UNPAVED ROAD DETECTION USING OPTIMIZED LOG GABOR FILTER BANKS

A Dissertation Presented to the Faculty of the

Graduate School University of Missouri

In partial fulfillment of the

requirements for the Degree

Doctor of Philosophy.

by

Pooparat Plodpradista

Dr. James M. Keller, Advisor

May 2021

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled

UNPAVED ROAD DETECTION USING OPTIMIZED

LOG GABOR FILTER BANKS

presented by Pooparat Plodpradista,

a candidate for the degree of doctor of philosophy,

and hereby certify that, in their opinion, it is worthy of acceptance.

____

Professor James M. Keller

____

Professor Dominic Ho

____

Professor Mihail Popescu

____

Professor Marjorie Skubic

Thank you to my mother who proved that there is no ceiling in life, to my father who showed me that any problems can be solved intellectually, to my older brother who paved a path for me, and to my loving sisters who always care for me. Lastly, this is for my wife Cha who think that Ph.D., in her own words, is ‘cool’ and gave her all to support me to pursue my degree.

ACKNOWLEDGMENTS

I cannot express enough thanks to my advisor Dr. James Keller for giving me an opportunity to join his research group. Throughout the years, I have received many advices, which helped me broaden my knowledge and improve my academic skills. Dr. Keller always shows passion in academic research and I have learned so much from talking to him. I am incredibly grateful for his understanding and support during the time of my life that seem impossible to move forward. The completion of this project would not be possible without his guidance and I am very thanksful to have him as my advisor.

I would also like to thank Dr. Dominic Ho for many of his academic advice as well as his assistance during the most difficult time in my life. Dr. Ho is an outstanding profressor and working with him has made me a better researcher. In addition, I want to thank other members of my committee, Dr. Mihail Popescu and Dr. Marjorie Skubic, for their time and support during my study at Mizzou. Lastly, I want to give thanks to my lab mates, Kevin, Brian, Eric and Drew, that I have worked with and known for many years.

Table of Contents

LIST OF ILLUSTRATIONS

LIST OF TABLES

UNPAVED ROAD DETECTION USING

OPTIMIZED LOG GABOR FILTER BANKS

Pooparat Plodpradista

Dr. James M. Keller, Disseration Supervisor

ABSTRACT

The revised unpaved road detection system (RURD) is a novel method for detecting unpaved roads in an arid environment from color imagery collected by a forward-looking camera mounted on a moving platform. The objective is to develop and validate a novel system with the ability to detect an unpaved road at a look-ahead distance up to 40 meters that does not utilize an expensive sensor, i.e., LIDAR but instead a low-cost color camera sensor. The RURD system is composed of two stages, the road region estimation (RRE) and the road model formation (RMF). The RRE stage classifies the image patches selected at 20-meter distance from the camera and labels them to either road or non-road. The classification result is used as a high confidence road area in the image, which is used in the RMF stage. The RMF stage uses log Gabor filter bank to extract road pixels that connect to the high confidence road region and generates a 3rd degree polynomial curve to represent the road model in a given image. The road model allows the system to extend the detection range from 20 meters to farther look-ahead distance.

The RURD system is evaluated with two-years worth of data collection that measures both spatial and temporal precisions. The system is also benchmarked against an algorithm from Rasmussen entitled “Grouping Dominant Orientations for Ill-Structured Roads Following”, which shown an average increase detection accuracy over 30%.

Chapter 1: Introduction

1.1.Motivation

Because of their vast number of applications, autonomous driving vehicles are becoming the future of road transportation. An autonomous driving vehicle (ADV) can fully assist the disabled or elderly who experience some driving difficulties by automating certain functions, such as parallel parking assistance and an early collision warning. In commercial transportation, an autonomous system will reduce operating costs by minimizing human errors and improving driver efficiency. Possibilities for new applications continue to grow, but perhaps the most consequential one pertains to military applications, because an ADV can help save lives by removing the need to put personnel behind the wheel in high-risk areas, i.e., war zones or areas where rioting or looting has occurred.

One of the most fundamental requirements for an ADV is the ability to detect roads laying in front of the vehicle. Therefore, a road detection system is the first and foremost component that an ADV needs to have. The sole function of a road detection system is to provide an ADV with a drivable path that is visible to the vehicle. Together with a navigation system, such as a global positioning system, the road detection system can work as the main guidance system for an ADV.

1.2. Adversity

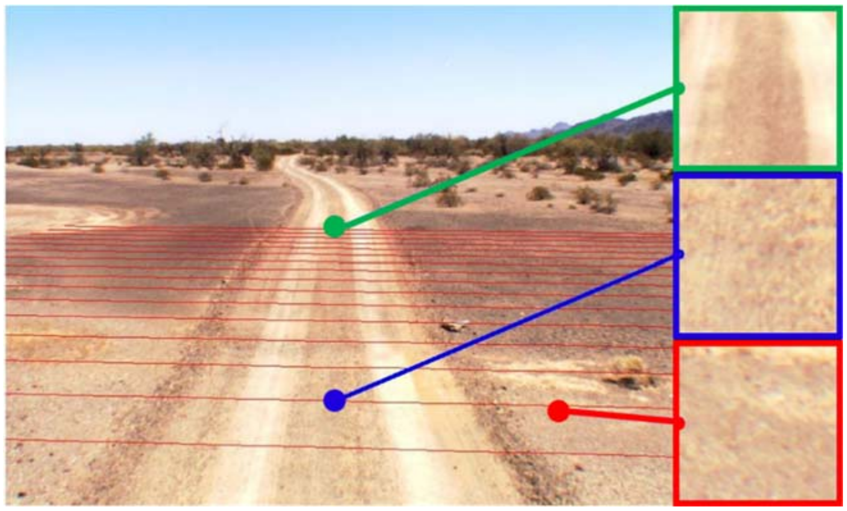


Figure 1-1. An unpaved road at a US. Army test site shows the road texture at close distance shown in the far right inset outlined in blue. This sample is much like the off-road texture outlined in red. The road texture at the farther distance is outlined in green.

Two common approaches are used for a road detection system:1) color cues in an image[1-3] and 2) illumination differences between the road and its background to detect lane markers.[4-6] However, in real-world applications, detecting an unpaved road is not an option for such road detection systems, especially when the road surface material matches that on the sides of the road. Even when a dirt road has shoulders with grass or a different colored gravel, fog, rain, and shadows can quickly make the road undetectable. These problems are especially common in military applications, where the ADV is likely to be operated in an underdeveloped area, as shown in Figure 1.1. The extent of these adversities was confirmed during the 2004 DARPA Grand Challenge, a competitive event to develop a driverless vehicle that could navigate a 300-mile course in the Mojave Desert. Out of the 106 participating teams, the farthest traveling vehicle was only able to travel slightly less than eight miles; thus, no winner was declared. The second attempt was held again in 2005. That year, the goals of the DARPA Grand Challenge were finally reached. However, only four out of the 195 participating teams were able to traverse the required 300 miles in less than10 hours.

1.3.Contributions

The contributions of this dissertation are the development and validation of a novel unpaved road detection system that utilizes a low-cost sensor with the ability to detect a road at a look-ahead distance of 20 meters or more. Even though the problem of detecting unpaved roads in arid environments was solved in the 2005 DARPA Grand Challenge, issues still had to be confronted as to which solution was worth pursuing further. Among the multitude of sensors used by the winning team as evidenced in Figure 1-2(a), the key component was a light detection and ranging (LIDAR) sensor[7] that has two major disadvantages: high cost and short distance detection (as shown in Figure 1-2(b). In contrast, a color camera is a low-cost device, which in theory, can ‘sense’ the road as far as its vanishing point. Therefore, an ADV should be able to eliminate the aforementioned disadvantages by using a color camera as the sole detection sensor for this novel system.

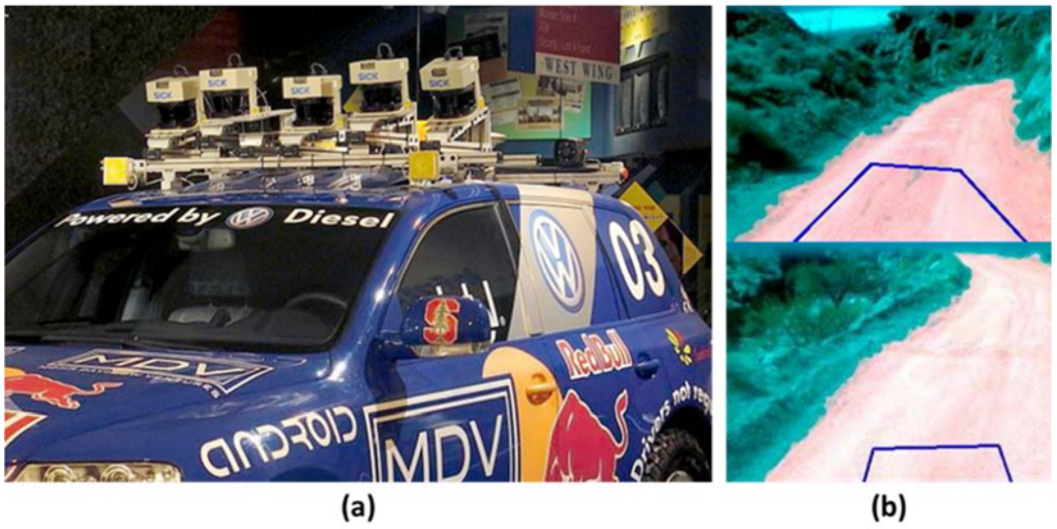


Figure 1-2. (a) An exhibit of sensors used by the winner of the 2005 DARPA Grand Challenge vehicle which has a total of five LIDAR modules mounted on its top and (b) examples of the LIDAR detection range.[8] The blue polygon represents a region of the road detected by the LIDAR. The detection range is relatively low when compared to the actual road region highlighted in pink.

The proposed unpaved road detection system presented in this dissertation is composed of two stages. The first stage is used to estimate the road region at a close distance by feature extraction and classification. This process uses a machine learning approach to create an algorithm that approximates the road region in a given color image frame. By using the labeled data of unpaved roads to train a road detection classifier model, the algorithm can correctly differentiate the road from the non-road image samples. These image samples are small images that have been cropped from each frame and captured by the RGB color camera. The cropping locations are evenly selected across the frame at a fixed distance ahead of the camera lens as demonstrated in Figure 1.3. However, the limitation of this approach is the variation in the image samples. This variation is not about the texture differences between samples but rather the environmental changes between each driving session. Depending on the time of day at which the system is used, changes in the environment, including weather conditions from one time to another, can differ greatly in terms of lighting conditions, shadow shapes, and road color. Consequently, the distribution of the training dataset can differ substantially from the testing dataset; moreover, traditional machine learning approaches cannot adequately handle this type of data variation.

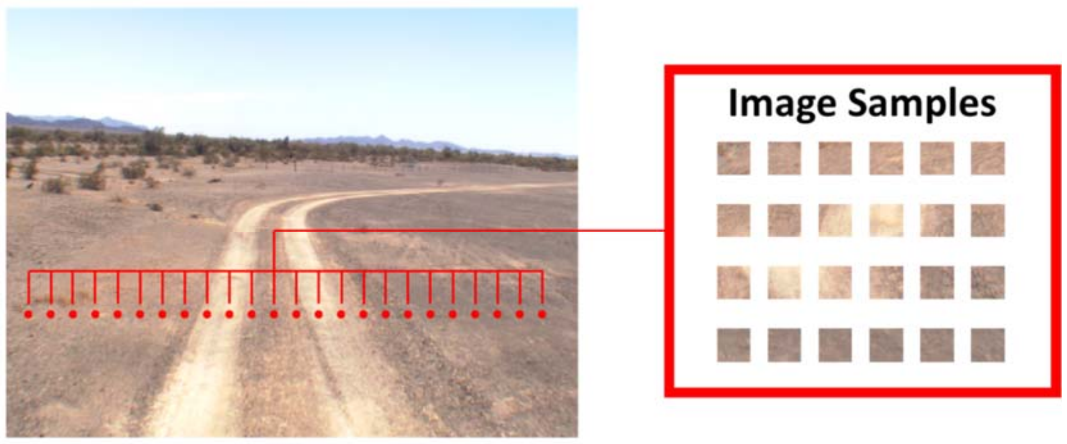


Figure 1-3. Example locations where the image samples from left to right are cropped from each frame as denoted by the red dots.

A well-known machine learning technique called transfer-learning[9] can handle the data variation issue. This method uses a small number of data samples from the testing dataset to adapt the developed model (a classifier trained with the training dataset from a different environment) so that the model is able to identify the differences between training and testing environments and improve the classification accuracy of the model. However, the benefit of transfer-learning is greatly limited by the method used to extract the information from the image sample (feature extraction) as well as the types of classifiers. To further understand this limitation, two experiments were conducted: The first was a performance analysis of multiple sets of features and the second experiment recorded variations in the image samples to determine classifier efficiency. The results of these experiments determined the parameters of the algorithm that was used in the first stage of the system in its proposal phase. Although the outcome of the first two experiments produced an algorithm that could accurately detect the unpaved road, the detection range was restricted to a low forward-reaching visual (look-ahead) distance because the image samples at the farther look-ahead distance had a significantly lower resolution and did not contain sufficient information. Thus, a second stage was developed to extend the detection range, which was achievable through an image processing approach. By using an optimization approach to identify the textures and patterns of an approximated road region in a given frame (the output of the first stage), the newly developed system was able to generate an optimal two-dimensional frequency filter bank (the log Gabor filter bank) that can either emphasize or suppress the road textures in a given frame. The optimized filter bank allows the system to generate a road model and extend the detection range to a practical look-ahead distance. The second stage of the unpaved road detection system included two experiments that determined 1) the filter bank optimization for road textures and non-road textures, 2) the impact of the filter bank size on the computation time 3) the trade-off between the extension of the look-ahead distance and the accuracy of the road model and 4) the improvement of an extended tailor filter bank over an optimized filter bank. After the experiments were completed, the final design of the unpaved road detection algorithm was determined and a final system evaluation was conducted.

The final phase of the dissertation project included four evaluations that assessed both the detection accuracy and real-time computation of the new system. The first two evaluations determined the accuracy of both the road region estimation and road model formation. The third evaluation measured the improvement of a consolidated system between road region estimation and road model formation, known as the revised unpaved road detection (RURD) system. Finally, the fourth evaluation focused on the comparison between the RURD system and a benchmark algorithm. The data used during the final system evaluation as well as during the experiments were collections of color images used in landmine detection research. These data were collected from an arid U.S. Army test site obtained from a fixed gimbal mounted on a landmine detection vehicle. However, the data collection contains multiple instances where the roads are not present in the camera frames. These occurred when the vehicle approached turns while the camera was aimed at a fixed forward position and the roads were out of the field of view. Since the evaluation process determines the success of the proposed system, it was essential that the verification data contained the road truth values needed to measure the system output in every single image frame. Therefore, the lane boundary truth of the verification data was prepared to ensure the presentation of a measurable truth value in every scene, even if the road was out of the field of view. Further details are discussed in Chapter 4 (Section 4.1).

1.4. List of Publications

This dissertation was built on works from the following publications:

1. **P. Plodpradista**, J. M. Keller, and M. Popescu "Road recognition in poor quality environments for forward looking buried object detection", Proc. SPIE 9072, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XIX, 90721A (29 May 2014); https://doi.org/10.1117/12.2049961

2. **P. Plodpradista**, J. M. Keller, and M. Popescu "An application of log-Gabor filter on road detection in arid environments for forward looking buried object detection", Proc. SPIE 9454, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XX, 94540R (14 May 2015); https://doi.org/10.1117/12.2177943
3. **P. Plodpradista**, J. M. Keller, and M. Popescu "Road detection in arid environments using uniformly distributed random based features", Proc. SPIE 9823, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXI, 982315 (3 May 2016); https://doi.org/10.1117/12.2224080
4. **P. Plodpradista**, J. M. Keller, D. K. C. Ho, and M. Popescu "Tuning log Gabor filter bank using genetic algorithm based optimization", Proc. SPIE 10182, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXII, 101821C (3 May 2017); https://doi.org/10.1117/12.2262615
5. **P. Plodpradista**, D. K. C. Ho, J. M. Keller, M. Popescu, and A. Buck "Analyzing three dimensional radar voxel data using the discrete Fourier transform for SAEH detection", Proc. SPIE 10628, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIII, 1062814 (30 April 2018); https://doi.org/10.1117/12.2304380
6. Dominic K. C. Ho and **Pooparat Plodpradista** "On the use of multiresolution analysis for subsurface object detection using deep ground penetrating radar", Proc. SPIE 11012, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIV, 1101209 (10 May 2019); https://doi.org/10.1117/12.2519389

Lastly, a potential paper will be writen based on the works in this dissertation, focusing on using an optimization apporach to automatically design a log Gabor filter bank that generates an effective road model. The paper will also introduce a casual system evluation method that penalizes an algorithm based on its computation time.

Chapter 2: Methodology

This chapter describes the methods used in the proposed system, which included: feature extraction, a classification algorithm, image processing, and optimization.

2.1. Feature Extraction

Feature extraction is an initial process of machine learning where we extract useful information from the data. In this case, we performed a texture analysis on the input images and obtained a descriptive vector that the proposed system could use to differentiate a road texture from its background. The texture analysis techniques used in this proposed system are local binary patterns, histograms of oriented gradients, gradient location and orientation histogram, and speeded up robust features. Since all feature sets were designed to work only with a grayscale image, the color camera sensor requires the input imagery be converted to a grayscale image before applying the feature extraction process.

2.1.1. Local Binary Patterns

A local binary pattern (LBP) technique was originally introduced as a texture spectrum model in 1990.[10] Because it is robust and computationally inexpensive, LBPs have been used in many applications. In the field of computer vision, LBP is a texture descriptor that describes the contrast of local variation between each pixel and its neighborhood. A basic version of the LBP descriptor is defined as when given neighborhood pixels, ***N***, center around pixel $\boldsymbol{P}$ in an image cell with a radius, $\boldsymbol{R}$, and a binary representation of a local pattern is computed using the following equation.

$$LBP(\boldsymbol{P}) = \sum_{i=1}^{N} B(I_i - I_{\boldsymbol{P}}) \times 2^i \qquad (2\text{-}1)$$

where:

$I_i = the\ intensity\ value\ of\ i^{th}\ neigborhood\ pixel.$

$I_P = the\ intensity\ value\ of\ center\ pixel.$

$$B(x) = \begin{cases} 1\ if\ x \geq 0 \\ 0\ if\ x < 0 \end{cases}$$

A key advantage of LBP is the illumination invariant,[11] meaning that LBP will give the same description of a texture regardless of the change in overall image intensities as illustrated in Figure 2-1.

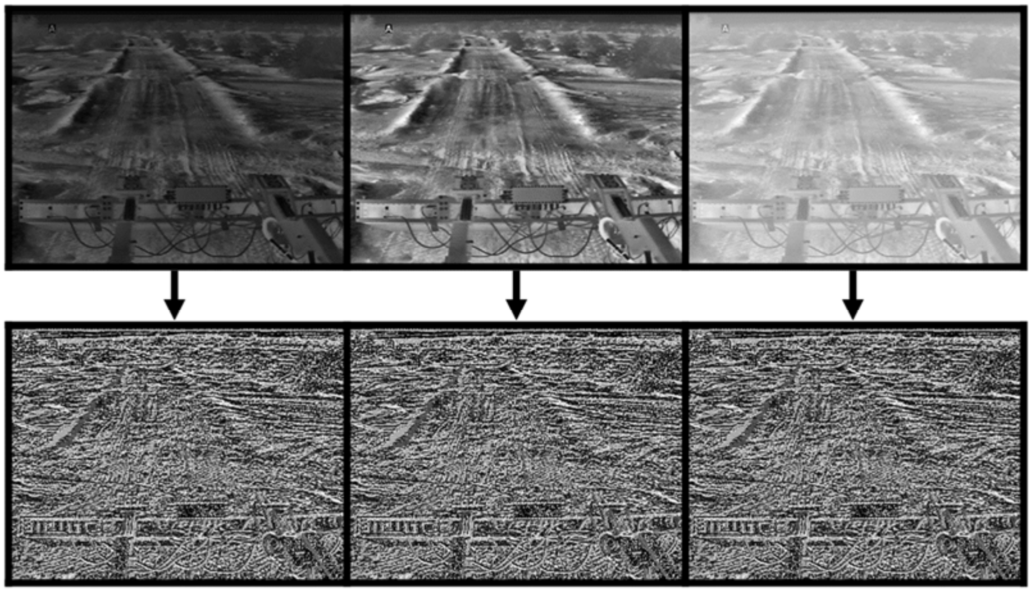


Figure 2-1. An illustration of the illumination invariant properties of LBP. The top row shows input images with varying illumination conditions from dark to bright. The bottom row shows the corresponding LBP images.

In this study, an extension version called the uniform rotation-invariant LBP ($\mathrm{LBP}_{\mathrm{URI}}$) will be used. When compared with the basic LBP, the $\mathrm{LBP}_{\mathrm{URI}}$ has an additional property of rotational invariance without any loss of discriminative power. Traditional rotation–invariant LBP can be achieved by using the Equations (2-2) and (2-3) as follows:

$$\mathrm{LBP}_{\mathrm{ROI}} = \mathrm{argmin}_{\mathrm{i}}\ \mathrm{BWS}(LBP_{binary}, \mathrm{i}) \tag{2-2}$$

$$LBP_{binary} = dec2bin(LBP) \tag{2-3}$$

where

$$BWS(B, i) = the\ bit\ wise\ shift\ on\ the\ binary\ number\ B, i\ times$$

$$dec2bin(x) = converting\ decimal\ to\ binary$$

This rotation invariant method outputs only a minimum LBP value for a texture. Thus, the texture information is compressed and distorted. To counter this problem, the uniform property of a texture is incorporated when computing LBP. [12, 13] The uniform property of LBP is defined such that a texture is uniform if there are at most two transitions between 0 and 1 or between 1 and 0 bits in a binary string representing the texture. For example, '00110000' represents a uniform texture since it has only two-bit transitions, at the third and fifth bit. Conversely, '0101000' is not a uniform texture since it has four-bit transitions. By incorporating the property of texture uniformity, the equation used to calculate LBP must be updated as shown in Equation (2-4) as follows.

$$LBP_{UROI}(\boldsymbol{P}) = \begin{cases} \sum_{i=1}^{N} B(I_i - I_{\boldsymbol{P}}) & if\ texture\ is\ uniform \\ P + 1 & if\ texture\ is\ not\ uniform \end{cases} \tag{2-4}$$

Once LBP_{UROI} values are computed for all the pixels in an image, a histogram size of $\boldsymbol{N} + 2$ bins will be built and used as an LBP descriptor. For a neighborhood size, $\boldsymbol{N} = 8$, the histogram size is 10 bins. The first 9 bins are the accumulation of the 9 uniform rotation-invariant textures (as shown in Figure 2-2) within an image while the last bin is the accumulation of non-uniform textures.

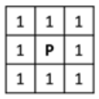
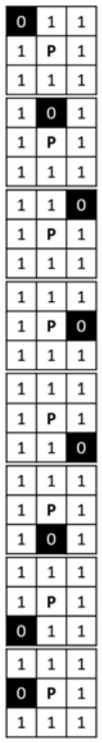
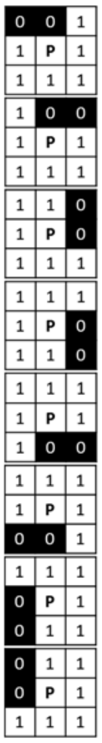
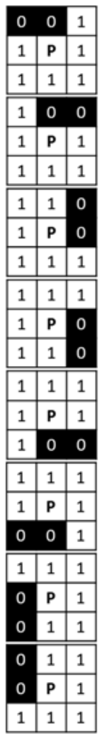
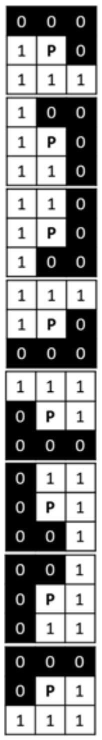
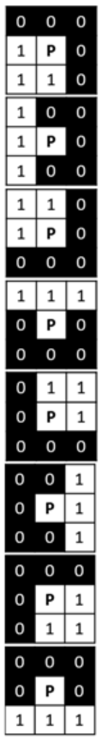
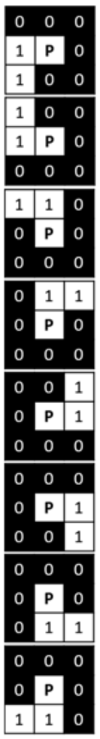
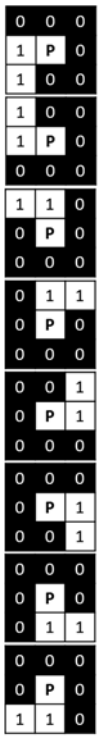
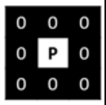
Bin 1	**Bin 2**	**Bin 3**	**Bin 4**	**Bin 5**	**Bin 6**	**Bin 7**	**Bin 8**	**Bin 9**	**Bin 10**
LBP_{UROI} **0**	LBP_{UROI} **1**	LBP_{UROI} **2**	LBP_{UROI} **3**	LBP_{UROI} **4**	LBP_{UROI} **5**	LBP_{UROI} **6**	LBP_{UROI} **7**	LBP_{UROI} **8**	LBP_{UROI} **9**
									**Any Non-Uniform Textures**

Figure 2-2. An illustration of the textures that can be accumulated in each bin of an LBP descriptor. P is the center pixel and the neighborhood pixel is 1 when its intensity is greater or equal to I_P; otherwise, the neighborhood pixel is 0.

2.1.2. Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) is a popular descriptor for object detection and recognition application. HOG was modernized and gained its reputation as one of the best descriptors based on work done by Dalal and Triggs.[14] HOG is derived from an edge detector technique, which finds the gradients of pixel intensities in both horizontal (∇I_x) and vertical (∇I_y) directions using the following equations:

$$\nabla I_x = I_{x+1} + I_{x-1} \tag{2-5}$$

$$\nabla I_y = I_{y+1} + I_{y-1} \tag{2-6}$$

In practice, to compute the horizontal and vertical gradients, an image will be convolved with a $[-1\ 0\ 1]$ gradient filter and a $[-1\ 0\ 1]^T$ gradient filter, respectively. Therefore, each pixel will have two values that represent a change in the intensities in both horizontal and vertical directions. From here, an absolute change in the intensities, the magnitude ($|\nabla I|$), and its orientation (θ) can be computed using the following equations:

$$|\nabla I| = \sqrt{\nabla {I_x}^2 + \nabla {I_y}^2} \tag{2-7}$$

$$\theta = \tan^{-1} \frac{\nabla I_y}{\nabla I_x} \tag{2-8}$$

Once the magnitude and orientation of the gradients on each pixel are obtained, the histograms of these values are built and used as a part of the HOG descriptor. A common practice used to generate the HOG descriptor is to separate the input image into cells and blocks, where a cell represents a group of pixels while a block is composed of multiple cells. The multiple-cell blocks can be overlapped with other blocks. For each cell, a histogram where the orientations of gradients are categorized into bins is generated by using the following equation:

$$Bin_i = \sum_{k=1}^{n_i} |\nabla I|_k \tag{2-9}$$

where n_i is a set of pixels that fall within the orientation of the *i*th bin.

Once the histograms of all the cells in a block have been extracted, they are concatenated into a single block histogram. To obtain local contrast information, the block histogram is then normalized by using the following equation:

$$Histogram_{Norm} = \frac{Histogram}{\|Histogram\|_2} \tag{2-10}$$

The final HOG descriptor that represents an image is the concatenation of all normalized block histograms. The number of histograms in a HOG descriptor depends on three variables: (1) the number of blocks in an image (N_{block}), (2) the number of cells in a block (N_{cell}), and (3) the number of histogram bins (N_{bin}). Then, the dimensionality of the HOG descriptor can be calculated as:

$$HOG\ Dimension = N_{block} \times N_{cell} \times N_{bin} \tag{2-11}$$

2.1.3. Gradient Location and Orientation Histogram

Mikolajczyk and Schmid (2005) proposed an extension of the scale-invariant feature transform (SIFT) feature called the gradient location and orientation histogram (GLOH).[15] Similar to the SIFT and HOG features, a GLOH feature is a concatenation of histograms of gradient orientations. These histograms are built from the gradient magnitude and orientation of each pixel calculated by using Equations (2-7) and (2-8) However, unlike SIFT and HOG, GLOH improves the robustness and increases the discriminative power by generating a histogram for each section in a log-polar location grid rather than in a Cartesian grid.[16-19]

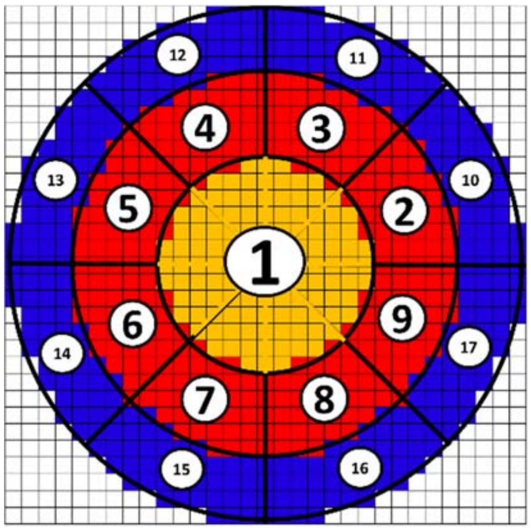


Figure 2-3. An illustration of a log-polar location grid divided into 17 sections, where the radius is set to 6, 11, and 15 pixels.

In the original GLOH paper,[15] the authors proposed a log-polar location grid that consists of three sections in a radial direction and eight sections in an angular direction, which is demonstrated in Figure 2-3. By quantizing gradient orientations in each section to 16 histogram bins, the size of the GLOH feature is started at 272 dimensions but is later reduced down to 128 dimensions through the use of principal component analysis (PCA).

2.1.4. Speeded-Up Robust Features

Speeded up-robust features (SURF) were proposed by Bay et al. as a SIFT-like algorithm.[20] The main idea of SURF is to reduce the computational cost while maintaining robustness through the use of the integral image technique.[21-23] This technique enables a fast computation of rectangular type convolution filters.[24] For a given image I, the integral image I_Σ can be computed by the following equation:

$$I_\Sigma(i,j) = \sum_{x=1}^{i} \sum_{y=1}^{j} I(i,j) \qquad (2\text{-}12)$$

Once the integral image I_Σ has been computed, the calculation for the sum of the intensities in a rectangular area on a given image are reduced down to only three arithmetic operations as demonstrated in Figure 2-4. Thus, the computation cost for convolving a rectangular filter with a given image is independent of its size.

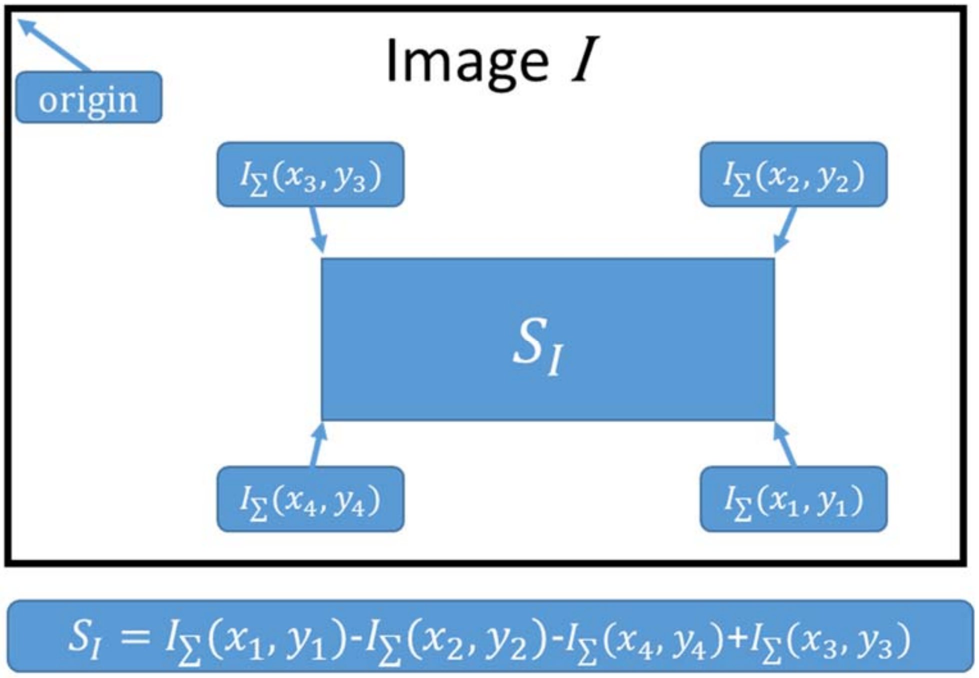


Figure 2-4. A demonstration for calculating the sum of the intensities in a rectangular area S_I by performing three arithmetic operations on the four values from the integral image.

The SURF algorithm consists of three steps, 1) detecting all interest points in a given image, 2) determining the major orientation of each interest point, and 3) extracting a SURF feature from the interest points. In the first step, the SURF interest points are defined as the extrema points in stacked blob response maps (a blob response volume). The blob response map is the determinant of the Hessian for a given scale.[25, 26] The Hessian $H(p_{ij}, \sigma)$ of a point (i, j) in a given image I at scale σ is defined as:

$$H(p_{ij}, \sigma) = \begin{bmatrix} L_{xx}(p_{ij}, \sigma) & L_{xy}(p_{ij}, \sigma) \\ L_{xy}(p_{ij}, \sigma) & L_{yy}(p_{ij}, \sigma) \end{bmatrix} \tag{2-13}$$

where:

$L_{xx}(p_{ij}, \sigma)$ denotes the convolution of $\frac{\partial^2}{\partial x^2} g(\sigma)$ with image I at point(i, j),

$L_{xy}(p_{ij}, \sigma)$ denotes the convolution of $\frac{\partial^2}{\partial xy} g(\sigma)$ with image I at point(i, j),

$L_{yy}(p_{ij}, \sigma)$ denotes the convolution of $\frac{\partial^2}{\partial y^2} g(\sigma)$ with image I at point(i, j), and

$\{\frac{\partial^2}{\partial x^2} g(\sigma), \frac{\partial^2}{\partial xy} g(\sigma), \frac{\partial^2}{\partial y^2} g(\sigma)\}$ denote the approximation for the second-order Gaussian partial derivative in the direction of x, xy, and y, respectively. These approximations are illustrated in Figure 2-5.

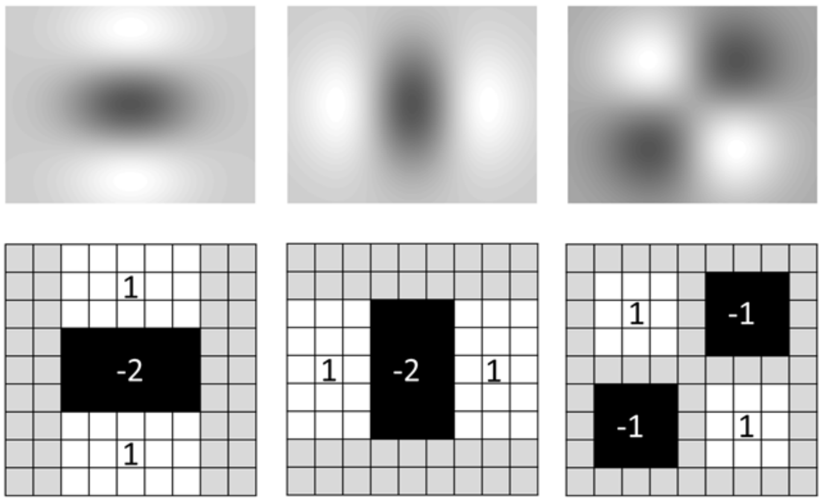
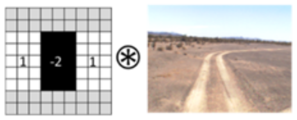


Figure 2-5. From left to right, the images on the top row are filters of the second-order Gaussian partial derivative from the y, x, and xy-direction, respectively. The bottom row shows the corresponding approximated version of the images on the top row.

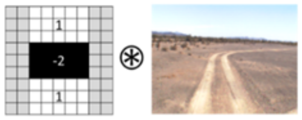
Due to the symmetry property of the Hessian matrix, only three elements in the matrix, which are $L_{xx}(\sigma)$, $L_{xy}(\sigma)$, and $L_{yy}(\sigma)$ need to be computed. Using the integral image technique, the calculation for the Hessian matrices of all pixels in an input image at a given scale is reduced down to several arithmetic operations as shown in Figure 2-6.

SURF interest points detector used Equation 2-14 to generate multiple blob response maps from the varied scale σ. By stacking the response maps on top of one another, the blob response volume is created. Afterward, the algorithm would select the interest point by using a 3D non-max suppression algorithm to select the extrema points in the blob response volume. These processes are repeated for a given number of octaves, and all the interest points in an image are detected.

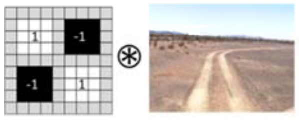
$$Map_{blob\ response} = \det\big(H(\sigma)\big) = L_{xx}(\sigma)L_{yy}(\sigma) - (0.9\ L_{xy}(\sigma))^2 \quad (2\text{-}14)$$



$$L_{xx}(\sigma) = \left(I_\Sigma\begin{bmatrix} i_{\Sigma_{3,7}} & \cdots & i_{\Sigma_{1018,7}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,766}} & \cdots & i_{\Sigma_{1018,766}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{3,3}} & \cdots & i_{\Sigma_{1018,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,762}} & \cdots & i_{\Sigma_{1018,762}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{1,7}} & \cdots & i_{\Sigma_{1016,7}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{1,766}} & \cdots & i_{\Sigma_{1016,766}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{1,3}} & \cdots & i_{\Sigma_{1016,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{1,762}} & \cdots & i_{\Sigma_{1016,762}} \end{bmatrix} \right)$$
$$-2\left(I_\Sigma\begin{bmatrix} i_{\Sigma_{6,7}} & \cdots & i_{\Sigma_{1021,7}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{6,766}} & \cdots & i_{\Sigma_{1021,766}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{6,3}} & \cdots & i_{\Sigma_{1021,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{6,762}} & \cdots & i_{\Sigma_{1021,762}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{4,7}} & \cdots & i_{\Sigma_{1019,7}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{4,766}} & \cdots & i_{\Sigma_{1019,766}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{4,3}} & \cdots & i_{\Sigma_{1019,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{4,762}} & \cdots & i_{\Sigma_{1019,762}} \end{bmatrix} \right)$$
$$+\left(I_\Sigma\begin{bmatrix} i_{\Sigma_{9,7}} & \cdots & i_{\Sigma_{1024,7}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{9,766}} & \cdots & i_{\Sigma_{1024,766}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{9,3}} & \cdots & i_{\Sigma_{1024,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{9,762}} & \cdots & i_{\Sigma_{1024,762}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{7,7}} & \cdots & i_{\Sigma_{1022,7}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,766}} & \cdots & i_{\Sigma_{1022,766}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{7,3}} & \cdots & i_{\Sigma_{1022,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,762}} & \cdots & i_{\Sigma_{1022,762}} \end{bmatrix} \right)$$



$$L_{yy}(\sigma) = \left(I_\Sigma\begin{bmatrix} i_{\Sigma_{7,3}} & \cdots & i_{\Sigma_{1022,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,762}} & \cdots & i_{\Sigma_{1022,762}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{7,1}} & \cdots & i_{\Sigma_{1022,1}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,760}} & \cdots & i_{\Sigma_{1022,760}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{3,3}} & \cdots & i_{\Sigma_{1018,3}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,762}} & \cdots & i_{\Sigma_{1018,762}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{3,1}} & \cdots & i_{\Sigma_{1018,}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,760}} & \cdots & i_{\Sigma_{1018,7}} \end{bmatrix}\right.$$
$$-2\left(I_\Sigma\begin{bmatrix} i_{\Sigma_{7,6}} & \cdots & i_{\Sigma_{1022,6}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,765}} & \cdots & i_{\Sigma_{1022,765}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{7,4}} & \cdots & i_{\Sigma_{1022,4}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,763}} & \cdots & i_{\Sigma_{1022,763}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{3,6}} & \cdots & i_{\Sigma_{1018,6}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,765}} & \cdots & i_{\Sigma_{1018,765}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{3,4}} & \cdots & i_{\Sigma_{1018}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,763}} & \cdots & i_{\Sigma_{1018,}} \end{bmatrix}\right.$$
$$+\left(I_\Sigma\begin{bmatrix} i_{\Sigma_{7,9}} & \cdots & i_{\Sigma_{1022,9}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,768}} & \cdots & i_{\Sigma_{1022,768}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{7,7}} & \cdots & i_{\Sigma_{1022,7}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{7,766}} & \cdots & i_{\Sigma_{1022,766}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{3,9}} & \cdots & i_{\Sigma_{1018,9}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,768}} & \cdots & i_{\Sigma_{1018,768}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{3,7}} & \cdots & i_{\Sigma_{1018}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{3,766}} & \cdots & i_{\Sigma_{1018,7}} \end{bmatrix}\right.$$



$$L_{xy}(\sigma) = \left(I_\Sigma\begin{bmatrix} i_{\Sigma_{4,4}} & \cdots & i_{\Sigma_{1019,4}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{4,763}} & \cdots & i_{\Sigma_{1019,763}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{4,2}} & \cdots & i_{\Sigma_{1019,2}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{4,761}} & \cdots & i_{\Sigma_{1019,761}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{2,4}} & \cdots & i_{\Sigma_{1017,4}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{2,763}} & \cdots & i_{\Sigma_{1017,763}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{2,2}} & \cdots & i_{\Sigma_{1017,}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{2,761}} & \cdots & i_{\Sigma_{1017,7}} \end{bmatrix}\right.$$
$$-\left(I_\Sigma\begin{bmatrix} i_{\Sigma_{8,4}} & \cdots & i_{\Sigma_{1023,4}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{8,763}} & \cdots & i_{\Sigma_{1023,763}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{8,2}} & \cdots & i_{\Sigma_{1023,2}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{8,761}} & \cdots & i_{\Sigma_{1023,761}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{6,4}} & \cdots & i_{\Sigma_{1021,4}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{6,763}} & \cdots & i_{\Sigma_{1021,763}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{6,2}} & \cdots & i_{\Sigma_{1021,}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{6,761}} & \cdots & i_{\Sigma_{1021,7}} \end{bmatrix}\right.$$
$$+\left(I_\Sigma\begin{bmatrix} i_{\Sigma_{8,8}} & \cdots & i_{\Sigma_{1023,8}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{8,767}} & \cdots & i_{\Sigma_{1023,767}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{8,6}} & \cdots & i_{\Sigma_{1023,6}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{8,765}} & \cdots & i_{\Sigma_{1023,765}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{6,8}} & \cdots & i_{\Sigma_{1021,8}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{6,767}} & \cdots & i_{\Sigma_{1021,767}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{6,6}} & \cdots & i_{\Sigma_{1021}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{6,765}} & \cdots & i_{\Sigma_{1021,}} \end{bmatrix}\right.$$
$$-\left(I_\Sigma\begin{bmatrix} i_{\Sigma_{4,8}} & \cdots & i_{\Sigma_{1019,8}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{4,767}} & \cdots & i_{\Sigma_{1019,767}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{4,6}} & \cdots & i_{\Sigma_{1019,6}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{4,765}} & \cdots & i_{\Sigma_{1019,765}} \end{bmatrix} - I_\Sigma\begin{bmatrix} i_{\Sigma_{2,8}} & \cdots & i_{\Sigma_{1017,8}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{2,767}} & \cdots & i_{\Sigma_{1017,767}} \end{bmatrix} + I_\Sigma\begin{bmatrix} i_{\Sigma_{2,6}} & \cdots & i_{\Sigma_{1017}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{2,765}} & \cdots & i_{\Sigma_{1017,}} \end{bmatrix}\right.$$

$I_\Sigma\begin{bmatrix} i_{\Sigma_{x_1,y_1}} & \cdots & i_{\Sigma_{x_2,y_1}} \\ \vdots & \ddots & \vdots \\ i_{\Sigma_{x_1,y_2}} & \cdots & i_{\Sigma_{x_2,y_2}} \end{bmatrix}$ represents a patch of an integral image formed from column x_1 to x_2 and row y_1 to y_2.

Figure 2-6. These three Hessian matrix element computations demonstrate the calculation process for all pixels in an input image size of 1024 x 768 pixels at scale $\sigma = 1.2$.

The second step of SURF is the orientation assignment. For each interest point, a circular area around the interest point is filtered with a Haar wavelet in both x and y directions. Once the wavelet responses are calculated, the dominant orientation of the interest point is estimated by summing all responses within a sliding orientation window sized at $\frac{\pi}{3}$. The two summed responses of the x and y direction will yield the local orientation vector. The longest orientation vector among all the windows will be defined as the interest point orientation.

Once the orientation is assigned to every interest point in a given image, the algorithm proceeds to the last step, which is the process of extracting a SURF feature for each interest point. Using the assigned orientation and the Haar wavelet responses from the second step, the SURF feature extraction algorithm is described in Table 2-1:

Table 2-1. SURF feature extraction.

- N is the number of interest points in an image and $i = \{1, \ldots, N\}$
- For each interest point, p_i
    - — Extract a square region R_i around p_i that orients with the assigned orientation.
    - — Split R_i into a 4x4 sub-regions
    - — For each sub-region, r_j
        - Compute the four dimensional descriptor X_j by

$$X_j = \{\sum dx, \sum|dx|, \sum dy, \sum|dy|\}$$

where:

$\sum dx$ denotes the sum of the horizontal Haar wavelet responses in the sub-region

$\sum|dx|$ denotes the sum of the absolute of the horizontal Haar wavelet responses in the sub-region

$\sum dy$ denotes the sum of the vertical Haar wavelet responses in the sub-region

$\sum|dy|$ denotes the sum of the absolute of the vertical Haar wavelet responses in the sub-region

— Compose the SURF feature of p_i by concatenating X_j where $j = \{1, \ldots, 16\}$ into a feature vector size of 64 dimensions.

After the feature extraction is completed, the interest points are ready for matching. To increase the speed of the matching step, the trace of the Hessian matrix of each interest point is included. Since the sign of a Hessian trace can be used to indicate whether the interest point is a dark blob on a light background or vice versa, the algorithm will only attempt to match the interest points that have the same sign. This has the potential to reduce the pairwise distance calculation number between interest points by half; thus, the computational cost of the matching step is significantly reduced.

2.2. Classifier

In machine learning, a classifier is an algorithm that separates data points into appropriate classes based on their attributes (features). The proposed system will use the supervised learning approach to train the classifier. This means that the classifier will analyze the training data and produce a model capable of classifying new data points.

2.2.1. Classification and Regression Tree (CART)

The classification and regression tree (CART) is a classical decision tree used in classification and regression problems.[27] A decision tree is a binary tree where each internal node will test the data points and split them based on their attributes (features). CART is a type of binary decision tree that uses the Gini diversity index (GDI) to grow (split) the decision tree. GDI is a measure used to evaluate the purity of a tree's node. The purity of a tree's node indicates the degree of the class mixture within that node, and the node is considered pure (GDI=0) when it contains data points of a single class. For a problem with C classes, the GDI of node n is calculated as:

$$GDI_n = 1 - \sum_{c=1}^{C} {p_c}^2$$
$$p_c = \frac{|D_c|}{|D|} \tag{2-15}$$

where:

D represents the set of all data points in node n, and

D_c denotes the data points in node n that belong to class $\boldsymbol{c}$.

To grow a decision tree using the CART algorithm, the data is optimally split in the top node, which results in two child nodes. These child nodes will be used as the parent nodes for the next split. By recursively splitting nodes, a decision tree is grown, and the process continues until all child nodes are pure. The optimal split is obtained by searching for threshold t of feature f that maximizes the impurity gain (ΔI). The impurity gain of a split at node n can be calculated as follows:

$$\Delta I_n = GDI_n - \left(\frac{|D_L|}{|D|} \times GDI_L\right) - \left(\frac{|D_R|}{|D|} \times GDI_R\right)$$
$$D_L = \{x_i : x_i^f \leq t\},$$
$$D_R = \{x_i : x_i^f > t\}, \tag{2-16}$$
$$D = \{x_1, x_2, x_3, \dots\}$$

where x_i^f represents the value of the f^{th} feature on data point x_i. One drawback of this training method is that it is likely to produce a decision tree that overfits the training data.

2.2.2. Random Forest

The idea of a random subspace decision tree was originally conceptualized by Ho[28] and further developed by Breiman[29] into a random forest classifier. Random forest is an ensemble of classification and regression trees (CARTs) where each CART is generated by using a bootstrapping feature set. The bootstrapping feature set is a training data subset where the size of its dimension has been reduced by random selection from the feature space (random subspace). Alternatively, random forest can be defined as:

$$RF = \{h_1, h_2, h_3, \dots, h_k\} \tag{2-17}$$

$$h_{\mathrm{k}} = H(X, \Theta_k)\ , X = \{x_1, x_2, x_3, \dots, x_n : x_n \in \mathbb{R}^D\}, |\Theta_k| < D$$

where:

h_{k} is a CART trained from a dataset X using only the features indicated by Θ_k.

X is a dataset with a D dimensional feature, and

Θ_k represents index features that are randomly selected from $\{1,2,3, \dots, D\}$.

A random forest classifies a testing data point by assigning a class label based on the majority vote. The main advantage of random forest is that it cannot overfit since Breiman's original work[29] has proved that the generalization error converges with the growing addition of CARTs.

2.2.3. Transfer Learning Random Forest

Transfer learning can be applied to the random forest classifier using two approaches, substitution and the mixed information gain method. The first approach substitutes trees in the random forest ensemble. Since the goal of transfer learning is to adapt a classifier that was trained in a source data set (i.e., training data) to a testing data domain by utilizing a small amount of testing data (denoted here as target data), the random forest classifier can be modified by replacing (substituting) the weaker trees with a new set of trees trained by the target data. The strength of each tree in the random forest is evaluated by the misclassification rate that the tree produces when testing with target data; moreover, the tree is considered weak if it has a high misclassification rate. The number of substituting trees can be determined with an experiment. Once the substitution step is completed, the resultant random forest will be able to correctly process the data in the target domain (i.e., testing data).

The second transfer learning approach (the mixed information gain method) is based on a work proposed by Norberto A. Goussies.[30] The concept of this approach is to use both the source data and target data to train a new random forest, unlike the first approach that attempts to adapt a pre-trained random forest. Typically, the amount of target data is minuscule when compared to the source data, therefore, a classifier that was directly trained from both data sets will be biased toward the source data and render the target data ineffective. To counter this problem, Goussies proposed that the trees in a decision forest be trained with the mixed information gain method. As mentioned in Section 2.2.1, the objective function for training a traditional CART is the maximization of the impurity gain, $argmax\ \Delta I$. In case of the mixed information gain (I_{mix}), the objective function is modified to the following equation:

$$\underset{t_f \in \mathbb{R}}{\operatorname{argmax}} I_{mix} \coloneqq (1-\gamma)\Delta I_t(t_f) + \gamma \Delta I_s(t_f)\ ,\ \{\gamma \in \mathbb{R} : 0 \leq \gamma \leq 1\} \tag{2-18}$$

where:

I_{mix} represents the mixed information gain,

I_t represents the information gain from the target data,

I_s represents the information gain from the source data,

t_f represents the threshold value of the f^{th} dimension,

γ represents a scalar parameter that encompasses the weight of information gain from both data.

By using the weight parameter, γ, a decision tree generated by the modified objective function is able to balance the influence so that both sets of data can create a proper transfer learning decision forest.

2.3. Image Processing

2.3.1. Log Gabor Filter Bank

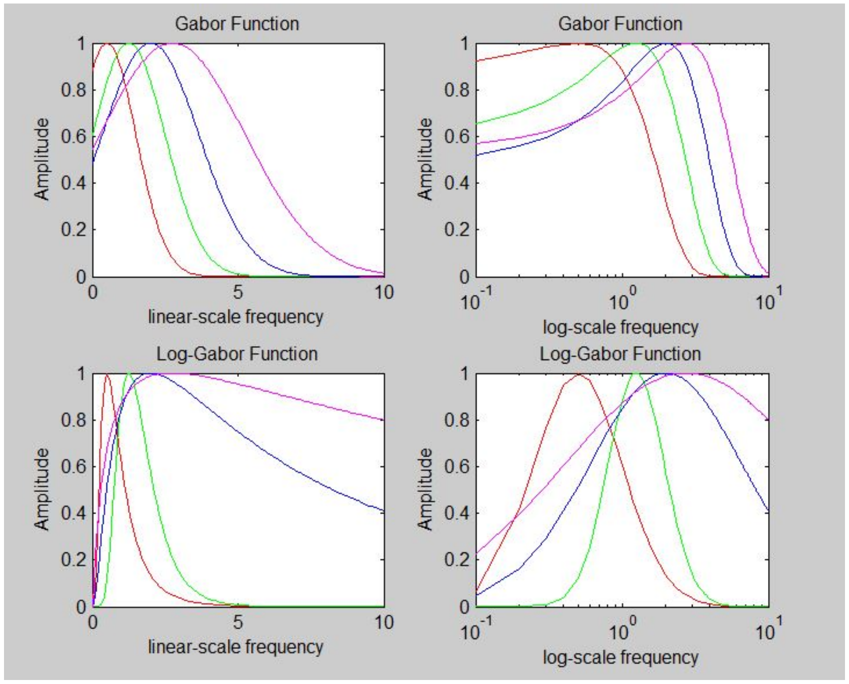


Figure 2-7. The comparison between the Gabor function versus the log Gabor function.

In 1987, David J. Field proposed the log Gabor filter (LGF) as an improvement over the original Gabor filter.[31] LGF is a signal processing technique based on the Gabor filter. Like the Gabor filter, it is capable of analyzing the frequency characteristics of data. However, Field[32] showed that LGF has a great advantage over the traditional Gabor filter in natural images. Compared to the original Gabor filters, which have Gaussian transfer functions in the linear scale frequency domain, LGF has the same transfer function in the log-scale frequency domain. The most important advantage of an LGF over the traditional Gabor filter is the zero DC component. As such, the response from filtering an input signal with LGFs is independent of the mean value of the input signal. As shown in Figure 2-7, with the same filter configuration, the Gabor filters have non-zero DC component while LGF filters have zero response at a DC frequency.

Due to the zero DC component, the analytic expression of LGF in a space domain is impossible to construct. With a two-dimensional (2D) LGF, which is an extension of the log Gabor filter, we can create a filter bank composed of multiple scales and orientations filters. In common practice, the LGF bank is generated by performing the numerical inverse Fourier transformation of the 2D log Gabor function in the frequency domain. In the dissertation, the implementation of the LGF proposed by Fischer[33] will be chosen. It is defined as follows:

$$G(\omega, \theta) = exp(-0.5(\frac{\omega-\omega_s}{\sigma_\omega})^2)exp(-0.5(\frac{\theta-\theta_s}{\sigma_\theta})^2) \tag{2.19}$$

where (ω, θ) represents the log-polar coordinate,

s represents the scale of the filter,

ω_s represents the center of filter frequency in log scale,

σ_ω represents the frequency's bandwidth parameter,

θ_s represents the orientation of the filter, and

σ_θ represents the orientation's bandwidth parameter.

For our application, we picked an LGF filter bank that covers all orientations. Thus, by specifying the total number of filters in each frequency scale, N_o, the orientation of the filters would be $\theta_s \in (0, \frac{\pi}{N_o}, \frac{2\pi}{N_o}, \ldots, \pi - \frac{\pi}{N_o})$. Figure 2-8 illustrates the LGF bank used in my previous work.[34]

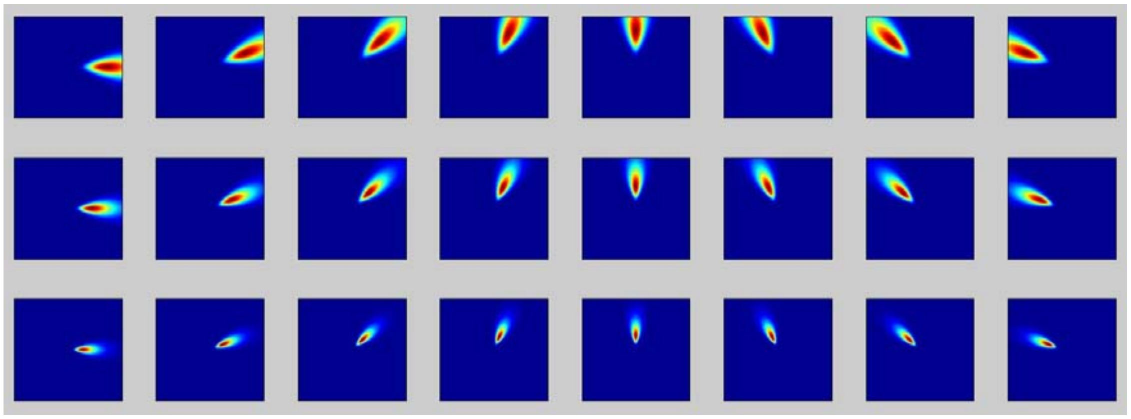


Figure 2-8. An example of an LGF bank with three scales and eight orientations

2.3.2. Histogram Equalization

Histogram equalization (HE) is an image processing technique that enhances a given image by improving its contrast. HE archives this by stretching the intensity range of the given image proportioning to the frequent of intensity values (histogram), which effectively increases the global contrast in the image. An implementation of HE for a given grayscale image I_g with intensity range from 0 to 255 is start by calculating the cumulative distribution function cdf.

$$cdf_{I_g}(v) = \sum_{i=0}^{v} P_i \ , v_{min} \leq v \leq v_{max} \tag{2-20}$$

$$P_i = \frac{n_i}{N} \ , 0 \leq i \leq 255$$

where v_{min} represents minimum intensity value in I_g,

v_{max} represents the maximum intensity value in I_g,

n_i represents the number of pixels with an intensity value equal to i, and

N represents the number of pixels in I_g.

Next, pixels that have v intensity value are replaced with $he_{I_g}(v)$, which defined as follows:

$$he_{I_g}(v) = round(\frac{cdf_{I_g}(v) \times N - cdf_{I_g}(v_{min})}{N - cdf_{I_g}(v_{min})} \times 255) \tag{2-21}$$

This process repeated until all pixels in I_g were replaced. An example of an image that has been processed by HE is shown in Figure 2-9.

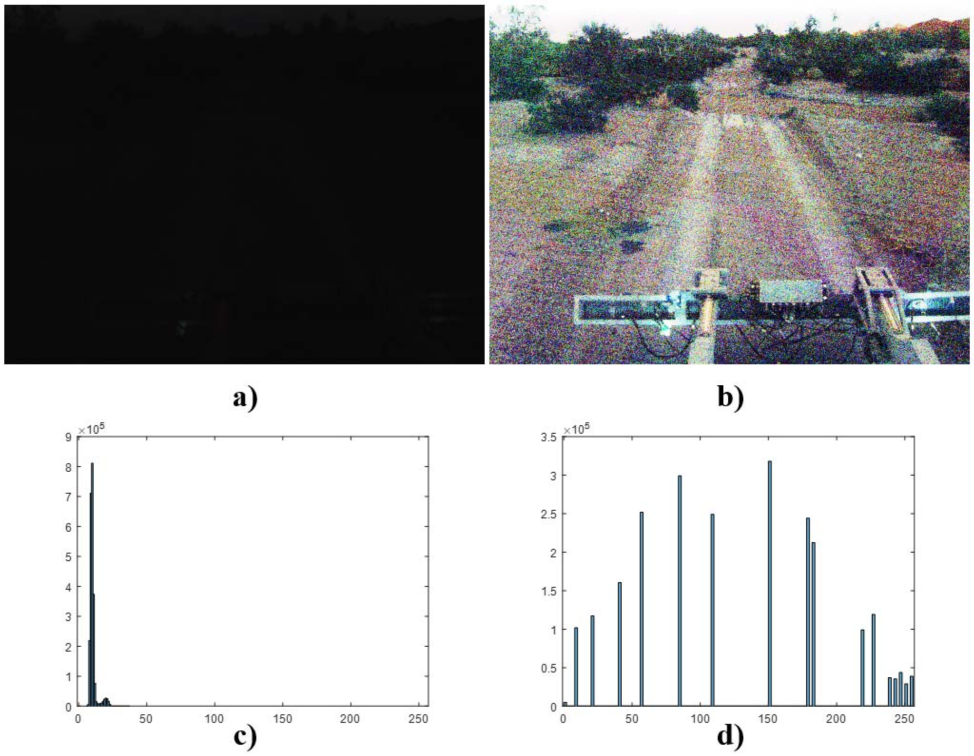


Figure 2-9. An example demonstrates the effectiveness of the histogram equalization technique. a) a severely underexposed image, b) histogram equalization (HE) enhanced image, c) the histogram of image a, and d) the histogram of image b.

2.4. Optimization

2.4.1. Genetic Algorithm

The genetic algorithm (GA) is an optimization algorithm inspired by the theory of natural selection. The concept of GA is based on the notation of the-survival-of-the-fittest, which follows the precept that only fit organisms can survive long enough to pass on their genetic traits and potentially produce fitter organisms for the next generation. GA optimization operates by representing the possible solutions as chromosomes, and an initial set of chromosomes (population pool) is a given number of chromosomes that are randomly generated. In each generation, parent chromosomes are selected (selection) from the population pool to enter the reproduction stage where they produce a new set of chromosomes through crossover and mutation operation. Crossover is an operation used to combine the genetic information from a pair of parent chromosomes and produces a new chromosome offspring. The mutation operation is a method that GA optimization uses to increase diversity in the population pool by making a small random change to the genes in the offspring chromosome. In this dissertation, deterministic tournament selection[35] and intermediate recombination[36] are the chosen selection and crossover operation, respectively. Deterministic tournament selection operates by randomly choosing a small number of chromosomes (in our case, we set this number to four) and select the highest fitness chromosome among the chosen chromosomes for the crossover operation. For intermediate recombination, let R represents an uniform randomized vector range from 0 to 1 that has the same length as chromosome C.

$$C_{offspring} = C_{parent1} + R * \left(C_{parent2} - C_{parent1}\right) \tag{2-22}$$

where $C_{parent1}$and $C_{parent2}$ are chromosomes that were selected from the population pool through the tournament selection method.

After the reproduction stage, both parent and offspring chromosomes are evaluated by the fitness function, which is the objective function of the optimizing problem. Lastly, a given number of the top strongest chromosomes (high fitness) are preserved in the next generation's population pool. By repeating this process, the chromosomes in the population pool move closer to an optimal solution. The iterative process terminates on some convergence criteria, i.e., reaching the maximum number of generations, or when little to no change in the population's fitness is found.

2.4.2. Multiple Objectives Genetic Algorithm

In a multi-objective optimization problem, the optimal solutions are defined as the solutions that dominate other solutions (*domination* scheme).[37] Based on the mathematical concept of partial ordering, the *domination* scheme has solution S_1 dominating solution S_2 if S_1 is not worse than S_2 in all objectives and S_1 is better than S_2 in at least one objective. In the dissertation, the nondominated sorting genetic algorithm (NSGA II)[38] was adopted for the multi-objective optimization due to its low computation cost.[39-41]

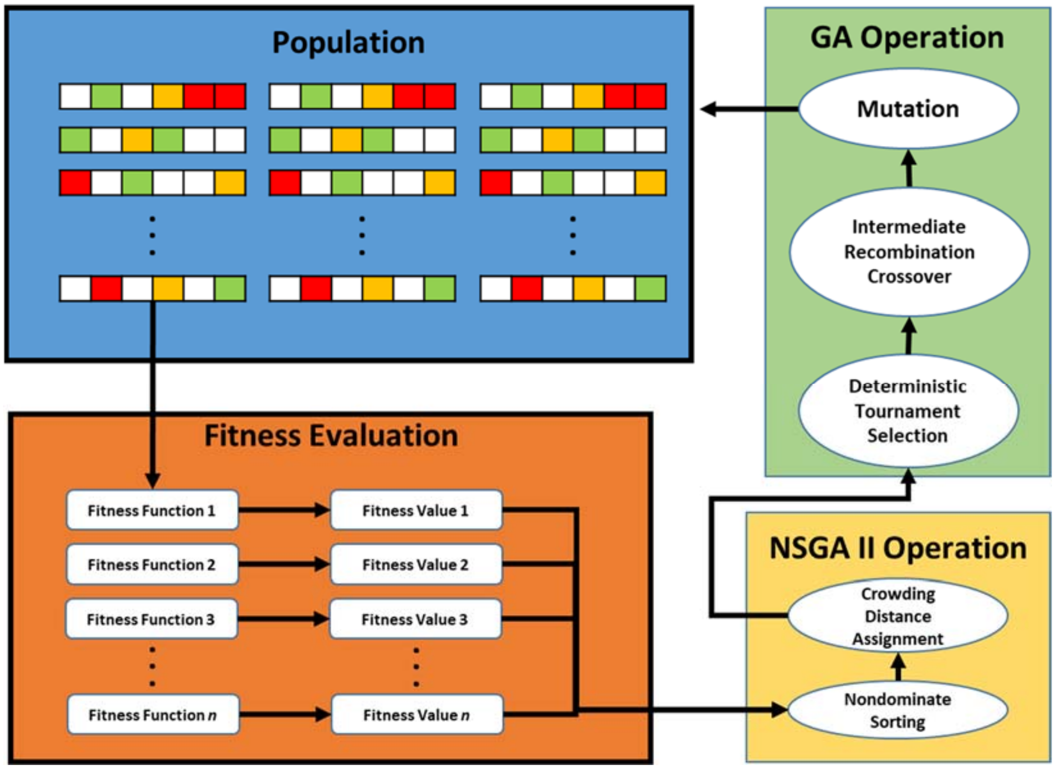


Figure 2-10. A flowchart of NSGA II optimization shows that the optimization has two additional processes to the traditional GA operations, nondominated sorting and crowding distance assignment.

The two advantages of NSGA II are the low computation cost and the preservation of solution diversity. As illustrated in Figure 2-10, the NSGA II algorithm finds the Pareto front of optimal solutions, which are solutions wherein the fitness value of any objective functions cannot be increased without reducing other objective values. This balance of objective functions is achieved by adding traditional GA operations to two new processes, the nondominated sorting and the crowding distance assignment. The nondominated sorting sorts the solutions by assigning a dominating rank to all solutions. The dominating rank is defined as the solution belonging to rank one if no other solution dominates it; meanwhile, the dominated solutions are stored for the next ranking. The ranking continues until all the solutions have a dominating rank. The solutions in rank one represent the dominating front, which moves closer to the optimal Pareto front as generations increase. The pseudo algorithm for nondominated sorting is demonstrated in Table 2-2.

Table 2-2. Nondominated sorting algorithm.

```
For each For each $c \in P$                  Each chromosome in the population pool
    $S_c = \emptyset$
    $n_c = 0$
    For each $\hat{c} \in P$
        If $c$ dominates $\hat{c}$ then
            $S_c = S_c \cup \{\hat{c}\}$         Add $\hat{c}$ to the set of solutions dominated by c
        Else if $\hat{c}$ dominates $c$
            $n_c = n_c + 1$
    If $n_c = 0$ then
    $c_{rank} = 1$                             c belongs to set of rank 1 solution
    $F_1 = F_1 \cup \{c\}$                       Add c to the Pareto front solution set
$i = 0$
While $F_i \neq \emptyset$
    $F_{tmp} = \emptyset$                      Temporarily store the solution of the next front
```

```
For each $p \in F_i$
    For each $q \in S_p$
        $n_q = n_q - 1$
        If $n_q = 0$ then
            $q_{rank} = i + 1$                    *q belongs to the next front*
            $F_{tmp} = F_{tmp} \cup \{q\}$
$i = i + 1$
$F_i = F_{tmp}$
```

In the second process, the crowding distance assignment is used to preserve the diversity of the solutions in the Pareto front. The crowding distance of a solution is defined as the sum of the Manhattan distances between it and the two adjacent solutions. Therefore, the more crowded solution would have a smaller crowding distance and vice versa. The algorithm for assigning the crowding distance is described in Table 2-3.

Table 2-3. Crowding distance assignment algorithm.

```
$N = |F_1|, I = F_1$                          *Number of solutions in the Pareto front*
For each $n, n = \{1, \dots, N\}$,
    $I[n]_{dist} = 0$                         *Initialized crowding distance of each solution.*
For each objective $m$
  $I = sort(I, m)$                            *Sort the solution based on objective m*
  $I[1]_{dist} = I[N]_{dist} = \infty$        *max and min solutions on the Pareto front*
  For $i$=2 to $(N - 1)$                      *that have been selected.*
    $I[i]_{dist} = I[i]_{dist} + (I[i+1]^m_{dist} - I[i-1]^m_{dist})/(max_m - min_m)$
```

During the selection process, NSGA II selects the solutions for the next generation based on the solutions ordering. The order of the solution is a prioritization of the lower dominating rank and the large crowding distance, which are defined as:

$$\{i_{order} > j_{order} | (i_{rank} \leq j_{rank}), (\, i_{dist} > j_{dist})\} \tag{2-23}$$

where:

i_{order}, j_{order} represent the order of solution i and j,

i_{rank}, j_{rank} represent the nondominated rank of solution i and j, and

i_{dist}, j_{dist} represent the crowding distance of solution i and j .

Chapter 3: Previous Work

Three previous papers by the author of this dissertation directly contributed to this work. The first one is titled "Road Recognition in Poor Quality Environments for Forward Looking Buried Object Detection,"[42] which proposed a road region estimation (REE) algorithm that provides accurate road detection with a closer view of the look-ahead distance. The second paper is titled "An Application of Log Gabor Filter on Road Detection in Arid Environments for Forward Looking Buried Object Detection,"[34] which is referred to as the road model formation (RMF) in this dissertation. The objective is to extend the detection range of the first work while retaining the detection accuracy. However, with the later experiments, it was discovered that the road model was not robust since the algorithm lacked the ability to adapt to environmental changes in a new set of data. Therefore, the third paper, "Tuning Log Gabor Filter Bank Using Genetic Algorithm Based Optimization,"[43] introduces an optimization method for the log Gabor filter bank (LGFB)and provides a road model algorithm that can adapt to and handle changes in the data environment. It is important to note that slight differences exist between the algorithm presented in these three papers and the methodology used in the final road detection system developed in the final stage of my Ph.D. graduate work. The algorithm mechanisms presented in the three previous papers are detailed in the next section.

3.1.Road Region Estimation

In the road region estimation (RRE) paper, Breiman's random forest[29] was used to classify the road samples in an image. The random forest classifier is a collection of classification and regression trees (CART). From the training data, each individual CART in the random forest classifier is generated by using a random set of features available in a predefined feature space. The reason for using the random forest classifier is its ability to never overfit data.[29] Furthermore, with the addition of a reinforcement technique, the algorithm of the random forest classifier was able to learn from its mistakes. With the assumption that the camera is always aimed forward, the algorithm was able to identify its past mistakes and used them to correct the classifier.

3.1.1. Training the Road vs. Non-road Classifier

The road vs. non-road classifier was trained by extracting features from image samples. The samples of each road and non-road class were uniformly selected across an image at a certain distance from the camera. Figure 3.1 shows the locations where the road samples, marked by a green-circle, and the non-road samples, marked by a blue-circle chain, were selected.

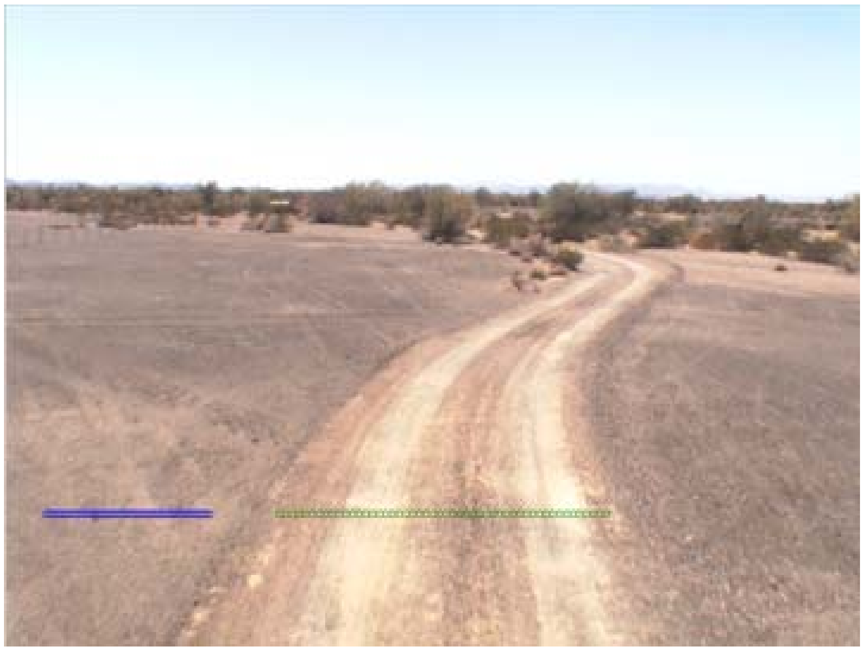


Figure 3-1. An illustration of a non-road and road image sample used during the training process of the random forest classifier.

In our original study, multiple features were extracted from each color channel in the RGB color images. These features are local binary patterns (LBPs),[11] histograms of oriented gradients (HOGs),[14] and histograms of the color values themselves.[44] Table 3-1 shows the list and size of features that were used. However, a further study has shown that both the color cues and the histograms features were ineffective and failed to improve the detection rate. For this reason, the histograms features have been omitted from the proposed system, and the RGB color images were converted to grayscale images prior to extracting the features. By bundling the feature vectors from all image samples in the training dataset, a feature matrix, where each row represented an image sample, was created. From this feature matrix, a random forest classifier was trained by generating multiple CARTs based on randomly selected features. The features were randomly selected each time a CART was created, and the number of CARTs was dependent on the data and the size of its features. For this research, 250 CARTs were determined reasonable for training the random forest classifier.

Table 3-1 Features list and size

Feature	Red	Green	Blue	Total
HOG	9	9	9	27
LBP	10	10	10	30
Histogram	10	10	10	30
Normalize Histogram	10	10	10	30
Total	39	39	39	117

3.1.2. Road Classification

It is computationally expensive and unreasonable to classify every pixel in a given image. Therefore, only a row of image samples at a fixed look-ahead distance in the image has been classified in my work. By cropping 100 image samples to 30×30 pixels evenly across the image at approximately the same look-ahead distance, the same feature extraction methods used during the training step were applied in order to generate an input feature vector. The next step was to use the trained random forest to classify the input feature vector. The classified result labeled each image sample as road or non-road. In our original study, the scale-invariant feature transform (SIFT)[45, 46] algorithm was used to track past classified results to the current image. The SIFT algorithm is commonly used for matching or tracking individual objects existing in two different images. In the same sense, by using SIFT to track the interest points of past images to the current image, the movement of the camera was identified, and then the transformation mapping used for tracking the past classified results to the current image were calculated. The number of tracked classified results from the previous frame was not fixed but dependent on whether the tracked results still existed in the current image. However, for this dissertation, we will replace the SIFT algorithm with SURF since multiple studies[21-23] have shown that SURF is more robust while lowering the computation cost. Figure 3-2 shows an example of classification results of both past and current frames.

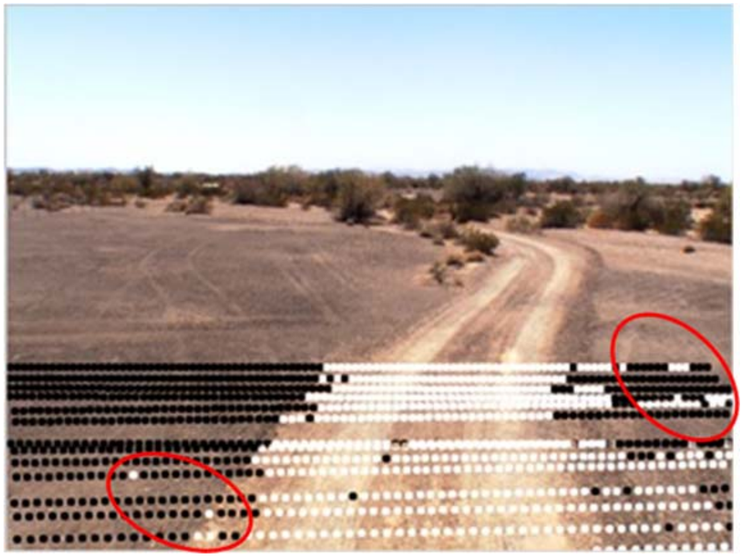


Figure 3-2. An example of the classified result of random forest. The white dots are road samples and the black dots are non-road samples. The top line is the current image while the rest are from past frames. The outlier results are circled in red.

In Figure 3-2, road classification results inform the 12 past frames, which are projected into the current frame using SIFT. Afterwards, the outlier results were removed by applying a connected component algorithm on the locations classified as road, and appropriate components were preserved. The appropriate components are defined as the largest connected component and any other components that have an area of at least 80% of the largest component. Finally, through using a convex hull algorithm,[47, 48] a road mask was generated by combining the classified results of the past and current images.

An example of a generated road mask is shown in the bottom right image of Figure 3-3. Notice that the output of the road classification is the road mask. Therefore, the road classification results are in pixel units (based on the fact that any pixels within the road mask are classified as road, and the rest are non-road).

3.1.3. Decision Fusion Classifier

To provide adaptability to road finding results, multiple sets of the random forests were used to classify the road at difference distances. Therefore, multiple road masks were generated for each image. Overlaying these masks together, a level of agreement was calculated for each pixel by using following equation:

$$\boldsymbol{C}_{x,y} = \left(\textstyle\sum_{d=1}^{n} \boldsymbol{B}_{x,y}^{d}\right)/n \qquad (3\text{-}1)$$

where:

$\boldsymbol{C}_{x,y} \in [0{:}\,1]$: Agreement level of pixel (x,y),

$\boldsymbol{B}_{x,y}^{d} \in \{0,1\}$: Value of d^{th} road mask image at (x,y) pixel, and

$n \in \mathbb{Z}$: Number of road mask generate per image.

Next, varied threshold values (agreement of multiple masks) were used to control the algorithm's result. By using a low threshold value, the algorithm not only produced higher detection rates but also more false alarms. On the other hand, higher threshold values lowered the false alarms, but the accuracy of the detection rates was reduced as shown below.

Decision fusion classifier pseudo code

```
— Initialize $n$, $t$ (threshold values), $\boldsymbol{R}$(result mask)=[0]
— for $d$ = 1 to $n$
        1. extract 100 image samples;
        2. extract features from samples;
        3. input feature to random forest classifier;
        4. SIFT transform past classified results to current image;
        5. dilate both past and current classifier results;
        6. use point connectivity to remove outlier result;
        7. generate road mask by creating the convex hull from remaining
           result, $\boldsymbol{B}^d$;
— end for
— use equation (3.1.1) to create road agreement image, $\boldsymbol{C}$
— if $\boldsymbol{C}_{x,y} \geq t$ then $\boldsymbol{R}_{x,y}$=1
```

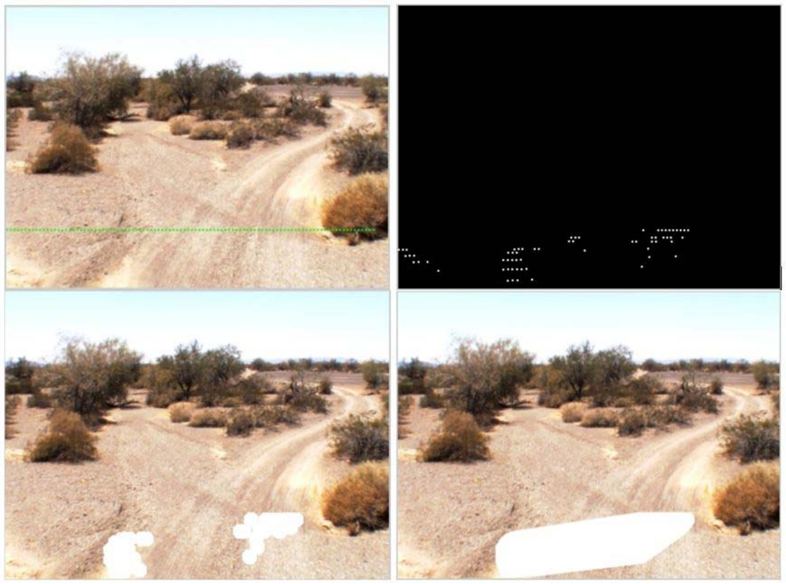


Figure 3-3. The results from each step of the decision fusion classifier algorithm at a look-ahead distance of 20 meters. The top left image shows Steps 1–2; the top right image shows Steps 3–4; the bottom left image shows Steps 5-6, and the bottom right shows Step 7.

The pseudo code for the combined road classification and decision fusion classifier algorithm is shown above. Furthermore, a reinforcement algorithm is provided at the end of Section 3.1.4 that illustrates the effect of different steps in the pseudo code.

3.1.4. Reinforcement Algorithm

The reinforcement algorithm strengthens the random forest classifier by learning from its mistakes. Assuming that the forward looking camera was always on the road, it is postulated that the area in the bottom center of an image consists of road pixels. Conversely, the pixels on both the left and the right bottom corners of the image are non-road pixels. Figure 3-4 shows an example of these specified areas as mentioned.

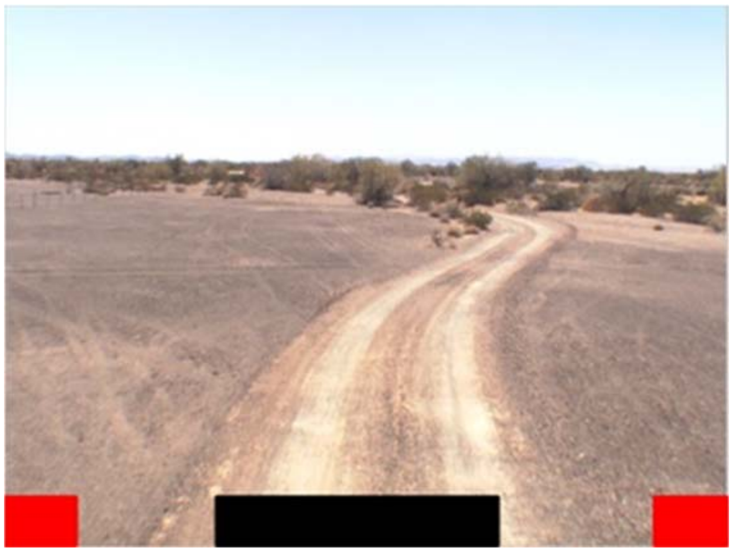


Figure 3-4. Region of A1 (black) contains road pixels and A2 (red) contains non-road pixels on an image.

By tracking all past classified results, we can identify which past results were misclassified, i.e., when those samples are at the bottom of the current frame. The next step is to extract features from these misclassified samples in order to generate a new random tree. Afterwards, each random tree in the forest is scored by reclassifying the misclassified samples and computing the number of incorrect votes for each tree. Finally, the tree with the worst score is replaced by the new one. A summary of the reinforcement algorithm is detailed below:

1. Specify range and area A1 for road region and A2 for non-road region;
2. Use SIFT to track all past classified results;
3. Collect any non-road results that fall in A1 and road results that fall in A2;
4. Extract features of the misclassified samples from the oldest image that contains these samples;
5. Create a new random tree once the feature extraction of misclassified samples reaches more than 100 samples;
6. Use the same set of misclassified samples as input to the random forest classifier to identify which tree was misclassified the most; and
7. Replace the worst tree with a new tree.

Figure 3-5 is an example of what happens when the reinforcement algorithm improves over a non-adaptive one by correctly classifying more road region as shown in the area outlined in red, and thus the road mask generated from its output is more accurate.

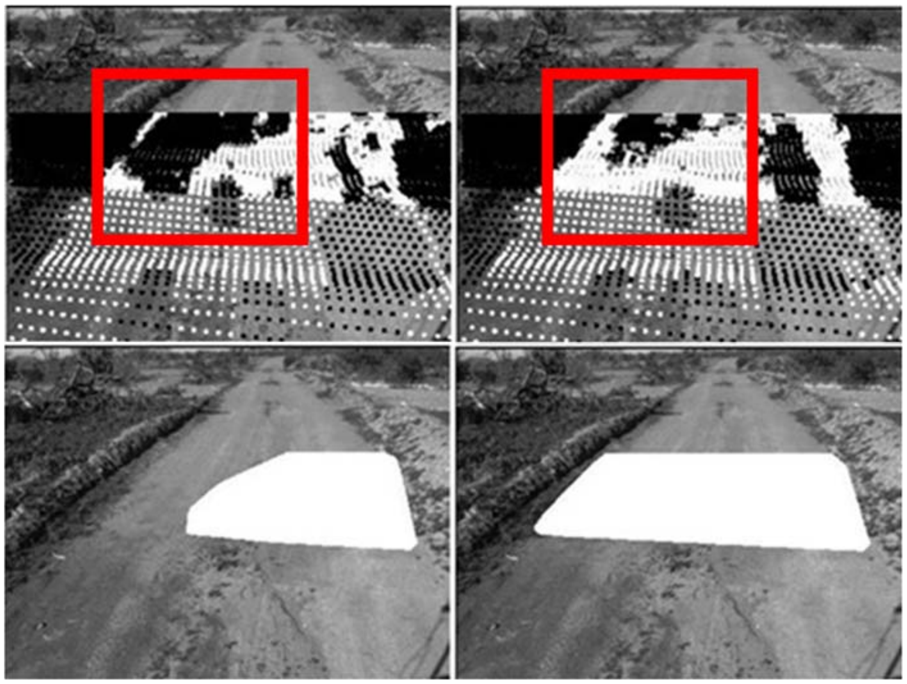


Figure 3-5. The left column is an example of the non-adaptive algorithm and the right column is an example of the reinforcement algorithm. The top row shows the classification results of each algorithm where the white dots are classified as road and the black dots are classified as non-road. The bottom row shows the corresponding road masks.

3.2.Road Model Formation

To detect the road in the image at a consistent distance, a model of a single road was chosen to represent the road center. The road model is a mathematical equation that describes the shape of the road that would fit with the given evidence. In the past, multiple road models have been proposed.[49-51] They essentially are polynomial functions with the look-ahead distance as their variable. To construct a road model for each given image, two processes are implemented: The first process is generating an initial image of the road assessment by finding all the apparent road pixels in the image, and the second process is optimizing the road model, which involves fitting a polynomial curve to the approximated shape of the road in the road assessment image.

3.2.1. Image of Road Assessment

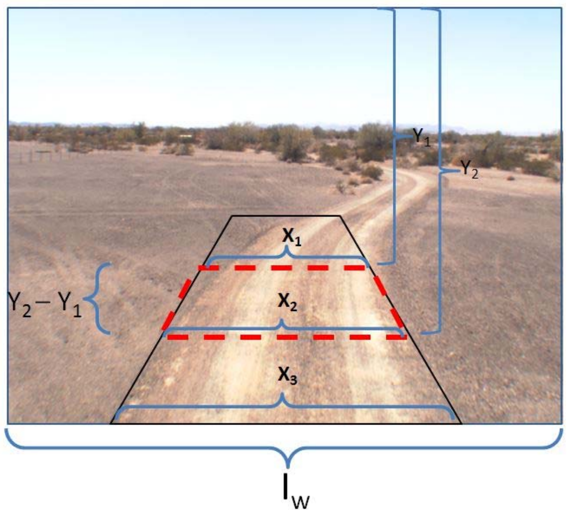
Compared to off-road conditions in the arid environment, the unpaved road shows a strong evidence of passing traffic (such as tire tracks, drag marks, etc.). Therefore, it is sensible to assume that the road area will have a smoother surface than that of the off-road area. We characterize the smooth texture and rough texture in the image as low frequency and high frequency data, respectively. With the use of a log Gabor filter bank, we can separate the low frequency from the high frequency in multiple orientations and scales. In other words, we can separate the road texture from off-road textures. A binary image that represents the approximate road region in a given image, called image of road assessment, can be obtained using the same concept. By filtering a given image with the log Gabor filter bank, road pixels in the image were mostly separated from non-road pixels as indicated by the filter responses. Finally, multiple image processing techniques were performed on the aggregated filter response in order to remove the outliers.

The algorithm for generating the road assessment image for the given image (I_m) is presented below and the outputs are depicted in Figure 3-7.

```
Step 1. During the initialization, generate the LGF filter bank from $N_s$ scales and
        $N_o$ orientations only once; a total of $N_f=N_s \times N_o$ filters will be obtained.
Step 2. If $I_m$ is a color image, convert it to a gray-scale image.
Step 3. Convolve $I_m$ with each filter in the LGF filter bank to obtain filtered
        images, $\{i_f^1, \dots, i_f^{N_f}\}$.
Step 4. Sum the filtered images:
                        $I_f=\sum_{k=1}^{N_f} i_f^k$                    (3-2)
Step 5. Horizontally normalize this result:
                        $I_{Hnorm}=(I_f-I_{Hmin})/(I_{Hmax}-I_{Hmin})$   (3-3)
        where
```

$$I_{Hmin} = \min_{u \in row} (I_f^u) \tag{3-4}$$

$$I_{Hmax} = \max_{u \in row} (I_f^u) \tag{3-5}$$



Step 6. Establish threshold I_{Hnorm} with the dynamic threshold value (DTV). DTV is the approximated percentage of road pixels on each row of an image. With the assumption that the road ahead in the image has the shape of the trapezoid, the DTV is calculated with three simple linear equations as presented below.

$$rate = \left(\frac{X_2 - X_1}{I_W}\right) \div (Y_2 - Y_1) \tag{3-6}$$

$$const = \left(\frac{X_2}{I_W}\right) - (rate \times Y_2) \tag{3-7}$$

$$DTV(Y_n) = (rate \times Y_n) + const \tag{3-8}$$

where:

n represents the arbitrary row in the image,

X_n represents the road width in pixel at the n^{th} row,

Y_n represents the number of rows from the top of the image to the n^{th} row, and

I_W represents the width of the image.

Step 7. Perform morphological image processing to the result image from Step 6 by:

7.1. Eroding the result image with following structure element, .

7.2. Dilating the result image with following structure element, .

`Step 8.` Finally, to generate the image of road assessment, we preserve only the connected component of the output of Step 7 that is most likely belong to a road. The concept is to find a high confidence road area (HCRA) in the image and then preserve the component from Step 7 that connects to that area. In our original work, two versions of HCRA were considered. The first version, which will be referred as a stand-alone algorithm, defines HCRA as a fixed region located near the bottom center of the image (directly in front of the vehicle). For the second version, called the auxiliary algorithm, the output of the road region estimation system described in Section 3.1 is used as the HCRA.

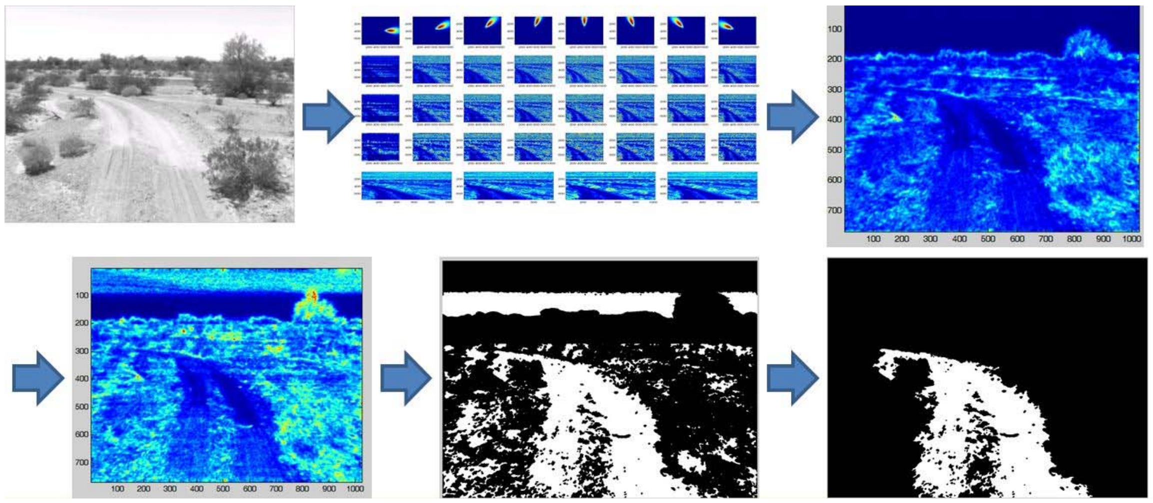


Figure 3-6. The results from each step of the road assessment algorithm. The top middle image shows Step 2-3; the top right shows Step 4; the bottom left shows Step 5; the bottom middle shows Step 6, and the bottom right image is the output of the above algorithm, which is the road assessment image.

3.2.2. Road Model Optimization

The road model is a polynomial function that best fits the image of road assessment, i.e., the output image from the previous section. An example of a road model is shown as the red polynomial curve in Figure 3-7. There are two approaches for generating the road model from the binary image of road assessment. The first approach is to transform the binary image from image space to world space (i.e., UTM coordinates) and then to optimize the coefficients of a polynomial curve (road model) to fit with the road evidence in the transformed image. The second approach is to optimize the road model directly onto the estimated road region in image space and then transform the results to world space. In preliminary experiments, it was found that both approaches yield similar results. Considering that the first approach was more computationally expensive (i.e., it would require the transformation of every pixel in the image of road assessment to world space), the second optimization approach was chosen.

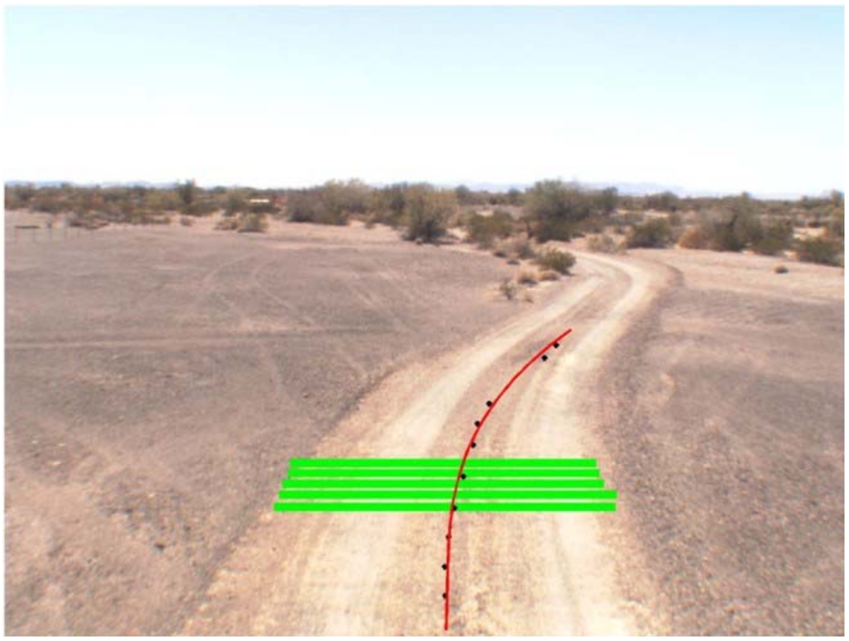


Figure 3-7. The road model (red line) was generated by the combination algorithm. The green lines are road estimation, which are the result from an algorithm in Section 3.1.

The following third degree polynomial function was adopted as the representation of our road model:

$$RM(L) = C_0L^0 + C_1L^1 + C_2L^2 + C_3L^3 \tag{3-9}$$

where:

L represents the look-ahead distance, and

$\{C_0, C_1, C_2, C_3\}$ represent the coefficients that need to be optimized.

The reason that a third degree polynomial was chosen is because both the first and second degree polynomial functions are insufficient to represent the curvature of a real-world road as demonstrated in Figure 3-8, while a polynomial function higher than the third degree is excessive and leads to a greater computational expense.

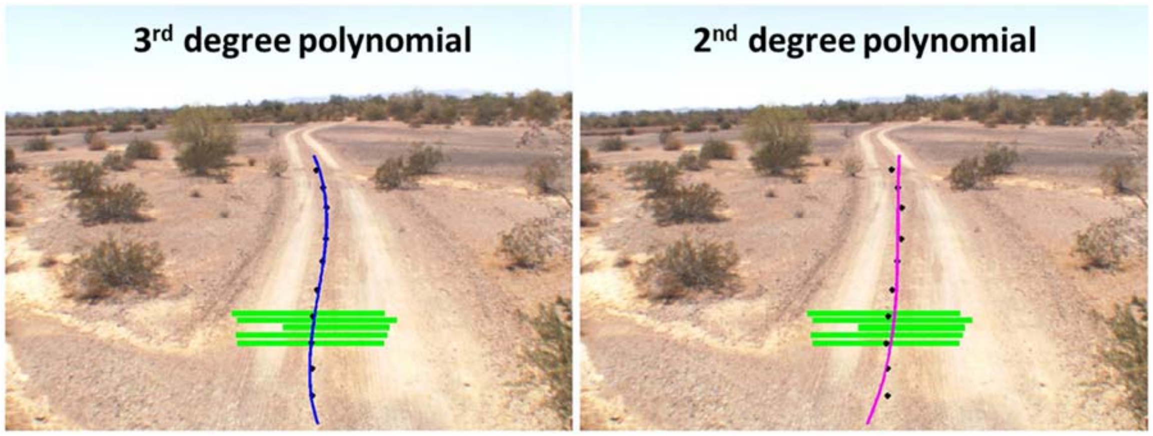


Figure 3-8. An example of a complex scene where the 2nd degree polynomial curve on the right image is inadequate to represent the road curvature, while the 3rd degree polynomial curve on the left image represents the road curvature better.

To optimize the road model, the least squares fitting technique was employed. The objective of the optimization was to minimize the squared distance between the road model and the estimated road region by utilizing the following objective function:

$$\min_{C_k} \| RM(C_k, L) - RR \|_2^2 = \min_{C_k} \sum_i (RM(C_k, L_i) - RR_i)^2 \quad (3\text{-}10)$$

where L_i is the look-ahead distance at ith pixel, and RR_i is the ith pixel in the estimated road region. The two commonly used algorithms for least squares optimization are the

Levenberg-Marquardt algorithm[52] and the trust region algorithm.[53] Both algorithms use the Newton-step method which achieves convergence in quadratic speed. However, with a poor initialization (i.e., when the initial solution is far from the optimal solution), it is possible that the Levenberg-Marquardt algorithm convergence speed becomes dramatically slower than that of the trust region algorithm.[54] Therefore, the trust region algorithm is the preferred method for optimizing the road model.

3.3. Log Gabor Filter Bank Optimization

3.3.1. Log Gabor filter (LGF) bank's parameters

Since a single LGF only covers a specific frequency range and orientation, information extracted from a single filter is likely not effective. A filter bank, which is composed of multiple LGFs, obviously has a larger coverage in both scales and orientations. From the implementation described in Section 2.3.1, the parameters used to generate the LGF bank were formatted to represent a solution (chromosome) in the optimization. The maximum size of the LGF bank was limited to nine scales, where the numbers of orientations in each scale varied from 0 to 8 orientations. However, the radial spread, the angular spread, and the scale spacing remained constant through all the scales. Hence, 12 parameters determined the design of the LGF bank, including the following:

- Parameters 1–9 determined the number of orientations for each scale.
- The 10th parameter determined the radial spread (σ_θ in Equation 2-19).
- The 11th parameter determined the angular spread (σ_ω in Equation 2-19).
- The 12th parameter determined the scale spacing.

3.3.2. Filter Bank Optimization

The design of the LGF bank is data driven, meaning that the composition of a good filter bank is largely dependent on the nature of the data. However, it is difficult to design a filter bank from just a visual inspection of the data. Two common approaches used by researchers are 1) to generate a filter bank with a large number of filters to make sure that its coverage in scales and orientations can extract all the useful information in the data and 2) to create a set of experiments to manually determine the best LGF bank parameters. The advantage of the second approach is its ability to create an efficient filter bank; however, it is labor-intensive. The disadvantage of the first approach is that there are more filters than necessary, which leads to a higher computation cost.

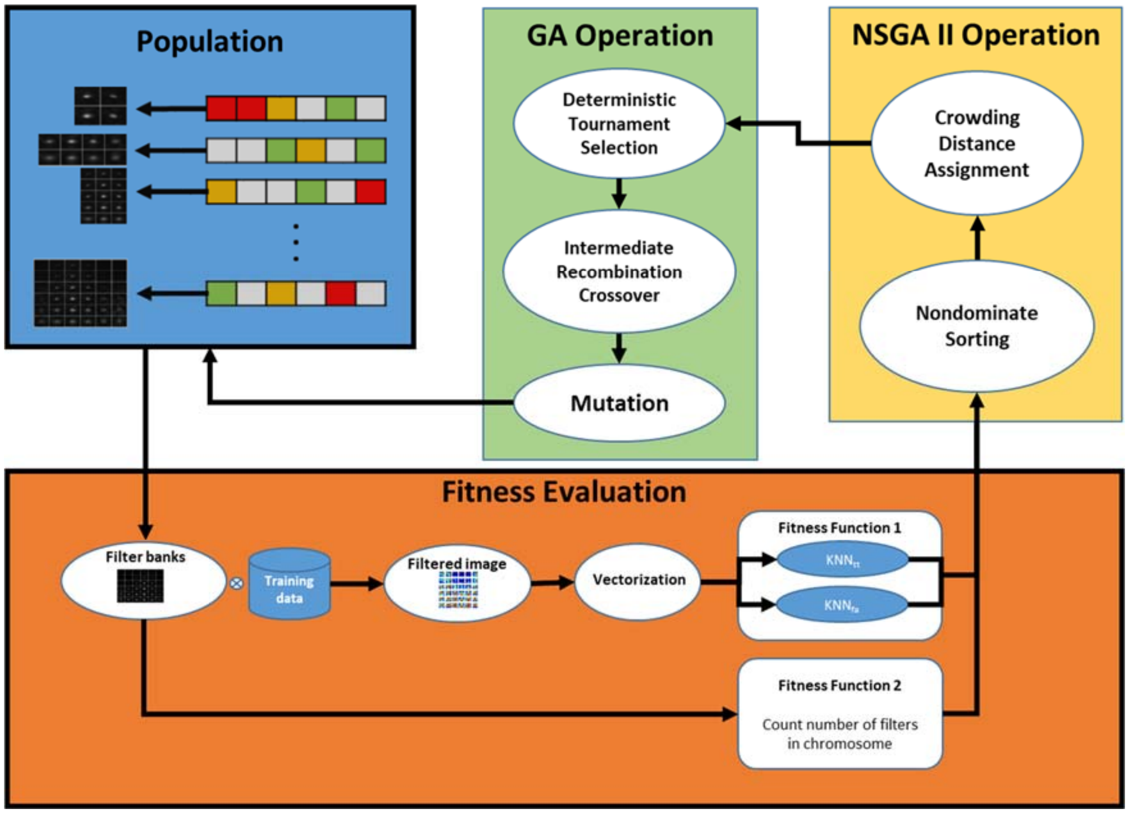


Figure 3-9. NSGA II optimization flowchart for the LGF bank optimization.

Table 3-2. NSGA II Parameters.

Parameter	***Value***
Chromosome	Vector size of 12 elements represents the LGF bank's parameters.
1st Objective	$argmin\ f(x) \coloneqq \sum_{i=1}^{nfa} KNN_{tt}\left(LoG(x, Ic_{fa}^{i})\right) + \sum_{i=1}^{ntt} KNN_{fa}\left(LoG(x, Ic_{tt}^{i})\right)$ where: x = Chromosome, Ic_{tt}^{i} = ith true target image chip, Ic_{fa}^{i} = ith false alarm image chip, ntt = number of true target chip, nfa = number of false alarm chip, $ntt = nfa$, KNN_{tt} = K-nearest neighbor to true target (output in fraction), KNN_{fa} = K-nearest neighbor to false alarm (output in fraction), K = One third of the number of data points, and LoG = Extract LGF features (vectorization).
2nd Objective	$argmin\ f(x) \coloneqq \lvert x \rvert$, where $\lvert x \rvert$ is the number of filter in the filter bank
Population size	100 chromosomes
Crossover function	child = parent1 + random * 0.8 * (parent2 - parent1)
Mutation function	Adaptive feasible[55]
Convergence	$\varepsilon < 10^{-4}$

Based on the second approach, instead of manually determining the best parameters through experiments, a genetic algorithm (GA) based optimization was tried instead. In our application, a multi-objective GA optimization, called NSGA II, was utilized to find optimal filter banks since the two main objectives had to be addressed before we could successfully complete the search. The first objective was to minimize the number of filters in the filter bank; the second objective was to maximize the discriminative power of the filter bank. Figure 3-9 illustrates the NSGA II optimization process of the log Gabor filter bank, and the parameters used in the optimization are shown in Table 3-2. Typically, once the optimization is completed, a set of Pareto optimal filter banks are found and the most suitable filter bank is chosen based on the user's preference. However, in the original study, we analyzed and tested each Pareto optimal filter bank. We found that the performance of each solution corresponded to its objective's ranking, meaning that a smaller filter bank solution (higher second objective ranking) would yield a lower classification rate than a bigger filter bank. Similarly, for this dissertation, we evaluated all the Pareto optimal filter banks and analyzed the trade-off between the detection accuracy and computation cost.

3.3.3. Vectorization

Vectorization is the process of transforming a filtered image into a feature vector. In this study, three vectorization methods were used for each filtered image chip: 1) the calculation of statistical moments, 2) the generation of local binary patterns, and 3) the calculation of the histogram of oriented gradient. The parameters of each vectorization, which are described below, were chosen with an intention to keep the computation cost at a minimum while extracting the necessary information.

- Statistical moments: In order to incorporate spatial information into the feature vector, the filtered images were equally divided into cells. We chose 3 x 3 cells configuration since it had a minimum number of cells while containing a cell at the center of the filtered image, where the most useful information was contained. The mean, standard deviation, skewness, kurtosis, and 2-norm of the intensity values in each cell were calculated. These values were concatenated into a single vector. The resulting feature vector had a size equal to $n_{LGF} \times 9 \times 5$, where n_{LGF} was the number of filters in the bank.

- Local binary patterns: These were obtained by using the rotation invariant LBP described in Section 2.1.1, where the number of neighbors was set to 8. However, unlike statistical moments vectorization, the filtered images were not divided into cells in order to keep the feature vector to a reasonable length. The size of the feature vector was $n_{LGF} \times 59$.
- Histogram of oriented gradients: The filtered images were equally divided into 2 x 2 cells rather than 3 x 3 cells because past experiments have shown that the 3 x 3 cell arrangement was not an improvement over the 2 x 2 cell arrangement, especially when considering that the feature vector length of 3 x 3 cells would be more than doubled. For each cell, the histogram with 18 bins was computed, where the orientation of the gradient ranged from -180 to 180 degrees. The size of the resulting feature vector was equal to $n_{LGF} \times 4 \times 18$.

Chapter 4: Revised Unpaved Road Detection System

In Chapter 3, a prototype unpaved road detection system that used a hand-crafted log Gabor filter bank showed the most success in a 2012 data collection. However, the prototype system struggled to make accurate road detections based on the data collection prepared in 2013. The main reason for this downfall was thought to be the lack of computational intelligence in a generation process of the log Gabor filter bank that was the core component of the road model formation. The original design of filters in a log Gabor filter bank was static in nature; therefore, it was incapable of adapting to the environmental changes in the 2013 data collection. The proposed solution for eliminating this problem was to utilize an optimization during the design process of the log Gabor filter bank. By optimizing the filter bank using a small number of samples from a test dataset, the system was able to generate an optimal log Gabor filter bank, which could adapt to changes in the 2013 dataset. Thus, the system was able to form a road model that correctly and accurately detected the details needed to characterize and analyze the texture properties of the unpaved road.

Figure 4-1 presents a flowchart of the unpaved road detection system and its outputs. The chart shows that the unpaved road detection system consists of two stages: a road region estimation and a road model formation. Both stages are similar to the systems described in the work detailed in Chapter 3. However, in this dissertation, the design of each algorithm is revised to form an improved and cohesive system. The revision is based on the outcomes of the experiments conducted in each system stage.

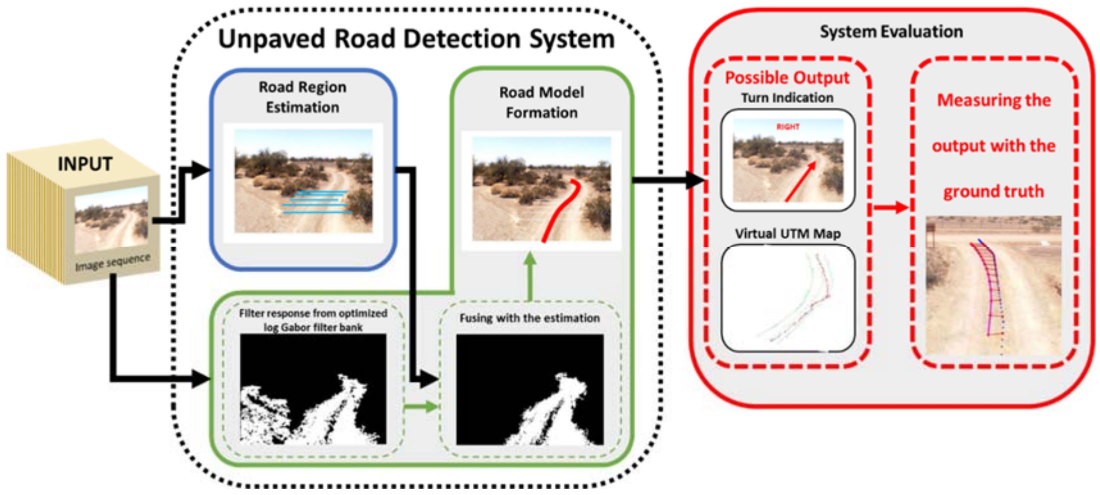

INPUT
Image sequence
Unpaved Road Detection System
Road Region Estimation
Road Model Formation
Filter response from optimized log Gabor filter bank
Fusing with the estimation
System Evaluation
Possible Output
Turn Indication
RIGHT
Virtual UTM Map
Measuring the output with the ground truth

Figure 4-1. Flow of the selected evaluation process for the unpaved road detection system. Three of the project's primary tasks are shown with a specific color outline around each one: 1) The road region estimation outlined in blue, 2) the road model formation flow chart outlined in green, and 3) the system evaluation flow outlined in red.

4.1.Verification Data

The verification data was a color image sequence collection of an arid test site captured by a US Army prototype vehicle, which provided the Universal Transverse Mercator (UTM) system coordinates of the vehicle at the time each frame was taken. This data was used in the experiments for revising the unpaved road detection system as well as for the evaluation of the final system design. The specification of the color image was 8 bits RGB per pixel (an 8-bit byte for each RGB value), i.e., 24-bit color with a resolution of 1024 x 768 pixels. Because the data must represent the environmental diversity of the real-world, the verification data consists of two years' worth of data collection. Each year, the data were gathered on multiple days at different times of the day. There are 16 runs of data collected from seven different lanes denoted as Lanes A through G. Lanes A, B, and C are mock lanes, which are short straight lanes with minimal real-world anomalies in the scene. In contrast, Lanes D, E, F, and G are long curvy roads with many real-world anomalies (i.e., rocks, grass, road wash-outs, etc.) within the lane boundaries, similar to the scene shown in Figure 4-2(a). Based on these characteristics, Lanes A through C will not be used in the evaluation process since they are straight and do not pose any challenges.

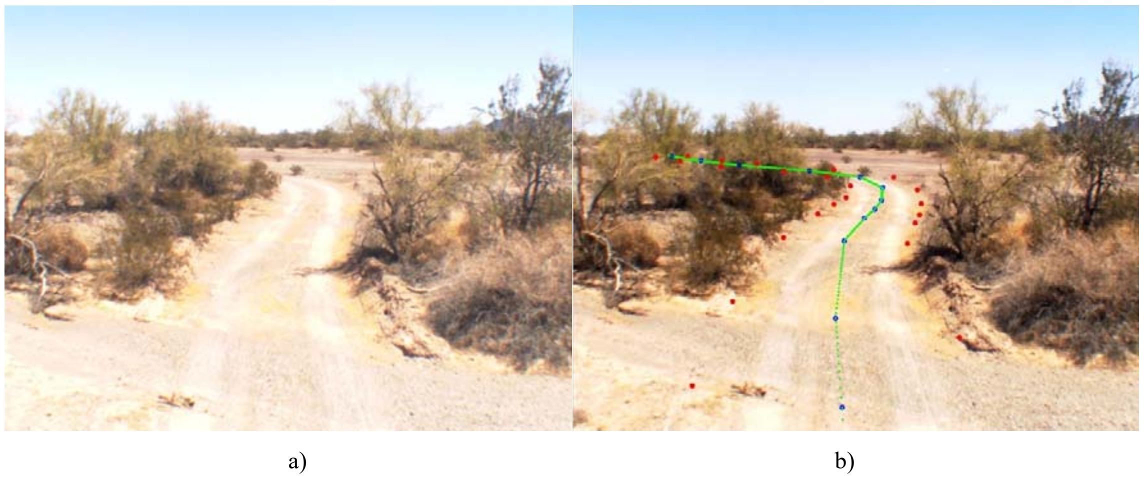


Figure 4-2. Example frames that are within the lane boundary. (a) a road that has been subjected to a wash-out with rocks and a shadow, and b) an illustration of the process of formulating the lane center truth: The red dots are the ground truth provided by the US Army. The blue dots are the lane centers before interpolation, and the green dots represent the interpolated lane center.

Table 4-1 presents details of the verification data distribution to be used in Section 4.2 and Section 4.3 experimentations as well as in the final system evaluation in Chapter 5.

Table 4-1. The distribution of the dataset used in Experiment 1 through 4 and the RURD system evaluation.

Run #	Collection Year	Date-Time	Lane	Image Count
1	2012	05/21/2012 – 08:17:51	**A**	854 images
2		05/21/2012 – 14:15:24	**B**	2,666 images
3		05/16/2012 – 13:41:07	**C**	3,546 images
4		05/17/2012 – 13:10:33	**D**	4,829 images
5		05/23/2012 – 13:14:37	**F**	8,355 images
6		05/23/2012 – 15:26:51	**G**	7,355 images
7	2013	12/05/2013 – 14:27:00	**E**	8,943 images
8		12/11/2013 – 11:23:00	**E**	8,763 images
9		12/11/2013 – 14:01:00	**E**	9,774 images
10		12/05/2013 – 13:57:00	**D**	8,839 images
11		12/11/2013 – 10:56:00	**D**	8,739 images
12		12/11/2013 – 13:29:00	**D**	8,938 images
13		12/11/2013 – 17:13:00	**D**	8,769 images
14		12/06/2013 – 14:27:00	**F**	15,822 images
15		12/04/2013 – 14:02:00	**G**	8,559 images
16		12/06/2013 – 13:39:00	**G**	16,302 images
Total				**131,053 images**

The ground truth of both the left and right boundaries of the lanes was provided by the U.S. Army in UTM coordinates. The rate at which the ground truth coordinates were recorded was approximately one per 11.65 meters. To obtain the lane center truth at multiple look-ahead distances, the lane boundaries truth coordinates were projected to each frame using the method proposed in Dr. Kevin Stone's dissertation,[56] and the lane center is the line generated by linear interpolation between the middle points of each pair of the projected lane boundary truths (Figure 4-2(b).

Finally, significant color distortions were found between each data run since the data collections were taken at different times over a multiple-day span. The color of some data runs are distorted to a point where they are visually unusable. An example of this issue is demonstrated in the top row of Figure 4-3. An image processing technique, called histogram equalization, was used to minimize the distortion. Histogram equalization enhances each image in the data collection by maximizing the image contrast, thereby rendering all images in the collection capable of achieving a similar level of light exposure (not too dark nor too bright). The effect of histogram equalization is shown on the bottom row images of Figure 4-3.

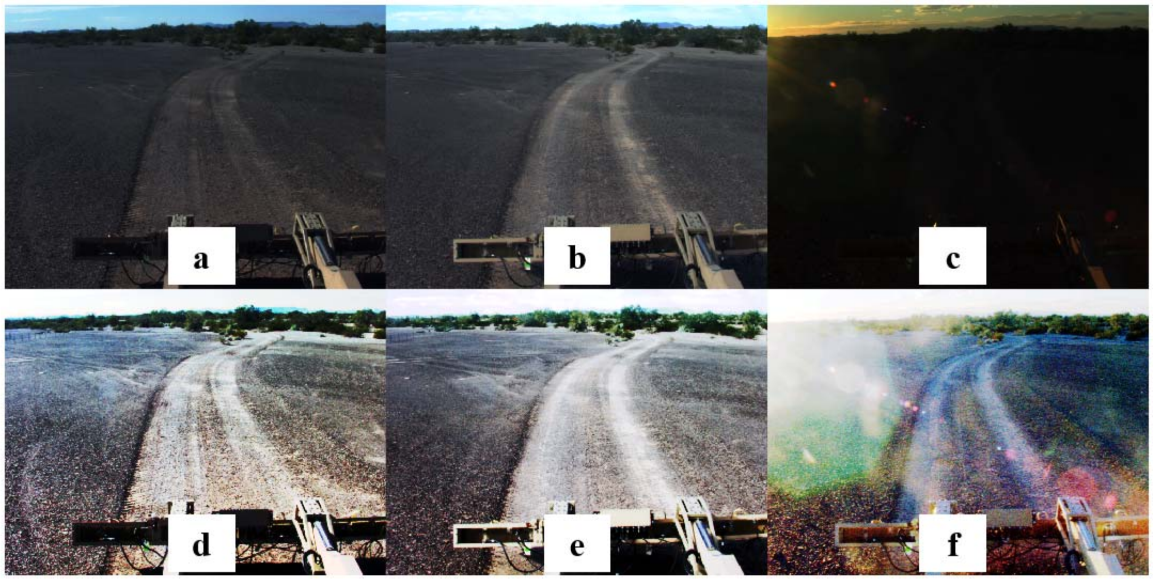


Figure 4-3. These illustrations show the color distortions in the same scene caused by the variations in the time and date that the images were taken; (a) and (b) were taken on different days, while (b) and (c) were taken on the same day but at different times of day. The top row images, (a)–(c), are unprocessed, while on the bottom row (d)–(f) are the corresponding images that have been processed by the histogram equalization method.

4.2 Road Region Estimation Algorithm Revision

In Section 3.1, an algorithm for the Road Region Estimation (RRE) was established. Experimental results from previous research has shown that the algorithm has a robust and adaptable design for detecting the unpaved road in an arid environment. By incorporating reinforcement techniques, the algorithm was intelligently improving itself while processing incoming data. Continued improvement came through the decision fusion classifier approach, which enabled us to control the optimism/pessimism level of the proposed algorithm. However, in the previous work, the notion of processing the data in real-time was not considered, and the validation data was limited to only two runs, which was insufficient to represent diversities in the real-world; thus, the RRE algorithm had to be improved.

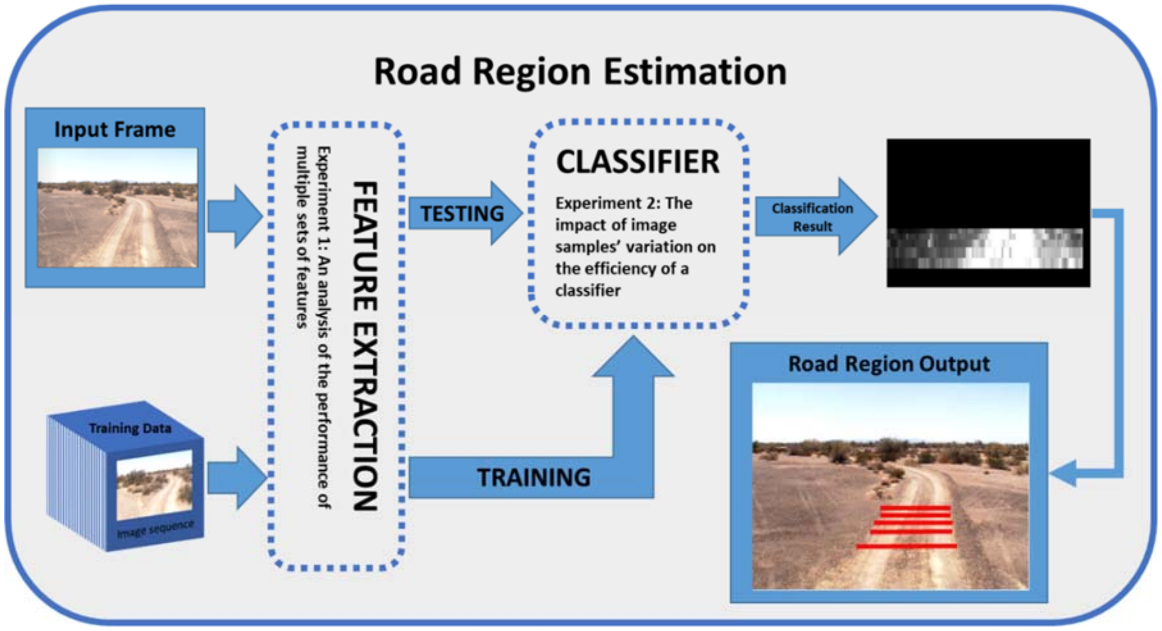

Road Region Estimation
Input Frame
Training Data
Image sequence
FEATURE EXTRACTION
Experiment 1: An analysis of the performance of multiple sets of features
TESTING
TRAINING
CLASSIFIER
Experiment 2: The impact of image samples' variation on the efficiency of a classifier
Classification Result
Road Region Output

Figure 4-4. A brief flow chart showing the road region revision plan, including two experiments—one for selecting an optimal set of features (Experiment 1) and another for tuning the transfer learning variables (Experiment 2).

As shown in Figure 4-4, the revision of the RRE algorithm was based on the conclusive results of two experiments. Experiment 1 determined the optimal parameters for extracting discriminative information from the data. Experiment 2 was set up to fine tune the algorithm during the classification step. Additionally, compared to the work in Section 3.1, the data in Table 4-1 represents both experiments, and is a significantly larger dataset that better represents the diversity of the environment in the real-world. For the record, from this point forward the revised RRE will be referred as RRE.

4.2.1 Data Preparation

The experiments for the RRE algorithm were set up as a classical binary classification machine learning problem, where the algorithm's accuracy was evaluated based on its ability to correctly label data as 'road' or 'non-road'. Therefore, the data images described in Section 4.1 had to be prepared. Each image in the dataset was cropped into smaller image patches and labeled with the name of one of the aforementioned classes. Multiple 'road' and 'non-road' image patches with varying sizes and look-ahead distances were cropped from the dataset so that they were matched with the experiment variables listed in Table 4-3. Data preparation was mainly tailored for Experiment 1, which analyzed the parameters that had an impact on the discriminative power of feature extraction. These parameters included the size of image patches and the look-ahead distance at which the image patches were taken. Figure 4-5 illustrates an example of the locations where the patches of both classes were cropped.

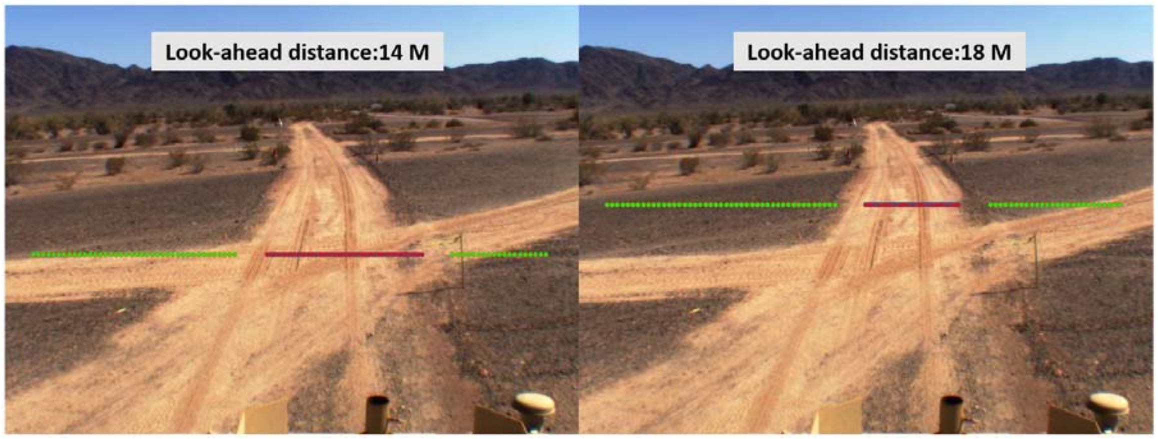


Figure 4-5. An illustration of the locations of image patches at two look-ahead distances, 14 meters (left) and 18 meters (right). Each green dot represents the non-road image patch while each red dot represents the center of the road image patch. At 14 meters, there are 59 green dots and 51 red dots, while there are 71 green dots and 51 red dots at 18 meters.

The numbers of image patches that were extracted from a data image depended on the look-ahead distance since there were more non-road areas than road areas in images with farther look-ahead distances. Thus, the total numbers of image patches extracted for each class varied by the look-ahead distance and these numbers are detailed in Table 4-2.

Table 4-2. Number of road and non-road image patches grouped by look-ahead distances.

Look-ahead Distance	Run#	Lane	Road Patches	Non-road Patches
14 meters	1	A	6,834	7,921
	2	B	21,471	24,876
	3	C	28,560	33,087
	4	D	71,094	85,167
	5	F	69,615	83,372
	6	G	77,775	93,174
	7	E	70,176	84,158
	8	E	69,462	83,166
	9	E	70,992	85,090
	10	D	69,666	83,382
	11	D	38,658	44,820
	12	D	66,912	77,925
	13	D	122,757	149,231
	14	F	58,956	68,828
	15	G	68,595	83,838
	16	G	125,919	152,996
	Sub Total		1,037,442	1,241,031
15 meters	1	A	6,834	8,277
	2	B	21,471	26,047
	3	C	28,560	34,566
	4	D	70,992	88,597
	5	F	69,564	86,755
	6	G	77,673	96,916
	7	E	70,125	87,624
	8	E	69,360	86,498
	9	E	70,890	88,496
	10	D	69,615	86,829
	11	D	38,862	47,143
	12	D	66,810	81,403
	13	D	122,043	154,302
	14	F	58,803	71,638
	15	G	68,340	87,379
	16	G	125,307	158,415
	Sub Total		1,035,249	1,290,885

Table 4-2. Number of road and non-road image patches grouped by look-ahead distances, continued

Look-ahead Distance	Run#	Lane	Road Patches	Non-road Patches
16 meters	1	A	6,834	8,598
	2	B	21,471	27,048
	3	C	28,560	35,890
	4	D	70,992	91,816
	5	F	69,513	89,867
	6	G	77,622	100,361
	7	E	70,125	90,862
	8	E	69,258	89,555
	9	E	70,890	91,780
	10	D	69,615	90,066
	11	D	38,862	48,963
	12	D	66,555	84,199
	13	D	121,380	158,801
	14	F	58,701	74,260
	15	G	68,187	90,349
	16	G	124,593	162,976
	Sub Total		1,033,158	1,335,391
17 meters	1	A	6,783	8,814
	2	B	21,471	27,904
	3	C	28,560	37,101
	4	D	70,890	94,443
	5	F	69,411	92,424
	6	G	77,571	103,339
	7	E	70,023	93,501
	8	E	69,258	92,322
	9	E	70,788	94,462
	10	D	69,615	92,880
	11	D	38,913	50,589
	12	D	66,453	86,783
	13	D	120,615	162,372
	14	F	58,497	76,202
	15	G	68,034	93,065
	16	G	123,879	166,732
	Sub Total		1,030,761	1,372,933

Table 4-2. Number of road and non-road image patches grouped by look-ahead distances, continued

18 meters	1	A	6,783	9,061
	2	B	21,471	28,698
	3	C	28,560	38,191
	4	D	70,788	96,750
	5	F	69,360	94,751
	6	G	77,469	105,855
	7	E	69,921	95,808
	8	E	69,156	94,595
	9	E	70,788	96,986
	10	D	69,462	95,122
	11	D	38,913	52,089
	12	D	66,249	89,055
	13	D	119,646	164,880
	14	F	58,446	78,437
	15	G	67,932	95,354
	16	G	123,165	169,834
	Sub Total		1,028,109	1,405,466
19 meters	1	A	6,783	9,268
	2	B	21,471	29,334
	3	C	28,560	39,039
	4	D	70,737	98,873
	5	F	69,309	96,828
	6	G	77,469	108,276
	7	E	69,870	97,954
	8	E	69,156	96,829
	9	E	70,737	99,157
	10	D	69,360	97,122
	11	D	38,913	53,288
	12	D	65,994	90,663
	13	D	119,136	167,717
	14	F	58,293	79,990
	15	G	67,932	97,477
	16	G	122,502	172,498
	Sub Total		1,026,222	1,434,313

Table 4-2. Number of road and non-road image patches grouped by look-ahead distances, continued

20 meters	1	A	5,729	7,790
	2	B	19,681	26,776
	3	C	26,473	35,952
	4	D	66,805	92,719
	5	F	65,327	90,563
	6	G	73,181	101,497
	7	E	65,419	90,838
	8	E	64,957	90,145
	9	E	66,020	91,656
	10	D	65,003	90,176
	11	D	36,267	49,166
	12	D	61,954	84,517
	13	D	107,092	149,332
	14	F	54,701	74,474
	15	G	61,215	87,408
	16	G	110,418	154,066
	Sub Total		950,242	1,317,076
Total			7,141,183	9,397,095

4.2.2 Experiment 1: Feature Extraction Variables

Experiment 1 analyzed the performance of multiple sets of features with the goal of identifying a feature set that maximized discriminative power while keeping the computational cost to a minimum. Experiment 1 was divided into three smaller analyses, Analysis 1.1, Analysis 1.2, and Analysis 1.3. These smaller experiments used different combinations of experiment variables listed in Table 4-3 and their conclusive results determined the optimal set of features as well as its parameters.

Table 4-3. List of Experiment 1 criteria.

Criteria	Description
Features	Any combination of the following features: • Image histogram (IH) • Local binary patterns (LBP),[12] • Histogram of oriented gradients (HOG),[14] • Gradient location and orientation histogram (GLOH),[15] and • Speeded-up robust features (SURF).[20]
Image patch sizes	Approximately 5%, 7.5%, and 10% of the image's widths, which are 50 x 50 pixels, 75 x 75 pixels, and 100 x 100 pixels, respectively
Look-ahead distance	From 14 meters to 20 meters with an increment of 1 meter

The objective of Analysis 1.1 was to assess the discriminative power of each feature extraction method in conjunction with the size of the image patch. The size of the image patch determined the amount of both the background and foreground information contained in each image patch. This variable had to be assessed along with different feature extraction methods since one method might be sensitive to background information and would, therefore, work better with a smaller image patch while other methods might work best with a larger patch that contained more information. The data used in this experiment were the image patches at a 20 meters look-ahead distance. The 20-meter distance was chosen for two reasons: 1) the data at that distance contained the highest number of image patches, while each image patch covered a greater area in front of the vehicle; thus, the data was more diverse. The validation scheme for Analysis 1.1 was lane-based cross-validation, where the data of each Lane, A to G was individually classified by a random forest (RF) classifier that was trained from the data of the 6 other non-testing lanes. The RF classifier used in this experiment and Analysis 1.2 was an ensemble of 250 random decision trees, where each tree was grown without pruning. The validation calculated the percentage at which the RF classifier could label the testing data samples correctly. To counterbalance the randomness of the RF classifier, each validation was trained and tested 10 times. The validation results detailed in Table 4-4 are the average percentages.

Three conclusions are drawn from Analysis 1.1 results. First, the SURF feature yielded significantly lower classification accuracy, and hence, it is not suitable for this application. Second, IH performed the best and the reason is that the road area appears drastically brighter than the non-road area after data images have been processed by histogram equalization. An example of this effect is shown in the bottom row of Figure 4-3. Lastly, a patch size of 100 x 100 pixels was the best size for feature extraction. This is expected since the largest patch size contains the most information. One drawback of using a larger patch size is the increased computation cost during the feature extraction. However, when measured, the feature extraction time for an image patch size of 100 x100 pixels took 40 milliseconds (ms) per patch. This was an increase of 3.2 and 0.16 ms per patch over the patch size of 50 x 50 pixels and 75 x 75 pixels while the accuracy of the IH feature increased by up to 6.66% and 2.66%, respectively. It was decided that the improvement in classification accuracy justifies the increase in computation time.

Table 4-4. Lane-based cross-validation results comparison between multiple feature extraction methods and varying image patch sizes.

Patch Size	Lane	Feature Extraction Methods				
		HOG	**LBP**	**IH**	**GLOH**	**SURF**
50 x 50 Pixels	**A**	76.56%	69.77%	72.83%	81.20%	63.97%
	B	84.75%	78.25%	77.33%	88.53%	62.55%
	C	64.67%	47.14%	72.28%	68.67%	56.60%
	D	83.89%	86.65%	84.65%	81.50%	61.98%
	E	75.57%	74.55%	77.28%	73.59%	60.48%
	F	73.77%	76.84%	82.36%	71.67%	62.61%
	G	69.04%	72.52%	82.20%	67.51%	61.65%
Average		**75.46%**	**72.25%**	**78.42%**	**76.10%**	**61.41%**
75 x 75 Pixels	**A**	72.36%	66.80%	79.32%	77.11%	63.97%
	B	81.94%	77.69%	85.52%	86.55%	63.14%
	C	59.85%	44.62%	77.13%	63.60%	56.67%
	D	88.66%	90.54%	86.86%	86.11%	61.51%
	E	79.07%	77.36%	79.00%	77.25%	60.34%
	F	67.38%	79.16%	85.04%	74.65%	62.17%
	G	71.59%	73.94%	84.09%	69.83%	61.30%
Average		**75.84%**	**72.87%**	**82.42%**	**76.44%**	**61.30%**
100 x 100 Pixels	**A**	77.23%	62.78%	82.31%	81.63%	65.35%
	B	87.44%	75.84%	92.74%	89.84%	62.95%
	C	64.75%	43.96%	80.02%	67.39%	57.19%
	D	89.49%	92.56%	88.24%	87.31%	61.12%
	E	80.22%	80.47%	80.69%	78.65%	59.93%
	F	68.80%	79.86%	86.66%	67.79%	61.92%
	G	65.28%	74.70%	84.89%	64.68%	61.39%
Average		**76.17%**	**72.88%**	**85.08%**	**76.76%**	**61.41%**

Applying the conclusions from the previous experiment, Analysis 1.2 used the patch size of 100 x 100 pixels for feature extraction. The experiment dropped the SURF feature while focusing on different combinations between IH and other features. The goal of Analysis 1.2 was to determine the combination of feature extraction methods that best improves classification accuracy. The combination of feature extraction methods produced a combined feature vector by concatenating multiple vectors that were extracted using different extraction methods. The combined experiments in Analysis 1.2 were as follows: 1) IH + LBP, 2) IH + HOG, 3) IH + GLOH, 4) IH + LBP + HOG, 5) IH + LBP + GLOH, and 6) IH + LBP + HOG + GLOH. Similar to the previous experiment, Analysis 1.2 used lane-based cross-validation and the validation results show the classification accuracy of each feature combination. The results are detailed in Table 4-5.

Table 4-5. Lane-based cross-validation results comparison between multiple combinations of feature extraction methods.

Lane	Feature Concatenation					
	IH+LBP	IH+HOG	IH+ GLOH	IH+LBP +HOG	IH+LBP+ GLOH	IH+LBP+HO G+GLOH
A	83.86%	83.16%	82.99%	84.59%	85.50%	85.26%
B	95.54%	98.31%	95.84%	96.63%	95.80%	97.01%
C	73.96%	77.14%	78.03%	74.58%	74.53%	74.11%
D	93.89%	91.46%	90.36%	93.96%	93.68%	93.78%
E	82.74%	84.72%	83.94%	82.89%	82.74%	82.67%
F	88.99%	85.88%	85.51%	88.72%	88.48%	88.40%
G	88.30%	85.04%	85.05%	88.05%	88.11%	87.99%
Average	**86.75%**	**86.53%**	**85.96%**	**87.06%**	**86.98%**	**87.03%**

The results in Table 4-5 shown that combining IH with other features increased the classification accuracy by 1.64% on average and the combination of IH + LBP + HOG yielded the largest increase of 1.98%. Also, IH + LBP + HOG + GLOH and IH + LBP + GLOH yielded results similar to IH + LBP + HOG, and their differences were within a margin of error. This indicates that the HOG and GLOH features are redundant to each other. However, HOG is a better feature since it has been well studied and its optimized source code is available on multiple platforms. Therefore, IH + LBP + HOG was chosen to represent the optimal combination of features.

Analysis 1.3 is a two-part analysis that was done to determine the optimal number of classification trees in random forest (RF) and the optimal look-ahead distance that features were extracted from. The first part was to determine the optimal number of trees, which studies the trade-off between maintaining the classification accuracy and decreasing classification cost by reducing the number of trees in RF. The range of the number of trees that would be used in the first part of this analysis was gauged by examining the out-of-bag classification error, which is one of the byproducts of training an RF classifier. Out-of-bag classification errors show that misclassification errors begin to show a trend as the number of trees grow in the RF classifier. Figure 4-6 illustrates an out-of-bag classification error trend that was obtained during the RF training step from Analysis 1.2. The trend in Figure 4-6 shows that the error significantly decreases when the number of trees is above 50. Therefore, the minimum and maximum number of trees used in this experiment were 50 trees and 250 trees, respectively.

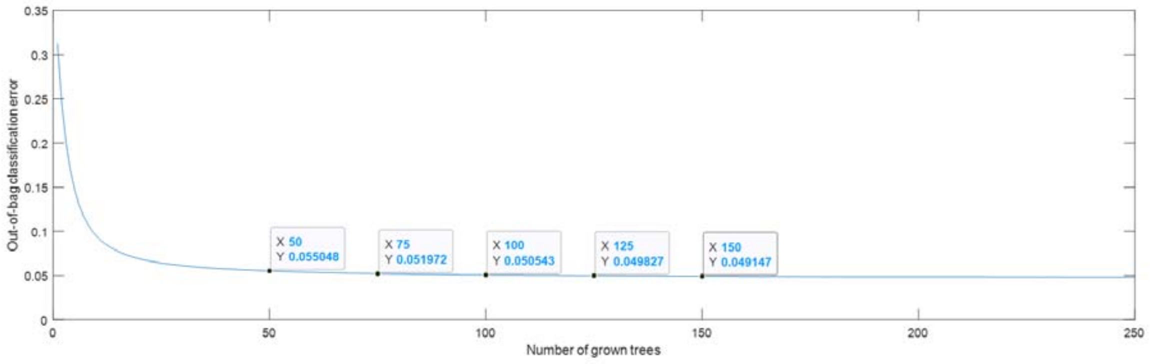


Figure 4-6. An illustration of the Analysis 1.2 out-of-bag classification error trend. It shows a 5.50% error at 50 trees, a 5.19% error at 75 trees, a 5.05% error at 100 trees, a 4.98% error at 125 trees, and a 4.91% error at 150 trees.

The second part of Analysis 1.3 studied the effect that lowering the look-ahead distance would have on classification accuracy. One hypothesis was that lowering the look-ahead distance would increase the resolution of image patches and should have maximized the information gain from feature extraction. This improves the classification accuracy. Using the same validation scheme as that in the previous two analyses, the results of the first and second parts are detailed in Table 4-6 and Table 4-7, respectively.

Table 4-6. The impact number of classification tree reduction has on classification accuracy.

Feature Concatenation: IH+LBP+HOG									
Lane	**Number of trees in Random Forest**						μ	σ	μ - σ
	250	**150**	**125**	**100**	**75**	**50**			
A	84.59%	84.67%	84.78%	84.85%	84.74%	84.46%	84.68%	0.14%	**84.54%**
B	96.63%	96.86%	96.62%	96.95%	96.56%	96.06%	96.61%	0.31%	**96.30%**
C	74.58%	74.38%	74.35%	74.67%	74.55%	73.85%	74.40%	0.29%	**74.11%**
D	93.96%	93.90%	93.90%	93.89%	93.93%	93.76%	93.89%	0.07%	**93.82%**
E	82.89%	82.89%	83.21%	83.20%	82.91%	82.79%	82.98%	0.18%	**82.80%**
F	88.72%	88.65%	88.66%	88.66%	88.58%	88.63%	88.65%	0.05%	**88.60%**
G	88.05%	87.96%	87.90%	87.90%	87.84%	87.86%	87.92%	0.08%	**87.84%**
Average	**87.06%**	**87.04%**	**87.06%**	**87.16%**	**87.01%**	**86.77%**			

Results in Table 4-6 show that reducing the number of trees from 250 trees down to 50 trees causes a slight decrease in classification accuracy. Without statistical analysis, it is visually impossible to determine the optimal number of trees in the classifier. In this dissertation, the optimal number of trees was defined as the least amount of trees that did not weaken the classifier. This weakening classifier was identified as the classifier with a classification accuracy below the accuracy mean minus the accuracy standard deviation of each lane (μ - σ), which in Table 4-6 are shaded in gray. As shown, the results identified 50 and 75 trees in the random forest (tRF) as the weakest classifiers and 100 trees seemed to be the optimal number. However, 75 tRF is not truly the weakest classifier since the results shown that the standard deviation, σ, of Lane F and G is very small when compared to other lanes and the poor classifier performance of 75 tRF on Lane F and G are considered outliers. Moreover, using a processor at a clock speed of 3.4 Ghz, the processing time of 75 tRF was 27 milliseconds (ms) per frame while 100 tRF was 37 ms per frame. By converting the unit from ms per frame to frames per second, it was determined that in one second 75 tRF could process 10 more frames than 100 tRF. After considering both the minimal loss in the classification accuracy and a noticeable reduction in computation cost, 75 was identified as the optimal number of trees for the RF classifier.

Table 4-7. The impact look-ahead distance has on classification accuracy

Lane	Look-Ahead Distance							
	14 M	**15 M**	**16 M**	**17 M**	**18 M**	**19 M**	**20 M**	**σ**
A	72.80%	74.22%	75.57%	77.85%	80.43%	83.05%	84.74%	**4.53%**
B	89.95%	91.95%	92.26%	92.60%	92.76%	93.63%	96.56%	**2.00%**
C	79.99%	77.72%	77.78%	77.32%	76.56%	75.45%	74.55%	**1.77%**
D	93.86%	93.84%	93.99%	94.11%	94.08%	93.99%	93.93%	**0.10%**
E	83.56%	83.42%	82.58%	82.90%	82.79%	83.22%	82.91%	**0.36%**
F	90.42%	90.36%	90.27%	90.13%	89.80%	89.49%	88.58%	**0.66%**
G	87.47%	87.55%	88.14%	88.17%	88.48%	88.71%	87.84%	**0.46%**
Average	**85.44%**	**85.58%**	**85.80%**	**86.15%**	**86.42%**	**86.79%**	**87.01%**	**0.60%**
Average (D to G)	**88.83%**	**88.79%**	**88.75%**	**88.83%**	**88.79%**	**88.86%**	**88.31%**	**0.19%**

Table 4-7 shows that the classification accuracy at further look-ahead distances were higher than the closer look-ahead distances, which contradicted the belief that the closer object (road) was easier to recognize. Upon closer inspection, it became clear that Lanes A and B were the cause of this abnormality. An explanation can be seen in Figure 4-7. Both Lane A and B were a 2-way straight lane, and the oncoming lane in a non-road area had a similar texture to the lane in the road area. The oncoming lane occupied more of the non-road area as the look-ahead distance drew closer, which means more image patches had been extracted from these oncoming lanes and were often misclassified as a road. Hence, lowering the look-ahead distance led to a higher number of misclassified image patches and a lower classification accuracy. Using the average classification accuracy from Lane D to G to determine the optimal look-ahead distance is more suitable since broad straight lanes such as those found in Lanes A, B, and C are arguably not a true representation of the unpaved roads encountered in the real-world, which are usually winding and narrow. Lane D to G represent the averaged classification accuracies shown on the bottom row of Table 4-7. This indicates that the look-ahead distance from 14 M to 19 M yielded almost the same accuracy. The differences are within the margin of error as shown by the standard deviation (σ) was only 0.19%. Hence, the 20 M look-ahead distance had a lower accuracy by approximately 0.5% that was more than 2σ in reduction. Therefore, 19 M was determined to be the optimal look-ahead distance.

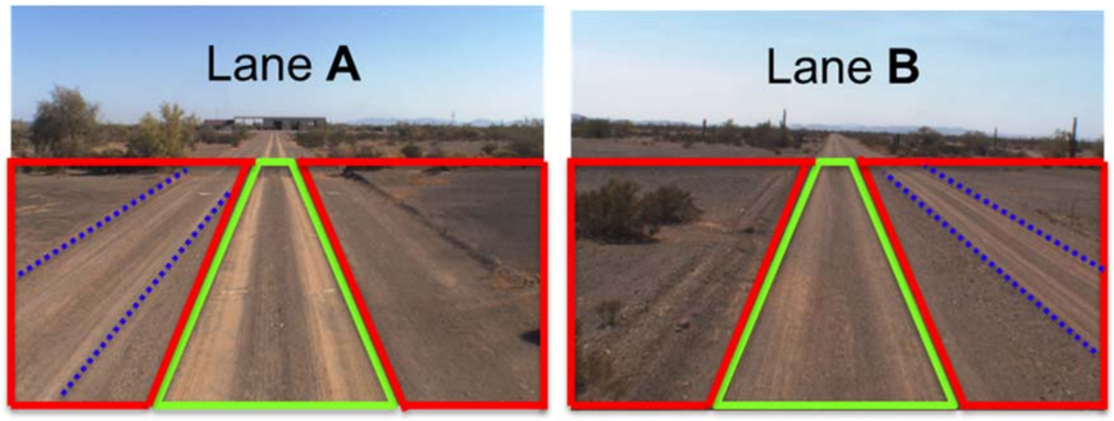


Figure 4-7. Example frames from lane A (left) and lane B (right) with the road area outlined in green, the non-road area outlined in red, and the non-road area, which is similar to the road area outlined in dotted blue lines.

To conclude all three analyses, the optimal parameters for training unpaved road classifiers are derived from the experiment results summarized in Table 4-8.

Table 4-8. Random forest optimal parameters for application of unpaved road detection.

Parameters	Optimal Values
Feature	Concatenation of image histogram, local binary patterns, and histogram of oriented gradients.
Image Patch Size	100 pixels by 100 pixels
Classifier	Random forest is composed of 75 random classification trees.
Look-ahead Distance	19 meters

4.2.3. Experiment 2: Classifier and Transfer Learning Variables

Experiment 2 analyzed the impact the image sample variations had on the efficiency of a classifier and the classification accuracy improvement by utilizing transfer learning. The verification data used in this dissertation were grouped into 16 runs of data and they were expected to be significantly different due to the varied times of day when the data were collected. Hence, it was perceivable that a classifier trained through a traditional validation scheme such as lane-based cross-validation would perform poorly. This issue was solved by utilizing the transfer learning method, a process that uses a small number of target samples such as data samples in the testing dataset to improve the performance of a pre-trained classifier.

Experiment 2 consisted of seven analyses, Analyses 2.1–2.7. These analyses were conducted to meet multiple criteria of Experiment 2, which are listed in Table 4-9. The purpose of Analysis 2.1 through Analysis 2.5 were to maximize the classification accuracy improvement of transfer learning random forest by finding the optimal transfer learning parameters. Meanwhile the purpose of Analysis 2.6 and Analysis 2.7 was to assess the performance of the transfer learning RF classifier by comparing it with two other transfer learning algorithms, the convolutional neural network (CNN) and large margin support vector machine (LMSVM), respectively. Derived from the conclusions of Experiment 1, the data format used in all seven analyses of this experiment was the histogram equalized image patches (size 100 x 100 pixels) that were extracted at the 19 meters look-ahead distance. A validation scheme used on all analyses was identical to Experiment 1, which was the lane-based cross-validation that measured classification accuracy.

Table 4-9. List of Experiment 2 criteria

Criteria	Description
Transfer learning a classifier	The two approaches for adapting random forest using the target samples can be described as follows: • Combine the random forest ensemble with a new set of decision trees that were trained using only the target samples. • Retrain each tree in the random forest ensemble using the mixed information gain[30] approach. The method that produces the highest classification accuracy will be chosen.
Transfer learning parameters	Determine the optimal parameters for successful transfer learning of a random forest classifier.
Benchmarking	The performance of transfer learning random forest will be compared with two other transfer learning algorithms: • the convolutional neural network and[57] • the large-margin support vector machine.[58]

As described in Chapter 2 (Section 2.2), the two transfer learning approaches for random forest were the reinforcement random forest (RRF) and mixed information gain (MIG). RRF was able to learn information from the testing lane by training a new classification tree using several frames of the testing lane that were previously processed by the RRF algorithm and stored in its memory. The new trees were then substituted for poorly performing trees in the random forest (RF) classifier. By design, RRF had three parameters that directly affected its performance, which were 1) the number of frames stored in the RRF algorithm's memory, 2) the number of substitute trees, and 3) the frequency with which the RRF algorithm reinforces the RF classifier. The initial work described in Section 3.1 stated that the RRF algorithm stores two seconds worth of previous frames for training new trees that will substitute 33% of the trees in the RF classifier at every half meter. Since the data were collected at the rate of 24 frames per second and Experiment 1 determined that the optimal RF size was 75 trees, the initial parameters used in Analysis 2.1 to 2.3 are as follow, 1) the RRF algorithm stored 48 frames in the memory, 2) the number of substituting trees was 25, and 3) the reinforcing frequency was every 12th frame.

The first analysis, Analysis 2.1 was conducted to identify the optimal number of frames stored in the algorithm memory. This number determines the amount of information from the testing lane that was used to reinforce the RF classifier since new classification trees were trained from the data stored in the algorithm memory. By setting the number of substituting trees to 25 and the reinforcing frequency to every 12th frame, Analysis 2.1 varied the number of stored frames by starting from 6 frames and then doubling the rate until reaching 96 frames. The results of this analysis are shown in Table 4-10.

Table 4-10. Lane-based cross-validation results from Analysis 2.1. The table shows the average detection accuracy from varying number of stored frames.

Substituting 25 Trees Every 12th Frame						
Lane	**Without Reinforcement**	**Number of Stored Frames (Amount of Reinforce Data)**				
		6	**12**	**24**	**48**	**96**
A	83.05%	96.75%	97.64%	97.60%	97.77%	97.06%
B	93.63%	98.70%	98.84%	98.79%	98.83%	98.58%
C	75.45%	98.26%	98.25%	98.82%	98.72%	98.27%
D	93.99%	95.76%	95.71%	95.65%	95.39%	95.03%
E	83.22%	95.90%	96.06%	96.24%	96.17%	96.14%
F	89.49%	93.50%	94.17%	94.85%	95.27%	95.12%
G	88.71%	92.35%	92.55%	92.99%	93.49%	93.29%
Average	**86.79%**	**95.89%**	**96.17%**	**96.42%**	**96.52%**	**96.21%**

Analysis 2.1 results in Table 4.10 show that on average, using the past 48 frames to reinforce RRF yielded the highest classification accuracy. This result is consistent with our initial work, which indicated that storing less than 48 frames or 2 seconds worth of reinforcing data is insufficient and causes RRF to yield a lower classification accuracy while a higher number of stored frames contains out-of-date information that negatively impacts the RRF accuracy. However, the computation cost of the RRF algorithm is immense and in order to run the algorithm in real-time, it is critical to minimize all possible computations. The RRF algorithm with six frames of memory yielded a 0.63% lower classification accuracy when compared to 48 frames of memory while the amount of data needed to train new trees was approximately eight times smaller. The smaller data size lowers the computation time, which allows the road detection system to utilize additional process to enhance the detection and ultimately increase the accuracy. Therefore, with an emphasis on minimizing the computation time, six frames are the optimal memory size for the RRF algorithm since a reduction of 0.63% in classification accuracy is well worth trade-off.

Next, Analysis 2.2 was conducted to determine the number of trees in the classifier that got substituted each time the RRF algorithm reinforced. In this analysis, the number of stored frames was set to six frames and the reinforcing frequency was set to every 12th frame while the number of substituting trees ranged from three trees to 45 trees. The results are shown in Table 4-11.

Table 4-11. Lane-based cross-validation results from Analysis 2.2. The table shows the average detection accuracy from varying number of substituting trees.

Reinforcing Every 12th Frame Using Data from Six Previous Frames							
Lane	**Without Reinforcement**	**Number of substituting trees**					
		3	**5**	**15**	**25**	**35**	**45**
A	83.05%	95.51%	96.51%	96.22%	96.75%	97.27%	97.50%
B	93.63%	97.40%	97.21%	98.30%	98.70%	98.89%	98.80%
C	75.45%	96.72%	97.48%	98.23%	98.26%	98.20%	98.33%
D	93.99%	91.66%	92.48%	94.52%	95.76%	96.51%	97.03%
E	83.22%	90.97%	91.67%	94.75%	95.90%	96.71%	97.08%
F	89.49%	81.97%	83.57%	90.81%	93.50%	94.05%	93.31%
G	88.71%	85.29%	86.50%	90.52%	92.35%	92.39%	92.20%
Average	**86.79%**	**91.36%**	**92.20%**	**94.76%**	**95.89%**	**96.29%**	**96.32%**

Analysis 2.2 yielded intuitive results where a larger number of substituting trees led to a higher classification accuracy. The results also show that there was a diminishing return as the number of substituting trees increased. A point where improvement in classification accuracy began to diminish can be identified by locating the knee of the curve on a plot showing the average classification accuracy versus the number of the substituting trees. As shown in Figure 4-8, the knee of the curve that represented the number of substituting trees could be either 15 trees or 25 trees. Therefore, both numbers of substituting trees, 15 and 25, would be re-evaluted in Analysis 2.3 to determine which one is the optimal number.

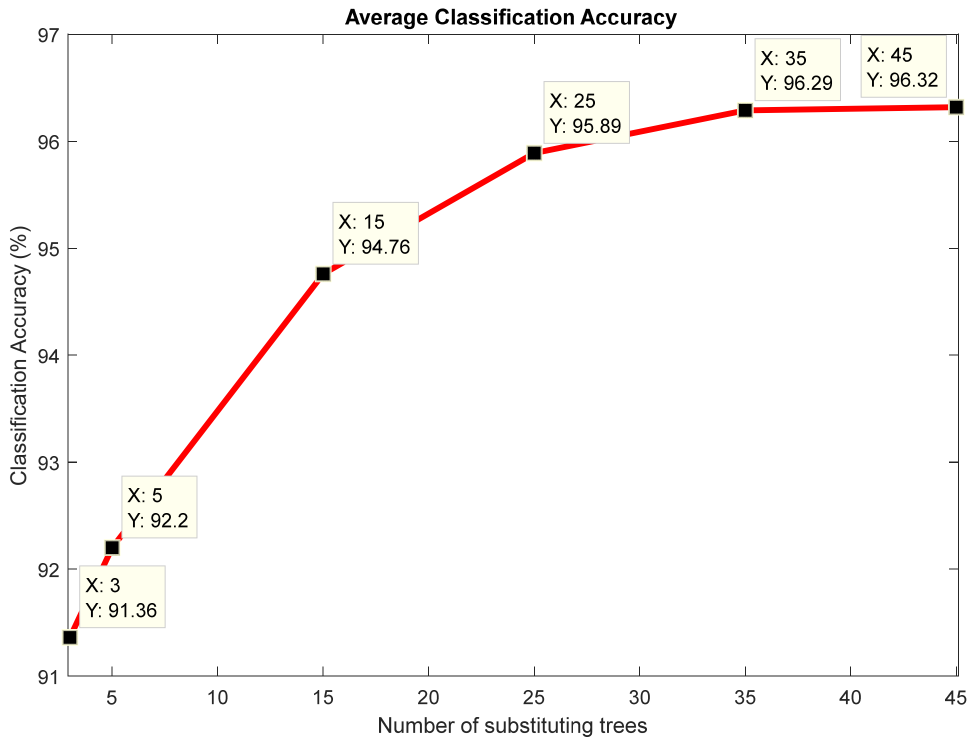


Figure 4-8. A curve showing the average classification accuracy (Y) by the number of substituting tree (X). The curve shows that both X=15 and X=25 can be considered as the knee of the curve.

The last analysis for RRF, Analysis 2.3 represented by Table 4-12, has the objective of determining the optimal reinforcement frequency, which is the number of frames skipped before the RRF algorithm reinforce the RF classifier. For this analysis, the number of stored frames was set to 6 frames and the number of substituting trees was set to both 15 trees and 25 trees. The reinforcement frequencies varied from 6, 12, 18, 30, 42, and 54. The results are shown in Table 4-12.

Table 4-12. Lane-based cross-validation results from Analysis 2.3. The table shows the average detection accuracy from varying reinforcement frequencies with 15 trees substitution (top-half) and 25 trees substitution (bottom-half).

Substituting 15 Trees Using Data From 6 Previous Frames							
Lane	**Reinforcement Frequency (Frames skipped)**						
	6	**12**	**18**	**24**	**30**	**42**	**54**
A	96.83%	96.22%	97.12%	96.87%	96.62%	96.47%	96.08%
B	98.84%	98.30%	98.04%	97.85%	97.56%	97.48%	97.65%
C	98.45%	98.23%	98.23%	97.95%	97.66%	96.76%	96.55%
D	96.37%	94.52%	93.48%	93.04%	92.60%	92.13%	92.07%
E	96.33%	94.75%	93.74%	92.98%	92.22%	91.39%	90.50%
F	93.40%	90.81%	89.06%	87.81%	86.50%	84.02%	82.39%
G	92.07%	90.52%	88.43%	87.66%	86.89%	85.14%	85.18%
Average	**96.04%**	**94.76%**	**94.02%**	**93.45%**	**92.86%**	**91.91%**	**91.49%**
Substituting 25 Trees Using Data From 6 Previous Frames							
Lane	**Reinforcement Frequency (Frames skipped)**						
	6	**12**	**18**	**24**	**30**	**42**	**54**
A	97.60%	96.75%	96.35%	96.70%	97.04%	96.41%	96.79%
B	99.08%	98.70%	98.45%	98.20%	97.96%	97.56%	97.63%
C	98.72%	98.26%	98.35%	98.04%	97.72%	97.93%	97.26%
D	97.38%	95.76%	94.84%	94.01%	93.15%	92.18%	92.07%
E	97.38%	95.90%	94.85%	94.21%	93.56%	92.44%	91.82%
F	94.50%	93.50%	91.82%	90.12%	88.43%	87.21%	85.25%
G	93.41%	92.35%	90.79%	89.79%	88.77%	86.54%	85.46%
Average	**96.87%**	**95.89%**	**95.07%**	**94.44%**	**93.80%**	**92.90%**	**92.33%**

Results of Analysis 2.3 were as anticipated. The RRF algorithm with more frequent reinforcement (lower number of frames skipped) yielded a higher classification accuracy. Reinforcing the classifier every 6th frame seems to best approach. However, a further inspection of the computational time revealed that the RRF algorithm took approximately 209 ms to substitute 15 trees and 270 ms to substitute 25 trees. After considering the additional computation costs from feature extraction and classification, the RRF algorithm with 15 substituting trees could reinforce once every 18th frame (maximum number), while substituting 25 trees can reinforce at most once every 24th frame. Analysis 2.3 indicates that substituting 25 trees with one second’s worth of reinforcing data every 24th frame was the optimal value since it had a higher classification accuracy than that achieved by substituting 15 trees every 18th frame.

The second transfer learning approach for RF was the mixed information gain (MIG). MIG used Equation (2-18) during the training step to incorporate the data from the training lane with a small amount of data from an initial part of the testing lane. This small amount of data was excluded from the validation step. Two analyses were conducted for the second approach to determine the optimal value for MIG's two transfer learning parameters, the amount target data and the weight of information gain (γ).

Analysis 2.4 (with results shown in Table 4.13) inspected the impact that the amount of target data had on MIG classification accuracy. The target data was a small initial part of the testing lane that was used in conjunction with the training data. In this analysis, γ was set to 0.5 and the amount of target data varied from 0% to 5% of the testing lane data.

Table 4-13. Lane-based cross-validation results from Analysis 2.4. The table shows the average detection accuracy from varying amount of target data.

Lane	**Amount of Target Data (% of testing data)**					
	0.0%	**0.1%**	**0.3%**	**0.7%**	**1.0%**	**5.0%**
A	83.05%	85.03%	86.63%	88.58%	88.87%	89.69%
B	93.63%	96.30%	97.06%	97.83%	98.01%	99.51%
C	75.45%	76.57%	80.59%	91.76%	94.23%	94.56%
D	93.99%	93.94%	94.04%	93.93%	93.93%	94.04%
E	83.22%	84.14%	84.07%	83.45%	84.06%	85.82%
F	89.49%	88.24%	88.20%	88.30%	88.44%	87.52%
G	88.71%	88.08%	88.04%	88.06%	88.18%	88.75%
Average	**86.79%**	**87.47%**	**88.38%**	**90.27%**	**90.82%**	**91.41%**

Results from Analysis 2.4 were similar to Analysis 2.2, which indicates that more target data lead to higher classification accuracy, and improvements diminish as the amount of target data increases. A curve representing the average classification accuracy by the amount of target data is plotted in Figure 4-9, which shows that 1% is the knee of the curve. This is the optimal percentage of the testing lane and will be used as target data in the next analysis.

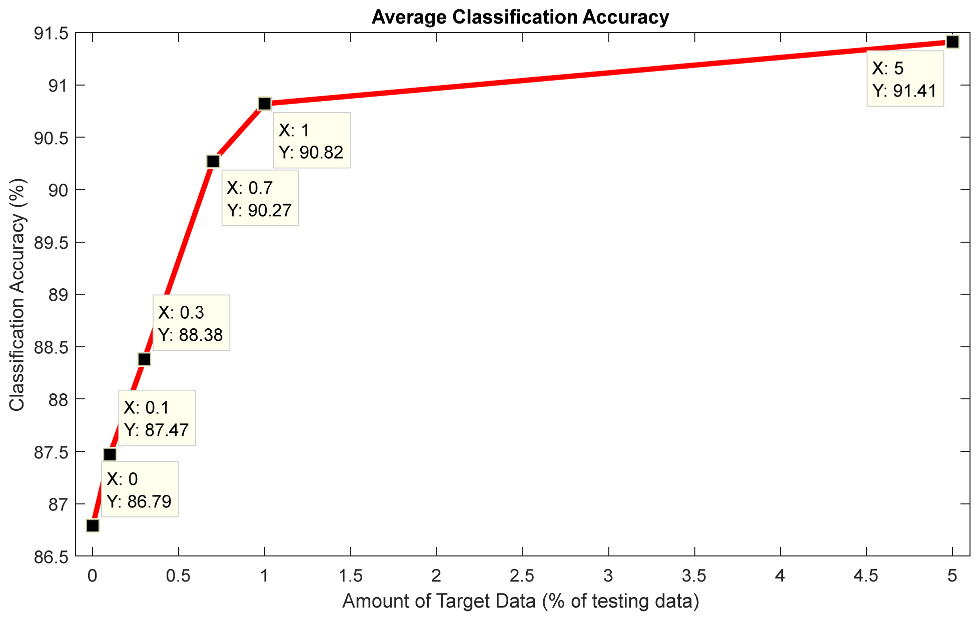


Figure 4-9. A curve of the average classification accuracy determined by the amount of target data. The plot shows that the knee of the curve is located at X=1%.

The last analysis for MIG, Analysis 2.5 (with results listed in Table 4-14), determined the optimal weight of information gain (γ) as described in Section 2.2.3. This analysis set the amount of target data to 1% of data on the testing lane while γ varied from 0.5 to 1.0 with an interval of 0.1. In Table 4-14, the results of this analysis shows the optimal weight of information gain to be 0.8.

Table 4-14. Lane-based cross-validation results from Analysis 2.5. The table shows the average detection accuracy from varying mixed information gain weights.

Lane	Mixed Information Gain Weight (γ)					
	1.0	**0.9**	**0.8**	**0.7**	**0.6**	**0.5**
A	83.05%	88.38%	89.20%	88.68%	88.61%	88.87%
B	93.63%	98.16%	98.01%	98.08%	98.14%	98.01%
C	75.45%	93.84%	94.13%	93.82%	94.23%	94.23%
D	93.99%	93.76%	93.85%	93.90%	93.89%	93.93%
E	83.22%	84.27%	84.28%	84.16%	83.92%	84.06%
F	89.49%	89.03%	88.82%	88.68%	88.58%	88.44%
G	88.71%	88.25%	88.18%	88.29%	88.22%	88.18%
Average	**86.79%**	**90.81%**	**90.93%**	**90.80%**	**90.80%**	**90.82%**

Analysis 2.1 to 2.5 have sequentially determined the optimal parameters for both RRF and MIG. Rearrangement of the experiment sequence can produce a difference set of optimal parameters. However, the most essential factor that determines a success of transfer learning is the quality and quantity of target data (data used for transfer learning a classifier). Setting the experiment for both RRF and MIG to determine this factor first is believed to be the best strategy.

The final two analyses, Analysis 2.6 and 2.7 were designed to benchmark transfer learning random forest against two other established transfer learning algorithms, LMSVM and CNN. Based on the conclusion from previous analyses, the amount of target data used in the transfer learning was the initial 1% of each testing lane.

Analysis 2.6 (with results in Table 4-15) evaluated the performance of the support vector machine (SVM) and large margin support vector machine (LMSVM). Similar to previous analyses, Analysis 2.6 measured the classification accuracy but instead of using the RF classifier, this analysis used SVM and LMSVM classifiers with a radial basis function (RBF) kernel. SVM was analyzed along with LMSVM as a baseline so that the effectiveness of LMSVM in transfer learning could be compared and quantified. Since the gamma parameter for the RBF kernel was an arbitrary number and could be vastly different for each problem, this analysis examined a wide range of gamma from 5 to 150.

Table 4-15. Lane-based cross-validation results from Analysis 2.6. The table shows the average detection accuracy from varying RBF kernel's gamma of SVM (top-half) and LMSVM classifier (bottom-half).

Lane	SVM RBF Kernel							
	5	10	25	50	75	100	125	150
A	57.63%	79.82%	80.93%	89.17%	92.85%	93.69%	94.38%	94.48%
B	57.63%	91.72%	94.84%	95.95%	96.81%	96.88%	96.93%	96.99%
C	57.59%	67.43%	71.40%	69.45%	69.01%	70.03%	71.36%	71.93%
D	61.80%	93.73%	92.62%	90.77%	90.00%	89.65%	89.33%	89.28%
E	59.60%	82.67%	85.84%	85.46%	84.82%	83.99%	83.81%	83.81%
F	66.61%	87.37%	86.72%	85.83%	85.54%	85.25%	84.88%	84.28%
G	63.93%	86.21%	86.91%	86.66%	86.24%	85.76%	84.92%	83.71%
Average	**60.68%**	**84.14%**	**85.61%**	**86.18%**	**86.47%**	**86.46%**	**86.51%**	**86.35%**
Lane	**LMSVM RBF Kernel**							
	5	**10**	**25**	**50**	**75**	**100**	**125**	**150**
A	57.63%	78.99%	80.82%	89.71%	92.90%	93.51%	93.51%	94.35%
B	57.63%	91.42%	94.53%	96.24%	96.91%	96.97%	96.97%	96.57%
C	57.59%	67.03%	71.17%	69.40%	69.03%	69.44%	69.44%	71.98%
D	58.23%	92.90%	91.88%	90.32%	89.61%	89.27%	89.27%	89.12%
E	58.13%	81.45%	85.67%	85.49%	84.60%	84.07%	84.07%	83.82%
F	61.35%	86.78%	86.30%	85.80%	85.27%	84.92%	84.92%	83.24%
G	60.46%	86.08%	86.78%	86.43%	86.04%	85.29%	85.29%	81.86%
Average	**58.72%**	**83.52%**	**85.31%**	**86.20%**	**86.34%**	**86.21%**	**86.21%**	**85.85%**

In Table 4-15, the results from Analysis 2.6 shows 75 as the best value for gamma with minimal performance differences between SVM and LMSVM. The analysis indicated that LMSVM failed to successfully transfer learning the problem of unpaved road detection, which was evident by Lane C's results. Lane C was an ideal indicator for determining the success of a transfer learning algorithm since Lane C had a unique road texture. Figure 4-10 illustrates an example frame from each lane. The figure shows a similarity between Lane A and Lane B, between Lane D and Lane E, and between Lane F and Lane G. Only Lane C had an overall scene that was vastly different from all the other lanes. Therefore, a transfer learning algorithm that achieved a relatively low detection accuracy on Lane C cannot be considered a success.

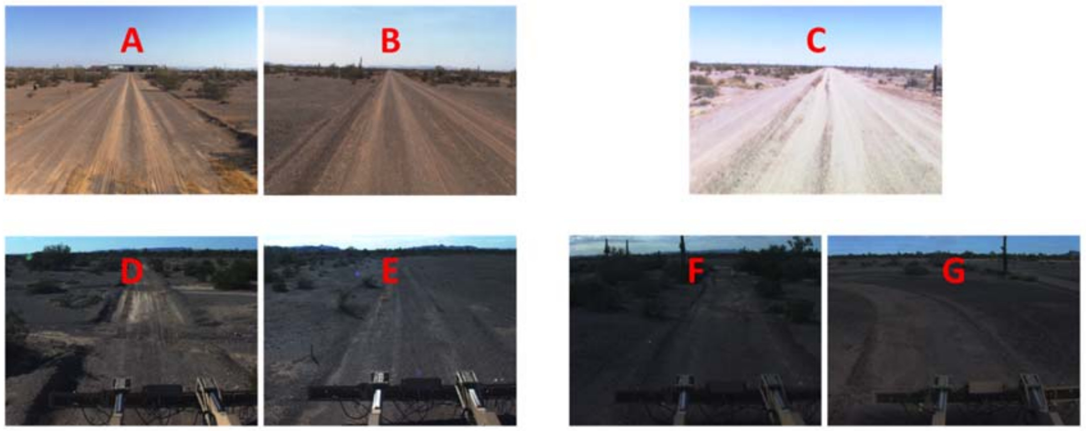


Figure 4-10. A sample frame from each lane, as shown in A to G, demonstrates the significant textural difference between Lane C and all the other lanes.

Lastly, Analysis 2.7 evaluated the transfer learning capability of CNN. Two different CNN transfer learning models were used. The first model was initially trained with training data and transfer learning by retraining the network with the target data. The second model was a transfer learning approach that retrains an established pre-train CNN network such as AlexNet[59] with a combined dataset from both the training data and the target data.

The first transfer learning model examined two variations of CNN architecture, a Shallow CNN[60] and a deep CNN.[61] These two architectures were chosen because they are based on literature[60, 61] that attempts to solve a similar problem, which identified texture anomalies on roads such as road markings or road cracks. The Shallow CNN only has one convolutional layer while the deep CNN has four convolutional layers. The details of each layer for both architectures are illustrated in Figure 4-11.

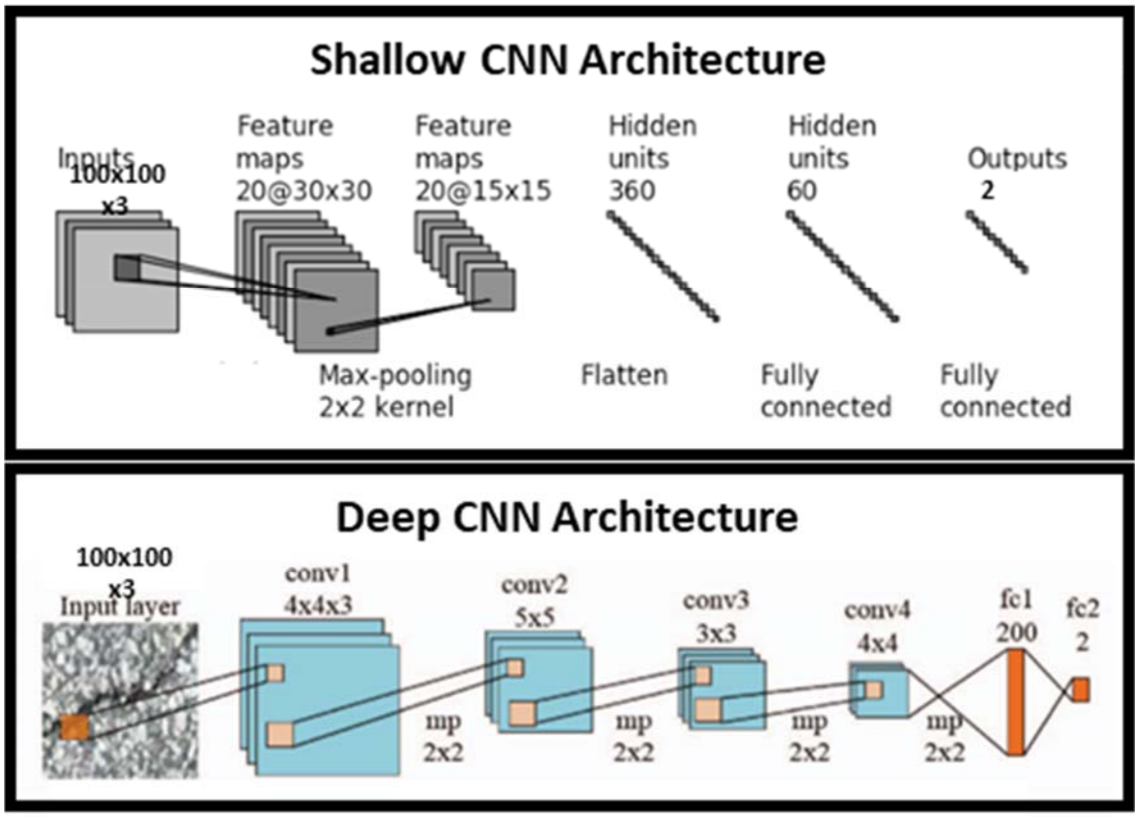


Figure 4-11. An illustration of a shallow CNN architecture (top) and a deep CNN architecture (bottom)

The networks for both architectures were trained until they converged without setting a maximum number of iterations. The networks reached their convergence points when the loss on the validation set was lesser than or equal to the previously smallest loss of the past 50 iterations. Additionally, during the retraining step, the first convolution layer (also known as the feature layer) was frozen. Data augmentations such as reflection, resizing, and rotation were also examined in this analysis. The results of both architectures are shown in Table 4-16, where CNN, TL-CNN, AUG-CNN and AUG-TL-CNN represents a convolutional neural network, transfer learning-convolutional neural network, and data augmentation-convolutional neural network, and data augmentation-transfer learning-convolutional neural network, respectively.

Table 4-16. Lane-based cross-validation results of a shallow CNN and a deep CNN. The table shows the average detection accuracy from multiple combinations of transfer learning (TL) and data augmentation (AUG).

Shallow Convolutional Neural Network				
Lane	**CNN**	**TL-CNN**	**AUG-CNN**	**AUG-TL-CNN**
A	62.20%	70.12%	63.23%	67.97%
B	58.76%	56.47%	63.47%	50.83%
C	58.75%	81.24%	41.61%	71.82%
D	59.22%	68.03%	59.22%	65.47%
E	59.25%	66.29%	64.42%	61.12%
F	64.98%	69.23%	65.64%	68.35%
G	59.15%	69.27%	59.32%	67.88%
Average	**60.33%**	**68.66%**	**59.56%**	**64.78%**
Deep Convolutional Neural Network				
Lane	**CNN**	**TL-CNN**	**AUG-CNN**	**AUG-TL-CNN**
A	77.99%	71.69%	67.17%	68.60%
B	59.29%	58.94%	59.31%	53.73%
C	41.39%	69.10%	41.63%	70.06%
D	71.09%	65.48%	68.15%	64.87%
E	66.52%	64.95%	65.81%	61.90%
F	70.90%	70.55%	67.05%	69.31%
G	72.00%	69.09%	66.58%	69.26%
Average	**65.60%**	**67.11%**	**62.24%**	**65.39%**

For the shallow CNN architecture, the results show successful transfer learning from the first 1% of the testing lane since the TL-CNN classification accuracy of Lane C had a 22.49% increase over the CNN and the average classification accuracy of all lanes increased by 8.33%. However, the data augmentations did not always improve the classification accuracy and severely lowered the accuracy on Lane C by 10% to 17%. As for the deep CNN architecture, the results show that transfer learning increased the accuracy on Lane C by 27.71% while reducing the accuracy on the other lanes. Data augmentation also reduced the accuracy on average by 2% or 3%. Figure 4-12 illustrates the feature layers of both TL-CNN architectures. Further inspection of the figure shows that the shallow CNN attempts to learn texture features while the first layer of the deep CNN learned a simple circular shape or a blob of varied colors intensities and the texture feature was only learned in the second layer. Nevertheless, the results in Table 4-16 prove that both CNN architectures were not suitable for this unpaved road detection problem since the classification accuracies were significantly lower than those produced by transfer learning random forest or support vector machine classification.

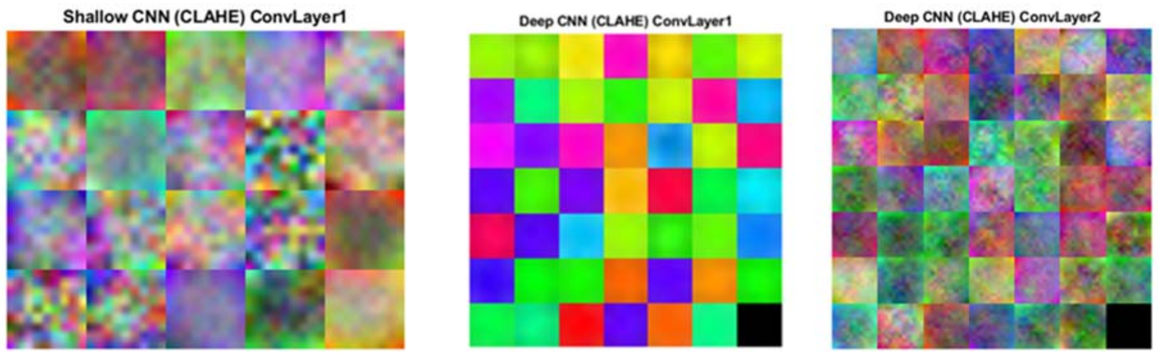

Shallow CNN (CLAHE) ConvLayer1
Deep CNN (CLAHE) ConvLayer1
Deep CNN (CLAHE) ConvLayer2

Figure 4-12. An illustration of the feature layers for both the shallow CNN and the deep CNN.

In the second CNN transfer learning model, the pre-trained network chosen for this analysis was AlexNet.[59] As shown in Figure 4-13, AlexNet is an 8-layer deep convolution neural network trained by the ImageNet[62] database. The input layer of AlexNet was replaced with a 100 x 100 x 3 convolutional layer and the output layer was changed from 1,000 classes to 2 classes.

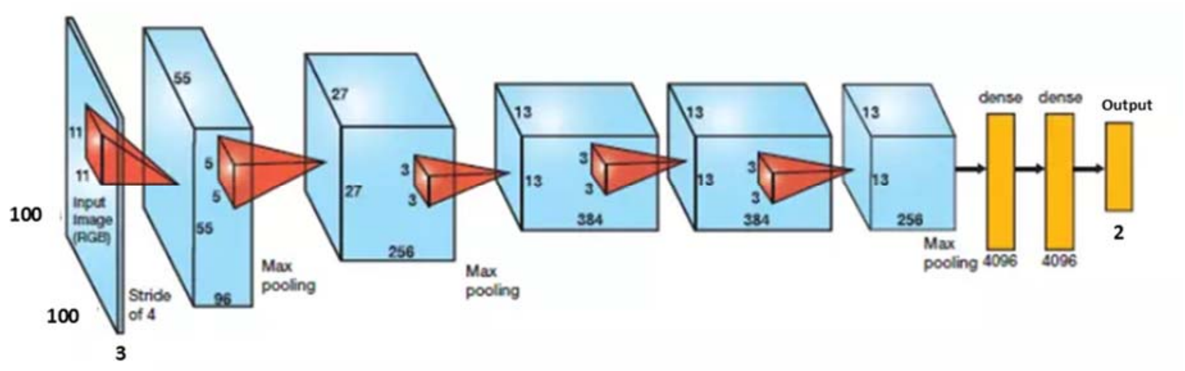


Figure 4-13. An illustration of AlexNet layers that has had both their input and output layer replaced to fit the unpaved road detection problem.

Given that each layer of the pre-trained CNN was learning different features of the image, the impact of freezing varied layers of the network during the retraining phase was first examined. Only the data from the training lanes were used to retrain AlexNet and the results are shown in Table 4-17.

Table 4-17. Lane-based cross-validation results from varying the number of freezing layers.

AlexNet					
Lane	**Number of Freezing Convolutional Layers**				
	1	**2**	**3**	**4**	**5**
A	87.69%	87.87%	87.82%	86.59%	90.23%
B	93.52%	92.16%	91.52%	93.60%	92.72%
C	43.76%	42.65%	42.30%	43.74%	42.93%
D	93.63%	92.58%	92.66%	92.33%	91.42%
E	84.37%	83.77%	83.56%	82.95%	81.96%
F	83.48%	83.40%	82.62%	83.53%	80.82%
G	85.41%	84.61%	84.88%	83.66%	82.22%
Average	**81.69%**	**81.01%**	**80.76%**	**80.91%**	**80.33%**

The results show that AlexNet achieved more than 80% classification accuracy, but the results also show that increasing the number of freezing layers reduces the accuracy. Therefore, in the next assessment, AlexnNet was retrained with the combined data from both training lanes and the initial 1% of the testing lane. Only the first layer was frozen during the retraining process. The results are shown in Table 4-18.

Table 4-18. Comparison of lane-based cross-validation results from AlexNet that was retrained with the training data versus the combined data.

AlexNet: After freezing the first layer		
Lane	**Only Training Data**	**Combined Data**
A	87.69%	89.41%
B	93.52%	93.18%
C	43.76%	82.62%
D	93.63%	93.79%
E	84.37%	84.99%
F	83.48%	86.44%
G	85.41%	86.73%
Average	**81.69%**	**88.16%**

The results show that AlexNet could successfully transfer learning the problem of unpaved road detection since it achieved an 82.64% classification accuracy on Lane C when using the combined dataset. This is a 38.86% increase from using only the data from the training lanes. Also, AlexNet achieved an 88.16% classification accuracy on average with a 6.47% improvement. Despite its success as a transfer learning CNN model, AlexNet still had a lower classification accuracy than both RRF and MIG.

In conclusion, Experiment 2 identified the optimal parameters for the two transfer learning random forest approaches and showed that transfer learning RF is superior to both LMSVM and CNN. Analyses 2.1 to 2.5 show that both RRF and MIG can increase the classification accuracy of the optimized RF classifier by 7.65% and 4.14%, respectively. Both approaches have advantages and disadvantages. RRF produced a classifier that yielded the best classification accuracy at the rate of 96.87%. However, due to RRF's high computation cost, the algorithm cannot reinforce an RF classifier in real-time unless the reinforcing frequency is reduced down to 1 Hz (one reinforcement per second). Still, RRF at 1 Hz reinforcing frequency can produce a classifier with a 94.44% classification accuracy, which is 3.51% higher than the MIG method. The advantage of MIG is that it is able to increase the classification accuracy of an RF classifier with virtually no increase in computation cost. Since the road region estimation process introduced in this dissertation is a fundamental component of the unpaved road detection system and the system requires real-time processing, further discussion is provided in Chapter 5 to determine whether the RRF or MIG approach is the most suitable one.

4.3.Road Model Formation Algorithm Revision

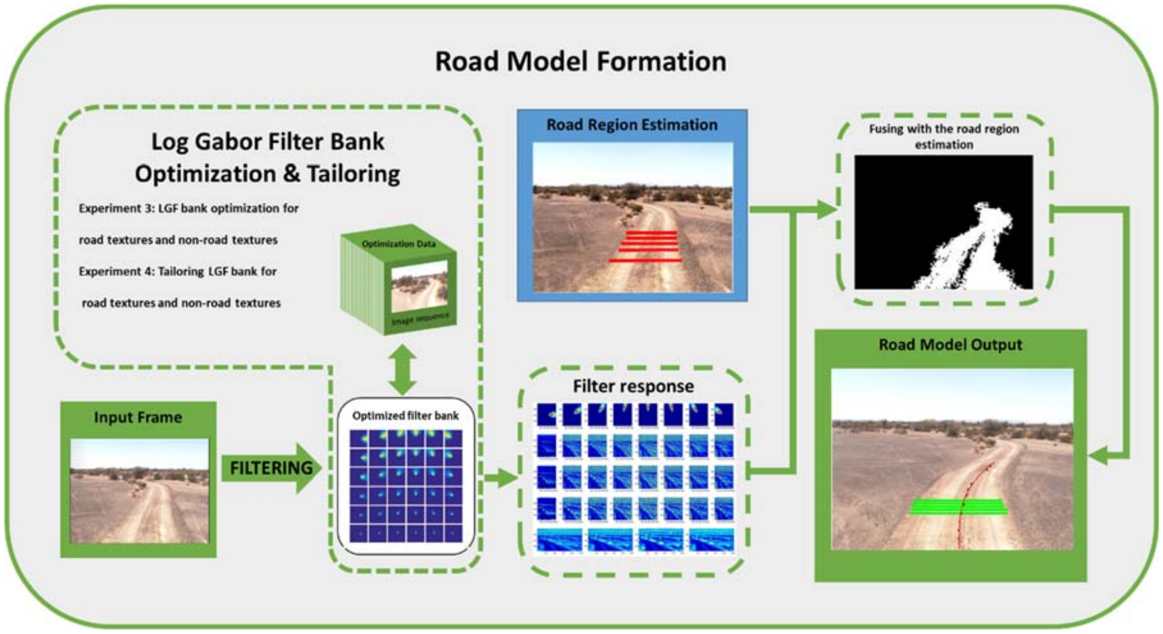

Road Model Formation
Log Gabor Filter Bank Optimization & Tailoring
Experiment 3: LGF bank optimization for road textures and non-road textures
Experiment 4: Tailoring LGF bank for road textures and non-road textures
Optimization Data
Image sequence
Road Region Estimation
Fusing with the road region estimation
Road Model Output
Filter response
Optimized filter bank
Input Frame
FILTERING

Figure 4-14. An overview flowchart of the road model formation revision stage. Experiments 3 and 4 determined the parameters used for generating an optimized and tailored log Gabor filter bank.

As described in Section 3.2, the original road model formation (RMF) algorithm consisted of two steps. The first step was to generate a binary image of road assessment in a given image by using a log Gabor filter bank to filter out the non-road pixels. The second step was to generate a road model by fitting a 3rd degree polynomial curve to the output of the first step. This polynomial curve represents the center of the unpaved road in a given image. However, the original algorithm used a fixed log Gabor filter bank that was only capable of filtering the non-road pixels in one specific run, #4 in the 2012 data collection. Therefore, the revised road model formation algorithm has an adaptable and flexible log Gabor filter bank design, which makes a positive difference. The flowchart in Figure 4-14 shows a total of three experiments that were conducted to finalize the design of the revised road model formation algorithm.

4.3.1. Ground Truth Preparation and Road Model Evaluation

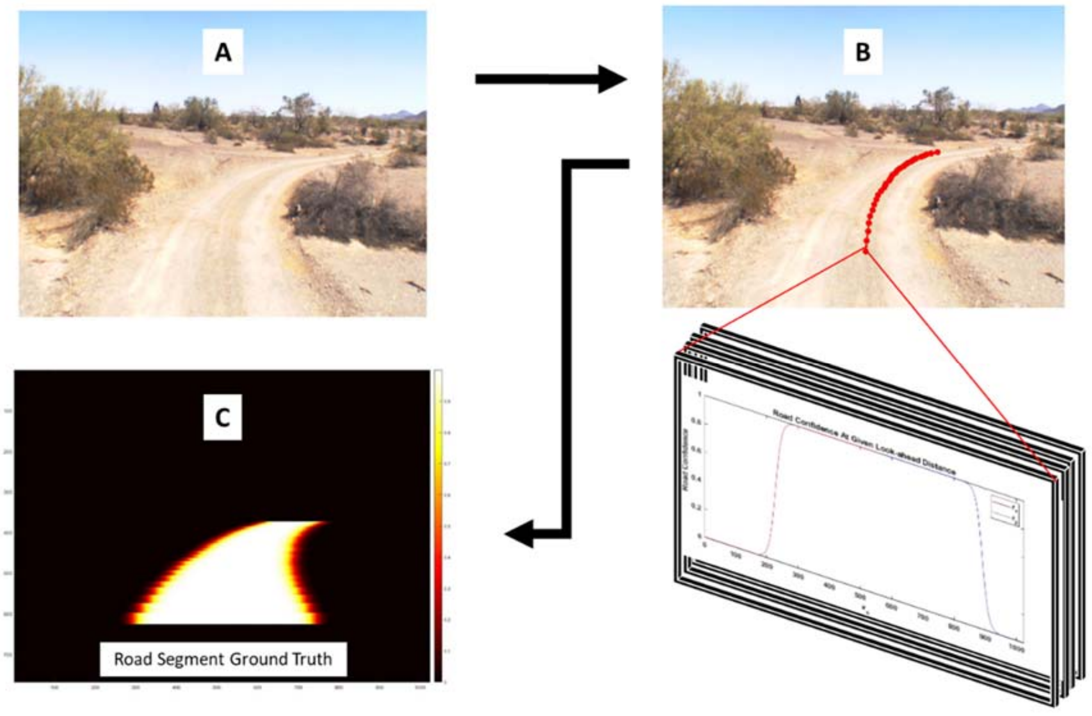

A
B
C
Road Segment Ground Truth
Road Confidence At Given Look-ahead Distance

Figure 4-15. A demonstration of generating a fuzzified road segment from a given frame. A) An example frame, B) UTM coordinates projected onto the frame as a red-dot chain, creating road confidence for each look-ahead distance, and C) all road confidences are consolidated, thereby generating a fuzzified road segment ground truth.

For the next experiments, the verification data from Section 4.1 was divided into two sets of data: 1) the optimization data used to optimize the log Gabor filter (LGF) and 2) the evaluation data used to evaluate the revised RMF algorithm. The optimization data consists of the first 10 frames from each run while the evaluation data covered the remaining frames from each run. Nevertheless, both sets of data have their ground truth for the road segmentation generated from the recorded UTM coordinates of the prototype vehicle. As demonstrated in Figure 4-15, the UTM coordinates were projected to a given frame and used to represent the center of the road. With an unpaved road width estimated at four meters wide, the road segment on each frame was a polygon that had its boundary extended two meters from both sides of the road center. However, because the unpaved road was in an arid environment, it did not have a clearly defined lane marking or boundary; thus, the ground truth of the road segment had to be fuzzified.

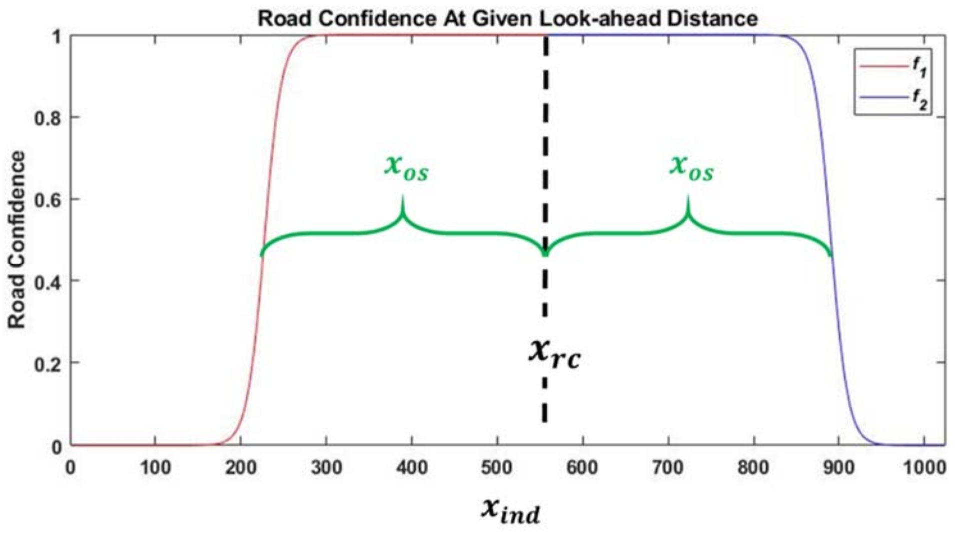
A fuzzified road segment ground truth of a given frame was composed of multiple road confidences. A road confidence represents a level of confidence that a pixel has on an unpaved road at specific look-ahead distances. The look-ahead distances were set to start at 12 meters and increase by 1 meter until reaching 40 meters. Once the road confidences from all 29 look-ahead distances of a given frame were generated, they were consolidated to create a fuzzified road segment ground truth. An example of ground truth is shown in Figure 4-15C. The road confidence at a given look-ahead distance is represented by using two connected Sigmoidal membership functions, fr_1 and fr_2 , which are defined as follows:

$$RC = f_1 + f_2 \tag{4-1}$$

$$f_{1,x_{ind1}}(x_{rc}\ , x_{os}) = \frac{1}{1+e^{-0.1(X_{ind1}-(x_{rc}-x_{os}))}} \tag{4-2}$$

$$f_{2,x_{ind2}}(x_{rc}\ , x_{os}) = \frac{1}{1+e^{0.1(X_{ind2}-(x_{rc}+x_{os}))}} \tag{4-3}$$

where:



RC : the road confidence

f_1: the left side of the road confidence,

f_2: the right side of the road confidence,

$x_{ind1} = \{1, 2, \ldots, x_{rc} - 1\}$, $x_{ind2} = \{x_{rc}, x_{rc} + 1, \ldots, 1024\}$,

$x_{ind} = \{x_{ind1},\ x_{ind2}\}$: pixel indices in x-axis,

x_{rc}: the road center coordinate in the x-axis at a given look-ahead distance, and

x_{os}: the number of offset pixels from the road center that is equal to two meters.

Once a fuzzified road segment ground truth was generated for every frame in the data, the ground truth was used to evaluate a predicted road model. The road model evaluation was conducted at every look-ahead distance. Using the pixel location of the predicted road model in the x-axis (x_{ind^*}), the evaluation calculated the road model error from the ground truth using the following equation:

$$error^{ld} = 1 - RC^{ld}_{x_{ind^*}} \tag{4-4}$$

where:

ld : a given look-ahead distance

RC^{ld} : the road confidence at the look-ahead ld, and

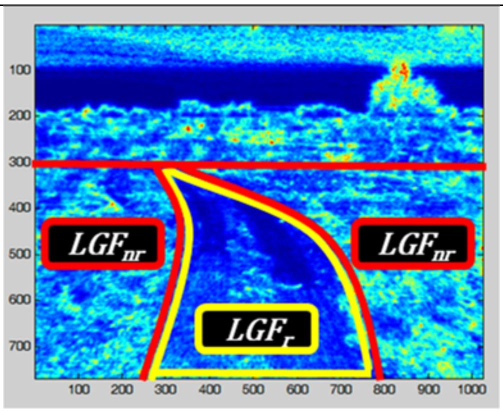
x_{ind^*} : the predicted road model pixel index.

4.3.2. Experiment 3: LGF Optimization

The first half of Experiment 3 was designed to revise the road model formation algorithm by exploring a log Gabor filter (LGF) bank optimization method to effectively filter road textures from non-road textures. The goal of this experiment was to find an optimal LGF bank capable of correctly filtering multiple data collections through an optimization algorithm. Similar to the previous work described in Section 3.3, the NSGA II optimization algorithm was applied to determine what LGF parameters were needed to generate the optimal filter bank. The chromosome used in the NSGA II optimization to represent the LGF parameters was described in Section 3.3.1 while the objective functions (a.k.a. fitness function) are the variable of this experiment, which are provided in Table 4-19.

Table 4-19. Objective functions for NSGA II optimization

Objective Functions	
A	$\begin{aligned} argmin\ f_1(X) &:= -\sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d} Stat\left(LGF_r^i(X,Im_j)\right)-\sum_{j=1}^{N_d} Stat\left(LGF_{nr}^i(X,Im_j)\right)\right\rvert \\ argmin\ f_2(X) &:= N_f \end{aligned}$
B	$\begin{aligned} argmin\ f_1(X) &:= -\bigvee_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_r^i(X,Im_j)\right\Vert_2-\sum_{j=1}^{N_d}\left\Vert LGF_{nr}^i(X,Im_j)\right\Vert_2\right\rvert \\ argmin\ f_2(X) &:= N_f \end{aligned}$
C	$\begin{aligned} argmin\ f_1(X) &:= -\sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_r^i(X,Im_j)\right\Vert_2-\sum_{j=1}^{N_d}\left\Vert LGF_{nr}^i(X,Im_j)\right\Vert_2\right\rvert \\ argmin\ f_2(X) &:= N_f \end{aligned}$
D	$\begin{aligned} argmin\ f_1(X) &:= -\sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_r^i(X,Im_j)\right\Vert_2\right\rvert \\ argmin\ f_2(X) &:= \sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_{nr}^i(X,Im_j)\right\Vert_2\right\rvert \end{aligned}$
E	$\begin{aligned} argmin\ f_1(X) &:= -\sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_r^i(X,Im_j)\right\Vert_2\right\rvert \\ argmin\ f_2(X) &:= \sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_{nr}^i(X,Im_j)\right\Vert_2\right\rvert \\ argmin\ f_3(X) &:= N_f \end{aligned}$
F	$\begin{aligned} argmin\ f_1(X) &:= \sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_r^i(X,Im_j)\right\Vert_2\right\rvert \\ argmin\ f_2(X) &:= -\sum_{i=1}^{N_f}\left\lvert\sum_{j=1}^{N_d}\left\Vert LGF_{nr}^i(X,Im_j)\right\Vert_2\right\rvert \\ argmin\ f_3(X) &:= N_f \end{aligned}$



where

$\boldsymbol{LGF^i_r}$ represents the pixels within the road segment in the ith filter response,
$\boldsymbol{LGF^i_{nr}}$ represents the pixels within the non-road segments in the ith filter response,
$\boldsymbol{N_f}$ represents the number of filters in the log Gabor filter bank,
$\boldsymbol{N_d}$ represents the number of images in the optimization dataset,
$\boldsymbol{Im_j}$ represents the jth image in the optimization dataset,
$\boldsymbol{X}$ represents the parameters of the log Gabor filter bank, and
$\boldsymbol{Stat(f)}$ represents a summation between the mean of $\boldsymbol{f}$ and the standard deviation of $\boldsymbol{f}$,
$\boldsymbol{Stat(f) = \mu_f + \sigma_f}$

In the first half of this experiment, the NSGA II used the optimization data to determine LGF parameters for each set of objective functions listed in Table 4-19. The optimization was individually run from Run #1 in the optimization data to Run #16. The solutions from the optimization output of Run #1 are shown in Figure 4-16 where the solution chosen from Pareto fronts are circled in black. The solutions of the other 15 runs are shown in Appendix A. For the objective functions set A to C, a solution was chosen by prioritizing the first objective to emphasize the LGF discriminative power rather than the filter bank's size. However, for the objective functions set from E to F, the first two objectives were weighted equally because both objectives were designed to measure the discriminative power of each solution. Finally, a total of 96 LGF banks were generated from the chosen solutions (optimized LGF parameters), with one bank for each data run and each set of objective functions. The specification of these LGF banks is shown in Table 4-20.

Table 4-20. Optimized LGF bank specifications grouped by each set of objective function.

Objective Function Set	Run #	LGF Parameters												Number of filters
		The number of orientation on each scale:									σ_θ	σ_ω	Scale Space	
		1	2	3	4	5	6	7	8	9				
A	**1**	6	3	4	7	7	6	7	7	7	1.291	3.162	1.054	**54**
	2	4	6	7	7	7	6	6	6	6	3.162	3.162	1.291	**55**
	3	6	6	5	5	2	4	5	5	5	2.236	3.162	1.118	**43**
	4	2	6	7	8	7	8	8	8	7	3.162	1.000	1.054	**61**
	5	5	5	3	4	7	8	8	8	8	3.162	3.162	0.816	**56**
	6	5	7	7	8	7	8	4	7	7	3.162	0.767	1.118	**60**
	7	5	6	7	8	8	4	4	5	5	2.236	3.162	1.195	**52**
	8	5	8	8	8	3	4	4	4	5	2.236	3.162	1.118	**49**
	9	7	4	4	4	5	5	5	5	5	3.162	3.162	1.291	**44**
	10	5	6	7	8	7	8	8	4	4	1.581	3.162	1.000	**57**
	11	6	8	8	5	6	8	3	4	5	1.826	3.162	1.118	**53**
	12	7	7	5	4	6	5	5	5	5	3.162	3.162	1.291	**49**
	13	7	6	6	6	7	6	7	7	7	3.162	3.162	1.195	**59**
	14	6	7	8	7	8	6	4	5	5	1.826	3.162	1.118	**56**
	15	7	7	3	4	5	5	6	6	5	3.162	3.162	1.195	**48**
	16	5	6	5	6	4	3	5	5	5	2.236	3.162	1.118	**44**
B	**1**	0	0	1	1	0	1	1	2	1	0.845	3.162	3.162	**7**
	2	1	0	0	0	1	0	1	1	1	1.195	3.162	2.236	**5**
	3	1	1	1	1	0	0	2	2	1	1.414	3.162	3.162	**9**
	4	0	0	0	0	1	0	3	2	2	0.520	3.162	2.236	**8**
	5	1	0	0	1	1	0	5	2	5	0.375	3.162	1.826	**15**
	6	0	0	0	1	1	2	1	2	2	0.370	3.162	2.236	**9**
	7	0	0	0	1	0	1	0	1	3	1.826	3.162	3.162	**6**
	8	0	0	0	1	0	0	0	0	0	1.581	3.162	3.162	**1**
	9	0	0	0	1	0	0	0	0	0	3.162	3.162	3.162	**1**
	10	0	1	0	1	0	1	0	0	0	1.195	3.162	1.581	**3**
	11	0	1	0	1	1	0	0	0	2	1.291	3.162	3.162	**5**
	12	0	1	0	1	0	1	1	1	1	0.745	2.236	3.162	**6**
	13	0	0	0	1	1	0	1	2	1	1.054	3.162	2.236	**6**
	14	1	1	1	1	1	1	1	1	1	1.195	3.162	3.162	**9**
	15	0	0	0	0	1	0	0	0	0	2.236	3.162	1.826	**1**
	16	0	0	0	0	1	0	1	1	1	2.236	3.162	2.236	**4**

Table 4-20. Optimized LGF bank specifications grouped by each set of objective function, continued

Objective Function Set	Run #	LGF Parameters												Number of filters
		The number of orientation on each scale:									σ_θ	σ_ω	Scale Space	
		1	2	3	4	5	6	7	8	9				
C	**1**	6	3	4	7	8	8	8	7	7	1.291	3.162	1.054	**58**
	2	5	7	7	8	6	6	6	6	6	3.162	3.162	1.291	**57**
	3	6	7	4	4	6	5	5	6	5	3.162	3.162	1.291	**48**
	4	1	5	8	7	8	7	8	7	7	3.162	0.913	1.000	**58**
	5	1	2	7	7	8	8	8	8	8	3.162	3.162	0.845	**57**
	6	1	5	8	6	7	7	7	7	7	3.162	0.767	1.000	**55**
	7	8	7	3	3	4	5	5	5	6	3.162	3.162	1.291	**46**
	8	8	7	8	3	5	5	5	5	5	3.162	3.162	1.414	**51**
	9	8	6	7	3	5	5	5	6	5	3.162	3.162	1.291	**50**
	10	4	6	6	5	8	8	4	5	5	1.826	3.162	1.118	**51**
	11	7	6	5	1	4	5	5	5	5	3.162	3.162	1.291	**43**
	12	6	5	2	6	6	5	6	6	5	3.162	3.162	1.291	**47**
	13	5	7	4	5	8	6	6	6	8	3.162	3.162	1.291	**55**
	14	5	8	7	8	7	4	5	6	6	2.236	3.162	1.195	**56**
	15	5	4	3	5	6	6	7	7	7	3.162	3.162	1.291	**50**
	16	5	7	6	3	5	6	6	6	6	3.162	3.162	1.291	**50**
D	**1**	2	2	3	3	3	2	2	2	3	1.195	1.291	0.674	**22**
	2	3	2	3	3	1	2	3	3	5	0.645	2.236	0.877	**25**
	3	1	2	0	0	1	1	2	3	2	1.414	3.162	1.195	**12**
	4	0	2	1	0	0	0	1	2	2	1.118	2.236	0.632	**8**
	5	0	1	3	0	1	2	2	4	3	1.414	1.826	0.707	**16**
	6	0	0	2	0	0	1	0	0	1	1.826	2.236	3.162	**4**
	7	0	2	0	0	2	2	2	4	3	2.236	2.236	1.581	**15**
	8	1	2	0	1	1	2	2	2	2	1.414	2.236	1.414	**13**
	9	2	1	1	1	2	2	3	3	1	1.581	1.826	1.581	**16**
	10	1	1	2	0	0	2	1	3	2	1.581	2.236	1.291	**12**
	11	1	1	0	0	2	1	2	2	1	1.826	2.236	1.581	**10**
	12	1	0	1	1	2	0	1	1	2	0.645	2.236	1.826	**9**
	13	2	1	0	1	3	1	1	2	2	1.581	1.414	0.690	**13**
	14	3	2	0	1	2	7	2	2	3	0.419	0.674	1.291	**22**
	15	1	0	0	2	0	1	0	2	2	0.953	1.826	2.236	**8**
	16	1	0	1	2	1	0	2	1	1	1.054	2.236	3.162	**9**

Table 4-20. Optimized LGF bank specifications grouped by each set of objective function, continued

Objective Function Set	Run #	LGF Parameters												Number of filters
		The number of orientation on each scale:									σ_θ	σ_ω	Scale Space	
		1	2	3	4	5	6	7	8	9				
E	1	2	3	2	3	2	1	3	5	5	0.953	1.581	0.745	**26**
	2	1	0	0	1	2	1	1	2	1	1.826	0.845	0.707	**9**
	3	0	1	1	2	1	1	0	1	0	1.195	1.581	2.236	**7**
	4	0	1	0	1	1	1	1	1	1	2.236	0.953	0.609	**7**
	5	0	3	1	1	0	0	0	2	0	3.162	2.236	2.236	**7**
	6	0	0	0	1	2	2	2	1	2	1.826	1.291	1.581	**10**
	7	0	0	0	0	1	0	2	0	0	1.118	2.236	2.236	**3**
	8	0	1	1	0	1	2	1	1	1	2.236	1.581	1.581	**8**
	9	0	0	0	0	1	0	0	0	0	3.162	3.162	3.162	**1**
	10	0	0	0	1	1	0	0	0	0	1.414	2.236	2.236	**2**
	11	1	0	0	1	1	2	1	1	1	2.236	3.162	1.581	**8**
	12	1	1	1	1	2	1	1	1	0	3.162	2.236	3.162	**9**
	13	2	1	1	1	2	0	1	1	1	1.414	1.581	0.877	**10**
	14	0	0	0	2	1	1	1	0	3	1.826	2.236	2.236	**8**
	15	0	0	0	0	1	1	0	1	0	1.826	2.236	1.826	**3**
	16	0	0	0	2	0	1	1	1	3	1.000	2.236	3.162	**8**
F	1	2	3	1	1	1	0	1	2	3	1.195	2.236	1.118	**14**
	2	0	1	1	0	0	0	0	0	1	2.236	3.162	1.195	**3**
	3	1	2	0	0	1	2	2	1	2	1.581	2.236	0.542	**11**
	4	0	0	0	0	2	0	0	2	2	0.645	2.236	2.236	**6**
	5	2	2	1	1	1	1	1	3	2	1.414	1.581	0.913	**14**
	6	0	0	1	1	1	1	2	0	3	0.477	1.581	3.162	**9**
	7	0	1	1	1	1	1	1	2	2	1.581	2.236	0.527	**10**
	8	2	1	2	1	1	1	1	3	1	1.826	2.236	0.482	**13**
	9	0	1	1	0	1	1	1	1	0	1.826	2.236	0.550	**6**
	10	0	1	1	1	2	1	1	2	1	1.581	1.581	0.620	**10**
	11	0	0	1	0	1	0	1	2	0	0.767	0.953	3.162	**5**
	12	1	2	1	2	1	1	2	1	1	1.581	2.236	0.632	**12**
	13	0	0	0	2	0	2	1	0	1	0.791	1.414	2.236	**6**
	14	2	1	2	1	1	1	1	3	2	1.826	1.581	0.598	**14**
	15	1	1	1	1	3	1	1	1	1	1.581	1.826	0.609	**11**
	16	1	0	1	1	2	1	1	2	2	1.826	2.236	0.506	**11**

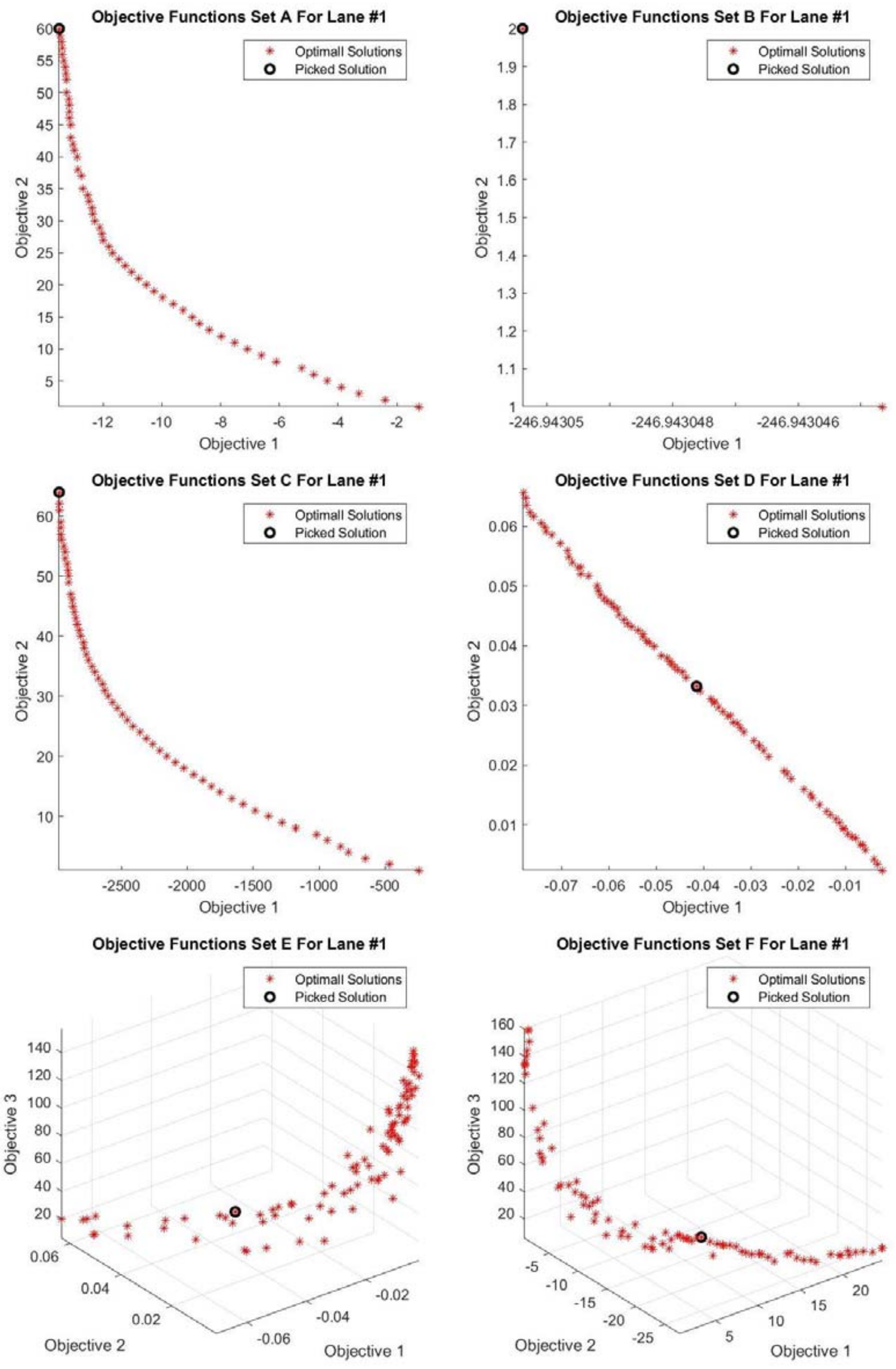


Figure 4-16. An illustration of six Pareto front outputs from NSGA II for Lane #1. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

The second half of Experiment 3 was to evaluate the optimization outputs. The optimized LGF bank evaluations used the same concept as the lane-based cross-validation. Rather than using the conventional method where the road model is directly generated from the training lanes data and then evaluated on a testing run, the road model was generated from multiple LGF banks that came from the outputs of NSGA II optimization on the training lanes. For example, the road model for testing Run #7 was generated from a combined LGF bank of Run #1 through #6 and Run #10 through #16. However, Run #8 and #9 were excluded since they were from the same lane as Run #7. Table 4-20 shows that the size of the combined LGF bank for Run #7 are as follows: 695 filters for the objective function set **A**, 87 filters for objective function set **B**, 685 filters for objective function set **C**, 170 filters for objective function set **D**, 114 filters for objective function set **E**, and 126 filters for objective function set **F**. The road model formation algorithm used these combined LGF banks to generate a road model on each frame. The road model was evaluated by measuring the error from the fuzzified ground truth using Equation (4-4). The evaluation was conducted by measuring the error of the road model on every frame in the evaluation data, which was grouped by data runs. The evaluation was also repeated for all six sets of objective functions. The results are summarized in Table 4-21 where each value is the average road model error of every frame in each data run and in all 29 look-ahead distances. The values in the table represent the average error of a specific data run when using a particular set of objective functions to optimize the LGF parameters. Appendix B presents results that show the average error per look-ahead distance (starting from 12 meters going up to 40 meters).

Table 4-21. Average detection accuracy results of the combined LGF bank generated road models.

The road model average error rate (10^{-2} or %)						
Run	**Objective Functions Set**					
	A	**B**	**C**	**D**	**E**	**F**
1	30.35	34.60	28.47	4.91	2.56	4.51
2	8.88	76.77	9.59	4.82	4.54	7.50
3	40.31	82.93	35.14	5.94	6.05	9.69
4	48.94	58.57	49.15	43.23	43.47	44.52
5	44.35	44.92	45.30	43.27	42.22	44.11
6	44.38	43.12	43.68	40.57	41.70	39.71
7	44.65	47.66	44.49	42.15	42.22	41.55
8	45.58	61.51	47.99	41.38	42.42	44.33
9	47.82	55.19	47.58	45.03	45.67	46.16
10	41.61	47.83	41.64	42.83	40.36	55.55
11	37.10	69.97	37.13	38.06	40.77	37.93
12	49.24	62.31	48.77	48.52	48.24	46.64
13	59.18	56.32	59.46	59.16	58.21	57.48
14	48.03	63.56	48.01	47.57	46.71	47.13
15	52.22	54.83	51.06	49.60	50.22	47.41
16	58.50	55.22	58.13	56.23	53.85	57.07
Total Average Error	**43.82**	**57.21**	**43.47**	**38.33**	**38.08**	**39.45**

One conclusion drawn from the above results is that the current approach cannot generate an effective road model. Even with the objective functions set **E** that yielded the lowest average error rate, the error was as high as 38.33×10^{-2}, which could be interpreted as a road model prediction with an approximately 38% chance of missing the entire road. Two issues were found to cause the algorithm to generate an ineffective road model. First, the overly large combined LGF bank is likely to contain many LGFs incapable of accurately filtering the road pixels, and hence, diminishing the effectiveness of the filter bank, which negatively impacts the road model performance. Second, there is no mechanism to incorporate the information from the testing lanes into the process of generating a combined LGF bank, which consequently causes the algorithm to produce unpredictable results. In the next section, these issues will be addressed by revising the method for generating a combined LGF bank.

4.3.3. Experiment 4: Tailoring the LGF Bank

Experiment 4 explored a method for tailoring the combined LGF bank from the previous experiment. The goal of this experiment was to increase the road model accuracy by improving the combined LGF bank effectiveness in separating the road texture from the non-road texture on the test run. An approach similar to transfer learning was used for tailoring the combined LGF bank to each test run. By adopting the transfer learning approach, data from the test run in the optimization data were used to screen out unsuitable filters from the combined LGF banks. Figure 4-17 presents a simplified flowchart of this process.

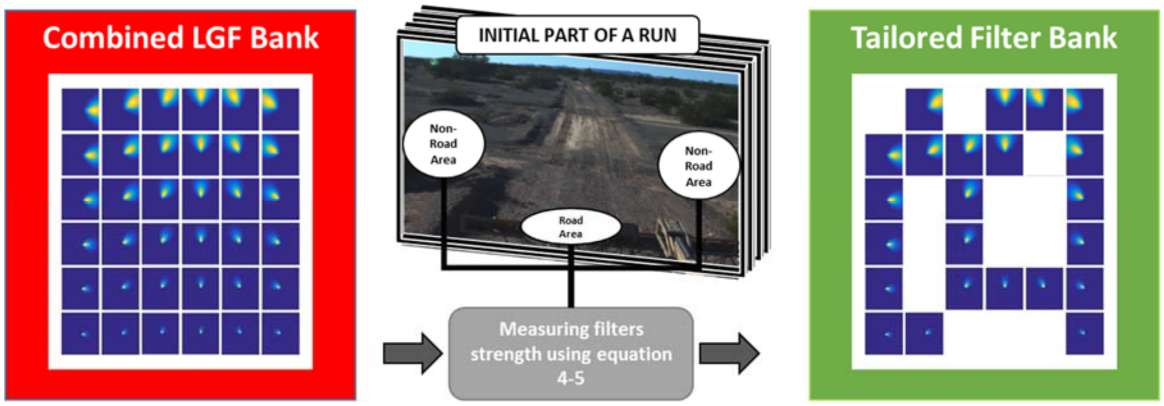


Figure 4-17. A flow chart for the process of tailoring the combined LGF bank on a given run.

The process of tailoring the LGF bank started at the initial part of the run with an assumption that the road was fairly straight. With a straight road, it was logical to presume that the bottom center region of an image was always a road area and the areas at the far left and far right of an image were non-road areas. Therefore, fairly accurate conjectural locations of the road and non-road locations on that initial part could be reasonably made. The initial part of the run in this experiment was the first 10 frames of the test run, which belonged to the optimization data. Using this initial part data, Equation (4-5) was applied to the conjectural areas to measure the discriminating strength of each filter in the combined LGF bank.

$$fs_i = \left| \left\| fr_r^i \right\|_2 - \left\| fr_{nr}^i \right\|_2 \right| \tag{4-5}$$

where:

fs_i = the filter strength of the ith filter in the bank,

fr_r^i = the ith filter response at the conjectural road area, and

fr_{nr}^i= the ith filter response at the conjectural non-road area.

Finally, the tailoring process discarded most filters and kept only a fixed number of top discriminating filters. The number of kept filters was the average number of filters produced by the optimization, and it was different for each set of objective functions. Based on the numbers in the last column of Table 4-20, the number of kept filters was 53 filters for objective set **A**, six filters for set **B**, 52 filters for set **C**, 14 filters for set **D**, 8 filters for set **E**, and 10 filters for set **F**. Once a tailored log Gabor Filter (TLGF) bank was generated for each set of objective functions, the second half of the previous experiment was repeated and the results are shown in Table 4-22.

Table 4-22. Average detection accuracy results of the road model generated by the tailored log Gabor filter (TLGF) bank.

The road model average error rate (10^{-2} or %)						
Run	**Objective Functions Set**					
	A	**B**	**C**	**D**	**E**	**F**
1	9.05	5.43	8.19	2.06	0.19	0.39
2	1.68	26.75	1.81	0.87	0.64	1.26
3	13.40	44.69	14.03	0.58	0.52	2.00
4	26.39	29.18	27.06	23.23	23.11	23.73
5	22.79	21.32	22.84	22.28	21.54	22.02
6	23.29	20.33	22.96	20.45	20.95	20.13
7	23.27	25.41	23.75	21.91	22.06	22.42
8	25.56	30.89	26.88	21.36	21.93	23.46
9	28.28	26.96	28.00	25.74	26.27	26.20
10	20.59	25.00	20.53	22.98	20.98	33.36
11	19.63	36.39	19.63	20.45	21.33	18.48
12	28.31	35.41	27.89	27.28	27.83	26.12
13	37.41	33.52	37.43	37.26	35.95	35.59
14	27.40	35.36	27.30	27.16	27.25	26.79
15	32.73	31.33	32.09	27.90	30.44	28.56
16	37.89	32.21	37.61	34.95	32.86	36.25
Total Average Error	**23.61**	**28.76**	**23.62**	**21.03**	**20.87**	**21.67**

With the removal of the weaker filters from the combined LGF bank, the results show that the TLGF bank was able to form a better road model. When compared to the combined LGF bank, the TLGF bank approach has reduced the average error rate by 17% to 28%. The TLGF bank approach has also lowered the computational cost for forming a road model by reducing the number of filters in the bank. However, due to the assumption made in this approach, the TLGF bank was biased toward the straight road. As evidenced by the results, the road model from the TLGF bank on Runs #1 to #3, which were mostly straight roads, had less than 1% error. However, Runs #4 to #16, which are curvy roads, had error rates higher than 20%. One method used to reduce this straight road bias issue was to extend the TLGF bank, so it covered more orientations. Past work has shown that an LGF bank with four orientations is enough to handle any road curvature. Therefore, the filters in the TLGF bank were increased to cover at least 4 orientations at every scale to minimize the issue. To verify the effectiveness of the extended TLGF bank, the previous experiment was repeated using the new method and the results are shown in Table 4-23.

Table 4-23. Average detection accuracy results of the road model generated by the extended TLGF bank.

The road model average error rate (10^{-2} or %)						
Run	**Objective Functions Set**					
	A	**B**	**C**	**D**	**E**	**F**
1	0.81	0.04	1.00	0.61	0.00	0.00
2	0.05	0.88	0.08	0.04	0.01	0.07
3	1.50	5.36	1.58	0.06	0.04	0.30
4	11.63	9.50	12.23	10.75	10.16	10.55
5	10.72	8.67	10.85	9.69	10.01	9.83
6	10.64	8.16	10.45	9.37	8.08	9.35
7	10.67	11.98	11.06	10.18	10.47	10.47
8	12.69	11.64	13.29	9.85	9.97	11.08
9	13.97	11.41	13.87	12.42	12.39	12.76
10	9.28	10.55	9.26	10.10	9.26	15.77
11	9.71	11.02	9.70	9.38	9.39	8.46
12	13.80	14.60	13.59	12.62	13.05	12.72
13	19.10	16.74	19.16	17.97	18.34	17.91
14	12.97	13.95	12.90	12.24	13.16	12.62
15	16.61	11.24	16.23	13.16	12.51	13.85
16	19.48	15.44	19.48	18.00	16.50	18.53
Total Average Error	**10.85**	**10.07**	**10.92**	**9.78**	**9.58**	**10.27**

The results shown that the extended TLGF bank was able to alleviate the issue of straight road bias and formed the road model more accurately since the average error rates were reduced 10% to 18%. Objective functions set E is chosen as the best set for NSGA II optimization since the optimization's output on average generated the most accurate road model. One disadvantage of the extended TLGF bank is the possibility of increasing the number of filters by 4 times. One last analysis was performed to determine the trade-off between classification accuracy and computation cost (with more filters in the bank). Multiple road models were generated from varying the sizes of the extended TLGF bank. Moreover, the effects the TLGF bank size reductions had on both the road model accuracy and the computation time were measured. The results are shown in Table 4-23.

Table 4-24. The trade-off between the road model accuracy and computational cost.

The road model average error rate (10^{-2} or %)						
Objective Functions Set E						
Run	**Number of filters in extended TLGF Bank**					
	8	**12**	**16**	**20**	**24**	**32**
1	7.92	23.11	13.53	6.84	1.19	0.00
2	4.63	1.81	1.80	0.59	0.06	0.01
3	6.45	2.49	10.15	0.52	0.05	0.04
4	44.93	32.98	30.34	20.33	10.61	10.16
5	45.78	36.74	27.13	20.49	10.11	10.01
6	42.58	30.14	23.08	16.33	8.75	8.08
7	43.18	34.40	26.38	17.84	10.63	10.47
8	44.06	40.07	33.35	18.63	10.13	9.97
9	46.93	38.68	31.62	22.51	12.53	12.39
10	43.43	33.86	25.58	19.47	9.82	9.26
11	41.30	29.83	22.17	17.21	9.49	9.39
12	48.56	37.31	29.22	22.64	13.24	13.05
13	58.31	46.64	38.47	30.08	18.35	18.34
14	48.70	36.79	30.66	23.82	13.42	13.16
15	52.67	42.14	33.37	26.15	14.84	12.51
16	56.73	45.25	36.13	29.00	18.49	16.50
Total Average Error	**39.76**	**32.02**	**25.81**	**18.28**	**10.11**	**9.58**
Computation Time (ms)	**28.1**	**53.5**	**71.6**	**93.4**	**118.4**	**135.5**

Based on the results from the above table, the best size for the extended TLGF bank is 24 filters. Compared to the extended 32-filter TLGF, the optimal 24 filters reduced the computation time by 17.1 milliseconds, which was approximately 13% lower, while the average error rate increase was less than 1%. Appendix C presents detailed results on the average errors per look-ahead distance for TLGF banks with 24 filters.

Chapter 5: System Evaluation & Conclusion

This chapter presents an evaluation of the proposed revised unpaved road detection (RURD) system. A performance comparison is also made between the proposed system and another algorithm from existing literature to further validate the proposed system. Conclusions are drawn and summarized at the end of this chapter.

5.1.RURD System Evaluation

An evaluation was designed to measure the detection accuracy of the RURD system design while monitoring the computation time. The evaluation was conducted using RURD's two stage system, the road region estimation (RRE) and the road model formation (RMF), as an individual algorithm as well as a combined system. The validation method used in this evaluation was the lane-based cross validation scheme, which meant that the data of a testing lane would not be used for training the two stages of the RURD system. However, in order to determine the real-time operation of the system, the evaluation also assessed the computation time of each stage by validating the RURD as a causal system. Unlike the validation method used in Experiments 1 through Experiment 4, where the ground truth was generated from the same frame as the one being tested, a causal system validates with a ground truth generated from a delayed frame. The amount of frame delay depends on the amount of time taken to run the algorithm. For example, testing Frame No. 1 from the RRE stage took 1 second to process and was validated with the ground truth of frame number 25 since the data was collected at 24 frames per second.

5.1.1. Standardized Algorithm Outputs

The road region estimation (RRE) stage output of a testing image is the classification result that estimates the road area in the frame, while the road model formation (RMF) stage output is a 3rd degree polynomial curve that represents the center line of the lane. No other evaluation method that can identically measure the outputs of both stages is currently available. Therefore, the output of each stage must be transformed into two standardized outputs so that both stages can be evaluated with the same measurement. The first standardized output, the directional line (DL), is a turn indicator depicting whether the road in the testing image is veering to the left, to the right, or going straight ahead. For the RRE stage, its output was transformed by forming a line between the bottom center point of the image and the center points of the estimated road region at various given look-ahead distances. The direction of the line was used to indicate the road direction. Similarly, for the RMF stage, the DL standardized output was formed by drawing a line from the bottom center point of the image to the points in the road model at various given look-ahead distances.

The second standardized output is a virtual UTM map (VUM), which is an output that shows the predicted road path in the UTM space. A VUM standardized output is the result of an affine transformation process. By using the provided transformation matrices to convert the estimated road region or the road model from image space to UTM space, a VUM standardized output is built as the vehicle progresses through the lane. Examples of the DL and VUM standardized outputs for both RRE and RMF stages are shown in Figure 5-1.

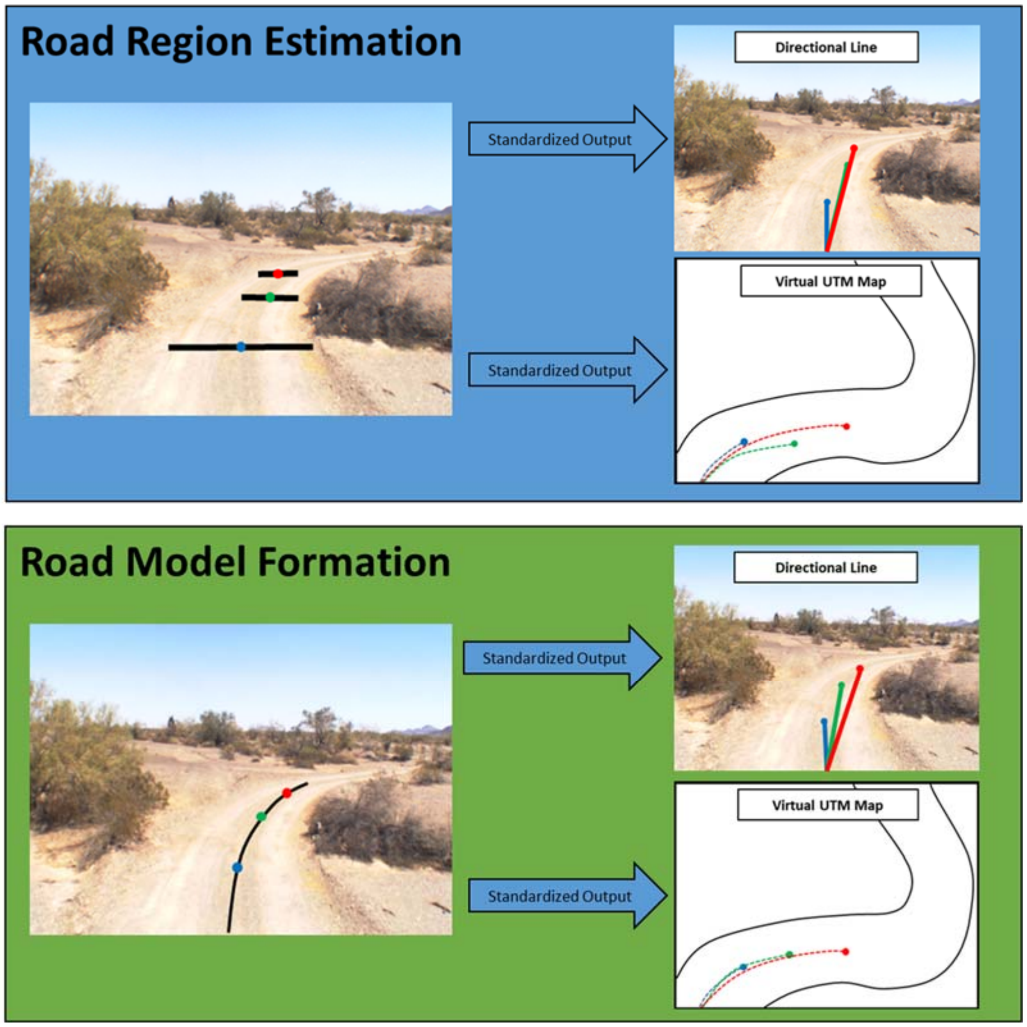


Figure 5-1. An example of directional line (DL) standardized outputs generated from both algorithms, RRE (left) and RMF(right). The three lines, blue, green, and red, indicate the direction of the road at three different look-ahead distances, 15 M, 20 M, and 25 M, respectively.

5.1.2. Standardized Ground Truth

For system evaluation, the ground truth needed to be transformed to the same format as the two standardized outputs. Standardized ground truth was generated only for Runs #4 through #16 because Runs #1 to #3 were collected from mock lanes, which were short straight roads. This evaluation also only used Runa #4 through #16 to validate the revised unpaved road detection (RURD) system.

Table 5-1. Directional line ground truth for Run #4 – 16. The table shows the number of frames in each class.

Run #	Number of frames				
	No GT		Left	Straight	Right
4	42	0.92%	519	3,468	558
5	355	4.47%	878	4,201	2,502
6	247	3.54%	963	3,973	1,804
7	169	1.99%	995	5,967	1,364
8	171	2.05%	948	5,930	1,275
9	171	1.84%	1,074	6,534	1,506
10	211	2.51%	1,283	5,607	1,296
11	181	2.18%	1,274	5,491	1,356
12	215	2.53%	1,484	5,568	1,224
13	202	2.42%	1,223	5,617	1,288
14	1,112	7.40%	3,313	7,039	3,567
15	155	1.91%	2,409	4,247	1,321
16	1,181	7.63%	3,558	7,349	3,398

The first standardized ground truth, the DL ground truth, is a record of a road direction in a frame. Each frame was vertically divided into three sections and the direction of the road, going straight, veering left or veering right, was indicated by the section where the ground truth was positioned. The 20 meters look-ahead distance was chosen for calculating the ground truth because it was the furthest distance that gave a decisive direction of the road while still remaining mostly in the field of view. Table 5-1 gives the detail of DL ground truth on 95% of the frames in each run. Sine the target data for the transfer learning came from the initial part of the runs, the first 5% of the frames on each run was omitted from the evaluation to ensure the fairness. The 'No GT' column shows the number and the percentage of frames where the road had a 20-meter look-ahead distance outside the field of view and the ground truth could not be obtained. The last three columns of the table show the number of frames exhibiting the road veering left, going straight, or veering right in each data run.

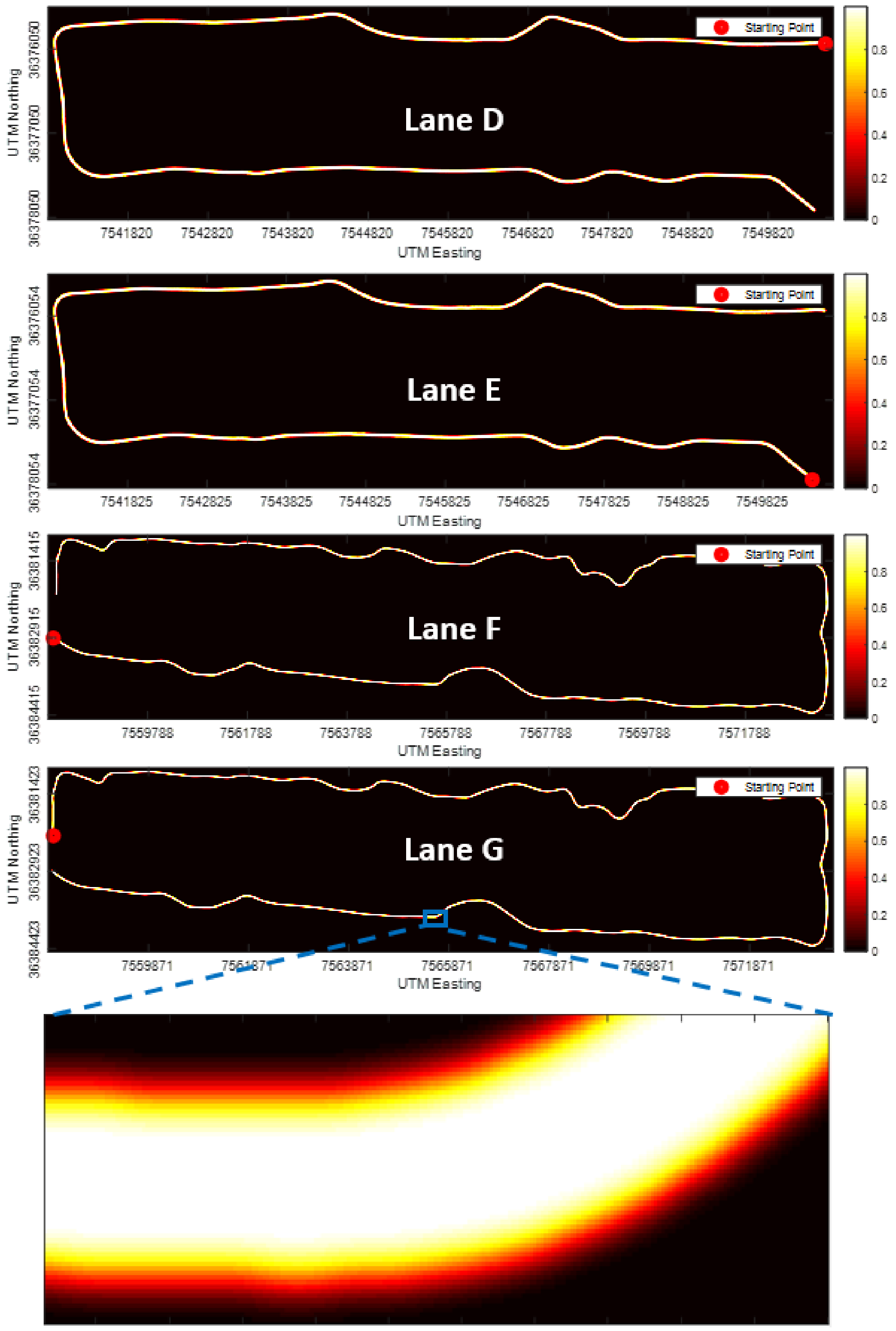


Figure 5-2. An illustration of the ground truth for a virtual UTM map (VUM). The top four images are the ground truths that have a red circle starting point from Lane D, E, F, and G. The bottom image is an example of a zoom-in section on Lane G that illustrates the ground truth has a fuzzified lane boundary. The color bars on the right indicate the color shade for the degree of road truth (a.k.a. the road confidence value).

As for the virtual UTM map (VUM), the ground truth is a confidence map of the road path in each data run. The ground truth was generated by using a method proposed by Dr. Kevin Stone[56] to project the road confidence described in Chapter 4.3.1 to the UTM coordinate space. The projection method is more accurate at a closer look-ahead distance, so the look-ahead distance of the VUM ground truth was 12 meters as it was the closest available distance. Similar to the DL ground truth, Runs #1 to #3 from Lane A, B, and C did not get standardized because they were mock lanes and the VUM ground truth was generated for Lane D, E, F, and G only. Figure 5-2 illustrates the VUM ground truth for all four lanes.

In our last step, the detection accuracy of the RRE stage, RMF stage, and RURD's combined system were evaluated by using the two standardized ground truths to measure the amount of errors they produced.

5.1.3. Individual Stage Evaluation

The first half of the evaluation was to calculate the detection accuracy of the road region estimation (RRE) stage and the Road Model Formation (RMF) stage. The discrepancies between the two standardized outputs and the two standardized ground truths were measured. For the DL standardized outputs, each measurement represents a calculated percentage based on how well each stage/algorithm could predict the direction best able to match the delayed ground truths. As stated earlier, the ground truth for the causal system had its frame delayed by the amount of time taken for the algorithm to run. For the VUM standardized outputs, the measurements provided the average road confidence values by matching the outputs with their ground truths. The road confidence value in the VUM ground truth is described in Equation 4-4 from Section 4.3.1. In practice, a road confidence value was determined for each testing frame by matching the UTM coordinate of the VUM output to the VUM ground truth. The validation results obtained using the VUM standardized method for each look-ahead distance represent the average value of the road confidences from every frame of the test run.

Evaluation based on the two standardized methods were interpreted as follows. The DL standardized method was an estimate of road detection accuracy in a casual manner since it only categorized the road into 3 directions (veering left, veering right, or going straight). Therefore, the DL method was more lax in terms of spatial precision but scrutinized the aspect of temporal precision. In contrast, the VUM standardized method only measured the spatial precision through the use of the road confidence values that were associated with their corresponding UTM coordinates.

Table 5-2. Average detection accuracy results from directional line standardized validation for the RRE stage.

Directional line detection accuracy (%)										
RUN#	**MIG Approach**					**RFF Approach**				
	Look-ahead distance									
	20M	**25M**	**30M**	**35M**	**40M**	**20M**	**25M**	**30M**	**35M**	**40M**
4	99.74	97.75	91.41	85.60	76.35	98.28	95.24	86.26	73.84	56.94
5	99.21	95.33	87.18	75.87	63.05	92.56	90.03	80.93	67.48	53.80
6	99.29	97.95	93.32	86.10	76.29	92.60	91.09	84.49	74.78	63.19
7	98.63	92.93	85.06	77.56	66.38	96.83	90.62	81.46	70.78	60.39
8	99.26	92.86	86.39	79.18	68.43	96.10	88.96	81.90	73.36	60.12
9	98.88	93.08	85.57	77.73	67.85	96.31	89.66	81.75	70.29	59.16
10	96.56	93.63	89.52	84.54	73.41	89.89	87.18	82.27	73.11	59.78
11	99.78	97.41	93.05	86.69	77.96	94.67	90.38	84.54	74.48	62.94
12	99.35	96.96	91.52	86.52	79.13	98.41	94.78	90.51	84.78	74.57
13	99.34	98.01	94.46	90.48	86.13	93.36	91.73	88.56	85.39	80.22
14	95.00	88.18	77.45	66.19	54.85	92.11	83.31	70.89	56.88	46.96
15	79.76	68.62	57.04	48.83	44.54	85.25	77.65	72.31	68.47	63.43
16	91.98	84.31	73.65	64.83	57.57	90.77	84.22	77.26	70.96	64.16
Avg.	**96.67**	**92.08**	**85.05**	**77.70**	**68.61**	**93.63**	**88.84**	**81.78**	**72.66**	**61.97**

Both the reinforcement random forest (RRF) and mixed information gain (MIG) approaches were reevaluated because Experiment 2 did not determine which approach was the most suitable for the road region estimation (RRE) stage. Experiment 2 shows that at a 19 meters look-ahead distance, both the RRF and MIG approaches are able to achieve a classification accuracy of 94.41% and 90.93%, respectively. Nevertheless, one of the objective in this evaluation was to compare the detection accuracy of the RRE stage to the road model formation (RMF) stage, which had a detection distance of up to 40 meters. Therefore, this evaluation tested the RRE stage at farther look-ahead distances that were ranged from 20 meters to 40 meters with an increment of 5 meters. Table 5-2 shows the results of the DL standardized method for both RRF and MIG. The results were the accuracy percentages measuring each approach that could correctly predict the road direction at varying look-ahead distances. Similarly, Table 5-3 shows the average road confidence results of the VUM standardized method at various look-ahead distances.

Table 5-3. Average detection accuracy results from virtual UTM map standardized validation for the RRE stage.

Virtual UTM Map detection accuracy (10^{-2})										
RUN#	**MIG Approach**					**RFF Approach**				
	Look-ahead distance									
	20M	**25M**	**30M**	**35M**	**40M**	**20M**	**25M**	**30M**	**35M**	**40M**
4	99.07	94.86	87.47	82.80	73.93	97.91	92.80	82.58	71.59	55.76
5	98.78	93.71	85.94	74.93	63.67	97.88	92.66	82.61	68.62	55.68
6	98.95	92.97	85.01	77.43	68.73	97.57	92.86	84.50	72.05	60.86
7	97.26	88.67	81.87	74.93	64.68	97.50	89.45	81.73	71.81	60.59
8	98.80	91.79	84.80	77.69	67.74	96.95	88.23	80.78	72.47	59.13
9	98.57	90.91	85.20	76.51	66.73	96.96	89.07	81.43	69.96	59.28
10	90.46	84.38	78.42	73.45	63.98	85.30	82.67	77.71	68.65	56.10
11	98.71	93.90	87.36	80.11	72.24	93.48	87.57	80.81	70.70	59.40
12	98.09	93.10	85.87	78.52	70.39	97.24	91.68	86.46	79.12	69.12
13	95.73	89.25	78.15	65.77	57.58	95.00	90.35	85.65	78.94	71.06
14	91.40	80.07	65.59	50.91	39.39	93.36	82.76	68.89	52.47	41.16
15	65.29	61.04	47.58	38.58	33.98	79.70	74.04	65.71	59.10	52.84
16	88.92	77.32	64.54	54.40	46.45	91.96	83.36	71.39	61.99	52.59
Avg.	**93.85**	**87.07**	**78.29**	**69.69**	**60.73**	**93.91**	**87.50**	**79.25**	**69.04**	**57.97**

The results from Table 5-2 and 5-3 share the same conclusion that the detection accuracy of road region estimation (RRE) stage is significantly reduced as the look-ahead distance increases. The rates of accuracy reduction for both the mixed information gain (MIG) and RRF approaches are immense. The RRE stage becomes impractical to use after the look-ahead distance extends beyond 30 meters since the error rates for both DL and VUM methods are significantly greater than 20%. At the look-ahead distance of 20 meters, the MIG approach achieved an average detection accuracy of 96.67% using the DL standardized method and 93.85% using the VUM standardized method. The RRF approach achieved a lower average detection accuracy of 93.63% with the DL standardized method and a slightly higher accuracy of 93.91% with the VUM standardized method. Only results from Run #15 were found to be an anomaly. A further inspection revealed that Run #15 had problems during the data collection such as a sensor dropout or the appearance of a road barricade. These problems required vehicle operator intervention in front of the camera while the data was collecting, which caused confusions to the detection algorithm. Unlike the conclusion from Experiment 2, the DL standardized method results show that the MIG approach yielded a higher accuracy than the RRF approach. The RRF lower accuracy is caused by its higher computation cost. RRF detected the road at a frequency of 1 Hz (one time per second), while MIG detected the road at 4 Hz. The RRF's lower detection frequency means that there is a greater chance that the road had changed its direction by the time the algorithm made a prediction, which is reflected in the lower detection accuracy. In contrast, the VUM standardized results show that the RRF was more accurate than MIG since the VUM method did not consider the causality of the system. However, the results from the VUM method also showed that the differences in detection accuracy between the RRF and MIG approach were minimal. Thus, this evaluation concludes that MIG is a more suitable approach for the RRE stage.

Table 5-4. Average detection accuracy results from DL and VUM standardized validation for RMF stage.

RUN#	**Directional Line detection accuracy (%)**					**Virtual UTM Map detection accuracy (10^{-2})**				
	Look-ahead distance									
	20M	**25M**	**30M**	**35M**	**40M**	**20M**	**25M**	**30M**	**35M**	**40M**
4	97.90	97.43	97.21	97.16	97.11	92.47	89.86	88.30	84.20	79.30
5	89.79	89.07	88.44	88.08	87.73	96.91	90.40	86.20	82.80	79.68
6	91.18	90.47	90.11	89.92	89.89	95.64	92.48	90.43	86.59	82.38
7	92.95	93.08	93.19	93.29	93.41	97.53	91.55	85.34	80.62	77.51
8	92.88	92.87	92.91	93.10	93.22	97.49	91.76	86.21	81.30	78.19
9	93.16	92.92	92.85	92.86	92.91	96.47	90.74	83.17	76.18	72.15
10	93.77	93.83	93.81	93.90	93.93	97.21	93.22	88.48	81.96	77.87
11	94.78	94.72	94.69	94.75	94.80	97.99	92.94	86.69	81.73	76.62
12	95.65	95.62	95.65	95.65	95.66	95.30	88.77	83.32	76.39	69.66
13	94.56	94.12	94.11	94.25	94.40	91.17	84.11	76.53	69.19	62.46
14	81.03	79.81	79.58	79.55	79.67	95.18	89.68	83.17	74.94	69.27
15	97.89	97.86	97.83	97.83	97.83	90.14	91.63	84.79	78.75	71.89
16	86.85	86.15	86.03	85.91	85.91	92.98	86.20	78.22	72.21	65.67
Avg.	**92.49**	**92.15**	**92.03**	**92.02**	**92.04**	**95.11**	**90.26**	**84.68**	**78.99**	**74.05**

Next, a similar evaluation was carried out in the RMF stage and the results for both the DL and VUM standardized methods are shown in Table 5-4. The RMF stage took approximately 1.25 seconds to process each frame, so there is a 30 frames delay between the testing frame and the ground truth frame for the DL standardized method. Comparable to the MIG approach in the RRE stage, the longer delay leads to lower averaged detection accuracies at 20 meters of look-ahead distance. However, despite of the delay, an RMF advantage is that its trajectory is consistent regardless of the look-ahead distances, which significantly lowers the accuracy reduction when detecting the road at great distances. The characteristics of the road model also minimizes the confusion caused by the appearances of the vehicle operator in Run #15 since the RMF stage processed the entire image at once, while the RRE stage processed an image in small image patches. Furthermore, the pseudo outputs of Run #4 from both stages were used as an example to corroborate with the results in Table 5-4. The pseudo output is an (X,Y) coordinate of an image that each stage produces as a predicted road location, and it can be different for each look-ahead distance. The top plot in Figure 5-3 is the RRE stage's pseudo output of Run #4. It shows high variations in the predicted locations between all five look-ahead distances, which leads to inconsistent detection accuracies. The bottom plot in Figure 5-3 is the pseudo outputs from the RMF stage of the same run. The plot shows that the RMF outputs are much more consistent, which supports the finding in Table 5-4.

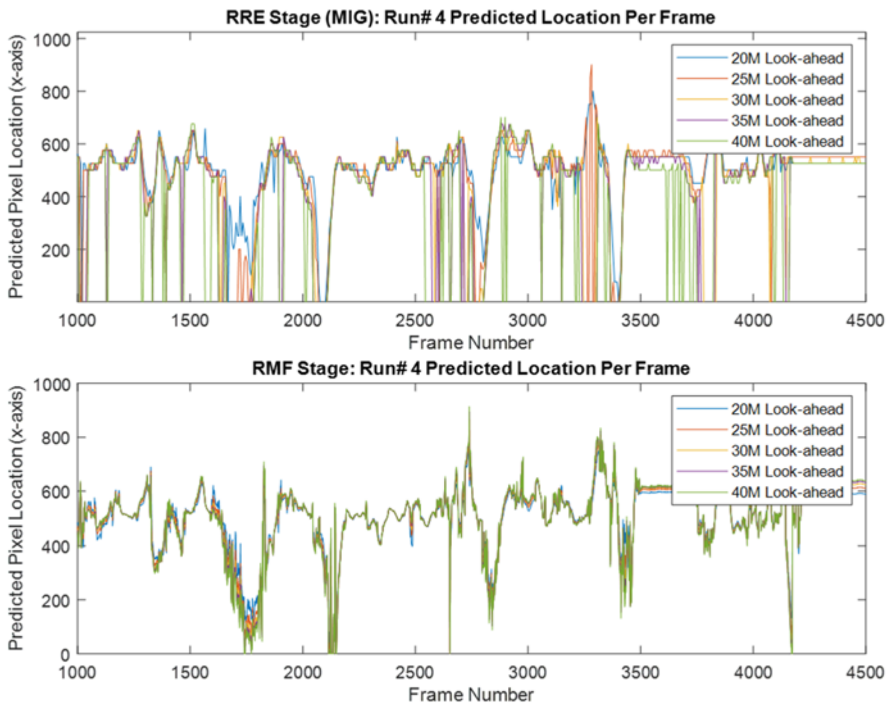


Figure 5-3. Illustration of the pseudo outputs of Run #4. The top plot is the outputs from the RRE stage, while the bottom plot is the outputs from RMF stage. Each line in the plots represents the output at varying look-ahead distances, starting from 20 meters to 40 meters.

5.1.4. Whole System Evaluation and Comparison

The second half of the evaluation was to calculate the detection accuracy of the RURD system as a whole and compare it against another unpaved road detection algorithm. The RURD system is composed of the RRE and RMF stages. As described in Section 3.2.1, both stages can work together by using the output of the RRE stage as a high confidence road area (HCRA). HCRA was used during the process of generating the image of road assessment in the RMF stage. Since the results from the previous section showed that the RRE stage was best at detecting the road at 20 meters of look-ahead distance, the detection output of the RRE stage at the 20-meter distance was selected as the HCRA for the RMF stage. With a consolidation of both stages, the detection accuracy is expected to improve over the results from the individual stages. The same evaluation used in the previous section was performed for the RURD system, and the results are shown in Table 5-5.

Table 5-5. Average detection accuracy results from DL and VUM standardized validation for RURD system.

RUN#	Directional Line detection accuracy (%)					Virtual UTM Map detection accuracy (10^{-2})				
	Look-ahead distance									
	20M	25M	30M	35M	40M	20M	25M	30M	35M	40M
4	97.02	97.22	96.97	96.51	96.09	99.32	99.40	98.16	95.05	92.10
5	88.06	88.08	88.06	88.34	87.57	98.99	98.50	95.14	90.49	83.52
6	91.23	90.91	90.59	90.07	89.67	99.28	98.96	95.87	88.66	78.82
7	97.68	97.85	97.53	97.09	96.79	98.44	97.35	92.78	87.88	83.38
8	97.80	97.93	97.52	97.72	97.61	99.16	98.23	94.28	88.19	83.01
9	93.75	95.87	97.00	97.78	97.48	98.85	97.05	88.69	74.98	58.85
10	98.66	99.01	98.11	97.23	96.50	92.01	95.75	90.93	86.03	83.12
11	98.58	98.77	98.87	98.96	98.82	98.69	99.46	95.46	87.35	82.16
12	99.17	99.44	99.44	99.20	99.06	98.07	99.12	96.56	87.17	83.44
13	98.50	98.88	98.54	98.39	98.24	96.97	98.21	94.15	87.94	83.71
14	85.78	88.75	89.27	89.69	89.09	93.14	92.07	78.79	55.75	40.76
15	97.68	97.61	96.36	83.43	78.41	75.92	77.34	77.09	80.81	83.01
16	88.42	89.39	90.33	89.71	88.26	90.15	89.14	76.97	60.01	47.68
Avg.	**94.79**	**95.36**	**95.27**	**94.16**	**93.35**	**95.31**	**95.43**	**90.38**	**82.33**	**75.66**

The computational times for RRE and RMF stages were approximately 0.25 and 1.25 seconds, respectively. Hence, it took the RURD system approximately 1.5 seconds to process a given frame. Therefore, the detection frequency went down from 0.8 Hz in the RMF stage to 0.67 Hz. For the DL standardized method, the lowered detection frequency increases the delay between the testing frame and the ground truth frame to 36 frames. The increased frame delay caused the detection accuracy at 20 meters to be slightly lower than the 25-meter result since the closer detection distance is more affected by the delay. Figure 5-4 shows an example where the increase in a frame's delay caused the RURD system to predict the road direction incorrectly at 20 meters out, while at farther distances, the system was able to make the correct predictions. Nevertheless, by consolidating both stages, the RURD system improved the detection accuracy at all look-ahead distances from any individual stage.

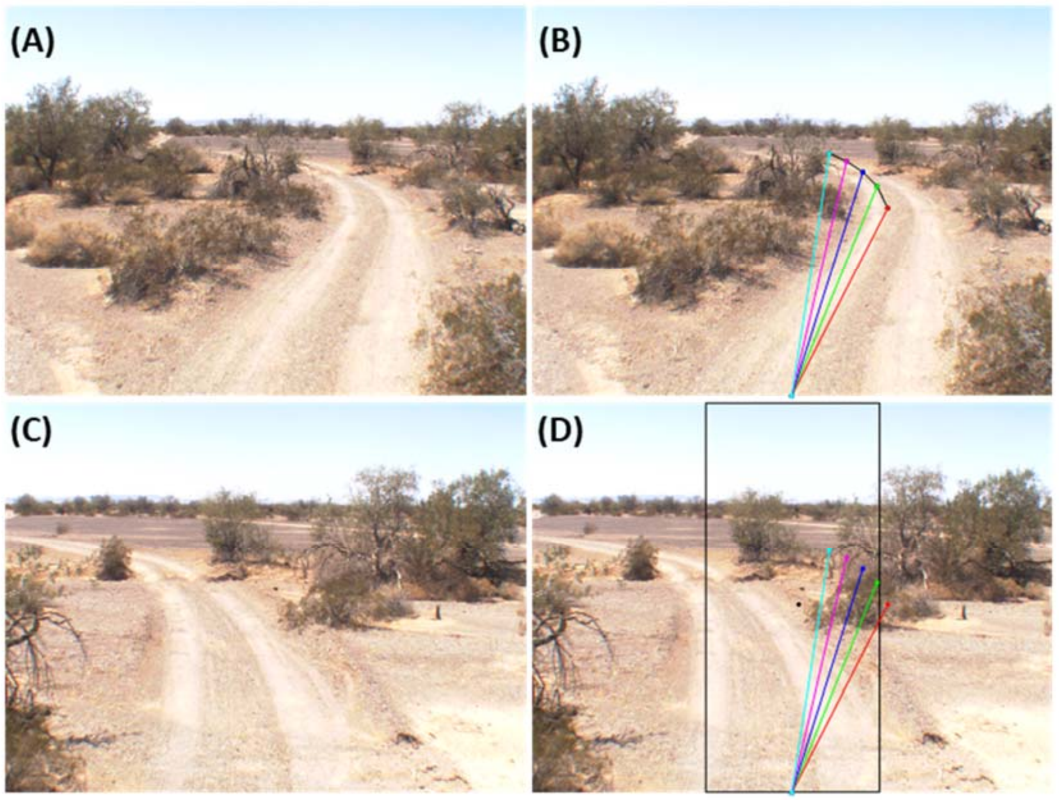


Figure 5-4. An example scenario of the DL standardized method showing the RURD system making an incorrect prediction of the road direction at a look-ahead distance of 20 meters while still correctly predicting the road at further distances. (A) is the testing image and (B) displays the predicted road direction at 20 m (red), 25 m (green), 30 m (blue), 35 m (magenta), and 40 m (cyan). (C) is the ground truth image, which is the 36th frame after the test image. (D) illustrates the DL truth section in the black rectangle with the predicted road directions from (B). It shows that only the predicted direction of 20 meters is outside the ground truth section.

Lastly, the result from the RURD system evaluation was used to compare an algorithm from Rasmussen.[63] The selected algorithm for benchmarking is a road following algorithm from a paper titled "Grouping Dominant Orientations for Ill-Structured Road Following," which from this point on will be referred to the GDOIS algorithm. The reasons for choosing the GDOIS algorithm are its relative high numbers of citations and its input/output similarity to the RURD system. While most unpaved roads have been reported as using different types of input data, such as a top-down view of the road or with LIDAR point cloud images, GDOIS was designed to follow the road in a perspective view image, which was the same as the verification data in this dissertation. The algorithm follows an unpaved road by computing the dominant texture orientations with the use of multi-scale Gabor wavelet filters, which then selects a consensus road vanishing point of a given image. For more detail about the GDOIS algorithm, a MATLAB version of the GDOIS code is provided in Appendix D.

The GDOIS outputs a road vanishing point, which is the point where the road in an image eventually ends. The vanishing point is used to generate the road direction line by forming a line between the vanishing point and the coordinate at the bottom center of an image. Ultimately, both DL and VUM standardized outputs were generated from GDOIS's road direction line. The same evaluation conducted for the RURD system was also carried out for GDOIS, and the evaluation results are shown in Table 5-6.

Table 5-6. Average detection accuracy results from DL and VUM standardized validation for GDOIS algorithm.

RUN#	**Directional Line detection accuracy (%)**					**Virtual UTM Map detection accuracy (10^{-2})**				
	Look-ahead distance									
	20M	**25M**	**30M**	**35M**	**40M**	**20M**	**25M**	**30M**	**35M**	**40M**
4	77.47	75.95	74.89	73.83	72.75	90.22	81.58	79.13	78.13	75.75
5	73.73	70.85	68.64	67.16	66.23	87.10	85.12	84.58	82.42	77.82
6	69.58	67.22	65.20	63.30	62.11	86.70	81.02	77.79	73.34	68.76
7	56.42	54.36	53.28	52.64	52.05	72.59	62.22	56.15	51.34	48.32
8	58.34	56.93	55.90	55.13	54.43	74.83	65.02	58.11	53.43	50.76
9	61.23	59.62	58.19	57.40	56.81	75.06	64.91	59.28	54.52	50.78
10	53.92	51.71	50.11	49.15	48.25	71.07	60.23	53.35	47.37	42.23
11	58.67	57.18	56.27	55.86	55.49	77.08	68.56	62.19	56.29	51.77
12	61.17	59.44	58.10	56.74	56.17	79.34	71.06	65.43	59.74	55.56
13	53.21	50.78	49.21	47.72	46.94	75.64	63.83	55.91	50.30	46.03
14	48.37	45.74	44.32	43.65	43.11	67.00	60.64	55.11	48.72	42.49
15	40.97	39.63	38.76	38.27	37.80	55.38	59.30	52.93	48.87	45.90
16	49.93	47.65	46.20	45.32	44.61	69.56	59.24	52.00	46.25	42.78
Avg.	**58.69**	**56.70**	**55.31**	**54.32**	**53.60**	**75.51**	**67.90**	**62.46**	**57.75**	**53.76**

The GDOIS algorithm took approximately 5 seconds to process one frame, which led to 120 delay frames between the testing frame and ground truth frame. The delay significantly reduced the detection accuracy of the DL standardized method to the point that the accuracy was lower than the VUM standardized method. This result suggests that the GDOIS computation time makes the algorithm impractical to use in a real-time application. An argument can be made to give a leniency to the GDOIS algorithm since the code used in this dissertation has not been optimized. However, the results from the VUM standardized method that does not account for the computational time are still significantly lower when compared to the results from the RURD system or any individual stage. There are two main reasons that render the GDOIS algorithm unsuitable for the verification data in this dissertation. First, the GDOIS algorithm has the same downfall as the previous version of the RMF algorithm, where the filter bank is static in nature and unresponsive to the texture diversities in the data. This reason was substantiated by the results from the VUM method. At a look-ahead distance of 20 meters, the results show that the GDOIS algorithm was effective at detecting less diverse road textures on Runs #4 to #6, but it was unable to detect more diverse road textures in Runs #7 to #16. Second, the verification data has many images that contain road curvatures that are too complex for an algorithm to accurately detect with the vanish point approach. Figure 5-5 shows two examples of complex road curves that render the vanish point approach ineffective.

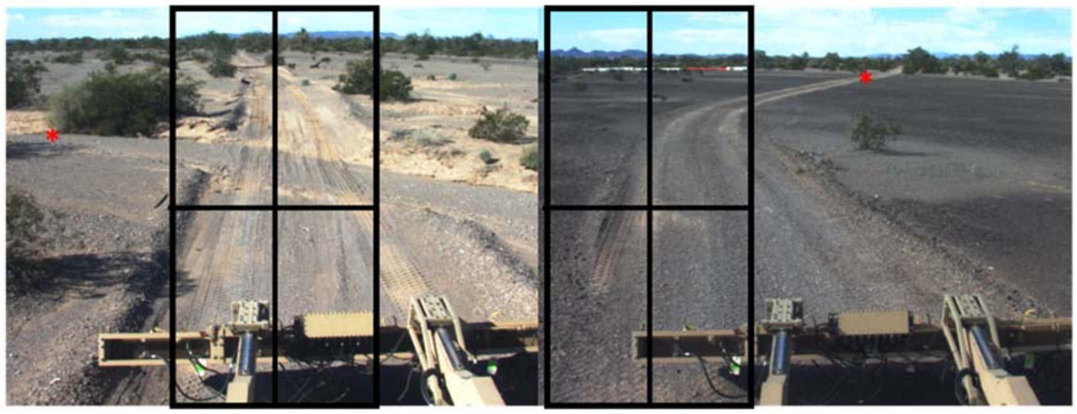


Figure 5-5. The left image shows the scenario where GDOIS predicted the vanishing point (red dot) in the road wash. The right image shows the scenario where the road vanishing point (red dot) is on the right side of the image. While it seems correct, the ground truth at 20-meter distance (black window) shows that the road in front is tilting left. This scenario shows a major disadvantage of the vanish point approach, which only indicates the end point of the road but does not consider the road right in front of the vehicle.

5.2. Conclusions

5.2.1. Summary

The revised unpaved road detection system (RURD) is a novel method for detecting unpaved roads in an arid environment from imagery collected by a stationary color camera mounted in front of a U.S. Army prototype vehicle. The method was proposed, revised, and evaluated for a real-time operation. It was also benchmarked against an algorithm from Rasmussen entitled "Grouping Dominant Orientations for Ill-Structured Roads Following (GDOIS).[63] A revision was made in the road region estimation (RRE) stage, with the use of a transfer learning approach called mixed information gain (MIG), used to maximize the detection accuracy while retaining a minimum computation cost. If the result from the problematic Run #15 was excluded, the RRE stage could achieve an average detection accuracy as high as 98.09% with the directional line (DL) standardized method and 96.23% with a virtual UTM map (VUM) standardized method at the look-ahead distance of 20 meters. For the road model formation (RMF) stage, a revision was made to the process of log Gabor filter (LGF) bank generation by utilizing the NSGA II optimization, so that the filter bank could intelligently adapt to more diverse road textures. The revision enables the RMF stage to effectively detect the unpaved road in both the 2012 and 2013 data collections, where the road textures are vastly different. The revised RMF yields a slightly lower detection accuracy at the look-ahead distance of 20 meters but can maintain significantly higher accuracies as the distance increases.

With the consolidation of the revised RRE and RMF stages, the DL standardized method shows that the RURD system can achieve a detection accuracy higher than 90% regardless of the look-ahead distances. For the VUM standardized method that emphasizes spatial precision, the RURD system also achieves more than 90% detection accuracy with a look-ahead distance of up to 30 meters. Furthermore, the RURD system significantly outperforms the benchmark algorithm, GDOIS, by attaining an approximately 20% to 40% higher detection accuracy. In conclusion, the proposed RURD system has fulfilled the goal of effectively detecting an unpaved road up to 40 meters distance in front of the vehicle.

5.2.2. Future work

Although, the RURD system can effectively detect the road in a given image, the system is still far from achieving 100% detection accuracy. A major goal for future work is to improve the system’s detection accuracy.

The RURD system evaluation has shown that an algorithm with higher detection frequency is likely to increase the detection accuracy in a causal system as well. The current implementation of the RURD system is in the MATLAB environment, which has a sizeable computation cost overhead. An obvious way to reduce the processing time and increase the detection frequency is to convert the RURD system implementation to C language since there is minimal overhead cost.

Another method to increase the detection accuracy is to improve each individual stage. For the RRE stage that classifies each image independently, a SIFT-like algorithm can be used to connect the detection results from multiple frames and acts as a temporal smoothing. The temporal smoothing should remove outlier detections and improve the overall accuracy. However, the current SIFT implementation took approximately 10 seconds to process each frame. The huge additional computation time would have a negative effect on detection accuracy. One possible future work is to use the SURF[64] algorithm, which is three times faster than SIFT. Implementing the SURF algorithm in C language should minimize the additional computation time caused by incorporating a temporal smoothing technique to the RRE stage. Similarly for the RMF stage, the detection frequency can be increased by utilizing the C implementation of the CUDA library, which can process LGF 14 times[65] faster compared to MATLAB version.

Lastly, another interesting approach to the RURD system is to further explore the deep learning approach potential. Rather than attempting to classify small image patches conducted in Experiment 2, deep learning can be implemented to process the entire image at once to segment road markings further out and with more detail.

Appendices

Appendix A: Illustrations of the NSGA II optimization output of Pareto fronts from Section 4.3.2.

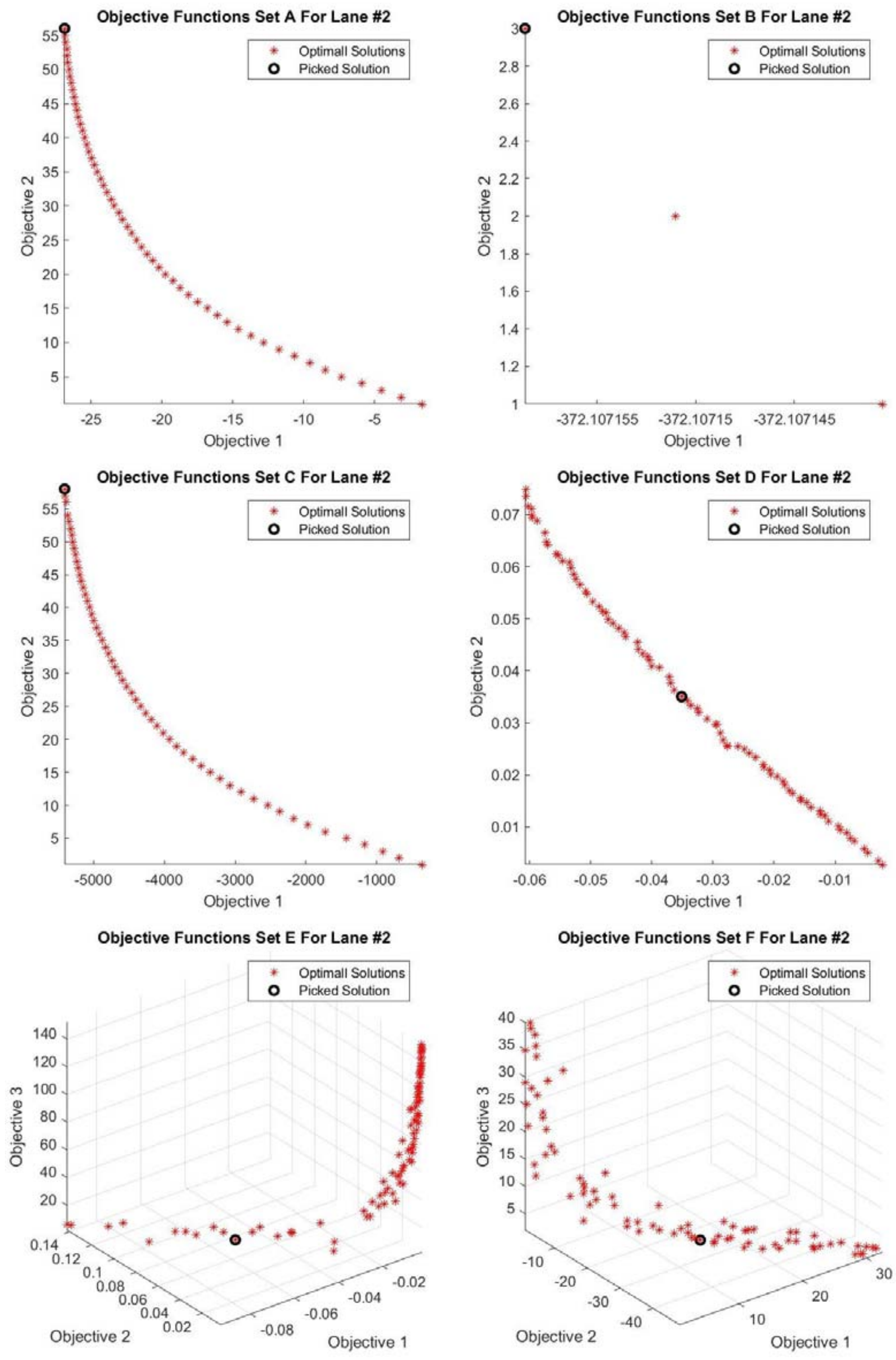


Figure A6-1. An illustration of six Pareto front outputs from NSGA II for Lane #2. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

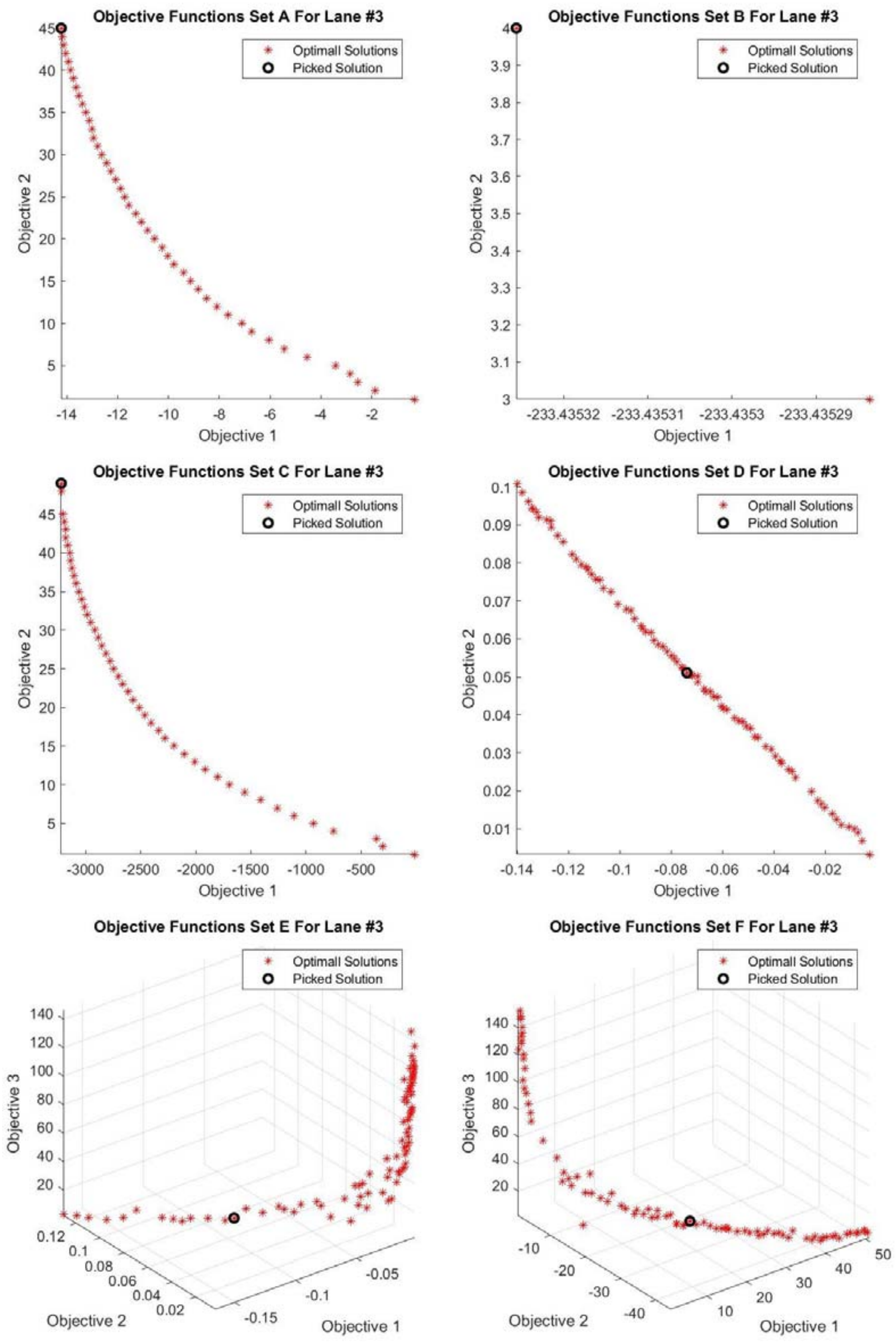


Figure A6-2. An illustration of six Pareto front outputs from NSGA II for Lane #3. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

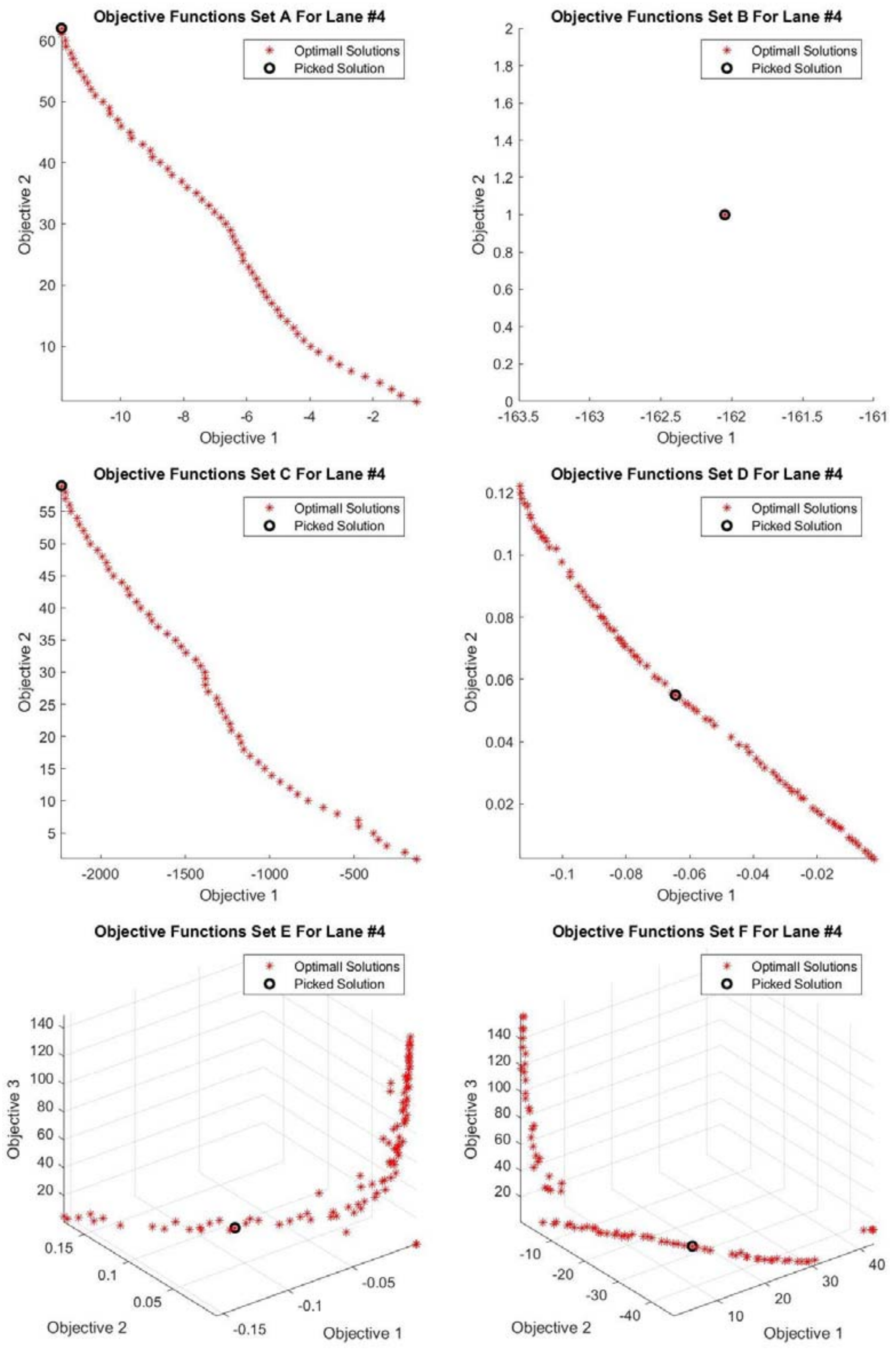


Figure A6-3. An illustration of six Pareto front outputs from NSGA II for Lane #4. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

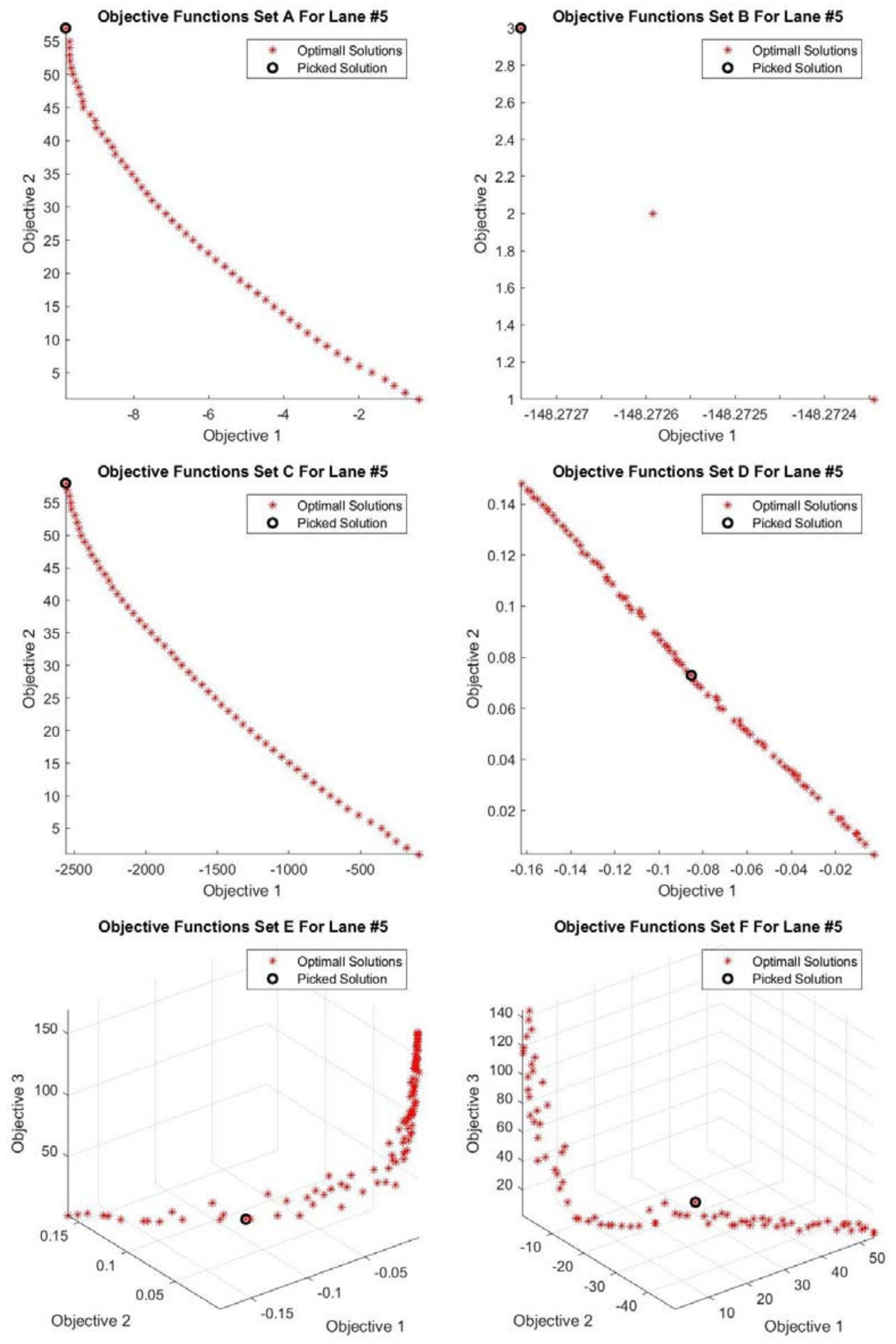


Figure A6-4. An illustration of six Pareto front outputs from NSGA II for Lane #5. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

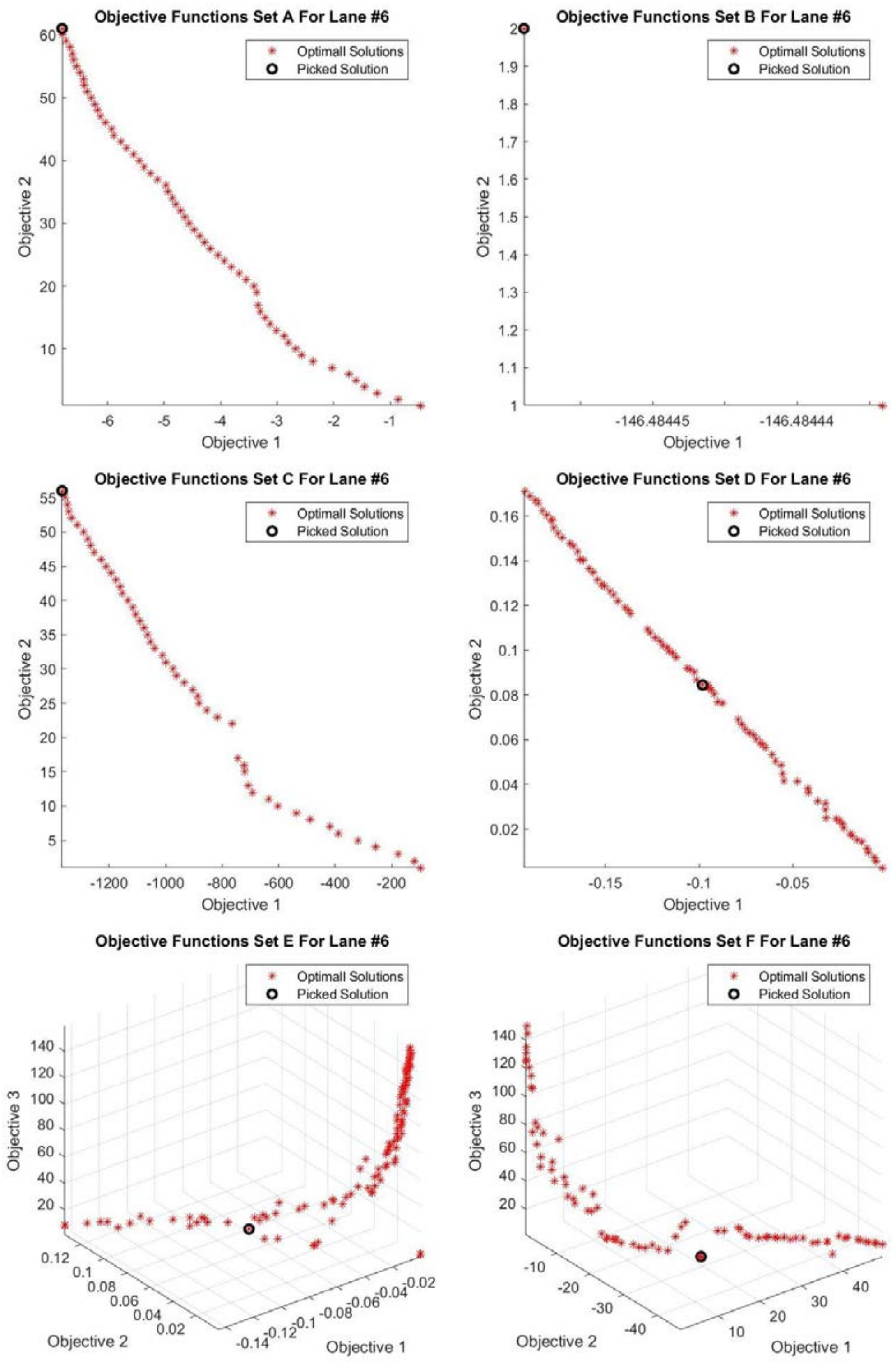


Figure A6-5. An illustration of six Pareto front outputs from NSGA II for Lane #6. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

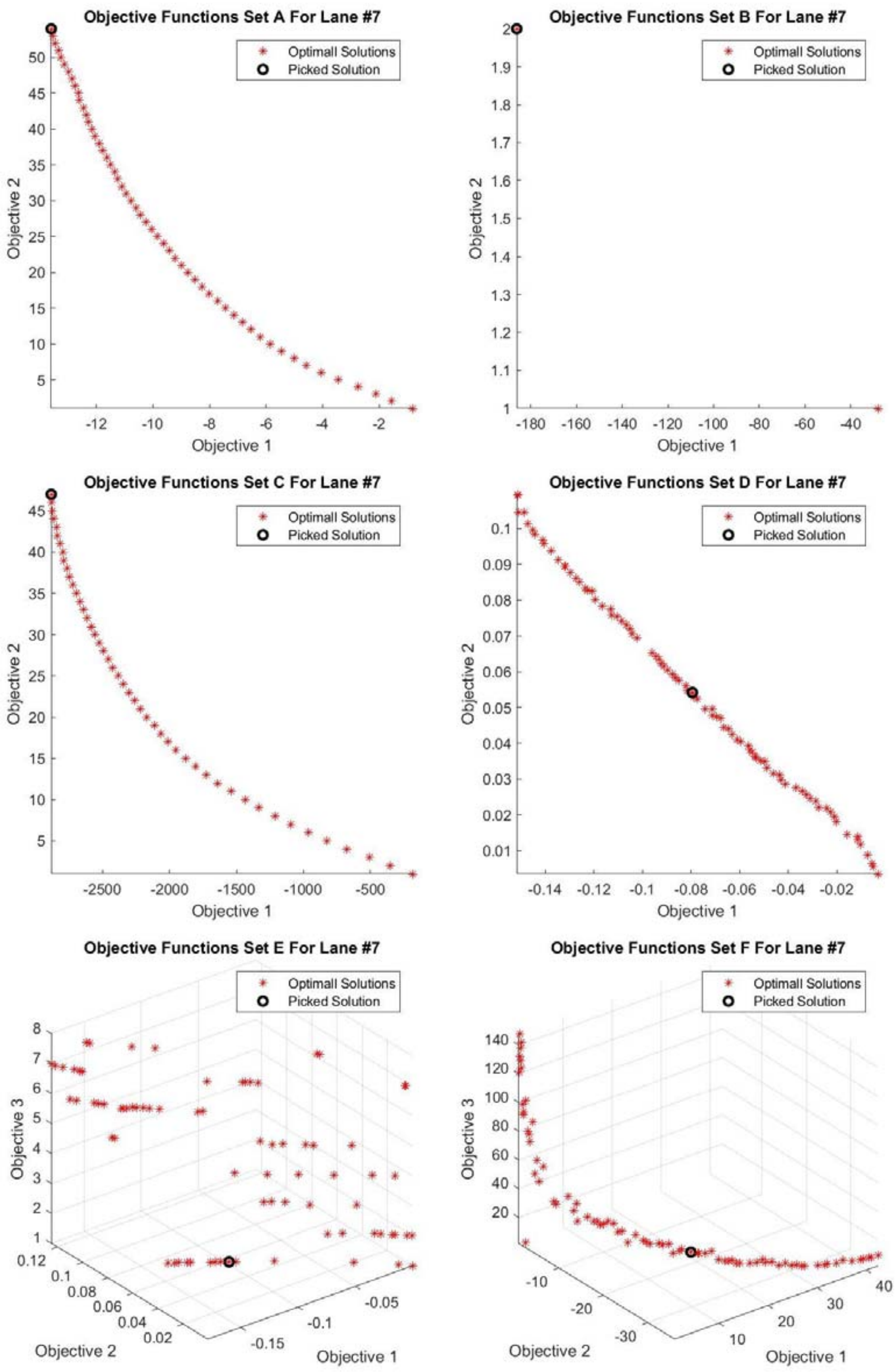


Figure A6-6. An illustration of six Pareto front outputs from NSGA II for Lane #7. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

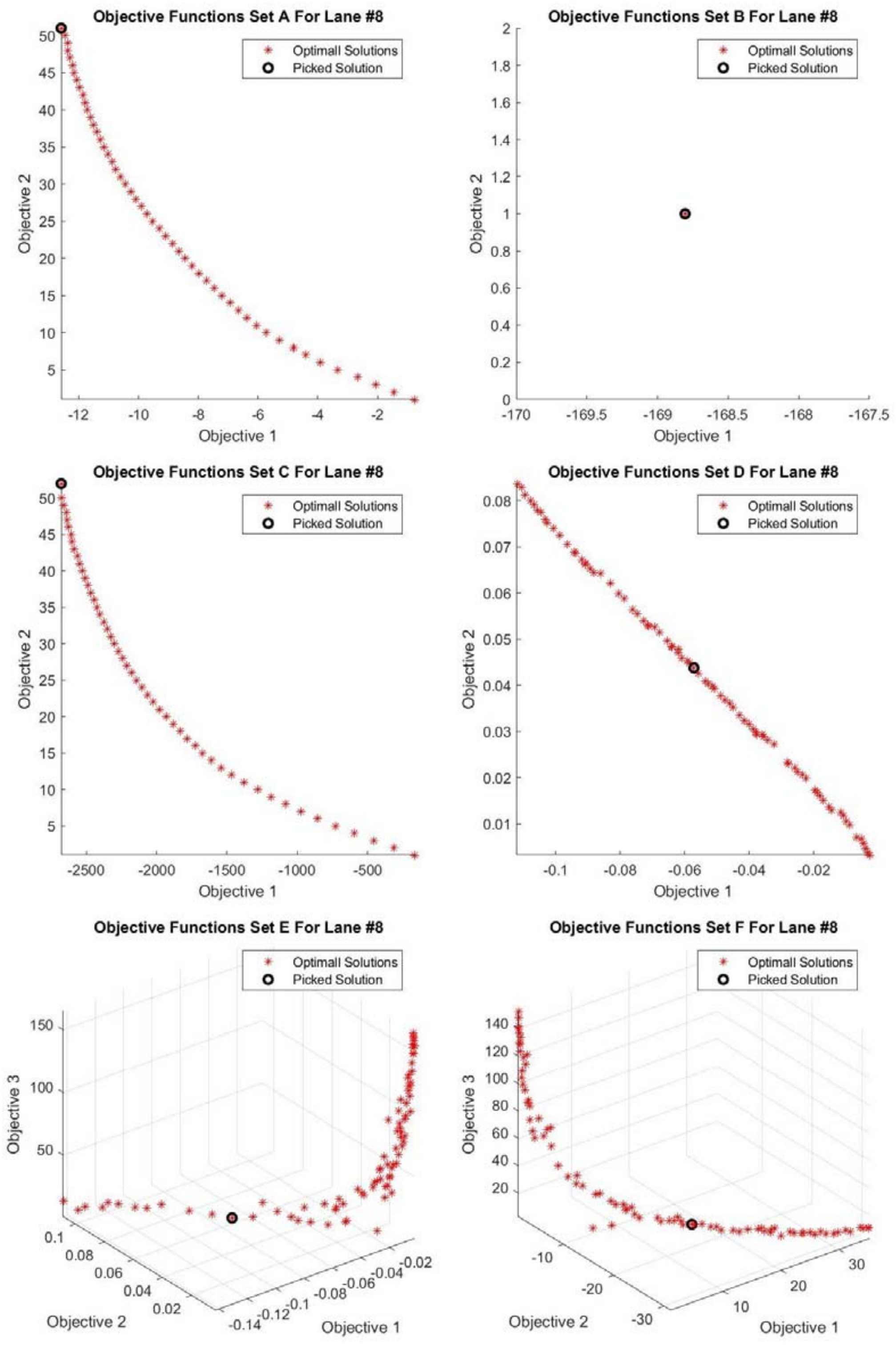


Figure A6-7. An illustration of six Pareto front outputs from NSGA II for Lane #8. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

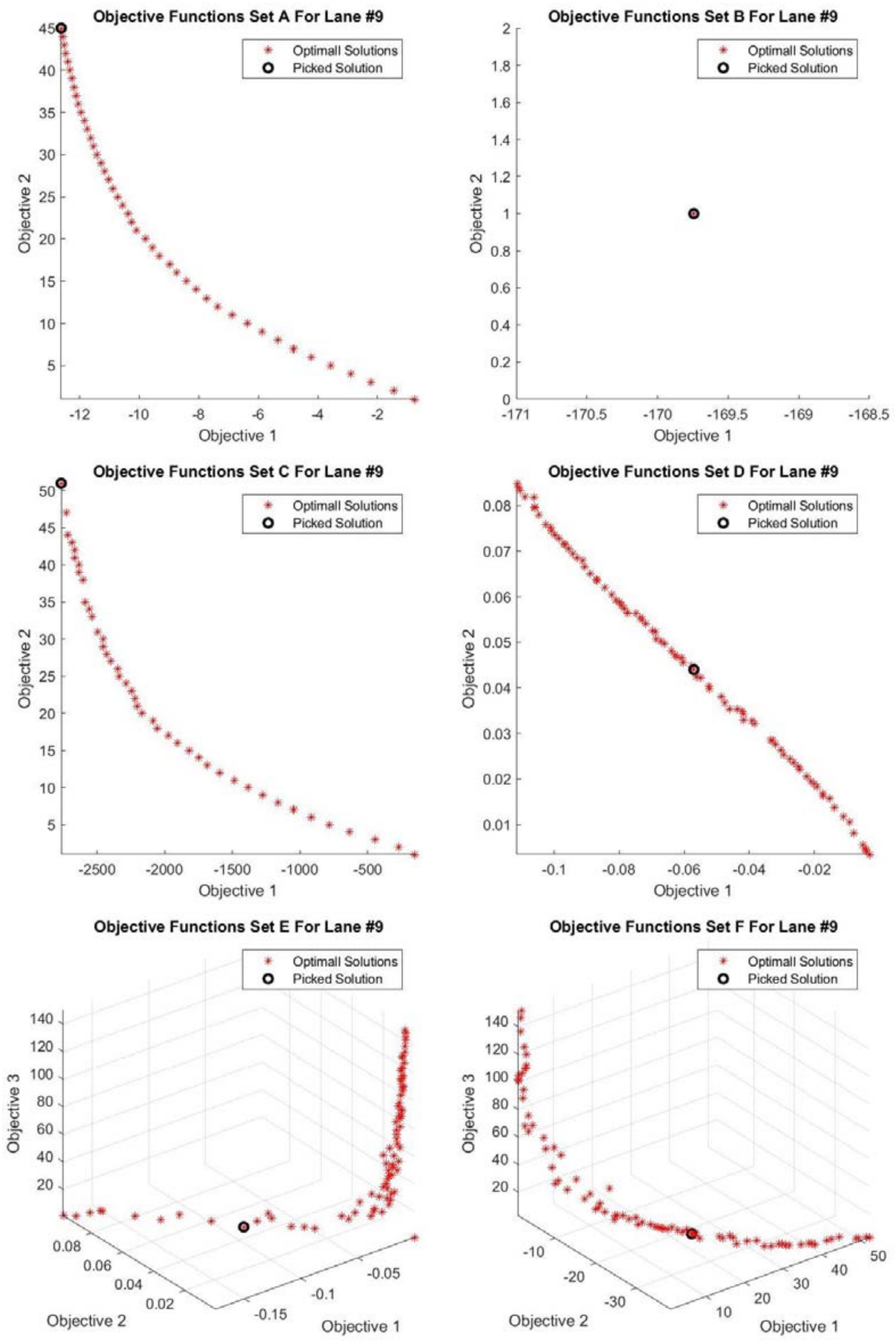


Figure A6-8. An illustration of six Pareto front outputs from NSGA II for Lane #9. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

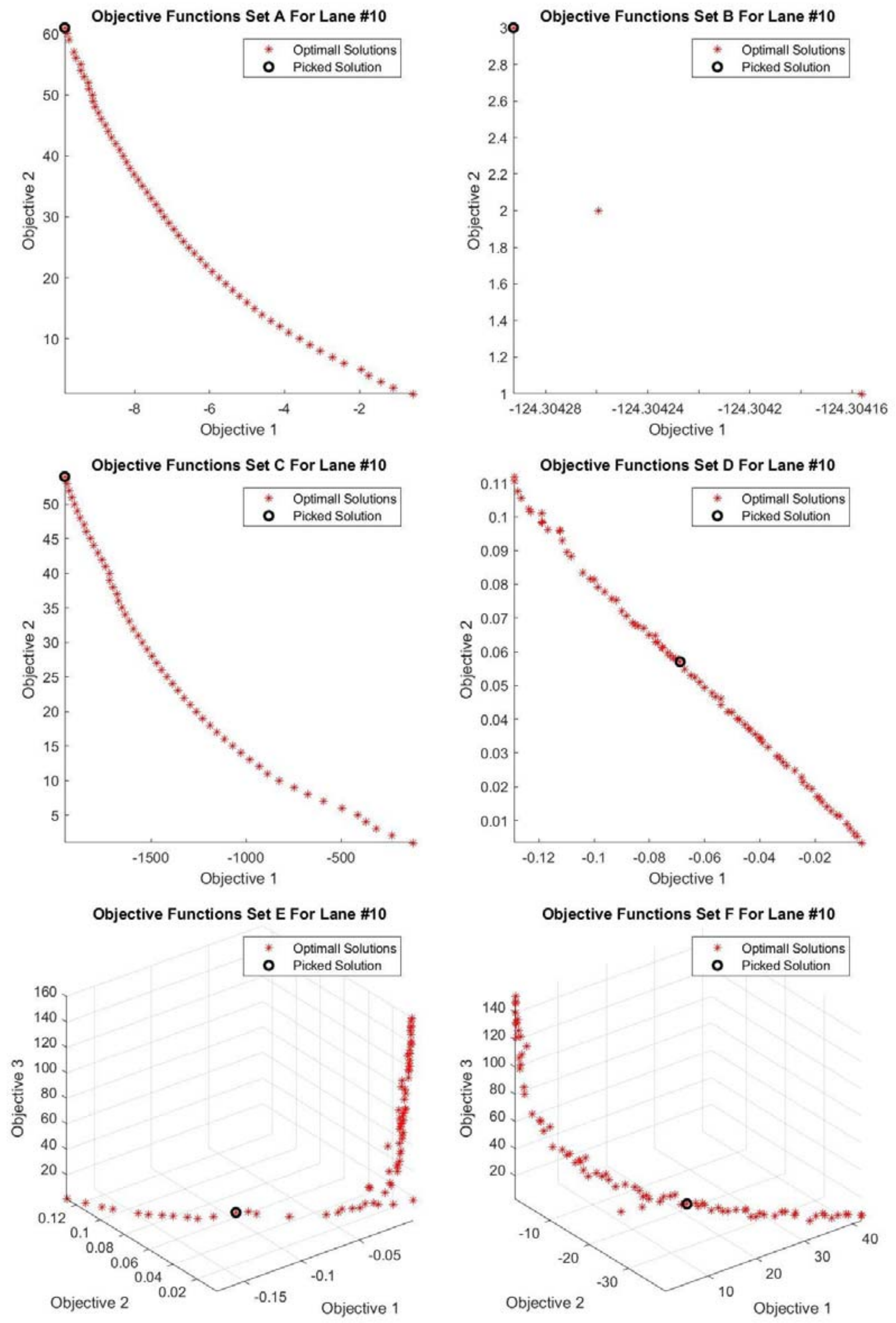


Figure A6-9. An illustration of six Pareto front outputs from NSGA II for Lane #10. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

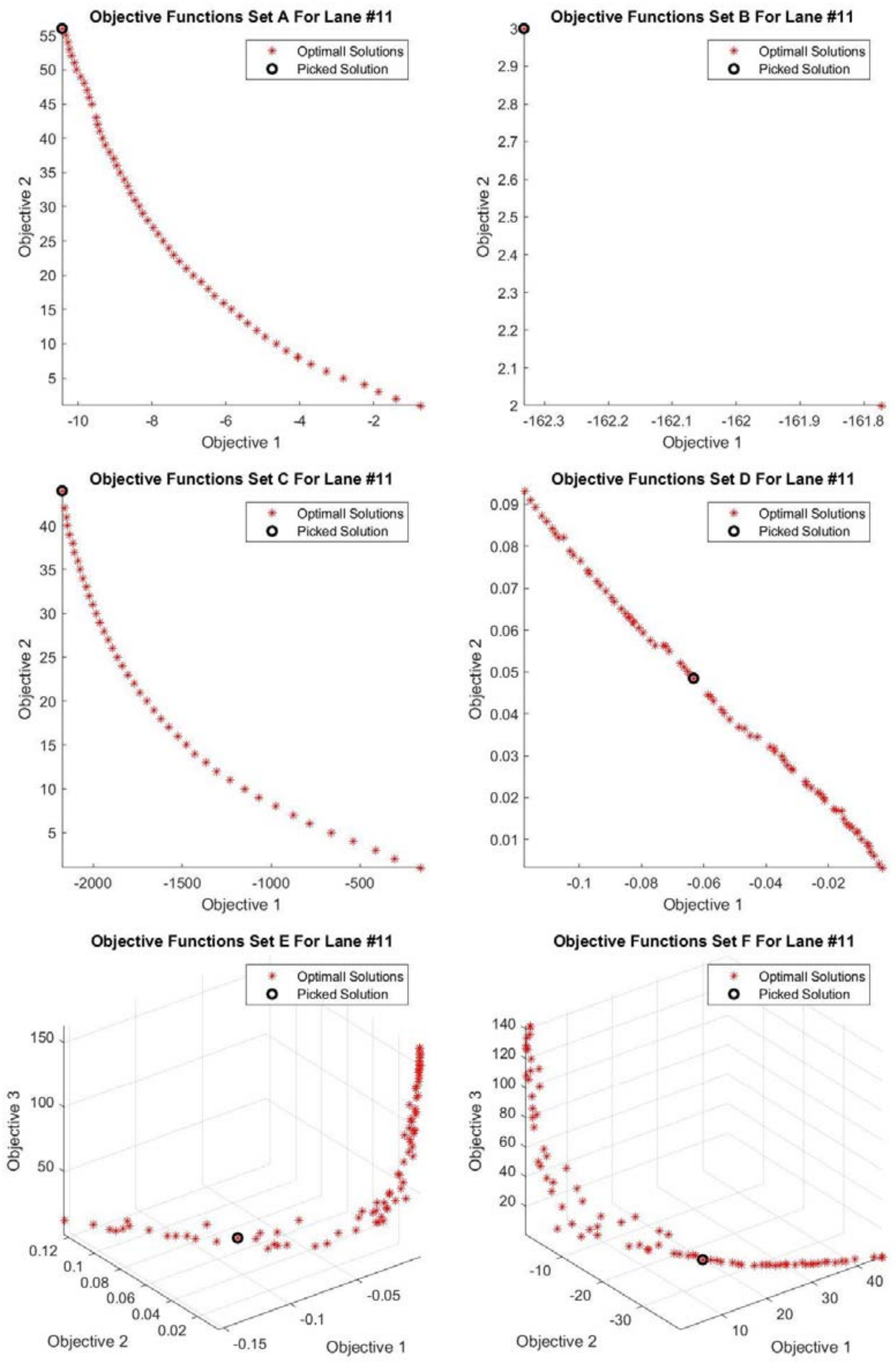


Figure A6-10. An illustration of six Pareto front outputs from NSGA II for Lane #11. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

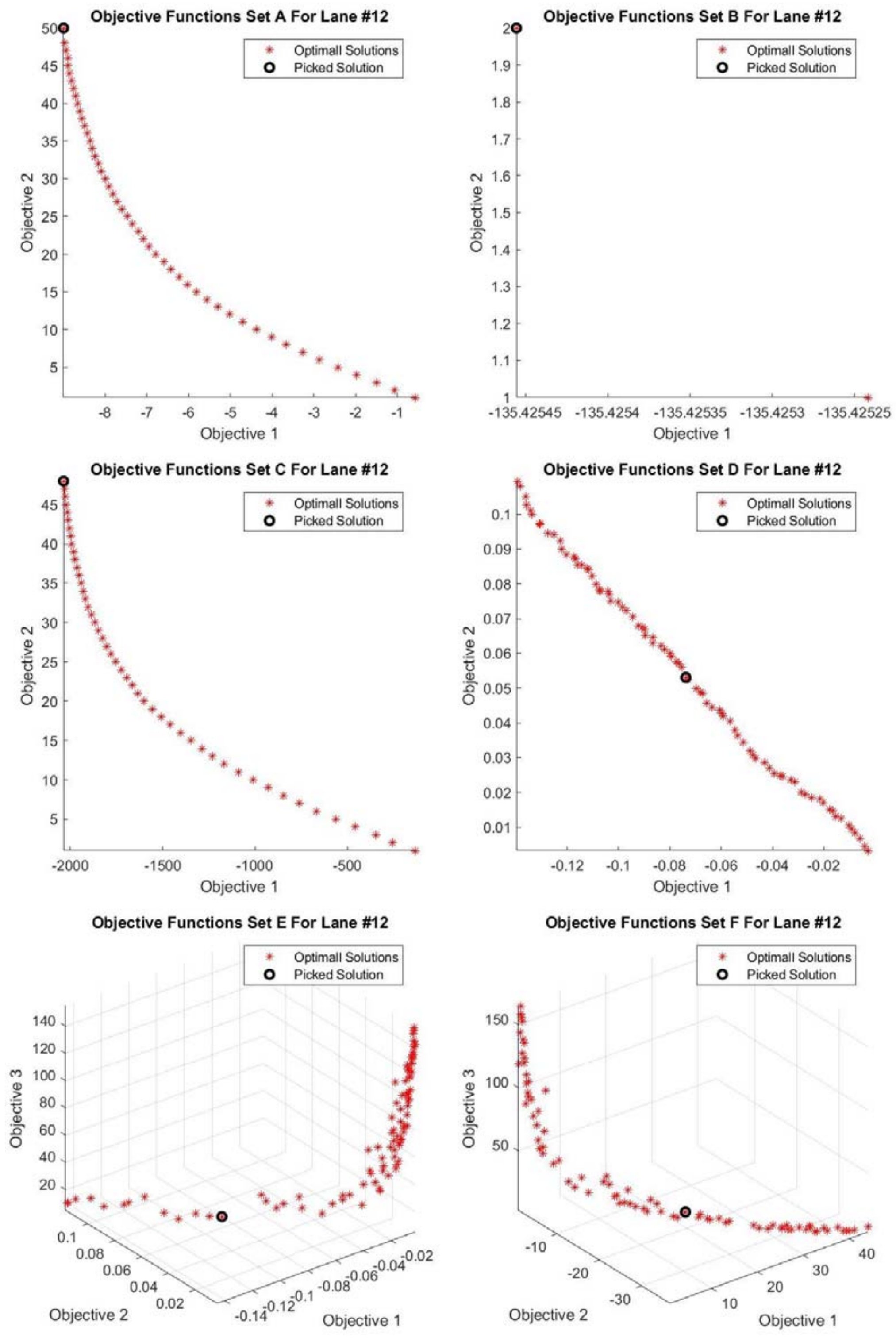


Figure A6-11. An illustration of six Pareto front outputs from NSGA II for Lane #12. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

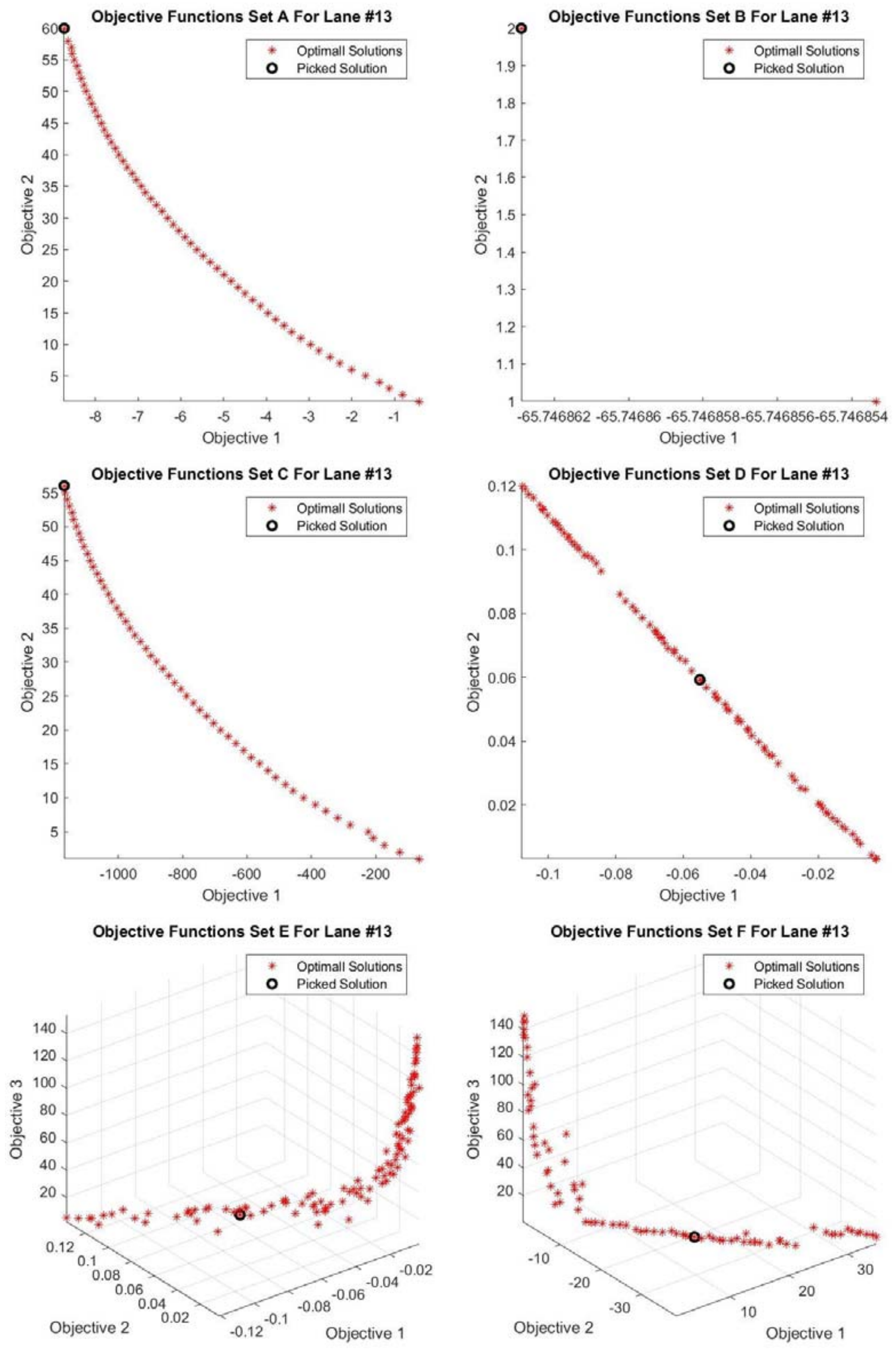


Figure A6-12. An illustration of six Pareto front outputs from NSGA II for Lane #13. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

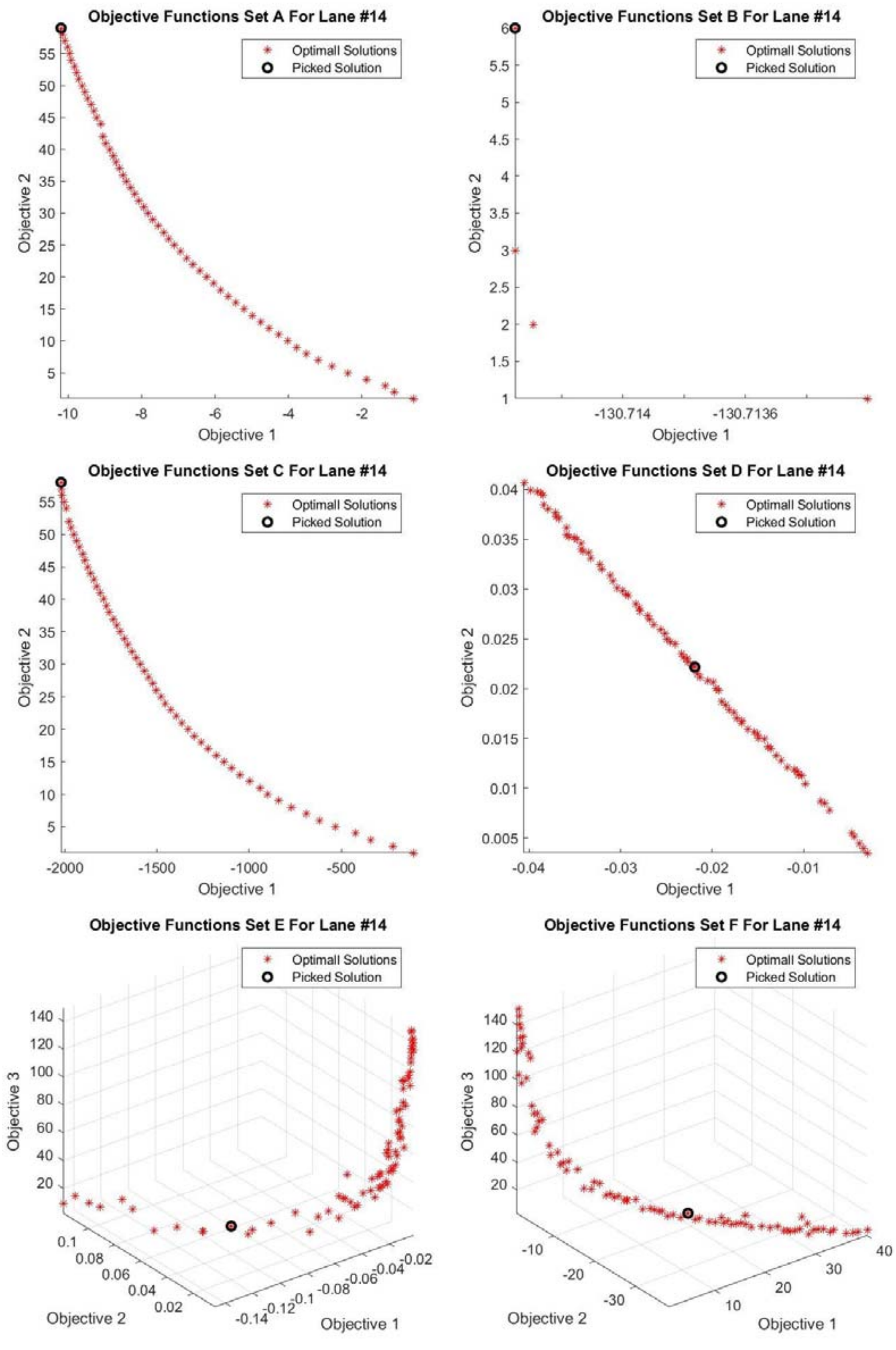


Figure A6-13. An illustration of six Pareto front outputs from NSGA II for Lane #14. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

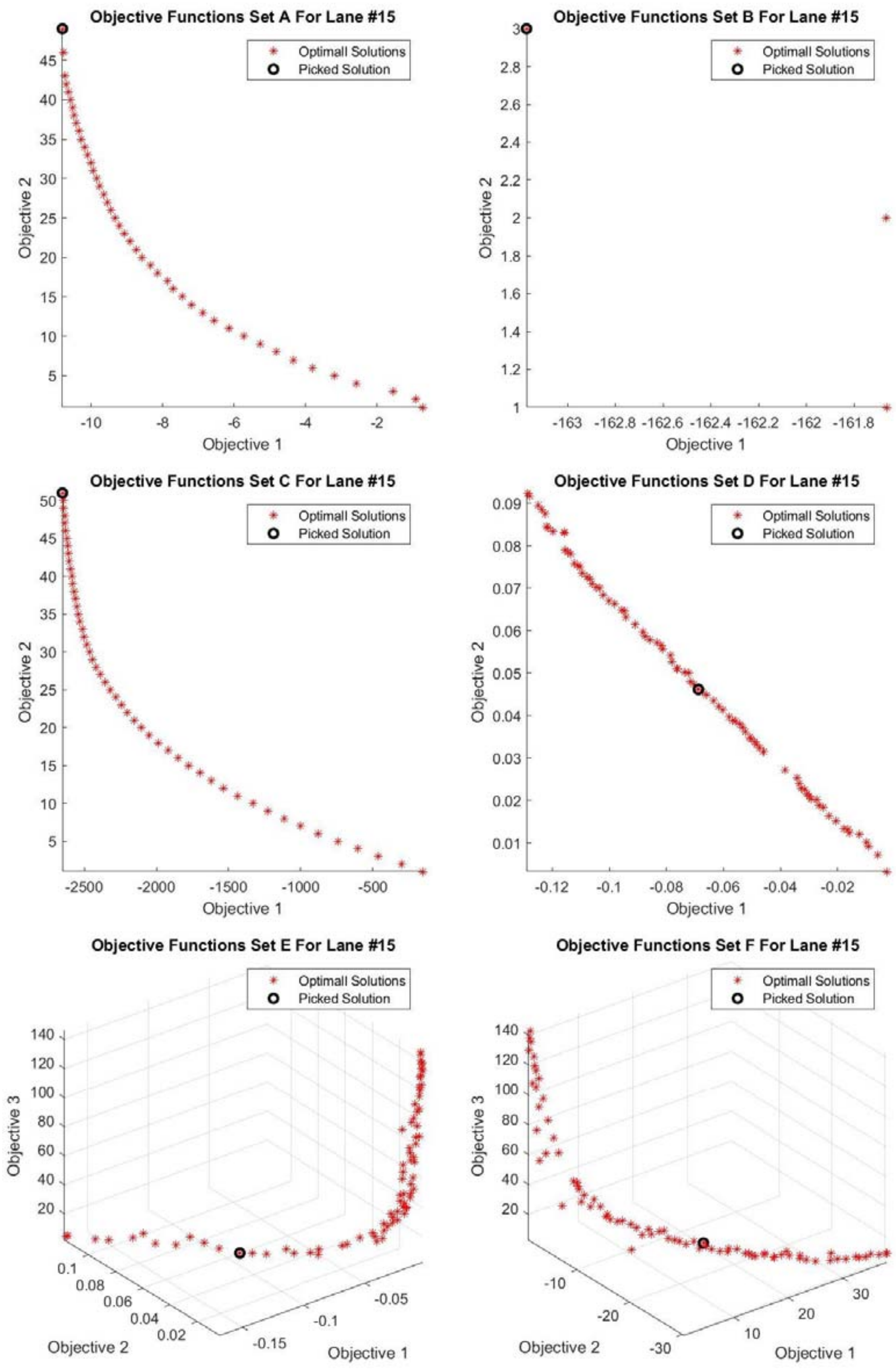


Figure A6-14. An illustration of six Pareto front outputs from NSGA II for Lane #15. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

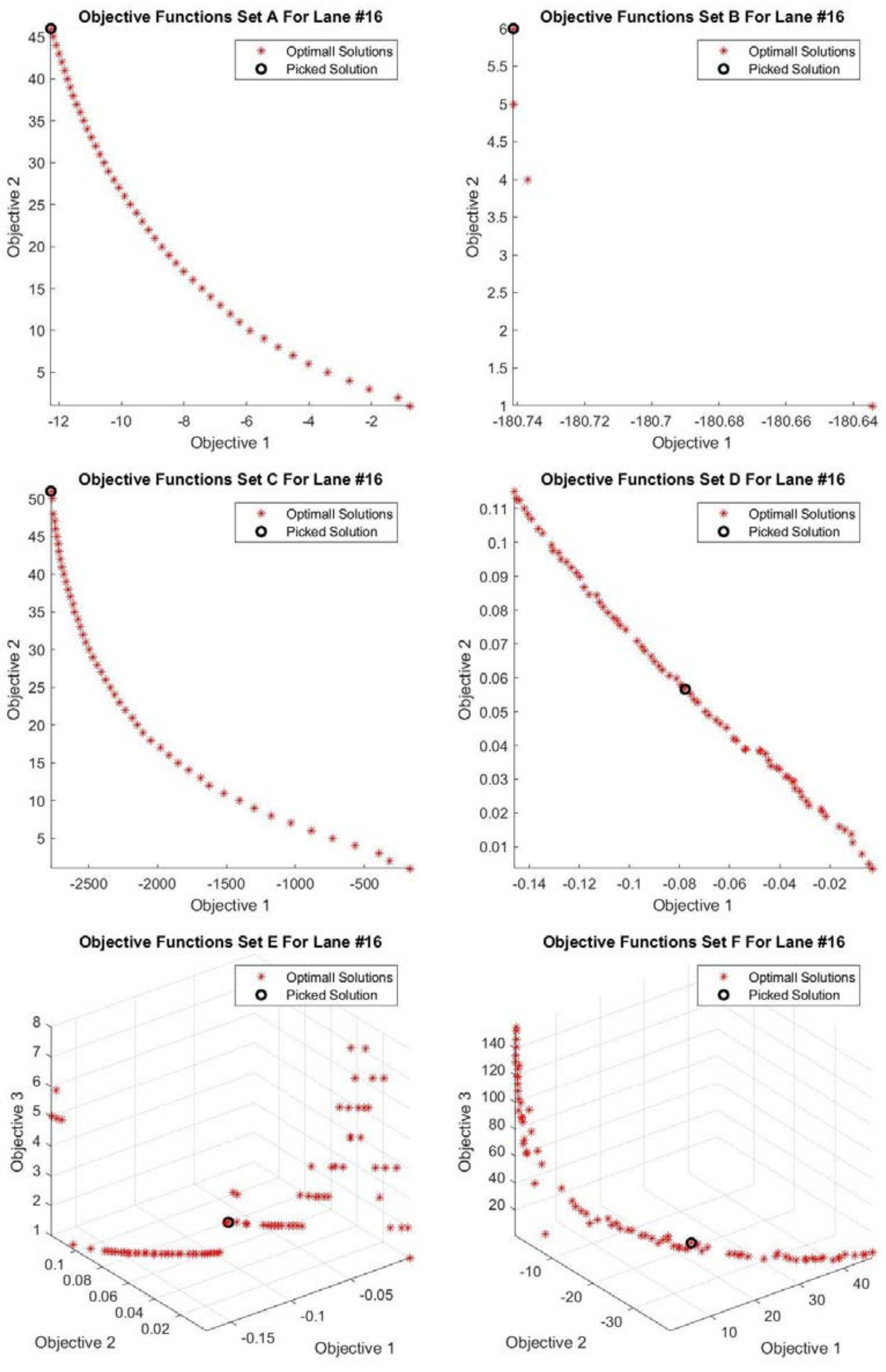


Figure A6-15. An illustration of six Pareto front outputs from NSGA II for Lane #16. Each Pareto front represents output solutions of the NSGA II algorithm after using six different sets of objective functions. The black circle is the chosen solution for generating the optimal LGF bank.

Appendix B: Experiment 3 Detailed Results

Table B1. The average road model error per look-ahead distance for Run #1

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
1	**12**	3.95	5.13	3.89	0.15	0.00	0.00
	13	4.52	6.61	4.61	0.30	0.00	0.00
	14	5.24	7.94	5.35	0.58	0.00	0.00
	15	6.37	9.32	6.27	1.12	0.00	0.00
	16	8.37	11.04	7.89	1.56	0.00	0.00
	17	11.04	13.09	10.00	1.76	0.01	0.00
	18	14.13	15.47	12.49	2.05	0.04	0.01
	19	17.03	17.58	14.63	2.42	0.12	0.05
	20	19.91	20.13	17.08	2.71	0.24	0.18
	21	22.89	23.21	19.69	3.01	0.36	0.43
	22	25.59	26.32	22.16	3.40	0.42	0.73
	23	28.12	28.96	24.53	3.82	0.48	1.01
	24	30.36	31.17	26.76	4.19	0.55	1.31
	25	32.86	34.34	29.45	4.62	0.70	1.80
	26	34.82	36.78	31.73	5.01	0.93	2.33
	27	36.57	39.41	33.81	5.36	1.25	2.98
	28	37.87	41.28	35.38	5.62	1.62	3.57
	29	39.12	43.25	36.89	5.88	2.05	4.24
	30	40.31	45.26	38.30	6.14	2.58	4.99
	31	41.50	47.24	39.66	6.43	3.18	5.82
	32	42.70	49.35	40.96	6.75	3.85	6.78
	33	43.60	50.67	41.91	7.04	4.39	7.61
	34	44.83	52.81	43.16	7.48	5.15	8.84
	35	45.75	54.21	44.11	7.86	5.82	9.90
	36	46.69	55.64	45.08	8.30	6.52	11.06
	37	47.58	57.11	46.02	8.78	7.24	12.26
	38	48.52	58.56	46.97	9.34	8.01	13.54
	39	49.52	60.02	47.97	9.98	8.89	14.93
	40	50.55	61.50	48.95	10.66	9.87	16.34

Table B2. The average road model error per look-ahead distance for Run #2

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
2	**12**	0.37	22.68	0.66	0.01	0.01	0.00
	13	1.06	33.23	1.50	0.12	0.07	0.03
	14	1.87	44.65	2.37	0.42	0.31	0.30
	15	2.54	54.20	3.05	0.68	0.58	0.63
	16	3.17	61.54	3.73	0.95	0.83	0.96
	17	3.77	66.69	4.37	1.34	1.08	1.32
	18	4.48	71.22	5.09	1.73	1.41	1.63
	19	5.15	74.10	5.71	2.02	1.71	1.84
	20	5.85	76.67	6.38	2.30	2.01	2.08
	21	6.59	78.95	7.07	2.56	2.31	2.43
	22	7.21	80.58	7.70	2.84	2.66	2.95
	23	7.69	81.76	8.23	3.10	2.93	3.59
	24	8.06	82.61	8.65	3.34	3.15	4.28
	25	8.59	83.63	9.25	3.66	3.44	5.19
	26	9.04	84.39	9.74	3.94	3.70	6.05
	27	9.52	85.08	10.29	4.28	4.02	7.01
	28	9.89	85.59	10.71	4.60	4.32	7.85
	29	10.31	86.07	11.15	4.98	4.67	8.73
	30	10.77	86.53	11.67	5.44	5.11	9.63
	31	11.31	86.99	12.24	6.00	5.64	10.61
	32	11.92	87.43	12.86	6.66	6.28	11.64
	33	12.34	87.73	13.31	7.18	6.78	12.45
	34	13.10	88.20	14.06	8.01	7.61	13.64
	35	13.68	88.51	14.62	8.64	8.25	14.54
	36	14.34	88.82	15.26	9.35	8.94	15.51
	37	15.03	89.13	15.91	10.11	9.70	16.50
	38	15.80	89.44	16.66	10.90	10.52	17.57
	39	16.62	89.73	17.46	11.77	11.39	18.66
	40	17.50	90.04	18.30	12.71	12.33	19.81

Table B3. The average road model error per look-ahead distance for Run #3

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
3	**12**	0.05	18.70	0.08	1.09	0.78	0.11
	13	0.28	29.94	0.29	1.10	0.68	0.22
	14	0.93	41.68	0.87	1.15	0.66	0.28
	15	2.26	52.12	2.07	1.17	0.64	0.32
	16	4.86	61.93	4.43	1.22	0.66	0.41
	17	9.17	70.57	8.59	1.34	0.75	0.68
	18	14.97	77.74	13.85	1.54	0.93	1.11
	19	20.87	82.80	18.82	1.76	1.11	1.58
	20	26.79	86.77	23.74	2.08	1.36	2.10
	21	32.42	89.56	28.50	2.46	1.68	2.73
	22	37.09	91.15	32.40	2.85	2.02	3.50
	23	40.97	92.07	35.47	3.22	2.39	4.34
	24	44.09	92.66	37.83	3.57	2.77	5.21
	25	47.26	93.20	40.13	4.11	3.37	6.39
	26	49.65	93.58	41.96	4.61	3.98	7.60
	27	51.75	93.92	43.69	5.20	4.71	8.91
	28	53.33	94.14	45.09	5.71	5.39	10.13
	29	54.84	94.38	46.56	6.35	6.23	11.47
	30	56.11	94.55	47.89	6.96	7.04	12.77
	31	57.39	94.74	49.29	7.70	8.01	14.20
	32	58.67	94.93	50.71	8.55	9.11	15.65
	33	59.60	95.04	51.78	9.19	9.96	16.79
	34	60.84	95.21	53.21	10.22	11.24	18.35
	35	61.77	95.32	54.27	10.96	12.23	19.56
	36	62.72	95.42	55.35	11.78	13.26	20.77
	37	63.67	95.53	56.42	12.65	14.35	22.02
	38	64.60	95.63	57.52	13.59	15.48	23.30
	39	65.57	95.74	58.62	14.57	16.68	24.63
	40	66.53	95.85	59.72	15.61	17.91	25.99

Table B4. The average road model error per look-ahead distance for Run #4

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
4	**12**	20.92	18.40	21.94	17.15	17.03	16.07
	13	24.58	24.64	25.49	20.21	20.21	19.36
	14	27.59	30.77	28.45	23.49	23.48	22.48
	15	30.39	36.28	31.18	26.06	26.00	25.23
	16	33.03	41.13	33.69	28.33	28.24	27.63
	17	35.26	45.38	35.88	30.19	30.13	29.74
	18	37.48	48.99	38.01	31.98	31.97	31.93
	19	39.70	51.92	40.10	33.91	33.90	34.19
	20	41.92	54.51	42.28	35.85	35.81	36.50
	21	44.05	56.87	44.44	37.77	37.67	38.89
	22	46.03	58.96	46.40	39.58	39.45	41.11
	23	47.89	60.80	48.24	41.37	41.31	43.18
	24	49.75	62.50	50.05	43.13	43.13	45.28
	25	51.32	63.80	51.45	44.65	44.75	46.98
	26	52.81	65.01	52.73	46.11	46.32	48.54
	27	54.13	66.05	53.90	47.40	47.70	49.98
	28	55.42	67.00	55.15	48.63	49.02	51.36
	29	56.31	67.52	56.01	49.53	50.00	52.27
	30	57.43	68.38	57.10	50.76	51.30	53.34
	31	58.29	68.96	57.98	51.73	52.28	54.16
	32	59.02	69.45	58.77	52.64	53.19	54.94
	33	59.57	69.86	59.35	53.41	53.98	55.57
	34	60.45	70.52	60.30	54.54	55.13	56.54
	35	61.07	70.85	60.98	55.40	56.03	57.29
	36	61.70	71.23	61.66	56.27	56.88	58.05
	37	62.45	71.71	62.52	57.25	57.83	59.04
	38	63.09	72.08	63.22	58.06	58.64	59.83
	39	63.54	72.37	63.73	58.79	59.38	60.49
	40	64.01	72.61	64.30	59.42	60.01	61.08

Table B5. The average road model error per look-ahead distance for Run #5

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
	12	18.65	10.65	18.53	15.67	14.99	15.41
	13	21.37	14.09	21.24	18.63	17.88	17.98
	14	23.69	17.82	23.69	21.05	20.25	20.34
	15	25.92	21.77	26.03	23.42	22.62	22.72
	16	28.03	25.38	28.39	25.90	25.07	25.33
	17	30.14	28.74	30.78	28.43	27.39	27.94
	18	32.33	32.06	33.18	30.88	29.54	30.60
	19	34.62	35.15	35.46	33.32	31.75	33.20
	20	36.85	38.01	37.54	35.56	33.94	35.71
	21	39.00	40.55	39.58	37.57	35.98	38.15
	22	40.99	42.88	41.62	39.48	37.88	40.40
	23	42.88	45.12	43.62	41.26	39.70	42.67
	24	44.58	47.16	45.48	43.00	41.52	44.67
	25	45.88	48.68	46.87	44.48	43.04	46.29
5	**26**	47.17	50.10	48.25	46.03	44.69	47.98
	27	48.39	51.38	49.55	47.64	46.44	49.69
	28	49.47	52.34	50.61	49.03	47.85	51.18
	29	50.56	53.30	51.63	50.28	49.13	52.42
	30	51.76	54.30	52.78	51.56	50.46	53.62
	31	53.07	55.43	54.15	52.96	51.91	54.86
	32	54.30	56.50	55.46	54.19	53.25	55.96
	33	55.40	57.45	56.65	55.19	54.37	56.87
	34	56.44	58.37	57.87	56.19	55.45	57.66
	35	57.26	58.99	58.75	56.95	56.25	58.21
	36	57.91	59.57	59.47	57.60	56.92	58.65
	37	58.64	60.37	60.28	58.34	57.69	59.21
	38	59.46	61.25	61.23	59.20	58.56	59.90
	39	60.40	62.28	62.21	60.14	59.50	60.56
	40	61.06	63.04	62.86	60.88	60.22	60.93

Table B6. The average road model error per look-ahead distance for Run #6

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
6	**12**	17.07	9.21	16.72	13.55	15.79	13.10
	13	20.35	10.99	20.05	16.97	19.03	16.42
	14	23.32	13.23	22.91	19.84	21.58	18.99
	15	26.22	16.16	25.57	22.16	23.66	21.29
	16	28.66	19.74	28.05	24.49	25.53	23.67
	17	30.94	23.71	30.41	26.48	27.31	25.95
	18	33.15	27.70	32.62	28.49	29.31	28.08
	19	35.37	31.48	34.78	30.57	31.56	30.08
	20	37.58	35.09	36.94	32.72	33.86	32.19
	21	39.62	38.31	38.88	34.74	36.03	34.22
	22	41.45	41.11	40.61	36.60	37.98	36.10
	23	43.16	43.70	42.21	38.36	39.85	37.94
	24	44.77	46.10	43.74	40.05	41.60	39.69
	25	46.02	47.82	44.95	41.35	42.92	40.98
	26	47.23	49.22	46.16	42.56	44.13	42.16
	27	48.77	50.83	47.69	44.09	45.59	43.73
	28	50.05	52.19	49.01	45.51	46.94	45.17
	29	50.92	53.17	49.95	46.68	47.95	46.27
	30	52.09	54.29	51.21	48.13	49.24	47.56
	31	53.28	55.36	52.46	49.64	50.50	48.72
	32	54.06	56.15	53.29	50.77	51.47	49.56
	33	54.91	56.95	54.19	51.97	52.48	50.47
	34	55.98	57.89	55.32	53.22	53.67	51.63
	35	56.88	58.65	56.27	54.24	54.70	52.63
	36	57.70	59.28	57.10	55.18	55.66	53.52
	37	58.38	59.74	57.78	55.96	56.47	54.28
	38	59.00	60.19	58.44	56.68	57.27	54.95
	39	59.68	60.72	59.21	57.43	58.15	55.70
	40	60.49	61.37	60.08	58.20	59.12	56.56

Table B7. The average road model error per look-ahead distance for Run #7

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
7	**12**	16.42	17.02	17.39	14.47	15.20	14.26
	13	19.48	20.88	20.10	17.44	18.22	17.15
	14	21.98	24.45	22.39	20.41	21.02	19.79
	15	24.36	27.75	24.61	22.85	23.40	22.17
	16	26.88	30.73	27.08	25.13	25.73	24.71
	17	29.35	33.40	29.44	27.31	28.03	27.21
	18	31.98	35.97	31.83	29.46	30.20	29.61
	19	34.37	38.32	34.09	31.60	32.31	31.76
	20	36.73	40.51	36.50	33.92	34.46	33.79
	21	38.87	42.58	38.76	36.09	36.41	35.81
	22	41.02	44.51	40.90	38.23	38.35	37.73
	23	43.21	46.52	43.06	40.42	40.52	39.86
	24	45.35	48.66	45.14	42.60	42.80	41.99
	25	47.31	50.67	47.00	44.47	44.90	43.95
	26	49.12	52.43	48.55	45.97	46.57	45.79
	27	50.97	54.19	50.20	47.32	48.11	47.48
	28	52.46	55.59	51.64	48.68	49.42	48.94
	29	53.77	56.86	52.88	49.93	50.55	50.30
	30	53.84	56.83	52.92	50.08	50.47	50.30
	31	54.80	57.66	53.90	51.29	51.42	51.16
	32	55.53	58.19	54.69	52.34	52.18	51.74
	33	56.37	58.80	55.59	53.44	52.92	52.46
	34	57.24	59.52	56.66	54.62	53.77	53.31
	35	57.89	60.17	57.64	55.63	54.59	54.10
	36	58.30	60.67	58.34	56.37	55.23	54.67
	37	58.64	61.25	58.85	57.01	55.85	55.19
	38	59.08	61.95	59.49	57.75	56.59	55.83
	39	59.48	62.64	60.00	58.42	57.25	56.50
	40	60.00	63.39	60.63	59.13	57.87	57.32

Table B8. The average road model error per look-ahead distance for Run #8

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
8	**12**	18.51	22.91	19.66	14.34	14.71	14.32
	13	22.06	27.52	23.51	17.53	17.89	18.12
	14	24.83	32.06	26.83	20.50	21.00	21.37
	15	27.24	36.73	29.59	23.13	23.84	24.24
	16	29.46	41.16	31.92	25.47	26.25	26.78
	17	31.63	45.19	34.05	27.74	28.59	29.19
	18	33.85	48.84	36.09	29.92	30.89	31.48
	19	36.07	52.27	38.15	32.03	33.06	33.74
	20	38.18	55.42	40.26	34.10	35.28	35.89
	21	39.72	57.87	41.77	35.59	36.97	37.49
	22	41.45	60.40	43.51	37.31	38.80	39.36
	23	43.08	62.66	45.28	38.91	40.43	41.19
	24	44.64	64.67	46.98	40.42	41.98	42.98
	25	46.18	66.45	48.64	41.87	43.40	44.76
	26	47.82	68.03	50.50	43.67	44.90	46.74
	27	49.52	69.48	52.33	45.41	46.51	48.56
	28	51.01	70.12	53.86	46.90	47.87	50.18
	29	52.34	71.27	55.26	48.26	49.20	51.72
	30	53.59	72.21	56.55	49.68	50.55	53.28
	31	54.65	73.34	57.67	50.95	51.79	54.66
	32	55.64	74.56	58.60	51.70	52.28	55.87
	33	56.67	75.17	59.51	51.97	52.80	56.98
	34	57.80	75.66	60.49	52.93	53.87	58.11
	35	58.79	75.94	61.35	53.79	54.81	58.99
	36	59.73	76.15	62.11	54.70	55.81	59.92
	37	60.69	76.47	63.05	55.89	57.05	61.01
	38	61.58	76.90	64.01	57.17	58.43	62.09
	39	62.27	77.05	64.77	58.40	59.97	62.87
	40	62.95	77.18	65.51	59.81	61.15	63.55

Table B9. The average road model error per look-ahead distance for Run #9

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
	12	19.42	18.29	19.28	17.37	17.82	15.80
	13	23.12	22.17	22.75	20.58	21.12	19.67
	14	26.61	26.81	26.08	23.85	24.35	23.09
	15	29.87	31.41	29.25	27.33	27.74	26.47
	16	32.92	35.44	32.27	30.07	30.51	29.96
	17	35.80	38.91	35.19	32.52	33.03	33.21
	18	38.52	42.37	37.99	34.91	35.49	36.01
	19	41.08	45.76	40.52	37.20	37.80	38.49
	20	43.40	49.09	42.86	39.53	40.15	40.97
	21	45.61	52.17	45.06	41.72	42.35	43.35
	22	47.44	54.90	46.99	43.71	44.30	45.40
	23	47.80	56.23	47.47	44.41	44.96	45.97
	24	49.14	58.30	48.89	45.88	46.41	47.43
	25	50.38	59.94	50.20	47.18	47.78	48.73
9	**26**	51.77	61.45	51.60	48.46	49.17	50.17
	27	52.99	62.74	52.84	49.67	50.49	51.53
	28	53.71	63.54	53.60	50.68	51.54	52.42
	29	54.73	64.62	54.60	51.81	52.64	53.48
	30	55.75	65.78	55.63	52.91	53.73	54.61
	31	56.34	66.54	56.21	53.63	54.43	55.33
	32	56.86	67.37	56.72	54.36	55.12	56.03
	33	57.53	68.11	57.40	55.17	55.91	56.82
	34	58.20	68.87	58.10	56.01	56.77	57.60
	35	58.74	69.41	58.72	56.70	57.46	58.26
	36	59.26	69.81	59.29	57.27	58.04	58.85
	37	59.66	70.07	59.75	57.80	58.50	59.35
	38	59.79	70.04	59.91	58.06	58.70	59.57
	39	60.02	70.04	60.15	58.34	58.92	59.85
	40	60.42	70.21	60.58	58.76	59.30	60.24

Table B10. The average road model error per look-ahead distance for Run #10

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
10	**12**	12.77	14.99	12.63	15.34	13.35	24.56
	13	15.86	18.85	15.69	18.85	16.73	30.50
	14	18.77	22.91	18.56	22.09	20.10	34.81
	15	21.29	26.63	21.14	24.97	23.06	38.24
	16	23.85	30.05	23.70	27.31	25.43	41.19
	17	26.40	33.20	26.15	29.48	27.38	43.92
	18	28.74	36.14	28.37	31.53	29.18	46.27
	19	30.97	38.89	30.51	33.46	31.06	48.43
	20	33.04	41.03	32.59	35.24	32.90	50.31
	21	35.03	43.17	34.63	36.97	34.58	52.11
	22	37.04	45.25	36.72	38.67	36.31	53.69
	23	39.14	47.20	38.92	40.54	38.21	55.31
	24	41.13	49.00	40.95	42.40	40.07	56.91
	25	42.87	50.70	42.78	44.08	41.77	58.45
	26	44.70	52.39	44.70	45.72	43.37	59.94
	27	46.48	53.94	46.59	47.43	45.05	61.35
	28	48.10	55.26	48.27	48.91	46.36	62.52
	29	49.57	56.44	49.76	50.07	47.02	63.11
	30	50.97	57.47	51.16	51.17	47.57	63.35
	31	52.08	58.23	52.22	51.96	48.28	63.82
	32	53.23	59.14	53.33	52.91	49.24	64.54
	33	54.37	60.17	54.45	53.97	50.73	65.30
	34	55.32	61.03	55.39	54.89	52.27	65.97
	35	56.17	61.81	56.23	55.78	53.23	66.49
	36	56.98	62.58	57.08	56.70	54.23	67.05
	37	57.58	61.90	57.74	56.39	54.07	67.39
	38	57.14	62.38	58.52	57.43	55.25	67.86
	39	58.09	62.96	59.48	58.53	56.42	68.44
	40	59.03	63.47	59.23	59.43	57.37	68.99

Table B11. The average road model error per look-ahead distance for Run #11

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
	12	6.79	24.55	6.64	6.93	7.44	4.89
	13	10.02	31.74	9.99	9.78	10.46	7.00
	14	13.28	39.29	13.35	13.85	13.97	10.71
	15	16.07	46.05	16.21	18.06	17.89	14.49
	16	18.53	52.12	18.67	21.41	21.89	17.85
	17	21.35	57.25	21.44	24.39	25.55	20.47
	18	23.97	61.51	24.02	26.97	28.75	22.83
	19	25.79	64.33	25.88	28.90	30.98	24.58
	20	27.78	67.17	27.92	31.06	33.30	26.80
	21	29.92	70.01	29.98	33.19	35.54	29.23
	22	32.09	72.30	32.07	35.10	37.64	31.70
	23	34.34	74.06	34.28	37.06	39.75	34.09
	24	36.36	75.43	36.30	38.71	41.48	36.18
	25	38.75	76.95	38.72	40.60	43.49	38.81
11	**26**	40.82	78.02	40.84	42.10	45.18	41.19
	27	42.75	78.79	42.76	43.50	46.77	43.56
	28	44.41	79.42	44.47	44.76	48.20	45.54
	29	45.97	79.96	46.09	45.94	49.59	47.44
	30	47.28	80.36	47.43	46.95	50.74	49.06
	31	48.43	80.66	48.60	47.90	51.76	50.61
	32	49.52	81.02	49.67	48.93	52.81	52.08
	33	50.17	81.23	50.27	49.55	53.41	53.03
	34	51.07	81.47	51.09	50.39	54.29	54.27
	35	51.60	81.65	51.59	50.99	54.90	55.09
	36	52.29	82.03	52.25	51.72	55.68	56.02
	37	52.96	82.31	52.89	52.42	56.39	56.82
	38	53.65	82.68	53.59	53.24	57.20	57.65
	39	54.51	83.20	54.41	54.17	58.21	58.54
	40	55.33	83.72	55.22	55.10	59.19	59.41

Table B12. The average road model error per look-ahead distance for Run #12

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
12	**12**	14.25	24.38	14.02	12.88	13.54	12.97
	13	19.33	30.15	19.16	18.30	18.75	17.67
	14	23.47	35.84	23.28	22.35	23.16	21.51
	15	26.99	41.10	26.61	25.69	26.79	24.99
	16	30.22	45.91	29.64	29.16	29.78	28.28
	17	33.16	50.01	32.56	32.79	32.75	31.34
	18	36.10	53.93	35.51	36.40	36.02	34.52
	19	38.79	56.96	38.18	39.24	38.64	37.02
	20	41.49	59.76	40.81	41.95	40.97	39.41
	21	43.99	62.03	43.30	44.36	43.07	41.54
	22	46.42	64.10	45.72	46.60	45.29	43.71
	23	48.47	65.69	47.85	48.58	47.23	45.64
	24	50.12	66.70	49.65	50.24	48.74	47.10
	25	52.02	67.65	51.59	52.04	50.52	48.77
	26	53.79	68.48	53.37	53.54	52.17	50.28
	27	55.52	69.29	55.11	54.91	53.89	51.83
	28	56.92	69.81	56.49	55.94	55.21	53.11
	29	58.14	70.29	57.74	56.89	56.39	54.41
	30	59.20	70.72	58.76	57.81	57.37	55.57
	31	60.47	71.47	60.04	59.13	58.66	56.95
	32	61.69	72.21	61.23	60.44	59.85	58.26
	33	62.50	72.35	62.02	61.05	60.66	58.90
	34	63.43	72.95	62.91	62.20	61.83	60.22
	35	64.08	73.32	63.55	62.89	62.64	61.12
	36	64.50	73.59	63.97	63.34	63.39	61.89
	37	64.89	73.94	64.44	63.79	64.16	62.67
	38	65.39	74.34	64.96	64.29	65.03	63.47
	39	66.17	74.84	65.68	64.97	65.88	64.40
	40	66.54	75.13	66.10	65.41	66.46	64.98

Table B13. The average road model error per look-ahead distance for Run #13

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
13	**12**	29.50	29.04	29.72	29.83	26.27	24.12
	13	33.38	33.89	33.68	33.94	30.60	28.39
	14	37.46	38.36	37.64	38.20	34.78	33.10
	15	40.97	42.10	41.27	41.75	38.42	37.05
	16	44.11	45.22	44.64	44.94	41.96	40.51
	17	46.93	47.58	47.57	47.75	45.24	43.53
	18	49.62	49.70	50.29	50.43	48.38	46.41
	19	51.97	51.52	52.58	52.79	51.14	48.96
	20	53.93	53.01	54.52	54.78	53.40	51.23
	21	55.89	54.39	56.42	56.68	55.55	53.54
	22	57.54	55.37	58.01	58.17	57.36	55.51
	23	59.13	56.37	59.60	59.64	59.02	57.49
	24	60.75	57.44	61.18	61.05	60.65	59.42
	25	62.27	58.48	62.64	62.40	62.14	61.25
	26	63.64	59.44	63.95	63.60	63.50	62.85
	27	64.77	60.31	65.02	64.59	64.70	64.19
	28	65.67	61.00	65.89	65.32	65.64	65.26
	29	66.38	61.48	66.59	65.83	66.32	66.11
	30	67.19	62.06	67.37	66.41	67.03	66.95
	31	67.91	62.72	68.11	67.03	67.67	67.67
	32	68.66	63.45	68.87	67.74	68.26	68.44
	33	69.28	64.10	69.49	68.36	68.68	69.04
	34	69.90	64.78	70.07	69.00	69.13	69.65
	35	70.47	65.38	70.60	69.60	69.54	70.15
	36	70.98	65.96	71.03	70.18	69.91	70.57
	37	71.50	66.68	71.53	70.76	70.31	71.00
	38	71.85	67.18	71.78	71.21	70.54	71.23
	39	72.20	67.85	72.08	71.66	70.83	71.54
	40	72.46	68.34	72.26	71.95	71.07	71.82

Table B14. The average road model error per look-ahead distance for Run #14

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
14	**12**	13.84	29.73	13.84	13.41	12.29	12.49
	13	17.06	35.70	17.06	16.69	15.93	15.80
	14	20.70	41.71	20.67	20.93	20.17	19.85
	15	24.77	46.94	24.70	24.98	24.60	24.14
	16	28.60	51.31	28.53	28.54	28.59	27.81
	17	31.94	54.82	31.87	31.57	31.94	31.03
	18	35.24	57.93	35.18	34.62	35.05	34.15
	19	38.01	59.87	38.04	37.08	37.42	36.64
	20	41.04	61.79	41.09	39.66	39.79	39.46
	21	44.06	63.50	43.98	42.41	42.21	42.33
	22	46.27	64.67	46.19	44.86	44.22	44.78
	23	48.26	65.67	48.18	47.11	46.18	47.02
	24	50.14	66.61	50.10	49.20	48.19	49.16
	25	52.02	67.70	51.97	51.23	50.22	51.17
	26	53.60	68.49	53.53	52.91	51.89	52.79
	27	54.81	69.10	54.77	54.13	53.21	53.97
	28	55.93	69.65	55.92	55.31	54.45	55.01
	29	56.84	70.02	56.87	56.23	55.44	55.82
	30	57.85	70.65	57.97	57.29	56.47	56.79
	31	58.78	71.23	59.00	58.37	57.48	57.76
	32	59.64	71.71	59.88	59.40	58.40	58.71
	33	60.34	72.11	60.54	60.17	59.05	59.43
	34	61.15	72.57	61.29	61.09	59.76	60.40
	35	61.94	72.89	61.99	61.97	60.44	61.34
	36	62.59	72.95	62.55	62.60	60.86	62.13
	37	63.44	73.20	63.30	63.43	61.55	63.08
	38	64.15	73.51	63.94	64.21	62.28	63.93
	39	64.66	73.61	64.37	64.74	62.88	64.53
	40	65.25	73.71	64.90	65.40	63.61	65.19

Table B15. The average road model error per look-ahead distance for Run #15

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
15	**12**	32.02	31.73	31.32	27.38	31.32	27.32
	13	37.04	36.94	35.81	32.21	34.12	30.30
	14	40.82	41.41	39.28	35.61	36.08	33.34
	15	44.03	45.60	42.33	39.16	38.17	36.53
	16	46.55	49.01	44.56	42.18	40.56	39.36
	17	49.64	52.17	47.19	45.08	44.00	42.67
	18	52.76	55.38	50.29	47.90	46.76	45.81
	19	54.89	58.00	52.86	50.32	48.89	48.47
	20	56.43	59.82	54.69	52.18	51.02	50.69
	21	57.88	61.43	56.33	53.78	53.99	52.56
	22	58.90	62.81	57.60	55.32	56.00	53.76
	23	47.53	51.70	46.37	44.51	45.10	42.27
	24	47.28	51.73	46.26	44.87	45.21	42.10
	25	48.30	53.04	47.37	46.43	46.54	43.33
	26	49.27	54.24	48.42	48.02	48.10	44.68
	27	50.43	55.34	49.57	49.80	49.73	46.31
	28	51.45	56.11	50.66	50.92	51.16	48.07
	29	52.61	56.93	51.82	52.33	52.69	49.58
	30	53.67	57.51	52.93	53.24	53.79	50.68
	31	54.73	58.11	54.04	54.06	54.84	51.63
	32	55.75	58.69	55.07	54.85	55.81	52.49
	33	56.77	59.24	56.06	55.58	56.83	53.35
	34	57.64	59.66	56.90	56.13	57.72	54.07
	35	58.24	59.85	57.46	56.51	58.28	54.57
	36	58.86	60.11	58.02	56.98	58.82	55.07
	37	59.49	60.48	58.62	57.54	59.43	55.65
	38	59.98	60.75	59.11	58.05	60.00	56.15
	39	60.51	61.06	59.63	58.57	60.57	56.80
	40	60.90	61.32	60.05	58.86	60.95	57.36

Table B16. The average road model error per look-ahead distance for Run #16

The road model average error rate (10^{-2})							
Run #	**Look-ahead Distance**	**Objective Functions Set**					
		A	**B**	**C**	**D**	**E**	**F**
16	**12**	33.82	23.19	33.76	24.47	21.07	30.47
	13	38.04	26.65	38.06	28.40	24.79	34.46
	14	41.32	30.07	41.38	32.56	28.82	37.72
	15	44.04	33.48	44.09	36.26	32.47	40.55
	16	46.24	36.78	46.31	39.77	35.98	43.07
	17	48.03	39.77	48.08	43.08	39.45	45.35
	18	49.66	42.76	49.68	46.07	42.87	47.61
	19	51.20	45.55	51.17	48.71	45.80	49.72
	20	52.82	48.33	52.69	51.30	48.41	51.75
	21	54.52	50.93	54.28	53.65	50.70	53.71
	22	55.89	53.16	55.58	55.47	52.59	55.24
	23	57.26	55.24	56.86	57.12	54.49	56.74
	24	58.64	57.16	58.18	58.69	56.25	58.19
	25	59.79	58.74	59.32	59.92	57.58	59.36
	26	61.00	60.27	60.53	61.06	58.82	60.52
	27	62.15	61.65	61.66	62.08	60.01	61.59
	28	63.28	62.84	62.79	63.03	61.05	62.57
	29	64.27	63.80	63.73	63.76	61.92	63.39
	30	65.28	64.74	64.68	64.46	62.80	64.27
	31	66.33	65.68	65.72	65.21	63.69	65.20
	32	67.19	66.53	66.57	65.83	64.44	66.04
	33	67.93	67.30	67.30	66.51	65.17	66.82
	34	68.56	68.00	67.90	67.28	65.90	67.46
	35	68.93	68.52	68.24	67.87	66.44	67.87
	36	69.41	69.16	68.70	68.54	67.07	68.39
	37	69.91	69.75	69.19	69.21	67.70	68.89
	38	70.17	70.10	69.48	69.65	68.10	69.12
	39	70.39	70.46	69.74	70.12	68.52	69.33
	40	70.56	70.69	69.97	70.46	68.87	69.50

Appendix C: Experiment 4 Detailed Results

Table C1. The average road model error per look-ahead distance for TLGF with 24 filters

The road model average error rate (10^{-2})								
Look-ahead Distance	**Run #**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
12	0.00	0.00	0.00	1.66	1.70	1.62	1.76	1.69
13	0.00	0.00	0.00	1.66	1.73	1.62	1.76	1.69
14	0.00	0.00	0.00	1.66	1.82	1.62	1.76	1.69
15	0.00	0.00	0.00	1.69	1.89	1.62	1.79	1.71
16	0.00	0.00	0.00	1.74	2.30	1.65	1.98	1.82
17	0.00	0.00	0.00	1.82	2.96	2.01	2.30	2.02
18	0.00	0.00	0.00	1.97	3.80	2.60	2.71	2.31
19	0.00	0.00	0.00	2.30	5.17	3.38	3.47	2.84
20	0.00	0.00	0.00	3.26	6.60	4.22	4.06	3.63
21	0.00	0.00	0.00	5.12	7.44	5.51	5.03	4.41
22	0.00	0.00	0.00	7.01	8.21	6.54	5.96	5.30
23	0.00	0.00	0.00	8.54	9.07	7.23	6.95	6.25
24	0.00	0.00	0.00	9.95	9.86	7.74	8.10	7.41
25	0.00	0.00	0.00	11.07	10.18	7.94	9.47	8.72
26	0.00	0.00	0.00	12.06	10.57	7.99	10.57	9.75
27	0.01	0.01	0.01	12.89	11.10	8.47	11.54	10.81
28	0.02	0.01	0.02	13.77	11.47	9.05	12.47	11.31
29	0.09	0.03	0.03	14.23	11.80	9.68	13.59	12.61
30	0.25	0.05	0.03	14.92	12.26	10.65	13.53	13.58
31	0.61	0.05	0.03	15.46	12.91	11.42	14.60	14.73
32	1.07	0.06	0.04	15.97	13.52	11.95	15.53	15.77
33	1.42	0.06	0.05	16.35	14.13	12.69	16.54	16.60
34	2.03	0.07	0.07	17.21	14.92	13.77	17.71	17.47
35	2.64	0.08	0.10	17.80	15.61	14.82	18.98	18.09
36	3.40	0.10	0.13	18.43	16.39	15.84	19.80	18.71
37	4.20	0.14	0.17	19.16	17.34	16.79	20.53	19.45
38	5.18	0.21	0.21	19.61	18.41	17.65	21.24	20.29
39	6.23	0.31	0.26	20.02	19.58	18.43	21.84	21.09
40	7.29	0.44	0.32	20.44	20.41	19.23	22.56	22.06

Table C1. The average road model error per look-ahead distance for TLGF with 24 filters, continued

Look-ahead Distance	Run #							
	9	10	11	12	13	14	15	16
12	1.74	1.71	0.00	1.28	3.50	0.96	0.76	3.66
13	1.74	1.71	0.00	1.28	3.75	0.98	0.77	3.86
14	1.76	1.71	0.00	1.39	4.27	1.18	0.83	4.23
15	1.78	1.72	0.02	1.56	4.82	1.54	1.14	4.75
16	1.80	1.74	0.08	1.54	5.65	1.89	3.10	5.49
17	1.90	1.81	0.52	1.64	6.46	2.45	5.25	6.21
18	2.11	1.90	1.01	2.00	7.55	3.38	6.36	7.03
19	2.42	2.12	1.34	2.59	8.80	4.22	8.56	7.77
20	3.02	2.58	1.81	3.46	10.02	5.26	10.43	8.65
21	3.10	3.24	2.09	4.65	11.39	6.24	12.57	9.74
22	5.10	4.05	2.36	6.31	12.30	7.06	14.24	10.93
23	5.76	4.83	3.13	7.62	13.32	8.12	7.53	12.47
24	7.96	5.92	4.45	8.82	14.60	9.89	9.25	14.24
25	9.80	7.45	6.24	10.55	15.83	11.73	11.47	15.82
26	11.53	8.72	8.09	12.08	17.04	13.18	13.23	17.63
27	12.93	10.15	9.89	13.95	18.24	14.46	14.70	19.44
28	14.02	11.13	11.58	15.47	19.40	15.54	16.12	21.20
29	15.84	12.28	13.00	17.01	20.60	16.44	17.58	22.78
30	17.57	13.12	14.05	18.26	22.08	17.27	18.99	24.32
31	19.05	14.40	15.13	19.68	23.67	18.42	20.36	25.86
32	20.55	15.65	16.36	21.17	25.41	20.12	21.58	27.27
33	21.96	16.84	17.13	21.98	27.11	21.73	22.83	28.58
34	23.26	18.01	17.98	23.50	28.97	23.44	24.02	29.84
35	24.37	18.86	18.75	24.59	30.63	24.82	25.01	30.98
36	25.29	19.67	19.81	25.73	32.28	25.86	26.21	32.36
37	26.01	20.13	20.90	27.05	33.93	26.97	27.48	33.67
38	26.41	20.57	21.92	28.40	35.33	27.97	28.72	34.78
39	26.91	21.12	23.16	29.77	36.88	28.71	30.11	35.87
40	27.66	21.76	24.36	30.73	38.18	29.43	31.29	36.94

Appendix D: Grouping Dominant Orientations for Ill-Structured Road Following MATLAB Code.

```
function [row, col] = findVanishingPoint(im)
    IM = fft2(im);
    ROWS = size(IM,1); COLS = size(IM,2);
    PERIOD = 2^floor(log2(COLS)-5)+2;
    SIZE = floor(10*PERIOD/pi);
    SIGMA = SIZE/9;
    NORIENT = 72;
    E = 8;
    [C, S] = createGaborBank(SIZE, PERIOD, SIGMA, NORIENT, ROWS, COLS, E);

    D = ones(ROWS, COLS);
    AMAX = ifftshift(real(ifft2(C{1}.*IM)).^2+real(ifft2(S{1}.*IM))).^2;
    for n=2:NORIENT
        A = ifftshift(real(ifft2(C{n}.*IM)).^2+real(ifft2(S{n}.*IM))).^2;
        D(find(A > AMAX)) = n;
        AMAX = max(A, AMAX);
    end
    T = mean(AMAX(:))-3*std(AMAX(:));
    VOTE = zeros(ROWS, COLS);
    for row=round(1+SIZE/2):round(ROWS-SIZE/2)
        for col=round(1+SIZE/2):round(COLS-SIZE/2)
            if (AMAX(row,col) > T)
                indices = lineBresenham(ROWS, COLS, col, row, D(row, col)*pi/NORIENT-pi/2);
                VOTE(indices) = VOTE(indices)+AMAX(row,col);
            end
        end
    end
    M=1;
    [~, index] = sort(-VOTE(:));
    col = floor((index(1:M)-1) / ROWS)+1;
    row = mod(index(1:M)-1, ROWS)+1;
    col = round(mean(col));
    row = round(mean(row));
end
```

```
function [C, S] = createGaborBank(SIZE, PERIOD, SIGMA, NORIENT, ROWS, COLS, E)
    orientations=[1:NORIENT];
    for n=orientations
        [C{n}, S{n}] = gabormask(SIZE, SIGMA, PERIOD, n*pi/NORIENT, E);
        C{n} = fft2(padWithZeros(C{n}, ROWS, COLS));
        S{n} = fft2(padWithZeros(S{n}, ROWS, COLS));
    end
end

function [cmask, smask] = gabormask(Size, sigma, period, orient, E)
if isempty(period) && isempty(sigma); sigma = 5; end
if isempty(period); period = sigma*2*sqrt(2); end
if isempty(sigma); sigma = period/(2*sqrt(2)); end
if isempty(Size); Size = 2*round(2.575*sigma) + 1; end

if length(Size) == 1
    sx = Size-1; sy = sx;
elseif all(size(Size) == [1 2])
    sy = Size(1)-1; sx = Size(2)-1;
else
    error('Size must be scalar or 1-by-2 vector');
end
hy = sy/2; hx = sx/2;
[x, y] = meshgrid(-hx:sx-hx, -hy:sy-hy);
omega = 2*pi/period;
cs = omega * cos(orient);
sn = omega * sin(orient);
k = -1/(E*sigma*sigma);
g = exp(k * (E*x.*x + y.*y));
xp = x * cs + y * sn;
cx = cos(xp);
cmask = g .* cx;
sx = sin(xp);
smask = g .* sx;
cmask = cmask - mean(cmask(:));
cmask = cmask/sum(abs(cmask(:)));
smask = smask - mean(smask(:));
smask = smask/sum(abs(smask(:)));
end
```

```
function [i] = lineBresenham(H,W,Sx,Sy,angle)
    k = tan(angle);
    if (angle == pi || angle == 0)
        Ex = W;
        Ey = Sy;
        Sx = 1;
    elseif (angle == pi/2)
```

```
        Ey = 1;
        i = (Sx-1)*H+[Ey:Sy];
        return;
    elseif k>0 && k < (Sy-1)/(W-Sx)
        Ex = W;
        Ey = round(Sy-tan(angle)*(Ex-Sx));
    elseif k < 0 && abs(k) < (Sy-1)/(Sx-1)
        Ex = 1;
        Ey = round(Sy-tan(angle)*(Ex-Sx));
    else
        Ey = 1;
        Ex = round((Sy-1)/tan(angle)+Sx);
    end
    Dx = Ex - Sx;
    Dy = Ey - Sy;
    iCoords=1;
    if(abs(Dy) <= abs(Dx))
        if(Ex >= Sx)
            D = 2*Dy + Dx;
            IncH = 2*Dy;
            IncD = 2*(Dy + Dx);
            X = Sx;
            Y = Sy;
            i(iCoords) = (Sx-1)*H+Sy;
            iCoords = iCoords + 1;
            while(X < Ex)
                if(D >= 0)
                    D = D + IncH;
                    X = X + 1;
                else
                    D = D + IncD;
                    X = X + 1;
                    Y = Y - 1;
                end
                i(iCoords) = (X-1)*H+Y;
                iCoords = iCoords + 1;
            end
        else
            D = -2*Dy + Dx;
            IncH = -2*Dy;
            IncD = 2*(-Dy + Dx);
            X = Sx;
            Y = Sy;
            i(iCoords) = (Sx-1)*H+Sy;
            iCoords = iCoords + 1;
            while(X > Ex)
                if(D <= 0)
                    D = D + IncH;
                    X = X - 1;
                else
                    D = D + IncD;
                    X = X - 1;
                    Y = Y - 1;
                end
                i(iCoords) = (X-1)*H+Y;
                iCoords = iCoords + 1;
            end
        end
    else
        Tmp = Ex;
        Ex = Ey;
        Ey = Tmp;
        Tmp = Sx;
        Sx = Sy;
        Sy = Tmp;
        Dx = Ex - Sx;
        Dy = Ey - Sy;
        if(Ex >= Sx)
            D = 2*Dy + Dx;
            IncH = 2*Dy;
            IncD = 2*(Dy + Dx);
            X = Sx;
            Y = Sy;
            i(iCoords) = (Sy-1)*H+Sx;
            iCoords = iCoords + 1;
            while(X < Ex)
                if(D >= 0)
                    D = D + IncH;
                    X = X + 1;
                else
                    D = D + IncD;
                    X = X + 1;
                    Y = Y - 1;
                end
                i(iCoords) = (Y-1)*H+X;
                iCoords = iCoords + 1;
            end
        else
            D = -2*Dy + Dx;
            IncH = -2*Dy;
            IncD = 2*(-Dy + Dx);
            X = Sx;
            Y = Sy;
            i(iCoords) = (Sy-1)*H+Sx;
            iCoords = iCoords + 1;
            while(X > Ex)
                if(D <= 0)
                    D = D + IncH;
                    X = X - 1;
                else
                    D = D + IncD;
                    X = X - 1;
                    Y = Y - 1;
                end
                i(iCoords) = (Y-1)*H+X;
                iCoords = iCoords + 1;
            end
        end
    end
end
```

References

[1] Davies, B., and Lienhart, R., editors. Using CART to segment road images. Electronic Imaging 2006; 2006: SPIE.

[2] Nefian, A. V., and Bradski, G. R., editors. Detection of Drivable Corridors for Off-Road Autonomous Navigation. 2006 International Conference on Image Processing; 2006 8-11 Oct. 2006.

[3] Crisman, J. D., and Thorpe, C. E., "SCARF: a color vision system that tracks roads and intersections," IEEE Transactions on Robotics and Automation, 9(1), 49-58 (1993).

[4] Jinyou, Z., and Nagel, H., editors. Texture-based segmentation of road images. Proceedings of the Intelligent Vehicles '94 Symposium; 1994 24-26 Oct. 1994.

[5] Paetzold, F., and Franke, U., "Road recognition in urban environment," Image and Vision Computing, 18(5), 377-387 (2000).

[6] Taylor, C. J., Malik, J., and Weber, J., editors. A real-time approach to stereopsis and lane-finding. Proceedings of Conference on Intelligent Vehicles; 1996 19-20 Sept. 1996.

[7] Thrun, S., Montemerlo, M., and Aron, A., editors. Probabilistic Terrain Analysis For High-Speed Desert Driving. Robotics: Science and Systems; 2006.

[8] Dahlkamp, H., Kaehler, A., Stavens, D. *et al.*, editors. Self-supervised Monocular Road Detection in Desert Terrain. Robotics: Science and Systems; 2006.

[9] Pan, S. J., and Yang, Q., "A Survey on Transfer Learning," IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345-1359 (2010).

[10] Dong-chen, H., and Li, W., "Texture Unit, Texture Spectrum, And Texture Analysis," IEEE Transactions on Geoscience and Remote Sensing, 28(4), 509-512 (1990).

[11] Ojala, T., Pietikainen, M., and Maenpaa, T., "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7), 971-987 (2002).

[12] Lahdenoja, O., Poikonen, J., and Laiho, M., "Towards Understanding the Formation of Uniform Local Binary Patterns," ISRN Machine Vision, 2013, 20 (2013).

[13] Ahonen, T., Hadid, A., and Pietikainen, M., "Face Description with Local Binary Patterns: Application to Face Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12), 2037-2041 (2006).

[14] Dalal, N., and Triggs, B., editors. Histograms of oriented gradients for human detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05); 2005 25-25 June 2005.

[15] Mikolajczyk, K., and Schmid, C., "A performance evaluation of local descriptors," IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10), 1615-1630 (2005).

[16] Bellavia, F., Tegolo, D., and Trucco, E., editors. Improving SIFT-based Descriptors Stability to Rotations. 2010 20th International Conference on Pattern Recognition; 2010 23-26 Aug. 2010.

[17] Liang, Y., Liu, L., Xu, Y. *et al.*, editors. Multi-task GLOH feature selection for human age estimation. 2011 18th IEEE International Conference on Image Processing; 2011 11-14 Sept. 2011.

[18] Abdulmunem, A., Lai, Y., and Sun, X., editors. 3D GLOH features for human action recognition. 2016 23rd International Conference on Pattern Recognition (ICPR); 2016 4-8 Dec. 2016.

[19] Yu, Q., Zhou, S., Wu, P. *et al.*, editors. High-performance SAR image registration algorithm using guided filter &amp;amp; ROEWA-based SIFT framework. 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS); 2017 6-9 Nov. 2017.

[20] Bay, H., Ess, A., Tuytelaars, T. *et al.*, "Speeded-Up Robust Features (SURF)," Computer Vision and Image Understanding*,* 110(3), 346-359 (2008).

[21] Szczuko, P., editor Influence of image transformations and quality degradations on SURF detector efficiency. 2013 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA); 2013 26-28 Sept. 2013.

[22] Cheng, C., Wang, X., and Li, X., editors. UAV image matching based on surf feature and harris corner algorithm. 4th International Conference on Smart and Sustainable City (ICSSC 2017); 2017 5-6 June 2017.

[23] Zhang, R., Ming, Y., and Sun, J., editors. Hand gesture recognition with SURF-BOF based on Gray threshold segmentation. 2016 IEEE 13th International Conference on Signal Processing (ICSP); 2016 6-10 Nov. 2016.

[24] Viola, P., and Jones, M., editors. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001; 2001 8-14 Dec. 2001.

[25] Mikolajczyk, K., and Schmid, C., editors. Indexing based on scale invariant interest points. Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on; 2001: IEEE.

[26] Lindeberg, T., "Feature detection with automatic scale selection," International journal of computer vision*,* 30(2), 79-116 (1998).

[27] Breiman, L., [Classification and regression trees] Wadsworth International Group, Belmont, Calif.(1984).

[28] Tin Kam, H., "The random subspace method for constructing decision forests," IEEE Transactions on Pattern Analysis and Machine Intelligence*,* 20(8), 832-844 (1998).

[29] Breiman, L., "Random Forests," Machine Learning*,* 45(1), 5-32 (2001).

[30] Goussies, N., Ubalde, S., and Mejail, M., [Transfer Learning Decision Forests for Gesture Recognition], (2017).

[31] Gabor, D., "Theory of communication. ," Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*,* 93(26), 429-457 (1946).

[32] Field, D., [Relations between the Statistics of Natural Images and the Response Properties of Cortical-Cells], (1988).

[33] Fischer, S., Šroubek, F., Perrinet, L. *et al.*, "Self-Invertible 2D Log-Gabor Wavelets," International Journal of Computer Vision*,* 75(2), 231-246 (2007).

[34] Plodpradista, P., Keller, J. M., and Popescu, M., editors. An application of log-Gabor filter on road detection in arid environments for forward looking buried object detection. SPIE Defense + Security; 2015: SPIE.

[35] L. Miller, B., and E. Goldberg, D., [Genetic Algorithms, Tournament Selection, and the Effects of Noise], (1995).

[36] Mühlenbein, H., and Schlierkamp-Voosen, D., "Predictive models for the breeder genetic algorithm i. continuous parameter optimization," Evolutionary computation*,* 1(1), 25-49 (1993).

[37] Deb, K., [Multiobjective Optimization Using Evolutionary Algorithms] Wiley, U.K. Chichester(2001).

[38] Deb, K., Pratap, A., Agarwal, S. *et al.*, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation*,* 6(2), 182-197 (2002).

[39] Ishibuchi, H., Imada, R., Setoguchi, Y. *et al.*, editors. Performance comparison of NSGA-II and NSGA-III on various many-objective test problems. 2016 IEEE Congress on Evolutionary Computation (CEC); 2016 24-29 July 2016.

[40] Jiangyun, L., Xiaobo, J., and Chaonan, T., editors. Modeling and simulation of VRP in wartime using NSGA II. 2012 24th Chinese Control and Decision Conference (CCDC); 2012 23-25 May 2012.

[41] Gong, J., Zhu, X., Hu, Q. *et al.*, editors. A fast optimized algorithm based on the NSGA — II for microwave windows. 2017 Eighteenth International Vacuum Electronics Conference (IVEC); 2017 24-26 April 2017.

[42] Plodpradista, P., Keller, J. M., and Popescu, M., editors. Road recognition in poor quality environments for forward looking buried object detection. SPIE Defense + Security; 2014: SPIE.

[43] Plodpradista, P., Keller, J. M., Ho, D. K. C. *et al.*, editors. Tuning log Gabor filter bank using genetic algorithm based optimization. SPIE Defense + Security; 2017: SPIE.

[44] Sergyan, S., editor Color histogram features based image classification in content-based image retrieval systems. 2008 6th International Symposium on Applied Machine Intelligence and Informatics; 2008 21-22 Jan. 2008.

[45] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision*,* 60(2), 91-110 (2004).

[46] Harris, C., and Stephens, M., [A combined corner and edge detector], (1988).

[47] Barber, C. B., Dobkin, D. P., Dobkin, D. P. *et al.*, "The quickhull algorithm for convex hulls," ACM Trans. Math. Softw.*,* 22(4), 469-483 (1996).

[48] Srungarapu, S., Reddy, D. P., Kothapalli, K. *et al.*, editors. Fast Two Dimensional Convex Hull on the GPU. 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications; 2011 22-25 March 2011.

[49] Loose, H., Franke, U., and Stiller, C., editors. Kalman Particle Filter for lane recognition on rural roads. 2009 IEEE Intelligent Vehicles Symposium; 2009 3-5 June 2009.

[50] Trzebiatowski, M. S. v., Gern, A., Franke, U. *et al.*, editors. Lane Recognition of Country Roads. IEEE Intelligent Vehicles Symposium, 2004; 2004 14-17 June 2004.

[51] Manz, M., Hundelshausen, F. v., and Wuensche, H., editors. A hybrid estimation approach for autonomous dirt road following using multiple clothoid segments. 2010 IEEE International Conference on Robotics and Automation; 2010 3-7 May 2010.

[52] Levenberg, K., "A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES," Quarterly of Applied Mathematics*,* 2(2), 164-168 (1944).

[53] Coleman, T. F., and li, Y., [An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds], (1996).

[54] Berghen, F. V., "Levenberg-Marquardt algorithms vs trust region algorithms," IRIDIA, Université Libre de Bruxelles, (2004).

[55] Marsili Libelli, S., and Alba, P., "Adaptive mutation in genetic algorithms," Soft Computing*,* 4(2), 76-80 (2000).

[56] Stone, K. E., [Improved geo-referencing and prescreening for detection of buried explosive hazards in forward-looking infrared imagery] University of Missouri, (2014).

[57] Szegedy, C., Liu, W., Jia, Y. *et al.*, editors. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition; 2015.

[58] Quanz, B., and Huan, J., editors. Large margin transductive transfer learning. Proceedings of the 18th ACM conference on Information and knowledge management; 2009: ACM.

[59] Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” Communications of the ACM*,* 60(6), 84-90 (2017).

[60] Bailo, O., Lee, S., Rameau, F. *et al.*, editors. Robust Road Marking Detection and Recognition Using Density-Based Grouping and Machine Learning Techniques. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV); 2017 24-31 March 2017.

[61] Zhang, L., Yang, F., Zhang, Y. D. *et al.*, editors. Road crack detection using deep convolutional neural network. 2016 IEEE International Conference on Image Processing (ICIP); 2016 25-28 Sept. 2016.

[62] Deng, J., Dong, W., Socher, R. *et al.*, editors. ImageNet: A Large-Scale Hierarchical Image Database2009 2009.

[63] Rasmussen, C., editor Grouping dominant orientations for ill-structured road following. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.; 2004 27 June-2 July 2004.

[64] Mistry, D., and Banerjee, A., “Comparison of Feature Detection and Matching Approaches: SIFT and SURF,” GRD Journals- Global Research and Development Journal for Engineering*,* 2, 7-13 (2017).

[65] Amir-khalili, A., Hodgson, A. J., and Abugharbieh, R., editors. Real-time extraction of local phase features from volumetric medical image data. 2013 IEEE 10th International Symposium on Biomedical Imaging; 2013 7-11 April 2013.

VITA

Pooparat Plodpradista

1025 S. Country Club Dr. Jefferson City, MO 65109

Pooparat Plodpradista was born in Denver, Colorado. He attended Kasetsart University in Thailand from 2001 to 2005 and received a Bachelor of Engineering in Computer Engineering in 2005. He worked as a software developer for four years before began to work toward a Master of Science in Computer Engineering at the University of Missouri-Columbia College of Engineering, in August 2009 as a part-time student. He continued to work as a software developer in Columbia, Missouri until he decided to pursue a Ph.D. study. Pooparat received the Master's Degree in Electrical and Computer Engineering in December 2012, and a Ph.D. in Electrical and Computer Engineering in May 2021. Pooparat was a research assistant to Dr. James Keller between 2012-2018 and Dr. Dominic Ho between 2018-2020 on Army Research Office grants. His research interests include computer vision, image and video processing, signal processing, machine learning, and computational intelligence.