consequence of ignoring the lightness component especially when comparing terrain types with mainly achromatic colour components as shown in Figure 3.4. The results of \( \Delta E(L^*a^*b^*) \) vs \( \Delta E(a^*b^*) \) in Table 3.2 show that lightness is the main contributor to colour difference for these terrain types. Thus, using \( \Delta E(a^*b^*) \) increases the chances of unwanted false positives, resulting in different terrain types with different traversability characteristics being grouped together in the same class. However, the chances of the preferred false negatives can be increased by using stricter parameter thresholds. False negatives are preferred over false positives because the samples are labelled by traversability characteristics during feature registration; hence, samples of the same terrain type identified as different terrain types by the image processing pipeline will ultimately have a high likelihood of being labelled as the same, e.g., rough, after registration.

On the other hand, for the terrain samples with mainly chromatic colour components shown in Figure 3.5, the principal contributors to \( \Delta E \) are the chromatic components—as shown in the results for \( \Delta E(L^*a^*b^*) \) vs \( \Delta E(a^*b^*) \) in Table 3.3. The naturally-occurring granite/sand mix was added for comparison. Note that the choice of mean or median pixel values for calculating \( \Delta E \) does not have a significant effect on the results; the median values are shown in the plots. Also, the spread in the data reflects the typical noise in real-world terrain data.

Because it is preferred that the \( \Delta E \) be as large as possible for different terrain types, the aggregate results so far suggest that the use of \( \Delta E(L^*a^*b^*) \) is better, but a major consideration for the choice of \( \Delta E(a^*b^*) \) is the effect of illumination changes on \( \Delta E \) for samples of the same terrain type, as discussed in Section 3.2.2.
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Figure 3.4: Box plot comparison of terrain samples with mainly achromatic colour components.

Table 3.2: Colour difference for the terrain types in Figure 3.4 with respect to the tilled soil terrain type.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>∆E Basis</th>
<th>Road(bright)</th>
<th>Loose granite</th>
<th>Loose pebbles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>L<em>a</em>b*</td>
<td>41.913</td>
<td>11.467</td>
<td>11.893</td>
</tr>
<tr>
<td></td>
<td>a<em>b</em></td>
<td>4.633</td>
<td>6.788</td>
<td>5.535</td>
</tr>
<tr>
<td>Median</td>
<td>L<em>a</em>b*</td>
<td>42.139</td>
<td>14.936</td>
<td>14.490</td>
</tr>
<tr>
<td></td>
<td>a<em>b</em></td>
<td>3.926</td>
<td>6.711</td>
<td>5.417</td>
</tr>
</tbody>
</table>
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Figure 3.5: Box plot comparison of terrain samples with mainly chromatic colour components.

Table 3.3: Colour difference for the terrain types in Figure 3.5 with respect to the tilled soil terrain type in Figure 3.4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>ΔE Basis</th>
<th>Sand (bright)</th>
<th>Red soil</th>
<th>Gran./sand</th>
<th>Grass (bright)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>L<em>a</em>b*</td>
<td>40.619</td>
<td>64.218</td>
<td>15.708</td>
<td>56.893</td>
</tr>
<tr>
<td></td>
<td>a<em>b</em></td>
<td>31.477</td>
<td>64.192</td>
<td>2.585</td>
<td>55.228</td>
</tr>
<tr>
<td>Median</td>
<td>L<em>a</em>b*</td>
<td>42.744</td>
<td>64.745</td>
<td>16.764</td>
<td>56.563</td>
</tr>
<tr>
<td></td>
<td>a<em>b</em></td>
<td>31.372</td>
<td>64.698</td>
<td>2.474</td>
<td>55.036</td>
</tr>
</tbody>
</table>
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

3.2.2 Colour Difference in Illumination Changes and $E_T$

When comparing samples of the same terrain type under different illumination conditions, the importance of using only the chromatic components for $\Delta E$ to reduce illumination effects becomes evident. In this case, it is preferred that the $\Delta E$ be as small as possible since the samples represent the same terrain type. Examples of this case are shown for grass, sand and road samples respectively in Figures 3.6 to 3.8. The $\Delta E$ comparison results in Tables 3.4 to 3.6 show that the colour difference is principally in the lightness component due to differences in illumination. As would be expected, the effect of the lightness component is higher in the terrain types with principally achromatic colour components, i.e., road and sand, compared to grass.

One interesting observation is the effect of the sun on reflective terrains. For example, for the samples of sand taken from images captured at different times of the day, when the sun was high in the sky and bright, versus at sunset when it casts a reddish-yellow hue, the difference is reflected in the $a^*$ (green-red) component.

The choice of $E_T$ in Table 3.1 reflects the results in this section, with the dark vs dark and light vs light samples used within reasonable limits. Overall, as a compromise, a practical strategy is to use $\Delta E(L^*a^*b^*)$ for the local clustering stages; i.e., SLIC, and $\Delta E(a^*b^*)$ when comparing clusters globally because effects of illumination changes will be more pronounced.

3.2.3 SLIC Parameter $p_s$

In Table 3.1, the parameter $k$, which represents the prescribed number of superpixels, is nominal, and depends on the choice of $p_s$ that represents the relative importance of colour difference vs proximity between pixels during SLIC clustering.
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Figure 3.6: Box plot comparison of grass samples at different illumination levels.

Table 3.4: Colour difference for the grass patches in Figure 3.6 with respect to the dark patch captured with the camera facing away from the sun.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>( \Delta E ) Basis</th>
<th>Dark (facing sun)</th>
<th>Bright (back to sun)</th>
<th>Bright (facing sun)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>( L^*a^<em>b^</em> )</td>
<td>15.356</td>
<td>67.267</td>
<td>70.329</td>
</tr>
<tr>
<td></td>
<td>( a^<em>b^</em> )</td>
<td>14.709</td>
<td>34.540</td>
<td>54.504</td>
</tr>
<tr>
<td>Median</td>
<td>( L^*a^<em>b^</em> )</td>
<td>15.842</td>
<td>67.962</td>
<td>70.421</td>
</tr>
<tr>
<td></td>
<td>( a^<em>b^</em> )</td>
<td>15.130</td>
<td>35.056</td>
<td>54.957</td>
</tr>
</tbody>
</table>
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Figure 3.7: Box plot comparison of sand samples at different illumination levels.

Table 3.5: Colour difference for the sand samples in Figure 3.7 with respect to the dark patch.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>ΔE Basis</th>
<th>Bright (Sample 1)</th>
<th>Bright (Sample 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean</strong></td>
<td>L<em>a</em>b*</td>
<td>40.991</td>
<td>38.813</td>
</tr>
<tr>
<td></td>
<td>a<em>b</em></td>
<td>9.041</td>
<td>8.577</td>
</tr>
<tr>
<td><strong>Median</strong></td>
<td>L<em>a</em>b*</td>
<td>43.150</td>
<td>41.614</td>
</tr>
<tr>
<td></td>
<td>a<em>b</em></td>
<td>9.713</td>
<td>8.595</td>
</tr>
</tbody>
</table>
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Figure 3.8: Box plot comparison of road samples at different illumination levels.

Table 3.6: Colour difference for the road samples in Figure 3.8.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>∆E Basis</th>
<th>∆E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>( L^*a^<em>b^</em> )</td>
<td>61.875</td>
</tr>
<tr>
<td>Mean</td>
<td>( a^<em>b^</em> )</td>
<td>18.580</td>
</tr>
<tr>
<td>Median</td>
<td>( L^*a^<em>b^</em> )</td>
<td>62.305</td>
</tr>
<tr>
<td>Median</td>
<td>( a^<em>b^</em> )</td>
<td>18.194</td>
</tr>
</tbody>
</table>

The goal is to have superpixels with as much similarity in constituent pixels as possible such that the consequent DBSCAN clusters will be unique in terrain types.
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

(a) $p_s = 1, k_a = 739.$ (b) Non-uniform DBSCAN clusters.

(c) $p_s = 5, k_a = 2094.$ (d) More uniform and unique DBSCAN clusters.

Figure 3.9: SLIC superpixels and resultant DBSCAN clusters for varying $p_s$. 

With $k = 3000$, $E_{DB} = 2.8$ and $E_I = 8.5$, sample results are shown for different values of $p_s$ in Figures 3.9 to 3.11 with the final number of superpixels formed, $k_a$. More superpixels are formed for higher values of $p_s$; thus, more unique DBSCAN clusters in terms of terrain type are produced. However, above $p_s = 20$, the change is not significant.
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Finally, the effect of $E_I$, the parameter that compares DBSCAN clusters in the same image, is presented in Figures 3.12 to 3.15. Compared to Figure 3.2, in Figure 3.12,
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Figure 3.11: SLIC superpixels and resultant DBSCAN clusters for varying $p_s$ (cont’d).

Figure 3.12: Clustering results for Figure 3.2 with $E_I = 17$ [87]. (© 2021 IEEE)

the slackened threshold is not sensitive enough to identify the much darker sections on the rough wood terrain caused by tyre wear. Using the fixed set of parameters in Figure 3.11(a), results are shown for different values of $E_I$ for the same terrain (Figures 3.13 to 3.15). Slackening the threshold increases false positives.

In summary, the results in Figures 3.16 and 3.17 demonstrate the clustering process for real world outdoor terrain based on the clustering parameters in Figure 3.13; using only the chromatic components better handles illumination changes and subtle
3.2. ANALYSIS OF IMAGE PROCESSING PARAMETERS

Figure 3.13: Detailed clustering results for Figure 3.11(a) with $E_I = 8.5$.

Figure 3.14: Detailed clustering results for Figure 3.11(a) with $E_I = 17$.

colour variations for the same terrain type within reasonable limits. Finally in Figure 3.18 (bottom), limitations of the clustering process for a case where the illumination changes are significant for a terrain type is shown.
3.3 Chapter Summary

In this chapter, the proposed image processing pipeline was studied. For a given terrain image, the goal of the pipeline is to identify clusters of unique terrain types in the 2D plane on which the vehicle navigates. A homography is employed to transform a fixed ROI from image space to the 2D world plane. Working in a uniform colour space, clustering is done by using the chromatic components only, to mitigate the effects of illumination changes. And by using a broad range of terrain types, clustering parameters were set empirically by comparing different samples of the same terrain type under different conditions. Conservative clustering thresholds were employed to encourage the preferable false negatives over false positives in the clustering process.
3.3. CHAPTER SUMMARY

(a) SLIC superpixels. (b) DBSCAN clusters. (c) Final results after clustering DBSCAN clusters using $E_I$.

Figure 3.16: Clustering results from including the achromatic component in the SLIC and DBSCAN clustering steps only.
3.3. CHAPTER SUMMARY

(a) SLIC superpixels.  
(b) DBSCAN clusters.  
(c) Final results after clustering DBSCAN clusters using $E_I$.

Figure 3.17: Clustering results from including the achromatic component in the SLIC clustering step only.
Figure 3.18: Results from the vision pipeline for some representative outdoor terrains. Each colour represents an identified terrain type. In the lower figure, the vision pipeline creates two clusters for the uniform grass terrain because of the significant illumination difference [87]. (© 2021 IEEE)
Chapter 4

Traversability Characterization, Feature

Registration, Path Planning and Cost Design

4.1 Traversability Characteristics and Estimation

Terrain traversability can be defined as a combination of the traversability characteristics of the constituent terrain types, such as sand, grass or rocks, and the unevenness of the overall terrain profile [39]. The traversability of the terrain types is often estimated in terms of properties such as roughness [3], [21], [20], [39], slip [32], and compliance (or hardness). Note that these characteristics are not independent and should be combined for a complete characterization of a terrain type. The overall terrain profile is often defined in terms of geometric unevenness [38], [39].

As discussed in Section 2.1, in practical terms, different terrain types can be grouped in the same traversability class, e.g., rough or smooth, if they are characterized to fall within the same pre-defined range for the particular traversability characteristic under consideration. The roughness of a terrain type can be theoretically defined as the standard deviation of a plane fit through stereo or LiDAR data obtained from a patch of the terrain type [39]; it is an inherent property that can
help to distinguish between terrain types e.g. grass from sand. However, as noted in Section 2.1, estimating roughness in the vision space does not provide enough information to distinguish between terrain types because terrain types with similar roughness profiles in the vision space can have significantly different physical traversability estimates because of compliance or hardness. Moreover, estimation of surface roughness is limited by the practical resolution of the stereocamera or 3D LiDAR sensor. Thus, proprioceptive sensing is necessary for estimating roughness.

Slip can be practically defined as a measure of the difference between commanded and achieved velocities of the vehicle as it drives through the terrain [39]; although slip is an inherent property of a terrain type, it is affected by the geometric unevenness of the terrain patch, and other considerations such as wet or dry conditions.

The geometric unevenness of the overall terrain is not an inherent property of a terrain type; it refers to the overall geometry of a patch of terrain (which may comprise different terrain types). Like roughness, geometric unevenness can be defined using the planar statistics of the terrain as the pitch and roll of a plane fit through 3D data obtained from the terrain patch [39].

As demonstrated in [38], a complete traversability estimate for a terrain should combine an estimate of the traversability characteristics of the inherent terrain types and an estimate of its overall geometric unevenness, however, in their work only geometric unevenness was considered in building planar cost maps for path planning. In a related work [39], more comprehensive planar cost maps were built for path planning in the image space by combining various costs representing roughness, slip, and unevenness. A similar work was done in [96]. However, in these works, expert supervision was required for vision-based classification of terrain types, and it was
assumed that the inherent characterisitics of the terrain types considered were known and previously modelled.

In this thesis, a self-supervised approach for building planar cost maps for planning paths towards goal-oriented navigation and active exploration in unknown environments comprising terrain types with unknown characteristics is proposed. The proposed approach does not require expert supervision for terrain classification or characterization. As a first approach, only inherent characteristics of terrain types are considered, but a method to include overall geometric unevenness information in the self-supervised framework is presented in Section 4.4 as an area for future work.

While a lot of work has been done in modelling slip [14], [32], it remains difficult to consistently estimate, particularly in a non-supervised paradigm. For simplicity, only roughness is considered in developing a traversability cost in this work; however, since the path-planning techniques presented in this chapter depend only on the traversability cost, the techniques also apply to a combined cost derived from multiple characteristics. The goal is not to obtain absolute quantitative measurements of surface roughness as was done in [97], [98], or indices like the standard International Roughness Index (IRI) and Present Serviceability Rating (PSR) indices used for assessing roads [99], but to get an indirect measure for practical application similar to the approach in [8], [10].

Many works in the off-road literature have assessed roughness using vibration-based approaches. One approach has been to choose a frequency that maximizes the data gain from a power spectrum of the time series data [27]. Using a similar approach, in [8], vibration data was acquired using a linear accelerometer; while in [26], vibration data was acquired from a contact microphone. A study of the
performance of different sensor modalities in the frequency domain including linear acceleration, angular rates, motor current, microphone and infrared data was done in [15]. But the case for performing the analysis in the time domain due to limitations in the frequency domain approach was made in [100]; it was argued that important cues are lost when using spectral analysis. Similarly, a time-domain based approach was used in [82] with angular sensors.

While roughness can be estimated in images in terms of height above a nominal plane [20], expert supervision is still required to attach meaning to the labels in terms of traversability [21]. Moreover, patches of the same terrain type, e.g., grass, will have varying pixel heights in image space, thus the roughness metric, in terms of height, will not map to terrain type. A reasonable mapping between metric and terrain type is a fundamental assumption for predictions using the near to far approach.

In this work, the analysis is done in the time-domain. Linear acceleration measurements from an IMU are chosen because they also encode jerk, which serves as a measure of traction at the typical low operating speeds in this exploratory approach. On rough terrain, vibrations are tri-axial but unlike in [45], [10], [35] [45], [21], the $z$ component is not used because it includes the gravity vector and considerable considerations must be included to abstract it [35]. Thus, the IMU’s linear acceleration readings are combined as: $a = [a_x, a_y]$. The $x$ and $y$ components encode jerk measurements that allow meaningful estimation of traction as a measure of traversability for nominally smooth terrain with negligible bumps e.g., road — and can help discriminate between terrain of similar roughness but different traction characteristics; e.g., dry vs wet road. For example, for the terrain types in (Figure 4.1), the faux grass terrain is rougher than the black tiles to the touch, but the acceleration data
4.1. TRAVERSABILITY CHARACTERISTICS AND ESTIMATION

Figure 4.1: Terrain types for terrain characterization tests: black tiles, granite, rough wood and faux grass.

in Figure 4.2 shows that the black tiles exhibits higher acceleration magnitudes. The indoor test terrain is described in Section 5.2.

Compared to the nominally rougher synthetic grass terrain, the tiles require more traction than grass because of higher frictional adhesion in its contact with the robot’s tyres. The difference in traction is further discussed in terms of jerk in Section 4.3.1; also, corresponding numerical values for Figure 4.2 are shown. Note that the bumps in acceleration at the beginning and end of each plot in Figure 4.2(a) represent the starting and stopping motion of the robot respectively. Even though angular rates have been considered [15], as shown in Figure 4.3, the readings are not as sensitive as the corresponding linear acceleration readings (Figure 4.2) across the tested terrain types. Similar results supporting the effectiveness of using the $x$ and $y$ linear accelerations over the $z$ component have been shown in [101], but in the frequency domain.

If $f_{T,i}$ represents the roughness cost for terrain type $i$, an empirical function $q$ can
Figure 4.2: Sample acceleration data for robot traversal over the indoor terrain types in Figure 4.1.
Figure 4.3: Corresponding angular rates for Figure 4.2.
be defined such that $q: a_i \rightarrow f_{T,i}$. Therefore, the traversability feature list is

$$F_T = [f_{T,i}, \ldots, f_{T,n}] \quad i \in [1, n], \quad (4.1)$$

where $n$ is the number of known unique terrain types in the terrain. The design of empirical function $q$ is discussed in Section 4.3.1.

### 4.2 Self-Supervised Feature Registration and the Specific Path Planning Requirements

Feature registration is the process of associating visual features and traversability metrics such that each terrain type, identified in the image space, is assigned a label that reflects its traversability characteristics. In previous works [7], [8], [9], [10], [41], [84] feature registration was achieved by an expert defining paths to guide the robot’s navigation, or teleoperating the robot, and then registering features by supervised spatio-temporal matching. In essence, the expert navigates the robot through the identified terrain types to acquire corresponding traversability characteristics.

A key contribution of this work is the proposed self-supervised feature registration approach that makes the system autonomous. In the proposed approach, representative clusters are chosen for each of the unique terrain types identified through the image processing pipeline introduced in Chapter 3. The representative clusters are the largest sub-cluster for each terrain type, or a sample section for larger clusters. For each representative cluster that meets a size and solidity threshold for robot traversal for data acquisition, the algorithm fits a bounding contour [102], and then fits a minimum area bounding box around the contour (Figure 4.4). In this context, solidity is defined as the ratio of pixels that represent the class under consideration
4.2. SELF-SUPERVISED FEATURE REGISTRATION AND THE SPECIFIC PATH PLANNING REQUIREMENTS

(a) Bounding boxes around representative samples of identified terrain types [87]. (© 2021 IEEE)

(b) Some paths for self-supervised feature registration.

Figure 4.4: The self-supervised feature registration process.

within the cluster area. Using zero traversal costs, paths are planned so that the robot drives longitudinally through the center of each box for self-supervised feature registration before proceeding to the interim goal position as shown in Figure 4.4(b). Zero costs are used at this stage because the traversability features are still unknown. The proposed approach achieves the dual purpose of actively exploring the terrain while
navigating to the goal autonomously. No expert labelling or supervision is required.

Thus, from (3.18) and (4.1), the proposed self-supervised feature registration approach provides a running list or dictionary of registered features

\[ F = \begin{bmatrix} F_V \\ F_T \end{bmatrix} \]  \hspace{1cm} (4.2)

The proposed self-supervised feature registration approach drives the active exploration aspect of the overall proposed navigation strategy and presents a unique path planning problem. In the typical path planning problem for autonomous driving, the goal is to find a path from start to goal around obstacles progressing longitudinally through the terrain, but as shown in Figure 4.4(b), the path planning problem in active exploration can require the planning of transverse sub-paths around the image in the forward and/or reverse direction. The overall path is a concatenation of sub-paths and paths through the terrain samples. Although the paths have sudden breaks, by rotating to the start pose of the next sub-path—at the end of each sub-path—path following is conveniently achieved by the robot (Algorithm 2 in Appendix A.2). The proposed motion planning strategy, which requires a highly-manoeuvrable robot, is summarized in Section 4.5.

4.3 Path Planning for Non-Binary Terrains

In autonomous driving, path planning is done at two levels. At the global level or long-range planning level, a path is planned to guide the overall navigation of the vehicle from start to goal through the environment. Long-range planning utilizes previously
built maps of the environment. On the other hand, at the lower level, for short-range planning, a local path planner actively plans paths around local and dynamic obstacles in the planned route and relies on scene understanding by using sensors such as cameras or LiDAR. Local path planning is critical for successful autonomous navigation.

At the global level, path planning algorithms such as Dijkstra and its variants e.g., A* are used; probabilistic road maps [103] and gradient approaches are also used [104]. Some global path planning methods are reviewed in [105]. These algorithms require a map of the environment. One approach is to employ a graph representation of the environment consisting of edges and nodes representing roads and intersections.

At the lower level, Dijkstra, A* or D* algorithms can also be used on a grid-based representation of the environment but with significant limitations, including: the generation of discontinuous paths with sudden changes in curvature, non-feasible choppy paths, and paths that are contiguous with obstacles. Although variants of these path planning algorithms such as any-angle path planning [106] exist, and modifications such as smoothing in a post-processing step can be done to improve the generated paths [107], [108], these post-processing techniques often require prior information from maps, and do not eliminate all the listed limitations. Importantly, these post-processing techniques have been principally developed for binary terrains, comprising free space and obstacles.

Another option for local path planning is search-based algorithms, which are predominantly based on RRTs [109]. While RRTs allow the inclusion of kinematic constraints in path generation, the resulting paths require post-processing for smoothness [108], [110], [111]. Moreover, RRTs are well-developed for binary grids, unlike the case
under consideration whereby the grids are discretised in terms of non-binary levels of traversability represented by varying costs. Extending search-based path planning algorithms to non-binary grids while maintaining near-optimality is non-trivial.

Although artificial potential field approaches have also been used [112], their inherent limitations, including the narrow passage problem and oscillations when repulsive and attractive forces cancel out, are well documented [113] and have a very high chance of occurring in unstructured terrain.

Due to the limitations of Dijkstra, RRT and their variants, practical approaches to motion planning and local path generation in structured and unstructured environments are predominantly based on a process of local path evaluation [114], [115], [116], sometimes referred to as sampling based motion planning (SBMP). In this process, several paths are proposed from the vehicle’s current position, typically defined by splines [117], clothoids [118] or bezier curves [119], [120], [121], and evaluated according to specified kinodynamic constraints and navigation criteria as was done in [41]. This practical approach emphasizes smooth paths that meet kinodynamic constraints over path optimality. To improve the sampling strategy, SBMP has been combined with supervised learning techniques, however, these techniques require demonstration data for training [122].

While kinodynamic constraints depend on the type of robot, exploratory robots are usually highly manoeuvrable, such as NASA’s all-wheel steered rovers, and the differential-drive robot provided for the DARPA LAGR challenge. In line with the exploratory context, the robot used in this work is a skid-steer robot; unlike all-wheel steered systems, the velocity is constrained to move along its axis—similar to Ackermann steered robots—however, it can execute a point turn, or rotate-in-place,
thus has reasonably high manoeuvrability. It must be noted that legged robots will play a significant role in future exploratory missions [123].

The path proposal aspect of SBMP has been implemented using three broadly related methods. The first method, based on the dynamic window approach (DWA) [115], proposes candidate trajectories as curves or splines in the position or velocity space and finds the local controls or parameterized velocity commands necessary to achieve the chosen trajectories by using the robot’s dynamics. The controls are parameterized in terms of time or arc length. In a broad sense, the model predictive control (MPC) approach [124], in which the search is done for parameterized velocity commands for a fixed distance or start and end position, can be grouped in this category. On the other hand, the second method directly searches and samples the generalized velocity space for feasible commands to generate candidate paths [41]. Using a purely geometric approach, the third method generates curves in the 2D plane and directly attempts to manipulate the parameters of the curve to keep within kinodynamic constraints [120].

To handle terrain undulations, the first two approaches often include some wheel terrain interaction models, thereby assuming some fore-knowledge of terrain layout and characteristics [125], [39], [96]. However, the three techniques will give good results for reasonably flat terrain with the local map covering a fairly small area. As noted in Section 3.1.1, terrains with significant undulations are a universal problem in off-road and structured environments and can be sensed and handled empirically [90], [125].

For the second step in SBMP, the choice of path evaluation approach to determine a suitable path from the proposed candidate paths depends on the specific mission
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

criteria. Specifically for off-road environments, some examples of evaluation functions can be seen in [41], [112], [96]; their functions encompass traversability, collision avoidance and distance measures.

Whereas in urban environments, lane lines and road boundaries, identified using a supervised scene understanding approach, can serve as references for generating feasible paths, these references are not available in off-road environments. And unstructured or off-road environments are typically unmapped, thus global planning in such environments is limited to assuming a straight heading from start to goal, represented in GNSS or map coordinates, with significant dependence on the local path planner for overall navigation and obstacle avoidance.

A consequence of relying on the local planner for navigation is that the vehicle can get stuck in local minima, such as cul-de-sacs, or in difficult terrain. But this problem is a fundamental challenge of active exploration in unmapped environments, as demonstrated in the DARPA LAGR program, whereby best results were achieved when maps of the unknown environment were built from multiple trial runs through the off-road test environment [41].

In this work, similar to [120], a geometric approach to path proposal is used because it offers high flexibility in curve generation over the local image and takes advantage of the high manoeuvrability of the exploratory robot. Importantly, the proposed path evaluation approach includes traversability considerations.

4.3.1 A Naive A* Path Planner and Traversability Cost Metric Design

For binary grids in the planar $\mathbb{R}^2$ space, the set of all points on the grid can be represented by $X = [X_{\text{free}}, X_{\text{obstacle}}]$; and the optimal path planning problem can be
defined as:

**Definition** (Path planning in binary grids). *Given a grid with* \((X_{\text{free}}, x_{\text{init}}, x_{\text{goal}})\) *find a path* \(\delta^*\) *using a cost function* \(f(\delta)\) *such that* \(f(\delta^*) = \min f(\delta)\), *where* \(\delta \in X_{\text{free}}\) *is a set of all feasible paths from* \(x_{\text{init}}\) *to* \(x_{\text{goal}}\).

Note that \(x_{\text{init}}\), \(x_{\text{goal}}\) are the current robot position and (interim) goal position. And rather than a goal region, a point goal is used in this case because active exploration in this scheme requires that the robot navigates to a specific point on the grid as shown in Figure 4.4(b).

An extension of the Dijkstra algorithm, the A* algorithm [126] uses a cost comprising traversal distance and a heuristic i.e.,

\[
f(n) = g(n) + h(n),
\]

(4.3)

to trace an optimal path from the start or robot position to the goal position. The cost of traversal \(g(n)\) typically represents the distance from the start, the heuristic \(h(n)\) represents an estimated distance from goal, and \(n\) represents a cell on the grid at position \((i, j)\). The Dijkstra algorithm uses only \(g(n)\) and is guaranteed to find a shortest path but can be relatively slow. And the faster greedy search algorithm uses \(h(n)\) only, but optimality is not guaranteed.

By combining \(h(n)\) and \(g(n)\), A* finds an optimal solution faster than Dijkstra but depends on a careful choice of heuristic and cost. To prevent A* from reverting to the slower Dijkstra or the sub-optimal greedy search algorithm, the magnitudes of \(h(n)\) and \(g(n)\) must be carefully chosen to scale appropriately. For example, cells farther away from the goal will have a higher heuristic value which can dominate the
cost and make A* sub-optimal in that region.

The cost \( g(n) \) represents the difficulty of traversal, and is typically represented in binary grids as the distance between neighbour cells in the grid starting from the start position, without regard to the terrain traversability features. This means that diagonal traversals have higher costs. As A* traces an optimal path to the goal from the start position, each cell will have a parent cell \( p \) at some position \( (r, s) \); thus, for cell \( n \) at position \( (i, j) \) in the grid

\[
g(n) = g_p + n_s \sqrt{(i - r)^2 + (j - s)^2 + c(n)}. \quad (4.4)
\]

Where \( n_s \) is a nominal cell size. Also \( c(n) \) is an intrinsic cell traversability cost: in binary grids, \( c(n) \) is set to a prohibitively high value for obstacle cells to represent the difficulty of traversal; and zero for free cells.

The heuristic \( h(n) \) can be Manhattan or Euclidean distance and must be optimistic i.e. it must not overestimate the actual cost to goal. The heuristic used in this work is similar to that used in [127]: given a uniform grid with \( x_{\text{goal}} \) at grid position \( (G_r, G_c) \), for cell \( n \) at position \( (i, j) \) in the grid, the heuristic is defined as

\[
\Delta = |i - G_r| - |j - G_c|,
\]

\[
h(n) = \begin{cases} 
\sqrt{2} \ n_s \ |i - G_r|, & \Delta = 0, \\
\sqrt{2} \ n_s \ (|j - G_c| + |\Delta|), & \Delta > 0, \\
\sqrt{2} \ n_s \ (|i - G_r| + |\Delta|), & \Delta < 0.
\end{cases} \quad (4.5)
\]

The A* algorithm has often been used for binary grids comprising obstacles
and free space only. In the binary case, $X$ and $c(n)$ can be equivalently represented as $X, c(n) = [0, 1]$, but this is not the case in the non-binary case because $X_{\text{free}} = [X_{\text{free1}}, X_{\text{free2}}, \ldots, X_{\text{freeN}}]$, consequently, $c(n) = [c_1, c_2, \ldots, c_N]$. Where $N$ is the number of traversable terrain types in the grid. For a discretized non-binary grid, ensuring that the costs scale appropriately with one another, and with the heuristic, such that near-optimal and feasible paths are planned can require an empirical approach involving tedious cost metric calibration.

In this self-supervised system, the intrinsic cell costs $c(n)$ must be mapped to the traversability costs of the constituent terrain types as estimated from the IMU linear acceleration readings; thus, the acceleration readings, $a \in \mathbb{R}^2$ in Section 4.1, must be mapped to scalar costs and appropriately scaled.

The IMU linear acceleration readings comprise both linear and angular components with respect to the robot, thus, the readings are position-sensitive. Three test configurations representing different positions of the IMU with respect to the centre of the robot were tested as shown in Figure 4.5; of interest in this section are Test Configurations 1 and 2. In Test Configuration 1, the IMU is located at the longitudinal centre of the robot but slightly offset transversely, while in Test Configuration 2, the IMU is offset longitudinally but at the centre transversely.

An approach to mapping $a$ to scalar costs before scaling is shown in Table 4.1 for Test Configuration 1; it uses the average of the Euclidean norm of $N$ readings of $a$ for each terrain type sample. Table 4.1 covers the terrain types in Figure 4.1. If the rough wood terrain is considered the lowest resolution for obstacle avoidance with reasonable precision using a LiDAR sensor or stereocamera, and the grass values as the smoothest terrain, the values in Table 4.1 can define a bounded operating range
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

Figure 4.5: Test configurations for traversability data collection.

Table 4.1: Traversal data for the indoor terrain types in Figure 4.1 at 0.5 m/s using Test Configuration 1 [87]. (© 2021 IEEE)

<table>
<thead>
<tr>
<th>Terrain</th>
<th>Faux Grass</th>
<th>Rough wood</th>
<th>Granite</th>
<th>Black tiles</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{a} = \frac{1}{N} \sum |a|_2$</td>
<td>0.683</td>
<td>2.723</td>
<td>1.262</td>
<td>0.948</td>
</tr>
<tr>
<td>$\bar{J} = \frac{1}{N} \sum |J|_2$</td>
<td>18.60</td>
<td>121.14</td>
<td>55.26</td>
<td>52.09</td>
</tr>
</tbody>
</table>

1 By acceleration, $a = [a_x, a_y]$, m/s$^2$. 2 Jerk, $J = [J_x, J_y]$, m/s$^3$.

for this scheme. The narrow range of values necessitates scaling.

Compared to Test Configuration 1, the results for the preferred Test Configuration 2 shown in Table 4.2 follow the same trend, but the linear accelerations are more pronounced for the rougher terrain. The IMU linear acceleration readings will also depend on robot-specific dynamics and operating speed [29], [35]. As noted in Section 2.1, the goal is not to obtain an absolute measure of roughness but an indirect measure, thus feature-registration is done at a fixed speed with a specific test robot and configuration.

Although the null hypothesis does not hold in an ANOVA (at $p = 0.05$) on the $\|a\|_2$ data from which Table 4.2 is derived, the trend is consistent, and the means
Table 4.2: Traversal data $\tilde{a}$ from different tests for the indoor terrain types in Figure 4.1 at 0.5 m/s using Test Configuration 2.

<table>
<thead>
<tr>
<th>Terrain</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>Run 4</th>
<th>Run 5</th>
<th>Run 6</th>
<th>Run 7</th>
<th>Run 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faux grass</td>
<td>0.659</td>
<td>0.746</td>
<td>0.590</td>
<td>0.763</td>
<td>0.695</td>
<td>0.733</td>
<td>0.793</td>
<td>0.694</td>
</tr>
<tr>
<td>Black tiles</td>
<td>1.062</td>
<td>0.948</td>
<td>1.169</td>
<td>0.797</td>
<td>1.148</td>
<td>0.855</td>
<td>1.252</td>
<td>0.882</td>
</tr>
<tr>
<td>Granite</td>
<td>2.835</td>
<td>2.851</td>
<td>3.007</td>
<td>2.700</td>
<td>3.293</td>
<td>2.521</td>
<td>2.867</td>
<td>2.357</td>
</tr>
<tr>
<td>Rough wood</td>
<td>4.416</td>
<td>4.404</td>
<td>4.746</td>
<td>5.849</td>
<td>5.271</td>
<td>5.524</td>
<td>5.441</td>
<td>5.437</td>
</tr>
</tbody>
</table>

across the runs for each terrain type are distinct enough for $\tilde{a}$ to serve as a practical metric. During deployment, one possible way to improve consistency is to average $\tilde{a}$ readings from multiple runs through the same bounding box.

One complication during deployment is the solidity of the bounding box from which data is collected, as described in Section 4.4. Although a minimum solidity threshold is used, higher solidity bounding boxes will give a better reflection of the traversability characteristics. However, due to the characteristic intermixed nature of off-road terrain, the predefined solidity threshold will barely be met in most cases. One way to make the data collection regions more representative is to use a different approach, for example the data collection appendage in NASA’s Mars rovers. Such an appendage will work better than the spaced robot wheels. The $\tilde{a}$ value can also be weighted based on the solidity of the bounding box from which it is derived. In addition, multiple runs over a terrain patch can be averaged.

Another consideration is that different samples of those terrain types with varying characteristics, like grass, will have different $\tilde{a}$. One way to handle this challenge is by updating the $\tilde{a}$ intermittently by averaging even when the terrain is identified in the dictionary of features by (3.22). This mitigating approach is especially relevant for the rougher terrain types that exhibit more variations as seen in Table 4.2.
Overall, the data collection method is complementary to the proposed approach and can be improved independently.

Further to the discussion in Section 4.1, Table 4.1 shows the advantage of using linear acceleration readings to characterize terrain traversability; in addition to encoding a measure of roughness, it also serves as an estimate of wheel traction as seen in the jerk values. Thus, it is a reasonably comprehensive measure of traversability. Although the black tiles terrain type appears smooth, its relatively high jerk reading reflects its higher traction due to frictional adhesion with the robot’s tyre material, and the discontinuities due to the holes in it.

Overall, as introduced in Section 4.1, an empirical function $q$ that maps and scales $a$ appropriately can be defined such that $q : a \rightarrow c$; (i.e. $c \equiv f_T$). Sample results for different candidate functions $q$ are shown in Figure 4.7 for the terrain in Figure 4.6 using the $\tilde{a}$ values in Table 4.1 and a smooth wood $\tilde{a}$ value of 0.810. Because some
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

Figure 4.7: Effect of cost function on generated path for the cost image in Figure 4.6 using A*.

$a$ values are less than one, a polynomial function is not suitable. And because of the narrow range of $a$ for the different terrain types with significantly different roughness characteristics, an exponential function is a good candidate. Relating Section 3.2 to Figure 4.7, note that the achromatic granite is identified along with some noisy false
positives around the dark borders of the grass section. Also, since traversability considerations have been included, i.e., distance is no longer the key metric, evaluating path optimality is no longer straightforward, and is discussed in Section 4.3.3. However, if path optimality is defined in strictly traversability cost terms, an optimal path should always choose a lowest traversal cost cell in any neighbourhood as it progresses towards the goal. The effect of the traversability cost function on path optimality in only traversability cost terms can be demonstrated by eliminating the heuristic and the distance component in (4.3) and (4.4) respectively to give a purely traversability cost-based Dijkstra path planner. Results from such a path planner using the cost functions in Figure 4.7 are shown in Figure 4.8. The ellipses in Figure 4.8 show regions of interest to demonstrate the effect of cost function on path optimality. The
red ellipses show sub-optimal regions where the path passes through higher traversal cost cells in a neighbourhood with lower traversal cost cells, and the green ellipses show corresponding optimal paths where the path always chooses lowest traversal cost cells. Compared to the linear functions, the exponential functions generate more optimal paths because the costs increase more significantly for a small change in $\tilde{a}$.

From Figures 4.7 and 4.8, the choice of cost function clearly affects the generated path and its traversal cost-based optimality. The choice of cost function also affects the speed of the algorithm. In addition, because the algorithm assumes that the robot is a point object, the generated path passes through narrow crevices; in reality the robot will straddle the narrow corridors with the tyres driving through the neighbouring high cost regions. Furthermore, the continuous path has sudden breaks that cannot be easily followed by a path following controller.

Finally, examples of disproportionate and unrealistic effects of small isolated clusters on the path can be seen in Figures 4.7(h) and 4.7(i) where the path avoids the narrow strip at the start. This can lead to choppy paths in highly unstructured terrain with ubiquitous intermingling small isolated clusters. The choice of costs for these small clusters will have a significant effect on path smoothness. One way to mitigate the effect of small isolated clusters is to consider them to be low cost areas, without regard to their actual cost. Practically, the goal is not to avoid every high cost cluster however small, but to generally avoid high cost areas. Although morphological processes are used in the vision pipeline to remove small isolated clusters, there is a limit to their use and effectiveness as seen in Figures 3.12 to 3.15 and Figure 4.6.

In the case of binary cost maps, the difficult class can be assigned an arbitrarily high relative cost for traversal cost-based path optimality; but in this case of multiple
classes with continuously varying costs (Table 4.1), a cost function must be carefully chosen for traversal cost-based optimality as shown in Figure 4.8. Since it is impossible to empirically tune an optimal cost function for every possible terrain configuration as was done in Figure 4.8, another approach [39] is to use a given cost function and locally find more optimal paths through the sub-optimal regions, e.g., the path sections inside the red ellipses in Figure 4.8, using sampling based methods. In a different approach [96], the variation in generated paths with changing cost functions was used as a candidate path generation tool for SBMP rather than using the curves reviewed in Sections 4.3 and 4.3.2. In any case, the overall path will remain choppy with sharp turns, and will closely follow the boundaries of higher cost regions. For these reasons, the direct SBMP approach is more practical.

4.3.2 Sampling Based Motion Planning (SBMP) for Traversability- and Workspace-Aware Path Planning

The geometry-based SBMP introduced in Section 4.3 can be implemented by using interpolation or approximation. Because higher order polynomial interpolation exhibits more Runge phenomenon effects, spline interpolation is preferred in generating smooth continuous paths. A key difference between interpolation and approximation is that in interpolation, the resulting curves pass through all the defined points, while in approximation, using Bezier curves, the resulting curve only passes through the first and last points with the intermediate points serving as guides.

In this section, the formulation for piecewise spline interpolation for the path proposal step in SBMP is presented. Derivations are shown to highlight spline properties,
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

and to demonstrate inherent limitations of using interpolated splines compared to approximation using Bezier curves for active exploration. Thereafter, a traversability-aware evaluation function for the path evaluation step in SBMP will be developed.

Consider Figure 4.9, the goal of SBMP cubic spline interpolation is to fit piecewise spline paths from the start position through the intermediate points to the goal position. Considering one path, the goal is to fit piecewise splines $S_x$ through $n + 1$ pre-defined points such that

$$S_x = \begin{cases} P_i(x), & x_{i-1} \leq x \leq x_i, \\ \vdots & \vdots \\ P_n(x), & x_{n-1} \leq x \leq x_n, \end{cases} \quad (4.6)$$

$$P_i = a_i x^3 + b_i x^2 + c_i x + d_i, \quad (4.7)$$

where $i = 1, \ldots, n$; subject to the following constraints for continuity and smoothness

$$P_i(x_i) = y_i, \quad P_i(x_{i-1}) = y_{i-1}, \quad \dot{P}_i(x_i) = \dot{P}_{i+1}(x_i), \quad \text{and} \quad \ddot{P}_i(x_i) = \ddot{P}_{i+1}(x_i). \quad (4.8)$$

In this case, the range of evenly-spaced intermediate points for each path is chosen according to the ratio: range = \frac{\text{start to goal distance}}{\text{image height}}. Also, the orientation of the line joining the intermediate points maintains orthogonality with the start-goal orientation as shown in Figures 4.9(a) and 4.9(b). Because the spline must pass through each point, fewer intermediate points produce smoother curves as shown in Figures 4.9(c) and 4.9(d) – one intermediate point is considered adequate in this work. When choosing multiple intermediate points, the condition for progression along the $x$-axis
Figure 4.9: Cubic splines interpolating prescribed points for generation of sample paths. The larger circles represent the start and end positions and the smaller circles represent intermediate points.
in (4.6) must be met. This deterministic approach pre-defines the position of the intermediate points, but a probabilistic approach can also be employed.

An alternative to the point goal used in this scheme (Figure 4.9) is a goal region which can be a line [41]. Using a goal region, can help constrain the length of each spline to a fixed constant thereby eliminating distance considerations in the path evaluation function. However, because of the strict position requirements during self-supervised feature registration, a point goal is used in this work; also, the absence of lane boundaries to guide navigation means that a point goal can help maintain a straight heading to the final goal position at each cycle. Since (4.6) is in Cartesian coordinates, the change of coordinates shown in Figure 4.10 between image and Cartesian is used.

The second derivative of (4.7) is linear, and thus corresponds to a linear fit between
the end points, $\ddot{P}_i(x_{i-1}) = M_{i-1}$ and $\ddot{P}_i(x_i) = M_i$, such that [128]

$$\frac{\ddot{P}_i(x) - M_{i-1}}{x - x_{i-1}} = \frac{M_i - M_{i-1}}{x_i - x_{i-1}}, \quad (4.9)$$

or

$$\ddot{P}_i(x) = \frac{x_i - x}{x_i - x_{i-1}} M_{i-1} + \frac{x - x_{i-1}}{x_i - x_{i-1}} M_i. \quad (4.10)$$

Hence to determine the parameters in (4.7), the constraints in (4.8) are used. From (4.10),

$$P_i(x) = \frac{(x_i - x)^3}{6(x_i - x_{i-1})} M_{i-1} + \frac{(x - x_{i-1})^3}{6(x_i - x_{i-1})} M_i + c_i x + d_i, \quad (4.11)$$

giving

$$P_i(x_{i-1}) = \frac{(x_i - x_{i-1})^2}{6} M_{i-1} + c_i x_{i-1} + d_i = y_{i-1}, \quad (4.12)$$

$$P_i(x_i) = \frac{(x_i - x_{i-1})^2}{6} M_i + c_i x_i + d_i = y_i. \quad (4.13)$$

Solving (4.12) and (4.13) gives

$$c_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} - \frac{x_i - x_{i-1}}{6} (M_i - M_{i-1}), \quad (4.14)$$

$$d_i = \frac{x_i y_{i-1} - x_{i-1} y_i}{x_i - x_{i-1}} - \frac{x_i - x_{i-1}}{6} (x_i M_{i-1} - x_{i-1} M_i). \quad (4.15)$$

Note that $\frac{y_i - y_{i-1}}{x_i - x_{i-1}}$ is the first divided difference. Substituting in (4.11), the polynomial
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

The equation is

\[ P_i(x) = \frac{(x_i - x)^3}{6h_i} M_{i-1} + \frac{(x - x_{i-1})^3}{6h_i} M_i + \left( \frac{y_{i-1}}{h_i} - \frac{M_{i-1} h_i}{6} \right) (x_i - x) \]

\[ + \left( \frac{y_i}{h_i} - \frac{M_i h_i}{6} \right) (x - x_{i-1}), \]  

(4.16)

\[ h_i = x_i - x_{i-1}. \]  

(4.17)

Similarly, to determine \( M_i (i = 1, \ldots, n-1) \), note that from (4.16),

\[ \dot{P}(x) = -\frac{(x_i - x)^2}{2h_i} M_{i-1} + \frac{(x - x_{i-1})^2}{2h_i} M_i + \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{6} (M_i - M_{i-1}); \]  

(4.18)

thus,

\[ \dot{P}_i(x_i) = \frac{h_i}{3} M_i + \frac{y_i - y_{i-1}}{h_i} + \frac{h_i}{6} M_{i-1}, \]  

(4.19)

\[ \dot{P}_i(x_{i-1}) = -\frac{h_i}{3} M_{i-1} + \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{6} M_i, \]  

(4.20)

and from (4.20),

\[ \dot{P}_{i+1}(x_i) = -\frac{h_{i+1}}{3} M_i + \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{6} M_{i+1}. \]  

(4.21)

In applying (4.8), it is convenient to express the spline as a matrix-vector product, and this can be achieved by multiplying both sides of the constraint \( \dot{P}_i(x_i) = \dot{P}_{i+1}(x_i) \) by \( \frac{6}{h_{i+1} + h_i} \) and rearranging to give

\[ \frac{h_i}{h_{i+1} + h_i} M_{i-1} + 2M_i + \frac{h_{i+1}}{h_{i+1} + h_i} M_{i+1} = \frac{6}{h_{i+1} + h_i} \left( \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right), \]  

(4.22)
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

note that the RHS is a product of the second divided difference, thus (4.22) can be
re-written as

$$\alpha_i M_{i-1} + 2M_i + \beta_i M_{i+1} = 6f[x_{i-1}, x_i, x_{i+1}],$$ (4.23)

$$\alpha_i = \frac{h_i}{h_{i+1} + h_i}, \quad \beta_i = \frac{h_{i+1}}{h_{i+1} + h_i} = 1 - \alpha_i.$$ 

Finally, for natural splines, the remaining unknowns, $M_0$ and $M_n$, are set to zero,
and when combined with (4.23) gives the system of equations:

$$\begin{bmatrix}
1 & 0 & \alpha_i & 2 & 1 - \alpha_i & \vdots & \alpha_n-1 & 2 & 1 - \alpha_n-1 & 0 \\
\alpha_i & 2 & 1 - \alpha_i & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 1 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix} \begin{bmatrix}
M_0 \\
M_1 \\
\vdots \\
M_{n-1} \\
M_n \\
\end{bmatrix} = \begin{bmatrix}
0 \\
6f[x_0, x_1, x_2] \\
\vdots \\
6f[x_{n-2}, x_{n-1}, x_n] \\
0 \\
\end{bmatrix}. (4.24)$$

The parameters for the spline (4.6), (4.16) are now completely determined.

Some examples of paths generated for different configurations are shown in Figure 4.11. In Figure 4.11(a) $x_{\text{start}} \approx x_{\text{goal}}$ and the paths are seen to circle back between the end points and intermediate points in a form of Runge phenomenon (Figure 4.11(b)); and in Figure 4.11(c), when $x_{\text{start}} = x_{\text{goal}}$ a singularity is introduced in the spline parameter $\alpha_i$. A singularity also exists in $P_i(x)$ when $x_{\text{start}} = x_{\text{mid point}}$. Otherwise smooth splines are produced (Figure 4.12). Note that paths that have a section outside the image boundaries are not shown, or used. As discussed in Section 4.2, the peculiarity of this active exploration scheme means that the start–goal positions can
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

(a) $x_{\text{start}} \approx x_{\text{goal}}$.

(b) Paths oscillate between mid point and end points.

(c) $x_{\text{start}} = x_{\text{goal}}$: due to a singularity, no paths generated.

Figure 4.11: Some inapplicable situations when using spline interpolation.

assume any configuration, and although the cases with singularities can be handled by using intermediate points, such an approach is inconvenient. A more flexible path generation approach is required.
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

(a) Full image traverse. (b) Smooth splines generated. (c) Partial image traverse with different start–goal orientation. (d) Smooth splines with asymmetric profiles.

Figure 4.12: Generated splines when $|x_{\text{start}} - x_{\text{goal}}| \gg 0$. 
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

Figure 4.13: Generating sample paths using five control points for quartic Bezier curves (see Figure 4.9).

Because of the limitations highlighted in Figure 4.11 with using spline interpolation in certain configurations, approximation based on Bezier curves become attractive. Consider Figure 4.13 with five control points; the generated quartic Bezier curve will interpolate or pass through the end points and approximate the internal control points. The intermediate points have been deterministically chosen as was done in the case of spline interpolation, but, by the nature of Bezier curves, there are more intermediate control points in this case to guide the curve. Bezier curves are parametric functions; for \( n + 1 \) control points, \( p_0(x_0, y_0), p_1(x_1, y_1), \ldots, p_n(x_n, y_n) \), the Bezier curve is defined as [129]:

\[
B(\tau) = \sum_{i=0}^{n} p_i B_{i,n}(\tau), \tag{4.25}
\]

\[
B_{i,n}(\tau) = \frac{n!}{(n-i)!i!} (1 - \tau)^{n-1} \tau^i \quad \tau \in [0,1]. \tag{4.26}
\]
Therefore, for the quartic case with five control points, some key properties can be summarized as

\[ B(\tau) = p_0(1 - \tau)^4 + 4p_1(1 - \tau)^3\tau + 6p_2(1 - \tau)^2\tau^2 + 4p_3(1 - \tau)\tau^3 + p_4\tau^4, \quad (4.27) \]

and the interpolation and endpoint tangent properties

\[ B(0) = p_0, \quad B(1) = p_4, \quad (4.28) \]
\[ B'(0) = 4(p_1 - p_0), \quad B'(1) = 4(p_4 - p_3). \quad (4.29) \]

From (4.29), by carefully choosing the location of the intermediate control points, the curvature at the end points can be controlled to achieve a desired pose at the goal [120], but since a highly-manoeuvrable robot is used in this work, this approach is not used. Also, since each control point has an effect on the overall curve, manipulating the location of the internal control points can have undesirable effects on the maximum internal curvature, making path following by the robot more difficult. Five control points are considered adequate for this problem; adding more control points can reduce smoothness in terms of curvature changes along the curve.

In Figures 4.14 and 4.15, Bezier curves for the situations previously considered for spline interpolation in Figures 4.11 and 4.12 are presented. Noting that, practically, there is no requirement for the curves to pass through the intermediate control points—which simply serve as guides for the sample paths, the generated Bezier curves for these intermediate control points are smooth and symmetric, and do not exhibit the limitations exhibited by spline interpolation in Figure 4.11.
4.3. PATH PLANNING FOR NON-BINARY TERRAINS

Figure 4.14: Sample paths generated by using quartic Bezier curves for the situations in Figure 4.11.

4.3.3 SBMP Evaluation Function

To choose a path from the proposed candidate paths, considerations include: collision detection, path curvature, path length, and, importantly, traversability.

As shown in Figure 4.16, for the purposes of collision detection and evaluating traversability, to account for the robot workspace, a circle of external and internal radii
Figure 4.15: Sample paths generated by using quartic Bezier curves for the situations in Figure 4.12.

based on the robot’s physical dimensions is simulated at steps along each candidate path on the cost image, and sample points (green dots within red circles in Figure 4.16) inside the circle and on its borders (red circles in Figure 4.16) are used in the analysis. Some allowance is given around the robot’s workspace, defined by its largest external dimensions, to allow for path following errors. For collision detection, if any
4.3. PATH PLANNING FOR NON-BINARY TERRAINs

After paths that collide with an obstacle have been eliminated, the remaining feasible paths $\delta$ are evaluated by using the following parameters:

1. Maximum path curvature: $\kappa_{\text{max}}$. 

$\kappa(\tau) = \frac{|x'(\tau)y''(\tau) - y'(\tau)x''(\tau)|}{(x'^2(\tau) + y'^2(\tau))^{\frac{3}{2}}} \quad \tau \in [0, 1]$ 

where $\kappa$ is the inverse of the radius of curvature, thus a smaller $\kappa$ is desirable. The unsigned curvature is used because the direction of approach to the goal is not important.
2. Path length: The path length can be approximated by the arc length as
\[ P \approx \sum_{i=1}^{n} \sqrt{(x(\tau_i) - x(\tau_{i-1}))^2 + (y(\tau_i) - y(\tau_{i-1}))^2}, \text{ given } n + 1 \text{ points.} \]

3. Traversability: \( T = \sum_{p=1}^{N.P} \sum_{q=1}^{S.P} c(p)(q) \); i.e., for each path point \( p \) at which the workspace is simulated, traversal costs \( c(n) \) from a total of \( S.P \) points are sampled within the workspace. The total number of path points \( N.P \) used in the simulation changes with path length.

The SBMP evaluation function for path \( i \) is thus
\[ \Lambda(\delta_i) = \lambda \frac{\kappa_{\max,i}}{\max(\kappa_{\max})} + \mu \frac{P_i}{\max(P)} + \nu \frac{T_i}{\max(T)}. \quad (4.31) \]

The normalizing parameters in (4.31) are the corresponding maximum values in \( \delta \).

To abstract path length from the traversability component, it is preferable to use \( \frac{T_i}{N.P \times S.P} \) instead of \( T_i \). Therefore, for \( p \) total feasible paths in \( \delta \), the chosen path \( \delta_m \) is path
\[ m : \Lambda(\delta_m) = \min_{\delta_i \in \delta}(\Lambda(\delta_i)) \quad \delta = [\delta_1, \ldots, \delta_p]. \quad (4.32) \]

It is convenient to set
\[ \lambda + \mu + \nu = 1 \quad \text{such that } \Lambda(\delta_i) = 1 \quad \text{for} \]
\[ \delta_i : (\kappa_{\max,i} = \max(\kappa_{\max}), \ P_i = \max(P), \ T_i = \max(T)). \quad (4.33) \]

And when choosing the parameters \( \lambda, \mu \) and \( \nu, \nu \) should dominate to emphasize traversability. One choice is \( \lambda = 0.2, \mu = 0.3 \) and \( \nu = 0.5 \); here curvature is given the least weighting because the operating speed in active exploration is typically low,
and all paths have been checked to be within maximum curvature limits; the path following controller can better negotiate difficult turns at lower speeds.

Because fixed pre-defined points are used in the shown examples, the curvature components of \( \Lambda \) are smooth for each candidate path. But if the intermediate points are probabilistic or pseudo-randomized, curvature direction can vary along each path. By tracking the signed curvature changes along each path, the relative smoothness of each path can be determined and included in \( \Lambda \). But, in this case, the pre-defined paths are all smooth, hence, relative smoothness is not included in \( \Lambda \).

In Figures 4.18 to 4.26, examples of chosen paths are shown for three samples terrains \( A, B, \) and \( C \). The traversal costs for the outdoor terrain types in Sample Terrain \( C \) are shown in Table 4.3, and the costs used for the indoor terrains are as shown in Table 4.1. Cluster details for each sample terrain are shown in Figures 4.6, 4.17 and 3.17 respectively. Similar to the relative parameter weightings, changing the cost function changes the relative class cost ratios, thus can affect the choice of path. Compared to the linear cost function, the exponential cost functions increasingly magnify cost differences between terrain types, making the chosen path increasingly avoid the comparatively higher costs regions—this effect increases with traversability weighting. From the results, a cost function in the range \( e^{5a} \) to \( e^{10a} \) is a good choice;
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Table 4.3: Traversal costs for outdoor Sample Terrain $C$.

<table>
<thead>
<tr>
<th>Terrain</th>
<th>Grass</th>
<th>Concrete</th>
<th>Sand</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{a}$</td>
<td>2.145</td>
<td>2.464</td>
<td>3.000</td>
</tr>
</tbody>
</table>

the large numbers can be appropriately scaled to limit large sums.

In the previous results, small isolated unlabeled clusters were given zero costs because data cannot be acquired from them. Consequently, paths that pass through such clusters will have an advantage. The alternative is to give such clusters very high costs thereby penalizing paths that pass through them as shown in Figures 4.27 and 4.29. The choice of strategy for handling small isolated unlabeled clusters will depend on the specific mission requirements and policy, but the former approach is preferred in this work because such small distributed clusters will have small overall effect on traversability.

4.4 Terrain Undulations, Obstacle Avoidance and Speed Adaptation

As discussed in Section 4.1, a complete characterization of the traversability of a terrain will include inherent characteristics of the constituent terrain types, e.g., roughness and slip, and characteristics related to the overall terrain profile or geometric unevenness, resulting in a combined terrain traversability cost map. In Section 4.2, a self-supervised approach for generating a cost map based on inherent characteristics of constituent terrain types only was proposed using the transformed image from Section 3.1.1 that represents a real world 2D planar map. This cost map can be further augmented with the overall terrain profile characteristics, such as undulations and obstacles (defined in Section 2.3), expressed in traversability cost terms as was done using supervised approaches with pre-learned models in [39]. Obtaining
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Figure 4.18: Chosen paths for Sample Terrain A when $\lambda = 0.2$, $\mu = 0.3$ and $v = 0.5$. 

$q = 10\tilde{a}$. $q = e^{5\tilde{a}}$. $q = e^{10\tilde{a}}$. 
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

$q = 10\hat{a}$.  
$q = c^{5\hat{a}}$.  
$q = c^{10\hat{a}}$.

Figure 4.19: Chosen paths for Sample Terrain A when $\lambda = 0.15$, $\mu = 0.25$ and $v = 0.6$. 
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

$q = 10\bar{a}$.

$q = 5\bar{a}$.

$q = 10\bar{a}$.

Figure 4.20: Chosen paths for Sample Terrain A when $\lambda = 0$, $\mu = 0$ and $\nu = 1$. 
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Figure 4.21: Chosen paths for Sample Terrain B when $\lambda = 0.2$, $\mu = 0.3$ and $\nu = 0.5$. 

$q = 10\tilde{a}$. $q = e^{5\tilde{a}}$. $q = e^{10\tilde{a}}$. 
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

\[ q = 10\tilde{a}. \]

\[ q = e^{5\tilde{a}}. \]

\[ q = e^{10\tilde{a}}. \]

Figure 4.22: Chosen paths for Sample Terrain B when \( \lambda = 0.15 \), \( \mu = 0.25 \) and \( \nu = 0.6 \).
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Figure 4.23: Chosen paths for Sample Terrain B when $\lambda = 0$, $\mu = 0$ and $\upsilon = 1$.

$q = 10\tilde{a}$.  
$q = e^{5\tilde{a}}$.  
$q = e^{10\tilde{a}}$.  

Figure 4.23: Chosen paths for Sample Terrain B when $\lambda = 0$, $\mu = 0$ and $\upsilon = 1$. 
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

$q = 10\tilde{a}$.  
$q = e^{5\tilde{a}}$.  
$q = e^{10\tilde{a}}$.  

Figure 4.24: Chosen paths for Sample Terrain C when $\lambda = 0.2$, $\mu = 0.3$ and $\nu = 0.5$. 
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Figure 4.25: Chosen paths for Sample Terrain C when $\lambda = 0.15$, $\mu = 0.25$ and $\nu = 0.6$.
Figure 4.26: Chosen paths for Sample Terrain C when $\lambda = 0$, $\mu = 0$ and $v = 1$. 

$q = 10\tilde{a}$. $q = e^{5\tilde{a}}$. $q = e^{10\tilde{a}}$. 
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Figure 4.27: Chosen paths for Figure 4.24 with small unlabeled clusters given high costs.

\[ q = e^{5\tilde{a}}. \quad q = e^{10\tilde{a}}. \]
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Figure 4.28: Chosen paths for Figure 4.25 with small unlabeled clusters given high costs.

$q = e^{5\alpha}$.  
$q = e^{10\alpha}$.  

Figure 4.28: Chosen paths for Figure 4.25 with small unlabeled clusters given high costs.
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

Figure 4.29: Chosen paths for Figure 4.26 with small unlabeled clusters given high costs.

\[ q = e^{5\alpha}. \quad q = e^{10\alpha}. \]
information about the terrain profile to express in cost terms requires complementary sensing using remote, exteroceptive 3D sensors, e.g., a stereocamera or LiDAR sensor. Remotely-obtained information relating to terrain unevenness characteristics such as pitch and roll for a terrain patch can be simply combined with the inherent characteristics of its constituent terrain types as a weighted linear sum as was done in (4.31) and [96] to form the cells in the cost map without the need for pre-learned models, unlike in [39]. In such a scheme, inherent terrain characteristics of the constituent terrain types will be acquired as discussed in Section 4.2 by using a dedicated mars rover-like appendage for slope-invariant terrain characterization; this is the subject of future work.

The image used to build the cost map in Section 4.2 can come from an aerial source, which will cover a wider area, or from a vehicle-mounted camera that only provides local information about the environment. Although the latter case is considered in this work, the perspective transformation method for ortho-projection in Section 3.1.1 applies in both cases, to handle the camera’s horizontal and vertical FOV and to appropriately scale the image [37], [130].

For the case of an image obtained from a vehicle-mounted camera, some obstacles can occlude certain areas in the image. In such cases, the perspective transform homography (3.13) fills out the occluded pixels with obstacle pixels as demonstrated in Figure 4.30(i). Intuitively, when the vehicle is surrounded by obstacles, e.g., trees in a forest, the occluded regions behind the trees that are within the ROI will be represented in the transformed image by tree pixels. One way to mark off obstacles and occluded areas is to use a line of sight approach that depends on complementary depth perception using a stereocamera or LiDAR scanner. In this approach, the pose
4.4. TERRAIN UNDULATIONS, OBSTACLE AVOIDANCE AND SPEED ADAPTATION

(a) Indoor terrain with obstacle. (b) 2D point cloud. (c) ROI+blind spot.

(d) Figure 4.30(c) (e) 20 pixel padding (f) Bounding box transformed to Tr per obstacle point. around padded obstacle points.

(g) Line of sight from robot. (h) Occluded area (ROI+blind spot).

Figure 4.30: An example of the process for incorporating obstacle avoidance in the proposed approach by using point clouds from a 2D LiDAR sensor.
of the robot and the boundaries of the obstacle are used to mark off the obstacle and occluded areas using a straight line of sight to obstacle edges. This process is demonstrated in Figure 4.30. The area of interest for obstacle avoidance is the ROI in addition to the camera blindspot—i.e., the area between the robot and the ROI. In Figure 4.30, a 2D LiDAR scan is used to generate a 2D point cloud of the environment (Figure 4.30(b)) which is transformed to pixel coordinates with some allowance as shown in Figure 4.30(d) to 4.30(e). A bounding box is subsequently fit around the obstacle external contours as discussed in Section 4.2. Then using a straight line of sight from the robot to the outer edges of the obstacle, the obstacle and occluded areas are marked off (Figure 4.30(h)), and mapped to the transformed ROI.

Finally, if the interim goal point is in the marked-off areas, the point is moved to a free area in the image and the marked-off region given a prohibitively high cost in the cost image for path planning. In the case of total occlusion, the robot rotates and restarts the motion planning process.

A limitation of using a 2D point cloud is that it provides limited information about the interconnections among constituent points. This limited information is shown in Figure 4.30(c) where the slanting handles in the lower corners of the obstacle in Figure 4.30(a) are only represented as separate points at the height of the 2D section used for generating the point cloud. While padding the obstacle points as shown in Figure 4.30(e) can help mitigate this incomplete information, a 3D point cloud will provide a better representation of the environment. 3D LiDAR can also be used for avoiding negative obstacles [131].
Apart from prioritizing low traversal cost terrain when navigating the vehicle, another goal of terrain-adaptive navigation is speed adaptation. In planetary robotics, which has motivated most of the near-to-far approaches, emphasis has been on terrain assessment to prevent the robot from getting stuck and potentially causing the premature end of costly space programs. Therefore, speed optimisation has not been explicitly explored in these near-to-far approaches, within the limits of this review. Due to their primary goal of discovery and data gathering, and distance limitations, planetary robots tend to be designed and operated to move slowly and deliberately. However, some works exist in the literature on speed optimization. An experience-based approach to speed optimization in which the robot learns optimal speeds in a repeating path by tuning pre-defined parameters has been proposed [132]. In [133], a shock-based adaptive method was employed to regulate speed in a terrain independent approach. Also, a terrain dependent approach was presented in [134]; it proposed a method to choose ride velocity based on terrain vibration data in a supervised scheme that used dictionary techniques. Speed optimization gets more difficult with more unstructured off-road terrain that exhibit significant intermingling between terrain types. It should be noted though that these mentioned works are not within the near-to-far paradigm.

4.5 Summary of the Motion Planning Strategy

In this chapter, the considerations for the choice of linear acceleration as the sensor modality for estimating traversability were detailed. The self-supervised feature registration process and its link to path planning was introduced. A cost function that maps linear acceleration to traversability costs was designed, and a practical
SBMP-based path planning approach that incorporates traversability evaluation was proposed.

The Dijkstra and A* algorithms can produce choppy paths that do not take cognizance of the robot’s workspace; and the use of naive spline interpolation in SBMP is limited to a fixed range of start–goal orientations. Hence, the flexibility and practicality of Bezier approximation makes it the choice for path planning using SBMP in the proposed approach.

The self-supervised feature registration step in the proposed approach is the most challenging step for motion planning. It can require sharp turns between the concatenated paths (Figure 4.4) that may not be feasible to follow in continuous driving. But the use of an exploratory skid-steer robots with high manoeuvrability enables the employment of a mitigating line and point turn approach for that step. The line and point turn approach entails rotating to the start pose of each path before path following.
Chapter 5

Experimental Methods and Results

This chapter describes the approach used for localization and navigation, and provides implementation details. Details of the test hardware are also given before results from indoor and outdoor experiments are presented and discussed.

5.1 Localization, Navigation and Control

The path points in image coordinates must be transformed to robot and map coordinates for the path following controller to navigate the robot through the terrain. Although wheel odometry is a convenient choice for localization because it can easily be used to track the robot in 2D Cartesian coordinates from the start position, slip and error accumulation presents a problem. Thus, it is necessary to use another measurement such as GNSS data and/or visual odometry [41] to augment wheel odometry measurements in a filter; in this work, GNSS data is used. Also, because the environment is unmapped, the navigation scheme relies principally on the local path planner that continually attempts to maintain a straight heading to the specified goal while avoiding obstacles and choosing low traversability cost terrain.

As shown in Figure 5.1, the start position of the robot is the origin of its global
map. The fixed position of the transformed ROI with respect to the robot \((x_{r,i,0}, y_{r,i,0})\) is determined by calibration. Thus, any waypoint in image coordinates \((u, v)\), can be transformed to robot and map coordinates \(S_r\) and \(S_m\), such that

\[
S = \begin{bmatrix} x_{r,i,0} & y_{r,i,0} \end{bmatrix}^T, \quad \text{and} \quad p = \begin{bmatrix} -v & -u & 1 \end{bmatrix}^T, \quad (5.1)
\]

\[
S_r = \begin{bmatrix} x_{r,i} & y_{r,i} & 1 \end{bmatrix}^T = \begin{bmatrix} I & S \\ 0 & 1 \end{bmatrix} p, \quad (5.2)
\]

\[
S_{xy} = \begin{bmatrix} R_{m,r,x} & R_{m,r,y} \end{bmatrix}^T, \quad (5.3)
\]

\[
S_m = \begin{bmatrix} x_{m,i} & y_{m,i} & 1 \end{bmatrix}^T = \begin{bmatrix} R(\theta) & S_{xy} \\ 0 & 1 \end{bmatrix} S_r, \quad (5.4)
\]
where $R(\cdot)$ is the rotation matrix.

GNSS coordinates (in longitude and latitude) are transformed by using Mercartor transformation to UTM and, as was done in (5.4), the $UTM$-$Map$ transformation is found using $(R(\phi), R_{utm,g})$. Thus, the goal is specified in Cartesian coordinates, or in GNSS coordinates that are transformed to Cartesian coordinates by using the $UTM$-$Map$ transformation. Details of the filter architecture are shown in Figure 5.2 according to open source implementations from [135].

In Section 4.3.1, it was noted that the traversability data acquisition method is a complementary consideration, similarly the choice of controller is complementary. While many path following controllers and strategies exist in the literature [136], [137], to show the robustness of the proposed approach, the simple feedback linearized (FBL) controller in [138] is employed for path following based on a simplified kinematic model. The errors are based on instantaneous closest path points as illustrated in Figure 5.1 (Left). Details of the controller are shown in Appendix B.2.
5.2 Experimental System Overview, Test Facilities and Locations

The test platform was a Clearpath Husky A200 unmanned ground vehicle, a 50 kg skid-steer robot with external dimensions $0.99 \times 0.67 \times 0.39$ m, and a maximum speed of 1 m/s. Husky was equipped with wheel encoders and a calibrated triple-axis LORD Microstrain 3DM-GX5-25 IMU comprising a magnetometer, gyroscope and accelerometer. Images were collected with an 8 MP monocamera mounted on Husky at a height of 0.5 to 1.2 m. To reduce light reflection from reflective classes e.g., gravel, the camera was set at about $0^\circ$ to $10^\circ$ from the horizontal towards the ground. Outdoors, a Swift GPS unit was used with real-time kinematic (RTK) corrections from a base station. The GNSS, wheel encoder and IMU sensor data were fused with an Extended Kalman Filter hosted on a Lenovo computer with an Intel Core i7 processor which runs the Robot Operating System (ROS) on Ubuntu for interacting with Husky. A Vicon motion camera tracker system provided pose estimates for localization during indoor testing. Hardware specifications are listed in Appendix C.

The indoor prototyping terrain is a square flat area of sides 8.6 m with a base synthetic black mat. Three reconfigurable terrain elements are used: (1) wood with distributed wooden protrusions (2) gravel on a wooden base and (3) synthetic grass on a wooden base. Each unit is square with 0.9 m sides. These four terrain types represent some of the typical classes found in an off-road environment and exhibit representative colour differences. The fixed ROI covers a square area of sides 3.2 m as empirically chosen to reduce interpolation effects from (3.13). The indoor terrain and hardware setup are shown in Figures 5.3 and 5.4.

The outdoor test was done in the terrain in Figure 5.5. This terrain was chosen because it had in close proximity four different terrain types: concrete, sand, grass
and dandelions. Apart from the similarity in colour between some of the constituent terrain types, the terrain was difficult because there was considerable intermixing.

5.3 Experimental Procedure and Results

As illustrated in Figure 2.7, three scenarios summarize the approach: (1) class generation; (2) class identification; and, (3) novelty detection.

In Scenario 1, at start up, the feature list is empty, the algorithm identifies the per-class representative clusters by size. For each representative cluster that meets a size and solidity threshold for robot traversal to collect data, the algorithm fits
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

Figure 5.4: The test platform, Husky.

Figure 5.5: The outdoor terrain for preliminary testing.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

a bounding contour [102], and then fits a minimum area bounding box around the contour. Using zero costs, paths are planned so that the robot drives longitudinally through the center of each box for feature registration before proceeding to the interim goal position (Figure 5.6, left top-bottom pair). Note that the bounding contour for the black tiles is the whole image, thus, a region near the robot is sampled. Also, note that the path sequence follows the terrain class numbers; this sequence is important for feature registration (4.2). The image waypoints are transformed to map coordinates for path following.

In Scenario 2 the robot has previously encountered all the terrain classes, so it uses the feature list to generate a labelled traversability cost image then plans a path to the interim goal as shown in Figure 5.6 (middle top-bottom pair).

Finally, in Scenario 3 (Figure 5.6, right top-bottom pair), the robot identifies the

Figure 5.6: The three test scenarios (L-R): Top: Source image with ROI. Bottom: Transformed ROI with paths from path planner [87]. (© 2021 IEEE)
new grass cluster, and fits a bounding box around it to which a path is planned using known costs. Lastly, a path is planned to the interim goal. In all cases, the robot assumes a direct heading to the specified final goal position—the interim goal position is the line of intersection with the image. In the indoor tests, the robot performed Scenarios 1 and 3 in a forward run away from the start position and Scenario 2 in a backward run towards the start position; all at 0.5 m/s. Between each scenario, the terrain elements were rearranged. Outdoors, all tests were done in a forward run.

5.3.1 Results from Indoor Tests with A* Path Planner

As discussed in Section 4.3.1, the paths in Figure 5.6 are choppy with sudden turns, but the line and point turn approach, introduced in Section 4.5, facilitated path following in Scenario 1. This strategy takes advantage of the manoeuvrability of the skid-steer robot particularly as speed is not critical in this context. Nonetheless, the basic FBL controller still struggles with the sharp turns as seen in Figure 5.7, although this is not the focus of this work. Also, the paths tend to closely follow high cost terrains, hence, the robot will partially drive over the edges of such terrains.

Furthermore, classes that are too small to be driven over to collect data were treated as low cost regions, otherwise they have unrealistic effects on path planning. For example, the grass section—which has the lowest cost—is walled off by its high cost wooden borders. However, the implication is that paths can be planned through unrealistic crevices such as between rough patches. Notice how the path is planned through the tyre burn marks in Figure 5.6 (middle) but not in the reverse run in Figure 5.6 (right). This is because the tyre burn marks were identified as a continuous section as shown in Figure 3.2(e) (second right) in the first case but as a discontinuous noisy
patch in the second case, all of unknown traversability. Smaller noisy clusters were removed morphologically.

The paths in Figure 5.6 (left), have been transformed to map coordinates using (5.4) and a straight heading from start across the blind spot has been concatenated. The FBL controller does not track the sharp turns as well as the straight paths.

In the reverse run for Scenario 2 in Figure 5.8, the path has fewer sharp turns because it comprises only two paths: a straight heading across the blind spot and the paths from Figure 5.6 (middle) all in map coordinates. The path following error is more pronounced during turning due to skidding and controller undercutting.

Finally, path following results for Scenario 3 are shown in Figure 5.9. In this case, the significant path following error at the end is caused by unintended skidding as the vehicle turns and drags the wooden terrain element along Figure 5.6 (right).
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

Figure 5.8: Scenario 2: Class identification [87]. (© 2021 IEEE)

Figure 5.9: Scenario 3: Novelty detection [87]. (© 2021 IEEE)
In Figure 5.7 to Figure 5.9, the controller path following errors are magnified by skidding as the robot drags the freely-moving indoor terrain elements along as it spins to turn. This was not intended. In addition, the controller errors are based on the instantaneous closest point to the vehicle which means sharp turns may be undercut—not diligently followed.

The foregoing results were from tests done using Test Configuration 3 in Figure 4.5 with a LORD Microstrain 3DM-GX3-25 IMU. In Section 4.3.1, the dependence of IMU mounting location on traversability costs was demonstrated in addition to the dependence of A* path optimality on traversability costs. To further demonstrate the effect of choice of IMU and its mounting location on traversability costs—hence A* path optimality—the preceding tests were repeated using Test Configuration 2 with the same cost function $q = e^{8\bar{a}}$.

Compared to the paths in Figure 5.6, the paths generated from Test Configuration 2 in Figure 5.11(b), Figure 5.12(b) and Figure 5.13(b) are more sub-optimal. However, in Figure 5.10(b), for which paths were planned using zero costs, the paths were much more optimal like the equivalent paths in Test Configuration 3. This reduced optimality means that the scaling between the components of the A* cost function in (4.3) are much more imbalanced in Test Configuration 2; similar to the initial path at the top of Figure 4.7(i) vs Figure 4.7(h) respectively, the contribution from the traversability cost is significantly higher than in the case of Test Configuration 3. Because of the relatively low speed, the controller was able to successfully navigate the sharp turns; however it had difficulty following the choppy sections in Figures 5.12(c) and 5.13(c). Subsequent tests were done using Test Configuration 2 and a LORD Microstrain 3DM-GX5-25 IMU.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

Figure 5.10: Scenario 1: All terrain types unknown; Test Configuration 2.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

(a) Source image with ROI.

(b) Transformed image and planned path.

(c) Path following results.

Figure 5.11: Scenario 2: All significant terrain types known; Test Configuration 2.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

(a) Source image with ROI.

(b) Transformed image and planned paths.

(c) Path following results.

Figure 5.12: Scenario 3: At least one unknown terrain type; Test Configuration 2.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

(a) Source image with ROI.
(b) Transformed image and planned path.
(c) Path following results.

Figure 5.13: Scenario 2: All significant terrain types known; Test Configuration 2.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

5.3.2 Results from Field Tests with A* Path Planner

Having successfully demonstrated the proposed scheme indoors, and shown that the vision pipeline works for outdoor terrain in Chapters 3 and 4, the main focus of the completed field testing was on the practical limitations of outdoor localization. The proposed scheme has a key dependence on reasonable localization estimates for self-supervised feature registration and path following. Indoors, the motion camera system gives high accuracy position estimates, but, outdoors, position estimates from the sensor suite can be inaccurate and require filtering multiple sensor inputs for reasonable accuracy.

In Figure 5.14 to Figure 5.15, results are shown for tests done using the general localization scheme illustrated in Figure 5.2; pose estimates are obtained from an Extended Kalman Filter (EKF) using a combination of corrected GNSS, wheel odometry and IMU inputs. In this outdoor test, the two steps cover only Scenarios 1 and 3 as dictated by the environment; both tests are done in a forward run. Compared to the cases in Figures 5.16 and 5.18 for wheel odometry and IMU inputs only, the initial errors are significant, but settle to similar ranges in the latter stages of the first step. The source of the initial error can be seen in Figure 5.19 where pose estimates were obtained from corrected GNSS and IMU fusion only; it must be noted that the base station position was not calibrated, which makes the RTK corrections inaccurate. Although the odometry and IMU only filter gave good pose estimates considering the multiple turns, it must be noted that tests were done in dry, high-traction conditions. Similar to the indoor case, note that the paths generated when known traversability costs were used are noticeably sub-optimal, in Figures 5.15(b) and 5.17(b).
Figure 5.14: Scenario 1: All terrain types unknown; GNSS + odometry + IMU.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

(a) Source image with ROI.

(b) Transformed image and planned paths.

(c) Path following results.

Figure 5.15: Scenario 3: At least one unknown terrain type; GNSS+odometry+IMU.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

Figure 5.16: Scenario 1: All terrain types unknown; odometry + IMU.
Figure 5.17: Scenario 2: All significant terrain types known; odometry + IMU.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

(a) Source image with ROI.

(b) Transformed image and planned paths.

(c) Path following results.

Figure 5.18: Scenario 1: All terrain types unknown; odometry + IMU.
5.3. EXPERIMENTAL PROCEDURE AND RESULTS

(a) Source image with ROI.

(b) Transformed image and planned paths.

(c) Path following results.

Figure 5.19: Scenario 1: All terrain types unknown; GNSS + IMU.
5.3.3 Results from Indoor Tests SBMP

In Chapter 4, and the foregoing section, the practical benefits of SBMP path planning based on Bezier curves over A* have been demonstrated. Thus, in this section, results are shown from tests done using SBMP. First, the feasibility of the path profiles for which analysis was done in Section 4.3.2 is demonstrated by the ability of the controller to follow the paths (Figure 5.20). After heading straight across the camera blind spot, the controller successfully guides the robot to follow the proposed paths cyclically, as shown in Figure 5.20(c).

The test results in Figures 5.21 to 5.24 show that the aims of the test scenarios are achieved with smooth workspace-aware paths. Overall, the SBMP approach is workspace aware and generates smooth paths that facilitate path following; however, it sacrifices some optimality in terms of path length, for practicality. The test results are for $\lambda = 0.1$, $\mu = 0.2$ and $\nu = 0.7$ in Eq. 4.31, and $q = e^{8a}$.

5.4 Training a CNN with the Labelled Images

To leverage the robustness and speed of supervised end-to-end schemes, a semantic segmentation network can be trained on the database of transformed ROI-cost image pairs. In addition to a generic obstacle class, clusters that do not meet the size limit for data collection can be treated as an unknown class; and the cost image thresholded into unique class labels in steps of 0.5 units (Table 4.2) similar to [21], [39]. Figures 5.25 and 4.4(a) show examples of clustered images for representative terrains. While the non-discretionary labelling of clusters results in naively labelled pixels, human-labelled data also struggles in these environments as seen in [139], or in [3] where fusion with high resolution LiDAR did not give good results for rough
5.4. TRAINING A CNN WITH THE LABELLED IMAGES

![Source image with ROI.](image1)

(a) Source image with ROI.

![Transformed image and planned paths.](image2)

(b) Transformed image and planned paths.

![Path following results.](image3)

(c) Path following results.

Figure 5.20: Path following tests for proposed paths.
Figure 5.21: Scenario 1: All terrain types unknown
5.4. TRAINING A CNN WITH THE LABELLED IMAGES

Figure 5.22: Scenario 2: All significant terrain types known.
Figure 5.23: Scenario 3: At least one unknown terrain type.
5.4. TRAINING A CNN WITH THE LABELLED IMAGES

Figure 5.24: Scenario 2: All significant terrain types known.
5.4. TRAINING A CNN WITH THE LABELLED IMAGES

(a) Terrain with sample ROI.

(b) Transformed ROI (Tr).

(c) Clusters identified as unique by the vision pipeline. Black sections are noisy clusters.

Figure 5.25: Classification of terrain types by the vision pipeline (see also Figure 4.4(a)).
terrain classification. It is important to note that the vision-based classification in these sample figures is nominal; nominally distinct classes can fall into the same traversal cost-based functional class after feature registration. Also note that the ROIs are not calibrated. From Figures 4.4(a) (upper figure) and Figure 5.25, the non-uniqueness in shape across classes due to the natural random distribution of terrain classes in off-road environments becomes evident. Similar problems have been tackled in the field of remote sensing and planetary robotics by using texture features to segment 2D colour and monochrome images [16], [31], [58]. If the ROI is well-sized, and the distortion caused by the transformation is considered consistent, then such texture features are also relevant here since CNNs are also sensitive to texture.

Although CNNs will learn some latent representation of the input data regardless of form, it remains important to consider the relative benefits of training with input data in the original image plane vs using the transformed ROI as input. Along these lines, in [140], distortions due to camera FOV and camera position were corrected with respect to human perception using the keystone technique, and it was shown that the results for the data of corrected images versus the data of distorted images were not significantly different. However, with particular reference to off-road terrains, ortho-projected images gave better results in [11].

5.5 Some Limitations of the Proposed Approach

As demonstrated in Chapter 3, the system requires systematic tuning of colour thresholds to achieve robustness; but in the absence of labelled data, or for terrain comprising classes with unknown traversal costs, it has good utility. Also, taking the classical approach of self-supervision as a tool for labelling data for training [6], [7], [8], without
5.5. SOME LIMITATIONS OF THE PROPOSED APPROACH

Simultaneously navigating to a goal, then robustness in the vision space is not critical. Strict colour thresholds will increase false negatives, which is preferred because labels are based on traversal costs; e.g., two grass clusters identified as different classes by colour will still have a high likelihood of having the same range of traversal cost. For example, the lower figure in Figure 4.4(a) shows the effect of illumination: although the terrain is all grass, a structure casts a shadow on a section, and the algorithm clusters it separately. But since labels are based on traversal costs, both sections will have the same label after feature registration. However, non-unique clusters reduce navigation efficiency.

Some difficult cases of significantly unstructured terrains are shown in Figure 5.26 [141]. In these cases, considerable intermingling between terrain types means that the robot will inevitably pass through all the terrain types. These cases show the benefits of SBMP, which emphasizes practicality over optimality; as was discussed in Section 4.3.1 using Figure 4.8, an optimal path in traversal-cost terms would meander impractically in these terrains. Because the boundaries of the multiple constituent terrain types with different traversal costs and a wide range of sizes and shapes are not clearly defined, they cannot be naively regularized as a means to smooth the paths. An interesting case is when the terrain is partially or fully covered with another terrain type, such as leaves, snow or dust. It is impossible for the system to tell what the base terrain is in such cases. However, it must be noted that an expert labeller, or a teleoperator, will find these cases similarly challenging if they are also constrained to strictly follow a low-cost path through such terrains.

An underlying assumption behind the proposed method, and colour-based near-to-far methods generally, is that a terrain type will have a narrow range of colour
5.5. SOME LIMITATIONS OF THE PROPOSED APPROACH

Figure 5.26: Some challenging cases because of significant intermixing between terrain types, and partial snow and dust cover.

variations in the same illumination conditions, and its traversability characteristics will vary reasonably. As discussed earlier in this section, the colour variation assumption is less of a challenge because the labelling is done by traversal cost; however, some terrain types, e.g., grass, can have a considerable range of traversal costs even before they are considered obstacles. To handle varying traversability characteristics in a terrain type, a plane can be fit to the concerned terrain patch in the image space to determine a representative height as was done in [20], [21]. This height can then be used to augment the previously registered costs for that terrain type during path
5.5. SOME LIMITATIONS OF THE PROPOSED APPROACH

planning. Another difficult case is loose vs wet sand or dry vs wet grass. A general strategy to ameliorate such difficulties is to intermittently sample known terrain types and update the characteristics as an average.
Chapter 6

Conclusions

This thesis proposed a method to improve navigation autonomy for autonomous vehicles in unknown off-road environments. The proposed method has been demonstrated through experiments in realistic and applicable environments using a representative exploratory robotic vehicle. The critical components of the proposed method are the: vision; traversability characterization; motion planning; and localization pipelines. The proposed vision pipeline takes an input image from a vehicle-mounted camera and identifies the constituent terrain types by colour in an output image. Also, an approach to traversability characterization was proposed to map planar linear acceleration readings to roughness-based traversability costs for motion planning. In addition, a dictionary-based feature registration process was developed to store corresponding image and traversability characteristics for each identified terrain type. Finally, using known traversability cost information when available, a sampling based motion planning method was developed to generate smooth, traversable and collision-free paths for a path following controller to guide simultaneous navigation and feature registration in active exploration.
The experimental results demonstrate the higher autonomy engendered by the proposed method compared to previous expert-guided schemes that also aim to achieve terrain adaptive navigation, and show the proposed method’s compatibility with contemporary end-to-end CNN-based navigation systems.

6.1 Contributions

A traversability- and workspace-aware self-supervised system for autonomous navigation is a complex system that includes multiple complementary parts and considerations. The key contributions in this thesis arose from the process of developing the components of the proposed approach, making it compatible with contemporary CNN-based end-to-end navigation approaches, and solving identified challenges. The contributions include:

1. A method to improve end-to-end semantic segmentation performance in off-road environments by RGB-LBP multimodal fusion [42]. The results show that LBP can provide added discriminating features for improved semantic segmentation performance in such environments.

2. A study of cross-domain performance of datasets and trained networks in off-road environments [42]. The main insight from this study is that off-road environments exhibit significant cross-domain differences, which makes a self-supervised approach attractive.

3. A new and effective self-supervised feature registration approach that does not require teleoperation or expert-defined paths [87].
4. An effective method for mapping sensor readings to costs for motion planning through off-road terrains [87]. Although this analysis was done with particular reference to terrain roughness, it can be extended to other traversability characteristics.

5. A new approach to SBMP that accounts for the high level of flexibility required for convenient and effective active exploration. This includes a cost function for evaluating sample paths in non-binary cost map representations of off-road terrain. In addition to traditional considerations of distance and path curvature, the proposed cost function accounts for traversability.

6. The experience from the tests done in colour-based clustering of terrain types, and from the tests of the complete system that demonstrated the feasibility of combining exteroceptive and proprioceptive sensors for autonomous self-supervised terrain-adaptive off-road navigation will provide useful insights to the robotics community [87].

6.2 Future Work

While the current results demonstrate the feasibility of the proposed approach, more extensive performance analysis in varying environmental conditions is required; a good test environment has been identified (Appendix D). In addition, it is important to evaluate CNNs trained on labelled images generated from tests using the proposed algorithm.

Also, to improve the precision of terrain characterization, a different proprioceptive sensing approach can be employed, e.g., using a dedicated Mars rover-like sampling appendage. The current approach of using the robot’s widely-spaced wheels for
6.2. FUTURE WORK

terrain sampling increases the likelihood of interference with other terrain types. A
dedicated appendage will have a much smaller footprint, can perform slope-invariant
terrain characterization, and can conveniently sample an identified terrain patch mul-
tiple times, and average the results.

In addition, it is important to include other traversability characteristics, e.g., slip
and compliance, in the characterization process. Although roughness is an important
characteristic, its vibration effects can be reasonably managed at low speeds using
mechanical devices; the effects of characteristics like slip and compliance on navigation
and the safety of the robot are much more critical.

Other possible considerations for future work to improve the overall performance
of the proposed algorithm can cover:

1. Including texture features for improved exteroceptive discrimination between
low chromaticity or monochrome terrain types, with inspiration from [16], [31]
and [58].

2. Implementing speed adaptation in reasonably unstructured environments by
using a trajectory tracking approach that allows the inclusion of a speed profile.

3. Expanding the approach to air–ground vehicle teaming as was done in [37], [86]
and [130]. The benefits will include the generation of local maps or images that
cover wider areas, and the generation of obstacle-based binary global maps.

4. Including terrain undulations and negative obstacles in the analysis by using
3D LiDAR and fitting a ground plane as was done in [125] and [131].
References


REFERENCES


Appendix A

System Flow Chart and Pseudocodes

A.1 System Flow Chart

A flowchart of the overall proposed method is shown in Figure A.1. The main subroutines are as shown in Figure 2.9, comprising the vision pipeline, path planning, self supervised feature registration and path following using the path following controller.

The flowchart does not consider obstacles, thus the interim goal positions correspond to the intersection between the image boundaries and the heading from the current robot position to the specified final goal position as shown in Figure 5.1. For the case where the traversal characteristics of all the terrain types in an image are known, the image is labelled with the known costs and stored with its source image for training a CNN; in other cases, the image can be labelled at the interim goal position after feature registration, and stored.

Pseudocodes for the path planning and path following subroutines are shown in Section A.2. The flowchart and pseudocodes are sample representations of the ROS architecture of the overall proposed algorithm.
A.1. SYSTEM FLOW CHART

Figure A.1: A representation of the proposed method.

Start

Rotate robot to specified goal heading

Acquire image from on-board camera

Vision pipeline:
Transform ROI and identify unique terrain classes by colour

Feature list empty?

No

Path planner:
Plan path to interim goal through representative per-class samples

Path following and feature registration:
Acquire traversal costs from samples while following path to interim goal

Store per-class colour features and traversal costs in list

Compare identified classes to list by colour

Unknown classes detected?

Yes

Number known classes by number in list

No

Number classes in image by most similar class in list

Save labelled image

Path planner:
Plan path to interim goal using known traversal costs

Path following:
Follow path to interim goal

Path following and feature registration:
Acquire traversal costs from samples while following path to interim goal

Update list with new colour features and traversal costs

Interim goal = specified goal?

Yes

Stop

No
A.2 Pseudocodes

Algorithm 1 Path planner: from robot position across blind spot to image start position through terrain samples to interim end position.

\[
p \leftarrow \text{start–end point pairs} \quad \triangleright \text{E.g., robot position to image start position etc}
\]

\[
\text{paths} \leftarrow \text{empty list} \quad \triangleright \text{For path points}
\]

\[
\text{for each start–end point pair in } p \text{ do}
\]

\[
\text{pathPoints} \leftarrow \text{SBMP}(\text{start, end})
\]

\[
\text{path} \leftarrow \text{pathPoints}
\]

\[
\text{Insert } \text{path} \text{ in } \text{paths}
\]

end for

Return paths

Algorithm 2 Path following and feature registration: using paths returned by Algorithm 1.

\[
\text{paths} \leftarrow \text{Paths list from path planner}
\]

\[
\text{traversalCosts} \leftarrow \text{empty list} \quad \triangleright \text{For per-class traversal costs}
\]

\[
\text{for each path in } \text{paths} \text{ do}
\]

\[
\text{Rotate robot to path start heading}
\]

\[
\text{while following path using controller do}
\]

\[
\text{if path is on sample terrain then}
\]

\[
\text{a} \leftarrow \text{IMU planar linear accelerations}
\]

end if

end while

\[
q : a \rightarrow \text{traversalCost}
\]

\[
\text{Insert } \text{traversalCost} \text{ in traversalCosts}
\]

end for

Return traversalCosts
Appendix B

Supplementary Algorithms

B.1 Nonlinear Least Squares for Homography

From (3.11), the transfer distance between the calibration pattern Tr and the corresponding measured ROI coordinates in the source image can be expressed as the Euclidean distance

\[ \sum_{i=1}^{n} d(p_i, HP_i)^2, \]  

(B.1)

where \( p_i = (u_i, v_i) \) and \( P_i = (U_i, V_i) \) are \( n \) pairs of corresponding points in \( Tr \) and the ROI from the source image respectively (Figure 3.1). The goal is to find \( \hat{H} \) that minimizes the sum of squares of residuals \( S(H) \), as in (3.12),

\[ S(H) = \sum_{i=1}^{n} [p_i - f(P_i, H)]^2. \]  

(B.2)

Starting with an initial guess for \( H \), \( \hat{H} \) is arrived at through a process of iteration with \( H \) updated by a small change \( \Delta \) at each step until convergence according to set criteria.
At each step of the iteration, the function \( f(P_i, H + \Delta) \) is approximated by linearization as

\[
f(P_i, H + \Delta) \approx f(P_i, H) + J_i \Delta, \quad (B.3)
\]

where \( J_i \) is the Jacobian

\[
J_i = \frac{\partial f(P_i, H)}{\partial H}. \quad (B.4)
\]

Thus,

\[
S(H + \Delta) \approx \sum_{i=1}^{n} [p_i - f(P_i, H) - J_i \Delta]^2, \quad (B.5)
\]

which can be expressed in vector terms and expanded to give

\[
S(H + \Delta) \approx \|p - f(H) - J \Delta\|^2
\]

\[
= [p - f(H)]^T [p - f(H)] - 2(p - f(H))^T J \Delta + \Delta^T J^T J \Delta. \quad (B.7)
\]

Setting the derivative of B.7 with respect to \( \Delta \) to zero gives

\[
(J^T J) \Delta = J^T [p - f(H)]. \quad (B.8)
\]

Eq. B.8 is equivalent to the Gauss-Newton method. For a more robust solution process, the Levenberg-Marquardt algorithm (LMA) introduces a damping term to give the augmented normal equation

\[
(J^T J + \lambda I) \Delta = J^T [p - f(H)]. \quad (B.9)
\]

Note that by adaptively changing \( \lambda \), the algorithm operates in between Gauss-Newton
and the gradient descent approach. The LMA approach is dependent on the initial guess of $H$; a good guess is the solution derived from the SVD process described in Section 3.1.1.

Since $P_i$ is a 2D vector, the minimization is done over $n$ 2D points and the parameters in $H$.

B.2 The Feedback Linearized Controller

The feedback linearized controller uses the same formulation as [138] but with a unicycle kinematic model. Consider the unicycle in Figure B.1 inclined at an angle $\theta$ with the reference $X$-axis with point of contact on the ground at $C$. Its linear velocity
B.2. THE FEEDBACK LINEARIZED CONTROLLER

is \( v_c \) and angular velocity is \( \omega = \dot{\theta} \). Thus,

\[
\begin{align*}
\dot{x}_c &= v_c \cos \theta, \\
\dot{y}_c &= v_c \sin \theta, \\
\dot{\theta} &= \omega.
\end{align*}
\]  

(B.10)  

(B.11)  

(B.12)

And the kinematic model of the unicycle can be written as

\[
\dot{q} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} v_c + \begin{bmatrix} 0 \\ v \\ 0 \end{bmatrix} \omega,
\]  

(B.13)

which is an affine system of general form

\[
\dot{q} = g_{o}(q) + \sum_{i=1}^{m} g_{i}(q)u_{i}
\]  

(B.14)

\[
= g_{o}(q) + G(q)u,
\]  

(B.15)

where \( q = [q_1, q_2, \ldots, q_n] \) and \( u = [u_1, u_2, \ldots, u_m] \) for \( n \) states and \( m \) action variables or control inputs; and \( G(q) = [g_1(q) \ g_2(q) \ \ldots \ g_m(q)] \).

Since \( m \) is two for the two control variables in this system, for a driftless system with \( g_{o}(q) = 0 \), it can be written that

\[
\dot{q} = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},
\]  

(B.16)
with \( v = v_c \), as in (B.13).

The goal of the feedback linearization process is to transform this driftless affine model to the linear form \( \dot{z} = Az + B\eta \). This transformation is achieved through a state transformation \( z = z(q) \) and an input transformation \( u = u(q, \eta) \).

Combining Figure 5.1 (lower left) and Figure B.1, assuming that the instantaneous tangent to the closest waypoint is always aligned with the reference \( X \)-axis, then the path following errors can be defined as a lateral error \( \varepsilon_L \) and an heading error \( \varepsilon_H \), such that

\[
\varepsilon_L = y, \quad \varepsilon_H = \theta, \tag{B.17}
\]

with error dynamics

\[
\dot{\varepsilon}_L = \dot{y} = v \sin \varepsilon_H, \quad \dot{\varepsilon}_H = \dot{\theta} = \omega. \tag{B.19}
\]

Now, the state transformation \( z = [z_1, z_2]^T \) is applied such that

\[
z_1 = \varepsilon_L = y, \quad z_2 = \dot{\varepsilon}_L = \dot{y}. \tag{B.21}
\]

And the state-space representation of the dynamics of the system in the new states
is

\[
\dot{z} = \begin{bmatrix}
\dot{z}_1 \\
\dot{z}_2
\end{bmatrix} = \begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
z_1 \\
z_2
\end{bmatrix} + \begin{bmatrix}
0 \\
1
\end{bmatrix} \eta.
\] (B.23)

The goal of representing the system in the linear form \( \dot{z} = Az + B\eta \) has thus been achieved. Note that for a fixed speed,

\[
\eta = \ddot{\varepsilon}_L = v\omega \cos \varepsilon_H,
\] (B.24)

which gives

\[
\omega = \frac{\eta}{v \cos \varepsilon_H}.
\] (B.25)

Eq. B.25 is clearly only feasible when the denominator is non-zero, i.e. when \( v \neq 0 \) and \( \varepsilon_H \in (-\pi/2, \pi/2) \).

At the lower level, a controller provides the appropriate electrical signals to the wheel motors to achieve the linear and rotational speeds.
Appendix C

Hardware Specifications

The listed specifications cover the test robot and on-board sensors. The Vicon motion camera specifications are also listed. Finally, specifications for the deep learning test rig used in Chapter 2 are listed.

Table C.1: Key specifications of Husky.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions (L × W × H)</td>
<td>0.990 × 0.670 × 0.390 m</td>
</tr>
<tr>
<td>Track</td>
<td>0.555 m</td>
</tr>
<tr>
<td>Weight</td>
<td>50 kg</td>
</tr>
<tr>
<td>Speed (max.)</td>
<td>1.0 m/s</td>
</tr>
<tr>
<td>Ground Clearance</td>
<td>0.130 m</td>
</tr>
<tr>
<td>Wheel Radius</td>
<td>0.165 m</td>
</tr>
<tr>
<td>Battery</td>
<td>24V 20Ah</td>
</tr>
<tr>
<td>Sensor</td>
<td>Specifications</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td><strong>Husky on-board Encoders</strong></td>
<td>Resolution = 78,000 ticks/m&lt;br&gt;( f_s = 10Hz )</td>
</tr>
<tr>
<td><strong>LORD Microstrain 3DM-GX3-25 IMU</strong></td>
<td>Gyroscope range = ±300°/s&lt;br&gt;( \sigma_{\text{gyro}}^2 = 0.012 )&lt;br&gt;( f_s = 100Hz )</td>
</tr>
<tr>
<td><strong>LORD Microstrain 3DM-GX5-25 IMU</strong></td>
<td>Gyroscope range = ±300°/s&lt;br&gt;( \sigma_{\text{gyro}}^2 = \sigma_{\text{magnetometer}}^2 = 0.010 )&lt;br&gt;Magnetometer heading accuracy = ±1.5°&lt;br&gt;( f_s = 45Hz )</td>
</tr>
<tr>
<td><strong>Duro RTK GPS</strong></td>
<td>RTK Accuracy (Horizontal) = 0.010m ± 1ppm&lt;br&gt;( f_s = 5Hz )</td>
</tr>
<tr>
<td><strong>Camera</strong></td>
<td>8 MP, f/2.2</td>
</tr>
<tr>
<td><strong>RTK Base Station set up.</strong></td>
<td>(<strong>GNSS RTK receiver</strong>)&lt;br&gt;Dimensions = 0.147m (L) × 0.113m (W) × 0.045 m (H)&lt;br&gt;Constellation: GPS, GLONASS, Galileo, BeiDou&lt;br&gt;Accuracy: L1 = 1.5 m; L1/L2 = 1.2 m&lt;br&gt;( f_s = 100Hz )</td>
</tr>
<tr>
<td><strong>Novatel FlexPak6 OEM628-1.01</strong></td>
<td><strong>GPS antenna</strong>)&lt;br&gt;Dimensions = 0.185 m diameter × 0.069 m&lt;br&gt;Weight = 0.5 kg&lt;br&gt;Constellation: GPS, GLONASS</td>
</tr>
<tr>
<td><strong>Microhard Systems VIP2-2400</strong></td>
<td><strong>Broadband ethernet bridge.</strong>)&lt;br&gt;1W 2.4 GHz</td>
</tr>
<tr>
<td><strong>L-com HGV-2406U</strong></td>
<td>(<strong>Antenna for VIP2400.</strong>)&lt;br&gt;2.4 GHz</td>
</tr>
</tbody>
</table>
Table C.3: Vicon motion camera system specifications.

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera specifications:</td>
</tr>
<tr>
<td>Resolution = 2.2 (2048 × 1088) MP</td>
</tr>
<tr>
<td>Max Frame Rate (Hz) = 330 @ 2.2 MP</td>
</tr>
<tr>
<td>Tracker: 12 × Vero v2.2 cameras</td>
</tr>
<tr>
<td>Minimum Standard FOV (H × V)° = 44.1 × 23.6 (tele)</td>
</tr>
<tr>
<td>Camera Latency = 3.6 ms</td>
</tr>
<tr>
<td>Dimensions (H × W × D) = 0.083 × 0.080 × 0.135 m</td>
</tr>
<tr>
<td>Weight = 0.57 kg</td>
</tr>
</tbody>
</table>
Table C.4: Deep learning test rig.

<table>
<thead>
<tr>
<th>PC</th>
<th>Dell Precision 5820 Tower X-Series:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Core i9 (3.00GHz), 32GB RAM, 512GB SSD</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA Quadro RTX 6000 GPU:</td>
</tr>
<tr>
<td></td>
<td>24GB GDDR6</td>
</tr>
<tr>
<td></td>
<td>72 NVIDIA RT Cores</td>
</tr>
<tr>
<td></td>
<td>4,608 CUDA Parallel-Processing Cores</td>
</tr>
<tr>
<td></td>
<td>576 NVIDIA Tensor Cores</td>
</tr>
<tr>
<td></td>
<td>FP32 Performance: 16.3 TFLOPS</td>
</tr>
<tr>
<td></td>
<td>Max Power Consumption: 295 W</td>
</tr>
</tbody>
</table>
Appendix D

Applicable Environments

Figure D.1 to Figure D.3 are from the Queen’s mining test site in Ontario, Canada; and Figure D.4 to Figure D.6 are from the Mars Curiosity Image Gallery [141].
Figure D.1: Sample Terrain 1 from Queen’s mining test site.
Figure D.2: Sample Terrain 2 from Queen’s mining test site.
Figure D.3: Sample Terrain 3 from Queen’s mining test site.
Figure D.4: Sample Terrain 1 from Mars Curiosity Image Gallery.

Figure D.5: Sample Terrain 2 from Mars Curiosity Image Gallery.
Figure D.6: Sample Terrain 3 from Mars Curiosity Image Gallery.