NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MACHINE LEARNING FOR SHIP VESSEL CLASSIFICATIONS AUGMENTED WITH SYNTHETIC IMAGES

by

Jun Wen Tang

September 2020

Thesis Advisor: Monique P. Fargues
Co-Advisor: Roberto Cristi

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
MACHINE LEARNING FOR SHIP VESSEL CLASSIFICATIONS AUGMENTED WITH SYNTHETIC IMAGES

5. FUNDING NUMBERS

6. AUTHOR(S) Jun Wen Tang

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Haze conditions have reportedly reduced visibility to about 3km in some of the busiest shipping lanes in the world. Haze conditions, including inclement weather conditions, are identified as a key challenge for autonomous vehicle operations. However, field data on poor weather conditions and ship images under hazy conditions may not be readily available to support research work aimed toward overcoming such challenges for autonomous vehicles. In this thesis, synthetic ship images are rendered under hazy conditions to augment a baseline dataset of haze-free ship images, in order to support our research on ship vessel classifications in a hazy environment using machine learning. The proposed feature extraction involves the counting of corner points detected using the Kanade Lucas Tomasi (KLT) technique to characterize the pattern of specific ship classes and computing of higher-order moments on the color planes on the ship structure detected in the images. Results show that the average ship classification accuracy rate is about 40% higher when the model is trained using a dataset augmented with synthetic hazy ship images; the classifier can classify for ship classes such as container ships, cargo ships, and sailing vessels, with an 80% average accuracy rate.

14. SUBJECT TERMS
Kanade Lucas Tomasi, KLT, synthetic hazy images, machine learning, ship vessels classification

15. NUMBER OF PAGES
97

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

MACHINE LEARNING FOR SHIP VESSEL CLASSIFICATIONS AUGMENTED WITH SYNTHETIC IMAGES

Jun Wen Tang
Major, Republic of Singapore Navy
BEE, Nanyang Technological University, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2020

Approved by: Monique P. Fargues
Advisor

Roberto Cristi
Co-Advisor

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Haze conditions have reportedly reduced visibility to about 3km in some of the busiest shipping lanes in the world. Haze conditions, including inclement weather conditions, are identified as a key challenge for autonomous vehicle operations. However, field data on poor weather conditions and ship images under hazy conditions may not be readily available to support research work aimed toward overcoming such challenges for autonomous vehicles. In this thesis, synthetic ship images are rendered under hazy conditions to augment a baseline dataset of haze-free ship images, in order to support our research on ship vessel classifications in a hazy environment using machine learning. The proposed feature extraction involves the counting of corner points detected using the Kanade Lucas Tomasi (KLT) technique to characterize the pattern of specific ship classes and computing of higher-order moments on the color planes on the ship structure detected in the images. Results show that the average ship classification accuracy rate is about 40% higher when the model is trained using a dataset augmented with synthetic hazy ship images; the classifier can classify for ship classes such as container ships, cargo ships, and sailing vessels, with an 80% average accuracy rate.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

2D	Two-dimensional
CMs	Higher order color moments
CPs	Corner Points
FBM	Fractional Brownian Motion
KLT	Kanade Lucas Tomasi
ML	Machine Learning
MRF	Markov Random Field
RF	Random Forests
RGB	red, green, blue

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to sincerely thank my thesis advisors, Dr. Monique P. Fargues and Dr. Roberto Cristi, for their advice and assistance in the completion of my thesis.

I would also like to thank the Republic of Singapore Navy for sponsoring me on this learning journey at the Naval Postgraduate School.

Last, but not least, and most importantly, I would like to thank my wife, Charmaine, for being here with me on this amazing yet challenging journey whilst tending our toddler daughter, Tessa. Without Charmaine's sacrifices, especially so during the COVID-19 situation, I would not have been able to fulfill this aspiration of mine.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Autonomous capabilities at sea are increasingly in demand given their potential in enhancing the efficiency of operational processes and minimizing human errors [1]. P. Koopman indicated that the key enabler for a potential widespread deployment of these autonomous capabilities thus lies in its assurance for safe operability [2]. Inclement weather conditions pose a significant challenge to autonomous vehicles due to performance degradation in contact sensing under such conditions [3]. Unfortunately, field data comprising poor weather conditions may not be readily available. This limitation may hinder studies that aim to overcome such a challenge.

For example, it was reported that the visibility was reduced to about 3km in some parts of Singapore when haze engulfed the island city in September 2019 [4]. The study of autonomous ship classification under such hazy conditions, without a database of ship images that were captured in such degraded environments, can prove to be an overly challenging task. P. Koopman also asserted that the challenge begins with enhancing system robustness for difficult environmental conditions in order to achieve ultra-dependable autonomous capabilities [2]. As sea lanes can potentially be hazy such as conditions encountered in Singapore, it is hence useful to account for hazy environmental conditions possibly faced by autonomous vehicles in the future.

In this thesis, we propose to augment an original dataset of haze-free ship images with a dataset comprising synthetic hazy ship images rendered based on the atmospheric scattering model. The resulting dataset includes ship images in both haze-free and hazy conditions and serves to facilitate our study on ship classification performance as part of the continual effort toward enhancing the safe operability of future autonomous capabilities in the naval domain.

B. OBJECTIVES AND THESIS ORGANIZATION

This thesis implements the atmospheric scattering model for rendering of synthetic ship images in hazy environments, and investigates ship classification performance

obtained with the dataset augmented with synthetic hazy ship images, using the Random Forests (RF) classification.

The remainder of the thesis is organized as follows. The conceptual overview of the atmospheric scattering model which forms the basis for our synthetic images rendering is introduced in Chapter II. A detailed approach in rendering synthetic hazy ship images using haze-free ship images is presented in Chapter III. In Chapter IV, we present our proposed approach to detect Corner Points (CPs) in ship images, using the Kanade Lucas Tomasi (KLT) technique, and the use of the detected CPs distribution as class features for classification. Chapter IV also discusses the extraction of higher order moments of the color planes of ship images as features for ship classification. Ship classification results obtained are presented and analyzed in Chapter V. Conclusions and recommendations are presented in Chapter VI.

II. ATMOSPHERIC SCATTERING MODEL

The atmospheric scattering model is widely used to characterize the formation of hazy and foggy images in numerous applications. F. Guo, J. Tang, and X. Xiao proposed realistic rendering of foggy images in game development and virtual reality applications [6]. E. Ullah, R. Nawaz, and J. Iqbal used the atmospheric scattering model for applications in image haze removal [7]. For the purpose of this thesis, the atmospheric scattering model is used to generate synthetic hazy images and implemented with MATLAB. The rendered synthetic hazy ship images will augment our original dataset of haze-free ship images and facilitate our ship classification study. This chapter provides the conceptual overview of the atmospheric scattering model (i.e., the basis for rendering of synthetic hazy images).

A. INTRODUCTION TO ATMOSPHERIC SCATTERING MODELS

The presence of particles in the air is an atmospheric phenomenon affected by weather conditions and air pollution which causes visibility degradation and possibly color changes [8]. Haze and fog are the most common atmospheric phenomena: haze is caused by air pollution such as smoke and dry particles (e.g., dust) suspending in the air, while fog is due to the presence of water vapour in the air. When light rays hit a particle (e.g., dust, water droplet), light scattering will occur in all directions with varying magnitude as the scattered rays moved away from the particle [9]. An imaging illustration based on the atmospheric scattering model with an example of the transmission map and synthetic hazy image rendered using our MATLAB implementation is shown in Figure 1.

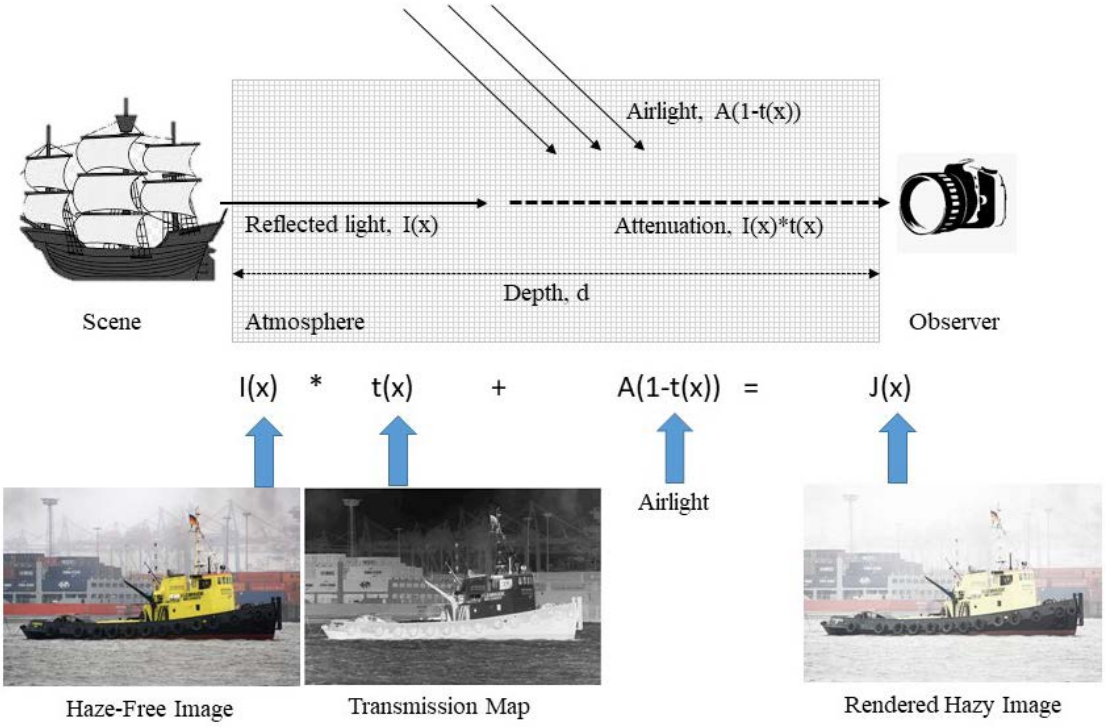


Figure 1. Atmospheric scattering model

The mathematical formula of the rendered hazy image *J(x)* in the atmospheric scattering model is given by [10, p. 6]

$$J(x) = I(x)t(x) + A(1-t(x)),$$

where *x* denotes a point in the two dimensional (2D) image plane, *I(x)* represents a haze-free image, (i.e., three 2D-matrices representing the red, green, blue (RGB) colored information), and *t(x)* is the transmission map of *I(x)*, which is the scalar representation of the variations in transmission due to the depth *d* between the scene and the observer. The parameter *A* represents the atmospheric light intensity, which is usually assumed to be a global constant throughout the scene and is often considered as the environmental illumination.

A physical view of the atmospheric scattering model is shown in Figure 1. The presence of haze particles and water vapor in the atmosphere scatters and absorbs the light transmitting through the atmosphere. The transmission of the reflected light from the scene

I(x) is partly attenuated due to the light scattering caused by the particles in the atmosphere. The transmission map *t(x)* represents the ratio of "the un-attenuated light that reaches the observer" to "the reflected light from the scene". The term *I(x)t(x)* is known as the attenuation. The term *A(1-t(x))* is called the airlight, which is the environmental illumination from several sources, including, diffused skylight, sunlight, and ground light [11, p. 5]. The rendered hazy image *J(x)* at the observer viewpoint would be the attenuated light with the additive of the airlight.

B. TRANSMISSION MAP

The transmission map represents "the portion of the light that is not scattered and reaches the observer" [6, p. 3], and since the transmission map is a "function of depth, it thus reflects the depth information in the scene" [6, p. 3]. Let us assume *d(x)* is the distance from a scene point at position *x* to the observer, with *d* denoted as the depth of the particular scene point. The transmission map *t(x)* is related to the depth *d*, and is given by

$$t(x) = e^{-\beta d(x)},$$

where β represents the extinction coefficient of the atmosphere. The parameter β is determined by the physical properties of the particles residing in the atmosphere, such as the particle size, material, shape and density, and function of the wavelength λ and wavelength selectivity γ, given by [11, p. 6]

$$\beta(\lambda) = \frac{\text{Constant}}{\lambda^{\gamma}}.$$

Under a homogeneous foggy or hazy environment, $\gamma \approx 0$ [12]; hence the extinction coefficient β is a spatial constant [10]. Various values for visibility distances and extinction coefficients β corresponding to different weather conditions according to the international visibility grades [6, p. 7] are listed in Table 1.

Table 1. International visibility grades. Adapted from [13].

Grade	Weather	Visibility distance	Extinction Coefficient β
0	Dense fog	<50m	>78.2m^{-1}
1	Thick fog	50–200m	78.2–19.6m^{-1}
2	Moderate fog	200–500m	19.6–7.82m^{-1}
3	Light fog	500m–1km	7.82–3.91m^{-1}
4	Thin fog	1–2km	3.91–1.96m^{-1}
5	Haze	2–4km	1.96–0.954m^{-1}
6	Light haze	4–10km	0.954–0.391m^{-1}
7	Clear	10–20km	0.391–0.196m^{-1}
8	Very clear	20–50km	0.196–0.078m^{-1}
9	Extremely clear	>50km	0.0141m^{-1}

C. AIRLIGHT

Unlike the attenuation term $I(x)t(x)$ which causes the scene radiance to decrease along the depth, the airlight term $A(1-t(x))$ increases the scene radiance along the depth. According to [8, p. 3], in situations of bad weather (e.g., hazy or foggy environment), the sky is usually overcast and the atmospheric light intensity A can be assumed to be a global constant. For the purpose of synthetic hazy image rendering adopted in this thesis, the atmospheric light intensity A is chosen to be a constant value equals to 255, consistent with an 8-bit representation of light intensity, as referenced in [6, p. 6].

D. RENDERED HAZY IMAGE

The RGB components of the haze-free image $I(x)$ (i.e., $I_R(x)$, $I_G(x)$, *and* $I_B(x)$), are multiplied by the transmission map $t(x)$, then added to the airlight term $A(1-t(x))$. Selecting the atmospheric light intensity A value equal to 255 results in using white color as the atmospheric light intensity for all scenes (i.e., R=255, G=255, B=255). See Figure 2 for the composition of the rendered hazy image.

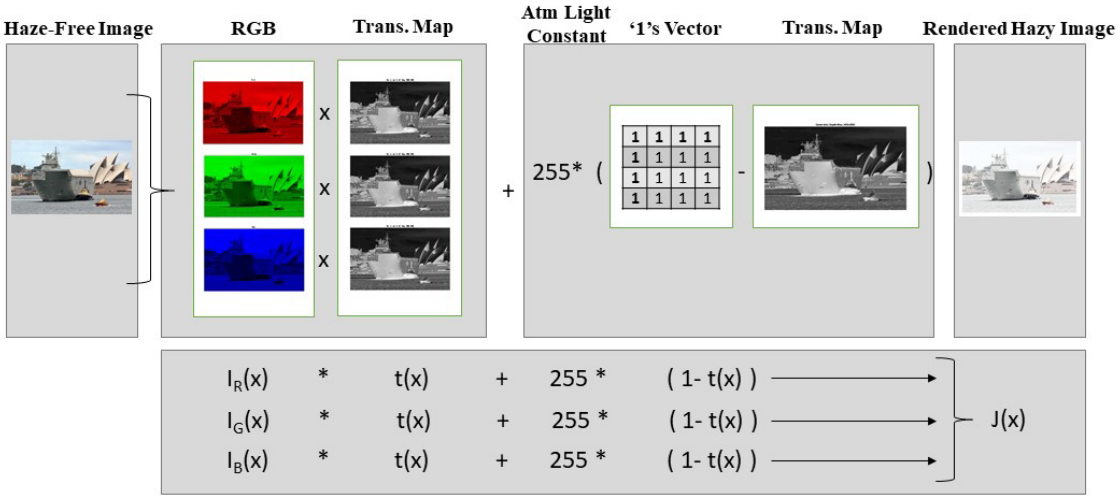


Figure 2. Composition of rendered hazy image

E. ATMOSPHERIC SCATTERING MODEL CONCLUSION

The atmospheric scattering model discussed in this chapter, a widely adopted model for characterization of hazy and foggy images applications [6], [7], is used in this thesis to generate synthetic hazy images for our ship classification study with ship images under hazy conditions. Benchmarking to the international visibility grades in Table 1, an extinction coefficient value equals to $1.96m^{-1}$ (worst case of Grade 5 for visibility distance between 2-4km) is chosen for the rendering of synthetic hazy images in this study. Such selection simulates the haze condition encountered by one of the busiest shipping lanes in the world, where the visibility range was reportedly reduced to 3km [4]. As haze particles may occur in spatially constant or varying density in the atmosphere [14], synthetic hazy images are rendered in both conditions, and termed in this thesis as homogenous synthetic hazy images and heterogeneous synthetic hazy images. Homogeneous synthetic hazy images are rendered with uniform density of haze particles in space. Heterogeneous synthetic hazy images are rendered with a non-uniform density of haze particles (i.e., non-homogeneous). The approach of the rendering of the homogeneous and heterogeneous synthetic hazy images will be addressed in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

III. APPROACH OF SYNTHETIC HAZY IMAGE RENDERING

The gaming industry has evolved from using software-based rendering tools, such as 3ds max and Photoshop, to a more efficient model-based rendering approach for foggy and hazy scenes rendering in their games development [6] (i.e., adopting the atmospheric scattering model described in Chapter II). Reference [6] also demonstrated the possibility to generate realistic foggy scenes with real-life images using the same approach implemented as that selected for foggy scene rendering in virtual environments. For this study, we will be using a similar approach for the rendering of synthetic hazy images, as previously adopted by [6]. This chapter presents the approach considered for rendering synthetic hazy image from haze-free images. The rendering processes are implemented using MATLAB, and detailed in Appendix A.

A. SYNTHETIC HAZY IMAGES RENDERING PROCESS FLOW

Rendering a synthetic hazy image requires three steps for the case of a generic heterogeneous hazy texture, following the approach presented in [6, p. 3]. Conversely, two steps are required to generate a synthetic image with a homogeneous hazy texture.

We first need to compute the transmission map with the Markov Random Field (MRF) model to generate homogeneous synthetic hazy images [15]. The goal in doing so is to assign each image pixel with an accurate label based on the minimum value of the energy function of the MRF. A label assigned with smaller value represents the image pixel at a deeper depth in the scene, while a larger value label corresponds to a scene point closer to the observer. An additional step is required to generate a heterogeneous noise image using the Fractional Brownian Motion (FBM) turbulence texture [16]. Finally, the synthetic hazy image can be rendered according to the atmospheric scattering model. The synthetic hazy image rendering flowchart is depicted in Figure 3.

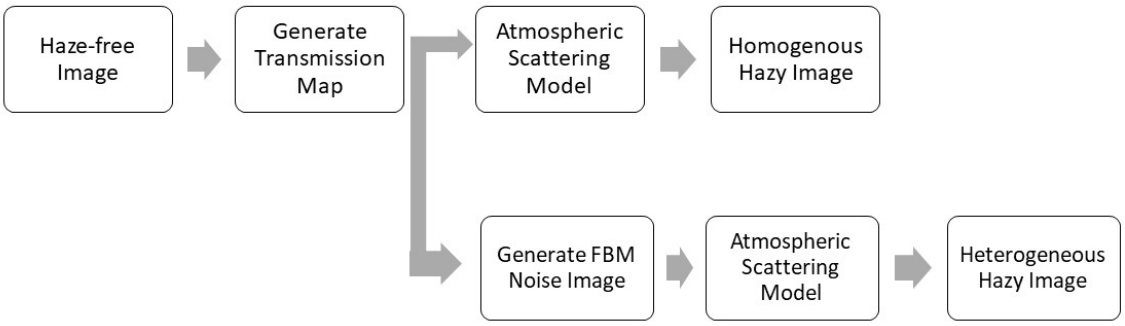


Figure 3. Synthetic hazy image rendering flowchart

B. TRANSMISSION MAP ESTIMATION

The most crucial step for synthetic hazy image rendering is the estimation of the transmission map. The MRF technique, which is a probabilistic-based graphical model, is commonly utilized for analyzing the spatial dependencies of a given image [15]. Here, we obtain the estimation of the transmission map using the MRF. To generate the transmission map, the input RGB image is first converted to a gray-level image, allocating 8-bits for the image intensity level representation. Referencing to the methodology of foggy scene images rendering adopted by the computer games industry [6, p. 4], the number of labels is set to l=32, where the set of labels L={1,2,3,…,l} represents the transmission values {0,1/(l-1),2/(l-1),…,1} [6, p. 4].

Table 2. Labels and corresponding label values

Label x_i	1	2	3	4	5	6	7	8
Label Value $L(x_i)$	0	1/31	2/31	3/31	4/31	5/31	6/31	7/31
x_i	9	10	11	12	13	14	15	16
$L(x_i)$	8/31	9/31	10/31	11/31	12/31	13/31	14/31	15/31
x_i	17	18	19	20	21	22	23	24
$L(x_i)$	16/31	17/31	18/31	19/31	20/31	21/31	22/31	23/31
x_i	25	26	27	28	29	30	31	32
$L(x_i)$	24/31	25/31	26/31	27/31	28/31	29/31	30/31	1

Each element t_i of the transmission map is assigned a label x_i. The labeling of the transmission map is estimated by minimizing the associated cost function *E(x)*

$$E(x) = \sum_{i \in P} E_i(x_i) + \sum_{(i,j) \in N} E_{ij}(x_i, x_j),$$

where *P* denotes the number of pixels in the image, and *N* denotes the collection of pairs of pixels within the cross-connect neighborhood configuration. The cross-connect neighborhood configuration using a 3x3 pixels image as an example is illustrated in Figure 4, where the number of pairs of pixels within a given neighborhood varies between *N*=2 at the corners of the image, *N*=3 along the edges of the image, and *N*=4 otherwise.

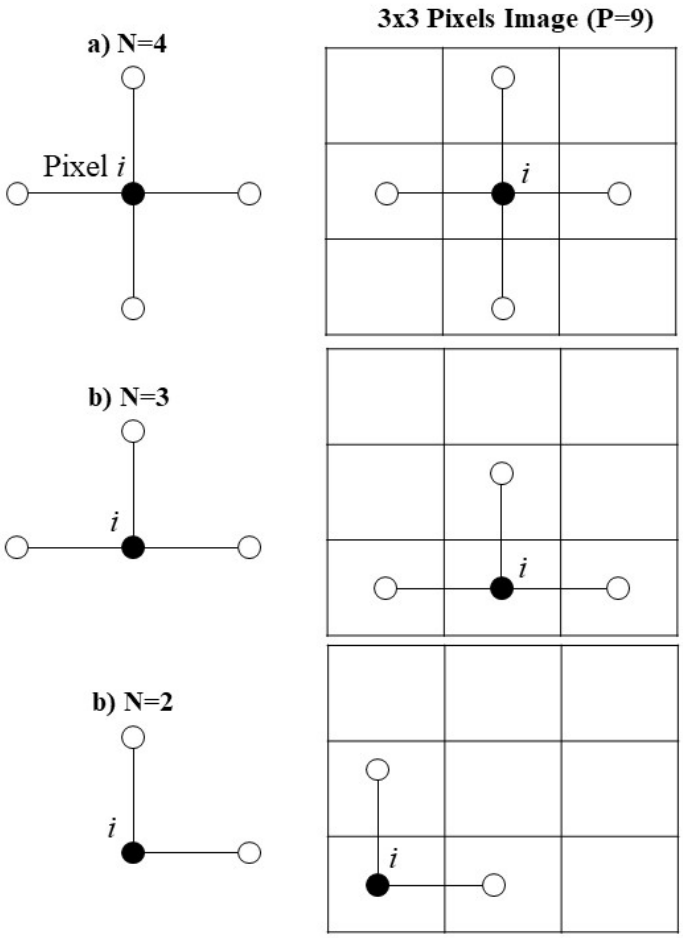


Figure 4. Cross-connect configuration on a 3x3 pixels image

The data term $E_i(x_i)$ is related to the probability of the transmission element t_i assigning to label x_i. The data term $E_i(x_i)$ is the absolute difference between the label value $L(x_i)$ and the intensity of each pixel value at the RGB-converted gray-level image I_i' (values ranging from 0 to 255) multiplied with a normalizing factor ω. The normalizing ω value is chosen to be 1/255 [6, p. 5] to ensure that the gray-level image I_i and label value $L(x_i)$ have the same order of magnitude, such that the data term $E_i(x_i)$ will always have a value between 0 and 1, and can be expressed as

$$E_i(x_i) = \left| I_i' * \omega - L(x_i) \right|.$$

The smooth term $E_{ij}(x_i,x_j)$ is a component associated with the probability of neighboring pixels (i.e., cross-connected pixels) surrounding pixel *i*, and value of the image strength control g=0.01 [6]. The smooth term $E_{ij}(x_i,x_j)$ expression, is given by

$$E_{ij}(x_i, x_j) = g \left| x_i - x_j \right|.$$

The cost function $E(x)$ can be estimated using the α-expansion algorithm [6], [15], which involves performing expansions for all labels of α, where the label values $L(x)$ are sequentially assigned to label α [17]. At each iteration of the α-expansion algorithm, the algorithm will determine if the pixel *i* should retain its existing label x_i, or replace with the label α [18]. The label for pixel *i* switches to label α when the cost function $E(x)$ is minimized, where the condition for label switching for the α-expansion algorithm [18], given by

$$\begin{gathered} E_{ij}(\alpha, \alpha) + E_{ij}(x_i, x_j) \leq Ei_j(x_i, \alpha) + E_{ij}(\alpha, x_j), \\ \forall i, j \in P, \\ x_i, x_j, \alpha \in \forall L(x). \end{gathered}$$

In this thesis, we adopted the GCO-v3.0 library [19], developed by O. Veksler and A. Delong, to implement the cost function minimization step using the α-expansion algorithm. The library function gets a haze-free image as input parameter and returns as output the estimation of each pixel label x_i in the transmission map. The pseudo-code using the GCO-v3.0 library for the estimation process is shown in Figure 5.

Psuedo-code: Label assignment using gco-v3.0 library

```
Input:      Haze-free image (RGB)
Output:     Each pixel label xi in transmission map

Step 1: // Create new object, h
      Set NumSites = M x N and NumLabels = 32;
      h = GCO_Create(NumSites, NumLabels);

Step 2: // Compute data term
      2.1 Convert input image from colored to gray-level image and reshape to one-dimensional vector
      2.2 for i=NumLabels
            for j=NumSites
                  L(i)= (i-1)/(NumLabels-1);
                  DataTerm(i,j) = | Data(j)*ω - L(i) |;
            end
          end
      2.3 GCO_SetDataCost(h, DataTerm);

Step 3: // Compute smooth term
      3.1 for i=1:NumLabels
            for j = 1:NumLabels
                  SmoothTerm(i,j) = g*|i-j|;
            end
          end
      3.2 GCO_SetSmoothCost(h, SmoothTerm);

Step 4: // Compute minimization of labels via α-expansion
      4.1 GCO_Expansion(h);
      4.2 Label = GCO_GetLabeling(h);
```

Figure 5. Pseudo-code for label estimation using GCO-v3.0 library

In Figure 5, parameters *M* and *N* denote the height and width of the input haze-free image, and the parameter *Data* represents the gray-level image reshaped into a one-dimensional vector. As a result, with the defined functions in the GCO-v3.0 library, each pixel label x_i in the transmission map is estimated. Implementation details are presented in Appendix A, which includes the MATLAB function *transmission.m* developed for generating the labels x_i for an image. Next, we need to translate the pixel labels x_i in the transmission map (values ranging from 0 to 1) into its corresponding gray-level intensity values $t_{int}(x)$ to display the transmission map as a gray-level image (values ranging from 0 to 255). The gray-level intensity values of the transmission map $t_{int}(x)$ can be as expressed [6, p. 6]

$$t_{int}(x) = 255 - (x_i - 1) * 8.$$

The transmission map *t(x)* is given by [15, p. 4]

$$t(x) = e^{-\beta d(x)},$$

where the depth of the transmission map *d(x)* given by

$$d(x) = -\frac{In\big(t(x)\big)}{\beta}.$$

The extinction coefficient β can be chosen in accordance with the desired weather condition listed in Table 1.

An example of the colored haze-free image (a), its converted gray-level image (b), and its corresponding intensity of the estimated transmission map (c) are shown in Figure 6.

Haze-free Image (a), gray-level image (b), and intensity of estimated transmission map (c).

Figure 6. Conversion to transmission map

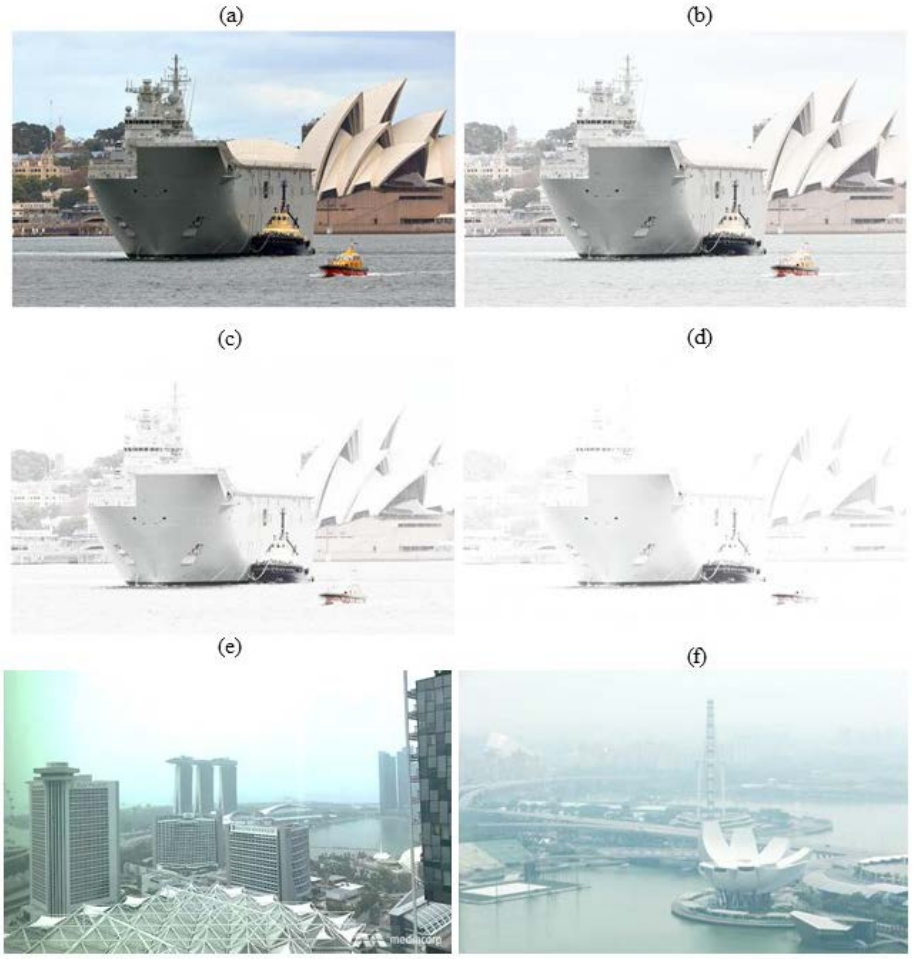
C. HOMOGENEOUS HAZY IMAGE SYNTHESIS

From the haze-free image *I(x)*, the transmission map *t(x)*, and the atmospheric light intensity constant *A*, the synthetic hazy image *J(x)* can be rendered using the atmospheric scattering model directly as

$$J(x) = I(x)t(x) + 255 * (1 - t(x)).$$

The atmospheric scattering model was implemented using MATLAB for rendering of synthetic hazy images, and code implementation details are presented in Appendix A (*setHaze.m*). Three different values for the extinction coefficient β were selected to render

synthetic homogeneous hazy images under various constant haze densities, and resulting synthetic homogeneous images rendered using the *setHaze.m* function are shown in Figure 7. By changing the β value, we can control the visibility distance of the objects in the image. Note that the haze effects look more natural in (b) than (c) and (d) when compared to real images in haze environment (e) and (f).



Original haze-free image (a). Synthetic image under homogeneous hazy condition β=2 (b). Synthetic image under homogeneous light fog condition β=4 (c). Synthetic image under homogeneous moderate fog condition β=8 (d). Singapore under haze on 23 Sep 2019 (e) Source: [4]. Singapore under haze on 13 Sep 2019 (f) Source: [20].

Figure 7. Synthetic and real-life images comparison

D. HETEROGENEOUS HAZY IMAGE SYNTHESIS

To render a heterogeneous hazy image, the noise image *n(x)* is added to the synthetic image rendered with the atmospheric scattering model. According to [6, p. 6], the process of adding the heterogeneous haze to the synthetic image can be written as

$$R(x) = J(x) + q * n(x),$$

where *n(x)* is the noise image, the parameter *q* is the gain coefficient to control the appearance of the noise image texture (*q*=0.15 in [6]), and *R(x)* represents the synthetic heterogeneous hazy image rendered. According to [6, p. 6], the hazy image *J(x)* (i.e., input for rendering of the heterogeneous hazy image), can be generated using the transmission map *t(x)*, atmospheric light intensity constant *A*, and hazing effect adjustment value t_0, as

$$J(x) = (I(x) - A) * \max(t(x), t_0) + A.$$

The Fractional Brownian Motion (FBM) technique can be used to generate the noise image *n(x)*, required for heterogeneous hazy image synthesis. Referencing to [16], the tiles of noise images are randomly generated. These tiles of noise images can be concatenated to form a baseline noise image, and can be cropped to the required noise image size. Examples of the *256x256* tiles of noise images are illustrated in Figure 8. The concatenated *1024x2048* baseline noise image is shown in Figure 9.

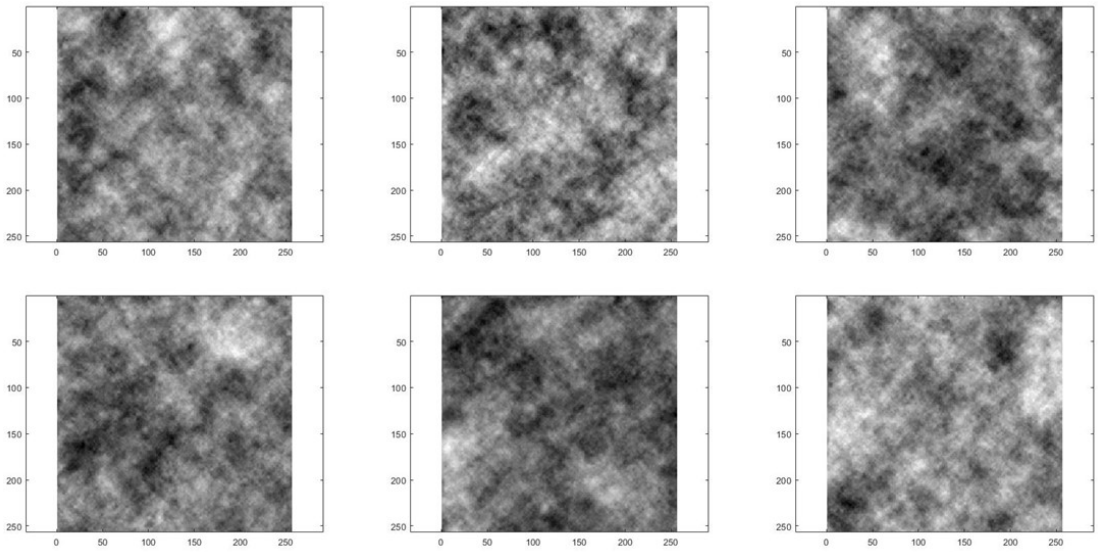


Figure 8. Tiles of noise images generated using FBM

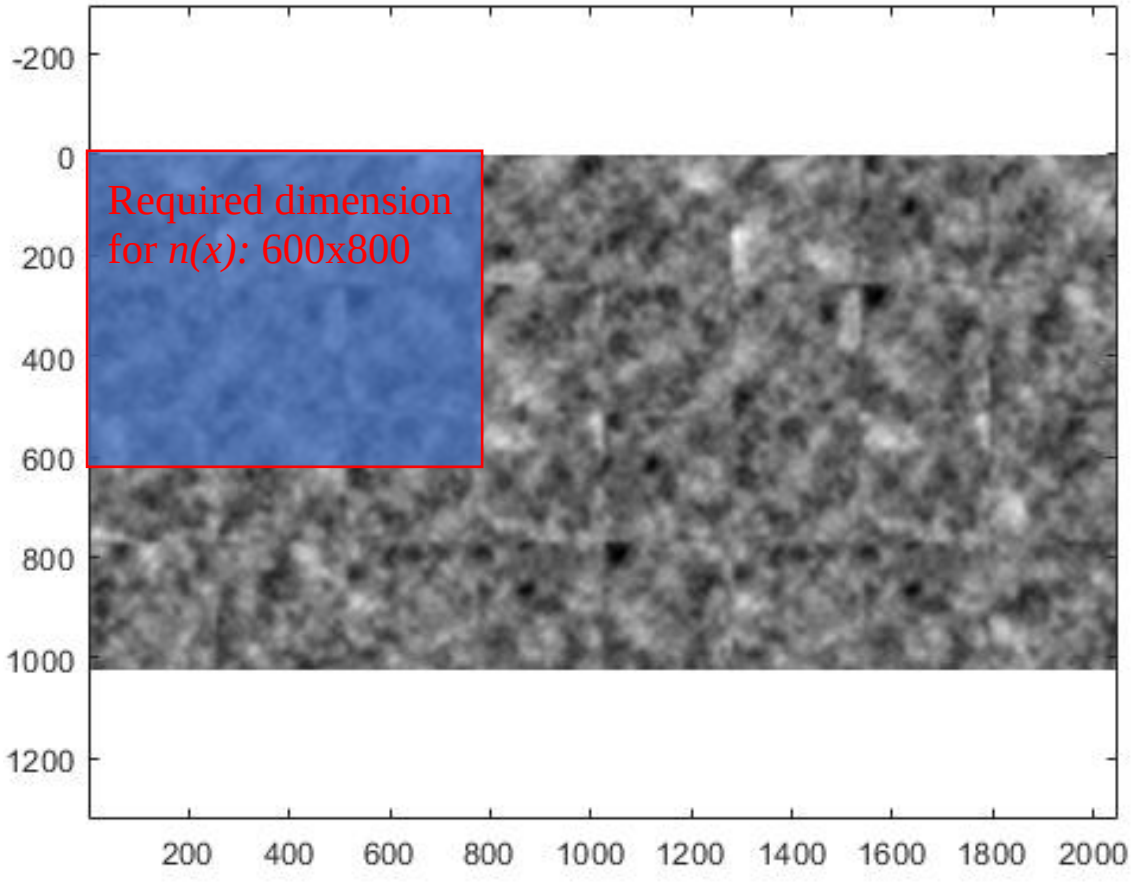
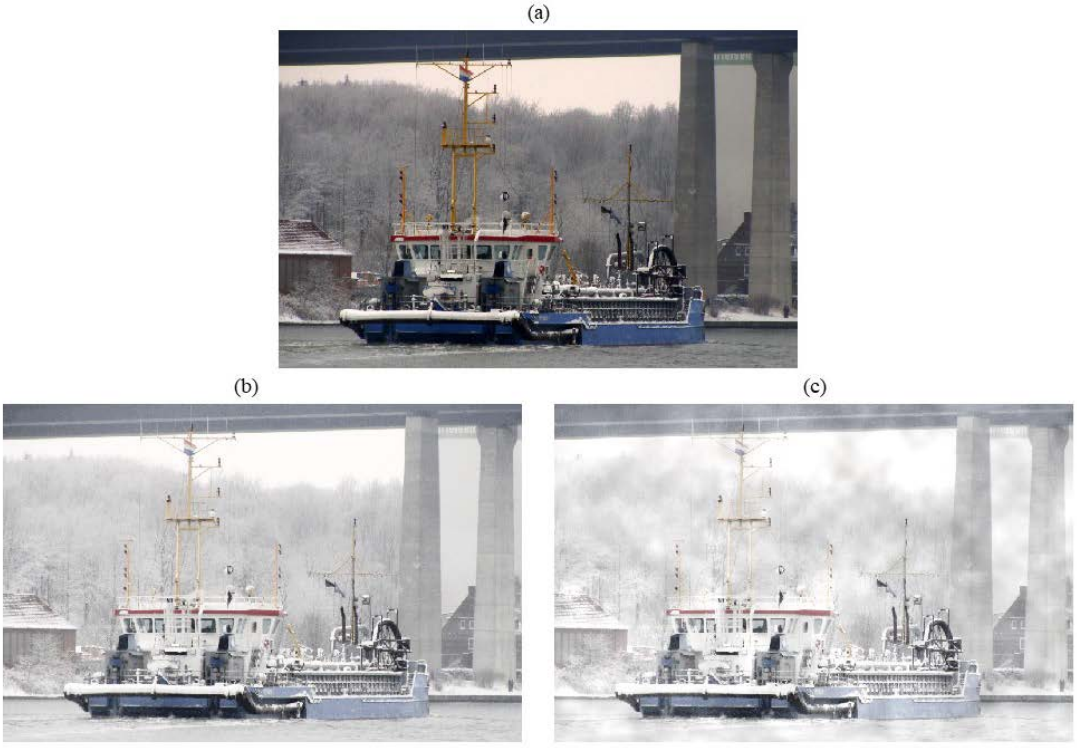


Figure 9. Baseline noise image (*1024x2048*)

The baseline noise image generated is cropped to the required dimensions (i.e., the size of the input image *J(x)*), prior to the heterogeneous hazy image synthesis processing. For example, the noise image *n(x)* will be cropped to the same dimensions as *J(x)* if *J(x)* has a dimension of *600x800*. The MATLAB function *FBM.m* developed to generate the noise image *n(x)* of required dimension, is provided in Appendix A. The synthetic heterogeneous hazy image *R(x)* can then be rendered by adding the input image *J(x)* and the noise image *n(x)*.

An example showing the visual comparison for a haze-free image, synthetic homogeneous hazy image, and synthetic heterogeneous hazy image is shown in Figure 10. Note the texture of the FBM noise image is visually apparent on the heterogeneous hazy image rendered.



Haze-free image (a), homogeneous hazy image (b), and heterogeneous hazy image (c).

Figure 10. Haze texture of heterogeneous hazy image

E. SYNTHETIC HAZY IMAGE RENDERING CONCLUSION

The rendering processes described in this chapter are used to generate the synthetic homogeneous and heterogeneous images to augment the size of a baseline dataset comprising haze-free ship images. The extinction coefficient was selected to be β=1.96 (worst case of Grade 5) to simulate the haze condition previously experienced in the Southeast Asia region, for our ship classification study. The MATLAB code *GenerateHazyImage.m*, rendering of synthetic hazy images in this thesis, could also be used for future studies that require rendering of synthetic images at a specific haze density by varying the extinction coefficient value β corresponding to the required visibility ranges.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. FEATURES FOR SHIP CLASSIFICATION

Coupled with the increased demand in autonomous systems and the availability of computing tools, reliable classification of our daily environment has been deemed as a potential technology enabler to enhance processes efficiency and minimize human errors [1], [2]. Ships are widely used for transportation, fishing and security in the oceans and seas. To potentially field autonomous capabilities in the waters, it is therefore of interest to embark on a study on ship classification using computer vision techniques and machine learning [3]. This chapter presents the use of the distribution of the detected Corner Points (CPs), and the extraction of higher order moments of the color planes of ship images, to extract class features for ship classification.

A. DATASET

The dataset used in this study comprises eight different classes of ships, namely container ships, cruises, cargo ships, tugs, yachts, sailing vessels, warships and fishing vessels. The baseline dataset with a total of 6,680 images was downloaded from [21]. The MATLAB function *flip.m* [22] was used to create horizontally flipped images from the baseline dataset. The homogeneous and heterogeneous hazy image synthesis processes described in Chapter III are also used to create synthetic images to augment the baseline dataset size.

A ship image example from the dataset is shown in Figure 11. By using image processing techniques such as the *flip.m* function and the hazy image synthesis, the ship images dataset is expanded to 6 times of the baseline dataset to a total of 40,080 images. The dataset distribution is summarized in Table 2.

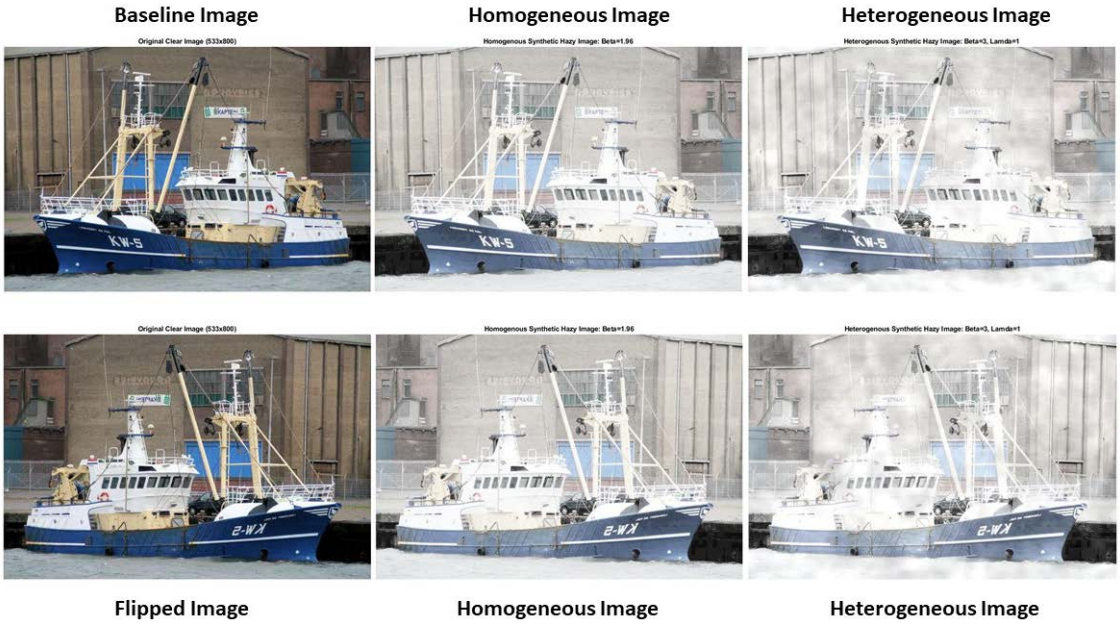


Figure 11. Example of ship images from dataset

Table 3. Dataset distribution

Classes	Baseline	Flipped	Homogeneous	Heterogeneous	Total
Container Ships	923	923	1,846	1,846	5,538
Cruises	855	855	1,710	1,710	5,130
Cargo Ships	931	931	1,862	1,862	5,586
Tugs	693	693	1,386	1,386	4,158
Yachts	800	800	1,600	1,600	4,800
Sailing Vessels	810	810	1,620	1,620	4,860
Warships	750	750	1,500	1,500	4,500
Fishing Vessels	918	918	1,836	1,836	5,508
Total	6,680	6,680	13,360	13,360	40,080

B. SHIP IMAGES PRE-PROCESSING

The dataset includes ship images of varying orientations, background, and distances from the observer. To identify the area of interest for our ship classification, pre-processing can be applied to detect the well-defined borders of the ship structure. We perform our ship structure detection by using the MATLAB function *edge.m* [23] available in the Image Processing Toolbox. The *edge.m* function receives as the input parameter a gray-level ship image, and outputs a binary image, where 1s (nonzero) indicate the edges detected, and 0s

elsewhere. In Figure 12, we show an RGB image and the resulting output binary image with detected edges indicated in "*white*."

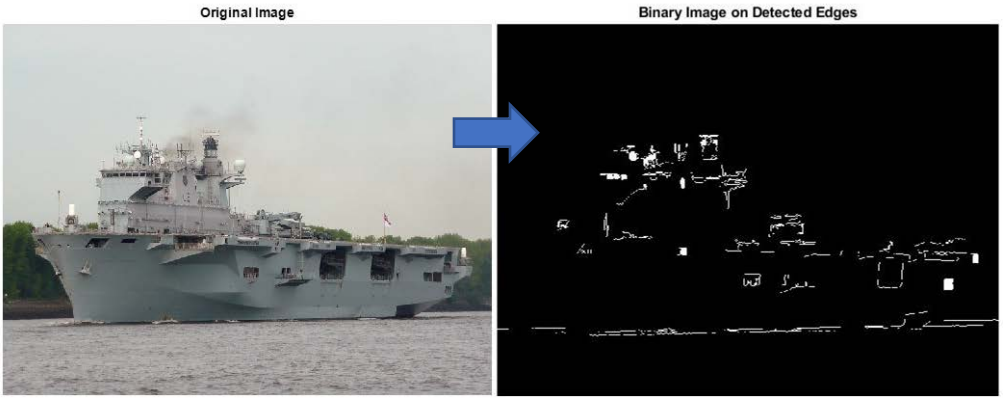


Figure 12. Detection of edges

The boundaries of the ship structure can be estimated from the maximum and minimum location of the edges in the binary image, which is given by

$$
\begin{aligned}
&[rows, columns] = find(binaryimage),\\
&TopRow = \min(rows),\\
&BottomRow = \max(rows),\\
&LeftColumn = \min(columns),\\
&RightColumn = \max(columns).
\end{aligned}
$$

Using the MATLAB function *find.m*, we can find the location of all the nonzero pixels in the binary image, and the boundaries (*Top-Row*, *Bottom-Row*, *Left-Column*, *Right-Column*) computed as described. Estimated boundaries of the detected ship structure presented in Figure 12, are shown in Figure 13. These estimated boundaries can then be used for further processing.

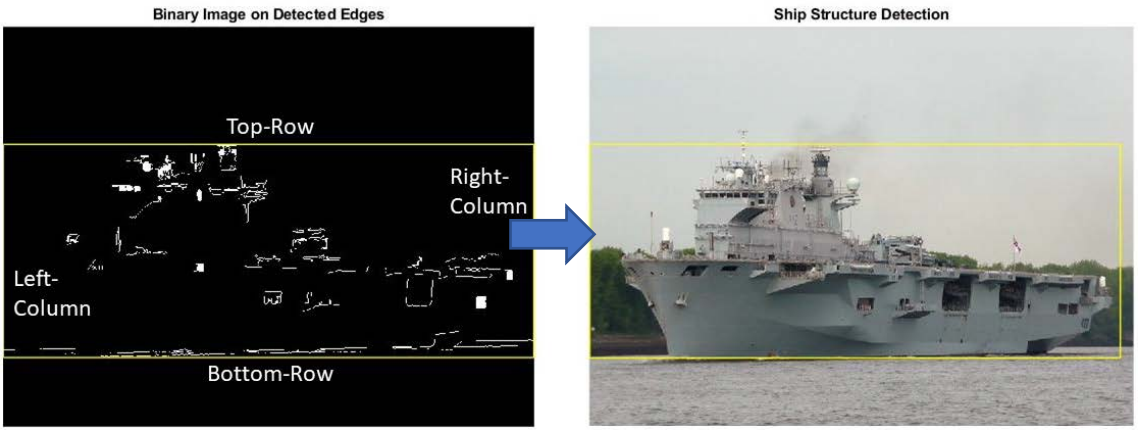


Figure 13. Estimated boundaries of ship structure

C. DETECTION AND DISTRIBUTION OF CORNER POINTS

After boundaries of the ship structure get estimated, features are extracted for classification processing. Our proposed feature extraction involves the detection of Corner Points (CPs) using the Kanade Lucas Tomasi (KLT) technique [24]. The KLT technique is a feature detection and tracking algorithm which first locates a CP by examining the eigenvalues of a *2x2* intensity gradient matrix within a defined neighbourhood, followed by pursuing the detected feature over time [25]. Given that the elements of the dataset of interest are still images, the application of KLT in this thesis only focuses on the CP detection (i.e., without application of tracking). The *2x2* gradient matrix *G* is given by [26]

$$G(x,y) = \sum_{u,v} I_{xy}(u,v) I^{T}{}_{xy}(u,v) z(x-u, y-u),$$

where the intensity gradient matrix of the gray-level image *I* is defined as

$$I^{T}{}_{xy}(x,y) = \left[\frac{\delta I(x,y)}{\delta x} \quad \frac{\delta I(x,y)}{\delta y} \right],$$

And the neighbourhood matrix *Z* is represented with

$$z(x,y) = \begin{cases} 1 & if\ (x,y) \in Z(0,0), \\ 0 & otherwise. \end{cases}$$

A pictorial illustration of the 5x5 neighborhood matrix *Z* is shown in Figure 14, where *Z(0,0)* represents a 5x5 neighborhood matrix, centered at the origin on the *x*-*y* axis (i.e., at (x_0=0, y_0=0)).

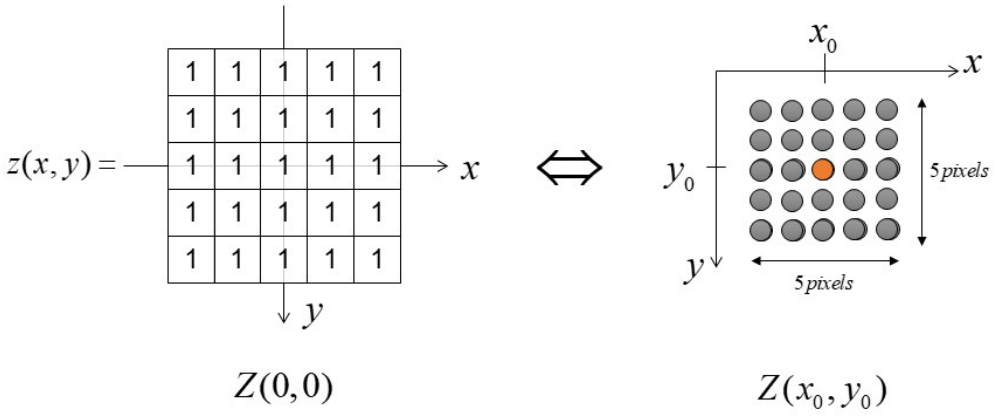


Figure 14. 5x5 neighborhood matrix *Z*

A discrete approximation can be used to compute the intensity gradient for each image pixel [27]. Using the 3x3 Sobel operators in the *x*-direction and *y*-direction, the 2D impulse responses h_x and h_y can be used to efficiently compute the intensity gradient I_x and I_y for an image [28], as

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix},$$

$$I_x = \frac{\delta I(x,y)}{\delta x} = conv2(I, h_x),$$
$$I_y = \frac{\delta I(x,y)}{\delta y} = conv2(I, h_y),$$

where the operation *conv2.m* performs the two-dimensional convolution of the intensity levels matrix *I* with the impulse responses h_x and h_y, respectively.

A corner of a 2D image is defined by a locality of intensity changes in both the *x*-direction and *y*-direction, hence the presence of corner points in an image can be estimated when gradient changes are detected in both directions [29]. The *2x2* gradient matrix *G* can be represented by the 2D convolutions between the power of the intensity gradients [30, p. 15], with 5x5 neighborhood matrix *Z* [31]. A 5x5 neighborhood matrix was chosen in this study because it can achieve a relatively smooth gradient detection [32] at the points of intensity changes, and it is not very sensitive to local noise within the neighbourhood [33].

$$G(x,y)=\begin{bmatrix} conv2(I_x(x,y)*I_x(x,y),Z(x,y)) & conv2(I_x(x,y)*I_y(x,y),Z(x,y)) \\ conv2(I_x(x,y)*I_y(x,y),Z(x,y)) & conv2(I_y(x,y)*I_y(x,y),Z(x,y)) \end{bmatrix},$$

with *Z* defined as

$$Z=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Using the minimum eigenvalue technique [24] [31], a CP can be said to be detected at location *(x,y)* when the eigenvalues of *G* are larger than a set threshold, where Λ=eigenvalues of *G*. The array containing the locations of the corner points *CP_location* can be computed using

$$if\ (\Lambda_{\max} \geq \Lambda_{\min}(x,y) > threhold),$$
$$CP\_location = (x,y).$$

Detected CPs (in *magenta*) using the KLT for the different ship images are illustrated in Figure 15. The CPs distribution is observed to represents characteristics for each specific classes of ships from images contained in our dataset. The MATLAB function *GenerateCornerPoints.m* for detection of corner points, is provided in Appendix A.

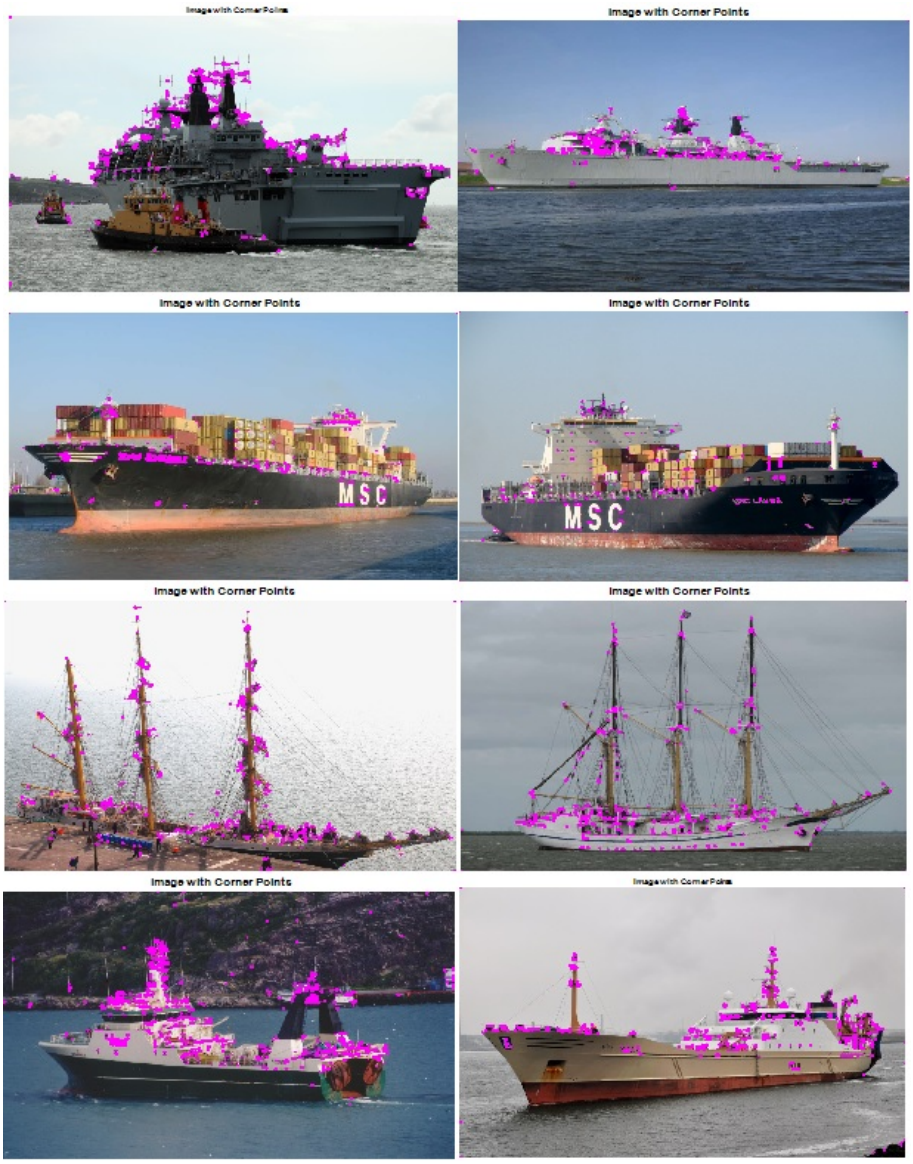


Figure 15. Detected CPs

To quantitatively tabulate the distribution of CPs for the different classes of ships, the estimated boundaries of the ship structure can be sub-divided into *R* rows and *C* columns of partitions. The number of CPs located within each partition are counted, thus allowing us to quantitatively tabulate the CPs distribution. Estimated boundaries of the ship structure segmented into *6x8* partitions is illustrated for one ship image in Figure 16. The MATLAB function *DetectShipAndPartition.m* implemented for the estimation of ship

structure boundaries of an image, and sub-division of the estimated boundaries into *R* rows and *C* columns of partitions, is provided in Appendix A.

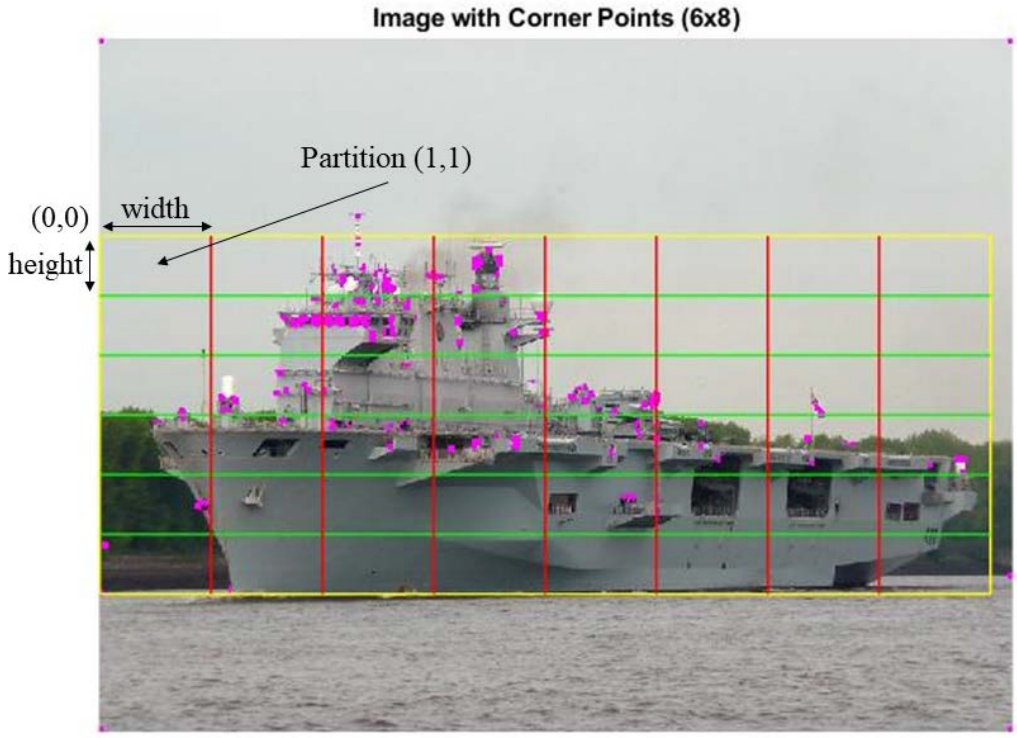


Figure 16. CPs distribution (*6x8* partitions)

The count of CPs located within each partition *C(r,c)* is given by

$$C(r,c) = \sum_{n=(r-1)*height,\, m=(c-1)*width}^{r*height,\, c*width} CP\_location(n,m),$$

where *r* and *c* represent the row and column positions, respectively, and parameters *width* and *height* are used to compute the number of CPs contained within a given partition. The array *CP_location* contains all the detected CPs locations *(x,y)* in the image. The sum of the number of CPs located within each partition is computed by the MATLAB function *CountingCPs.m*, code implementation details are provided in Appendix A.

D. IMAGE COLOR MOMENTS EXTRACTION

Another feature uses the color moments computed on the estimated boundaries of the ship structures. We take advantage of the fact that ships of the same class predominantly have similar color coding. For example, warships are typically grey, while cruise ships are generally white. For this thesis, the higher order moments of the color planes are extracted as features. The RGB image is vectorized into individual *Nx1* vectors, I_R, I_G, I_B, where N is the product of the image dimension.

The 1st moment is the mean color of the image μ, and defined as [34]

$$\mu_R = \frac{1}{N}\sum_{i=1}^{N} I_{Ri},$$
$$\mu_G = \frac{1}{N}\sum_{i=1}^{N} I_{Gi},$$
$$\mu_B = \frac{1}{N}\sum_{i=1}^{N} I_{Bi}.$$

The 2nd moment is the standard deviation σ, and given as [35]

$$\sigma_R = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left|I_{Ri} - \mu_R\right|^2},$$
$$\sigma_G = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left|I_{Gi} - \mu_G\right|^2},$$
$$\sigma_B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left|I_{Bi} - \mu_B\right|^2}.$$

The 3rd moment is the skewness *s*, which represents the degree of asymmetry around the mean value, given as [36]

$$s_R = \frac{\frac{1}{N}\sum_{i=1}^{N}(I_{Ri} - \mu_R)^3}{\sigma_R{}^3},$$
$$s_G = \frac{\frac{1}{N}\sum_{i=1}^{N}(I_{Gi} - \mu_G)^3}{\sigma_G{}^3},$$
$$s_B = \frac{\frac{1}{N}\sum_{i=1}^{N}(I_{Bi} - \mu_B)^3}{\sigma_B{}^3}.$$

The 4th moment is the kurtosis *k* which characterizes how outlier-prone the color of the image is, and can be computed as [37]

$$k_R = \frac{\frac{1}{N}\sum_{i=1}^{N}(I_{Ri} - \mu_R)^4}{\sigma_R{}^4},$$
$$k_G = \frac{\frac{1}{N}\sum_{i=1}^{N}(I_{Gi} - \mu_G)^4}{\sigma_G{}^4},$$
$$k_B = \frac{\frac{1}{N}\sum_{i=1}^{N}(I_{Bi} - \mu_B)^4}{\sigma_B{}^4}.$$

The MATLAB function *ExtractColorMoments.m* for computation of color moments information of a specified boundaries of an image, is provided in Appendix A.

E. FEATURES SUMMARY

Relevant features are selected as the count of CPs in image partitions and image color moments. From the count of CPs in each defined partitions of *R*x*C* dimension, we have *R*x*C* extracted class features, and there are 12 class features from the extraction of the image color moments. Therefore, we have (*R*x*C*+12) features to apply to the ship classification stage.

V. RESULTS AND ANALYSIS

In this chapter, we investigate ship classification under hazy conditions using Random Forests (RF) classification. Two models are considered: The first model is trained with the baseline dataset that is comprised of only the haze-free ship images. The second model is trained with our dataset augmented with synthetic ship images under haze conditions. The objective is to quantitatively compare the ship classification results between classifier models that were trained with /and without ship images under haze conditions. Classification rates are computed using a common test dataset consisting of flipped haze-free ship images and synthetic hazy ship images, during the testing phase. This chapter presents the ship classification performance results for these two considered models.

A. TERMINOLOGY

The terminologies used in this ship classification study includes:

1. Training Sets, Test Set and Models

The baseline images dataset consisting of 6,680 haze-free ship images, is termed the baseline training set. This baseline training set contains no prior information on ship images under haze conditions. The *Baseline* model is created and trained using the baseline training set.

The synthetic images dataset (total: 33,400 images) includes flipped haze-free ships images, homogeneous hazy ship images and heterogeneous hazy ship images. The function *cvpartition.m* is used to randomly partition the synthetic images dataset into a synthetic training set and a test set with stratification. A 50/50 ratio is set for partitioning; thus we have a synthetic training set and a test set each consisting of 16,700 images, respectively. The *Augmented* model is created and trained using the combination of the baseline training set and synthetic training set, which consists of haze-free ship images and ship images under haze conditions.

This test set comprising flipped haze-free ship images and ship images under hazy conditions will be used for the testing of the *Baseline* model and *Augmented* model. A graphical depiction of the training sets, test set and the models involved, is shown in Figure 17.

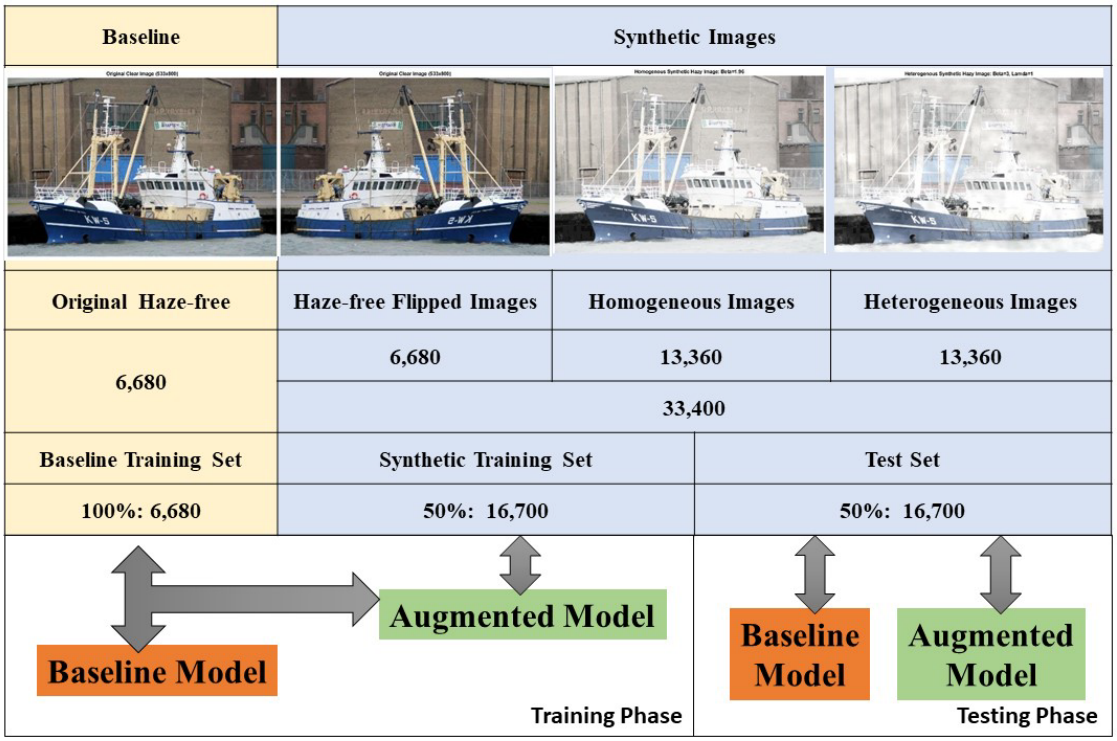


Figure 17. Training sets, test set, and models

2. Class Features

Three different configurations of class features corresponding to three different image partitions (i.e., *4x3*, *6x8*, and *8x8*), are considered in our study to get CPs features. In addition, we extract 12 class features from the image higher order color moments (CMs). The summary of class features for the different configurations is shown in Figure 18.

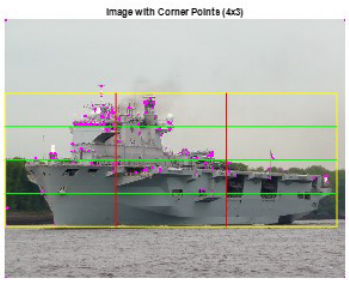
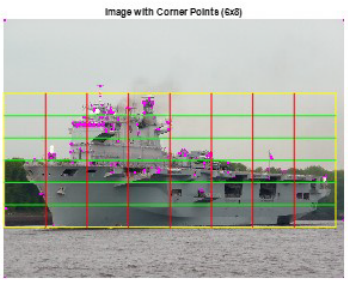
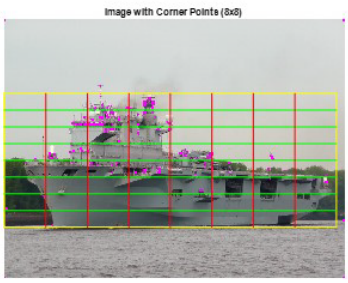
Class Features Configuration		
4x3	**6x8**	**8x8**
		
12 CPs features	48 CPs features	64 CPs features
12 CMs features	12 CMs features	12 CMs features
Total: 24 class features	**Total: 60 class features**	**Total: 76 class features**

Figure 18. Class features configurations

3. Classification Rate

The success rate of correct classification (or classification rate), used to evaluate classifier performance, is defined as the average percentages of correctly classified observations for all classes. A confusion matrix example that displays the total number of observations in each cell is shown in Figure 19. The rows of the confusion matrix correspond to the true class, and the columns correspond to the predicted class. Diagonal and off-diagonal cells correspond to correctly and incorrectly classified observations, respectively. The row-normalized row summary displays the percentages of correctly (i.e., column in blue) and incorrectly (i.e., column in beige) classified observations for each true class. The classification rate of the example in Figure 18 is therefore the average percentages of 87.9%, 82.4% and 80.9% (i.e., equals to 83.73%).

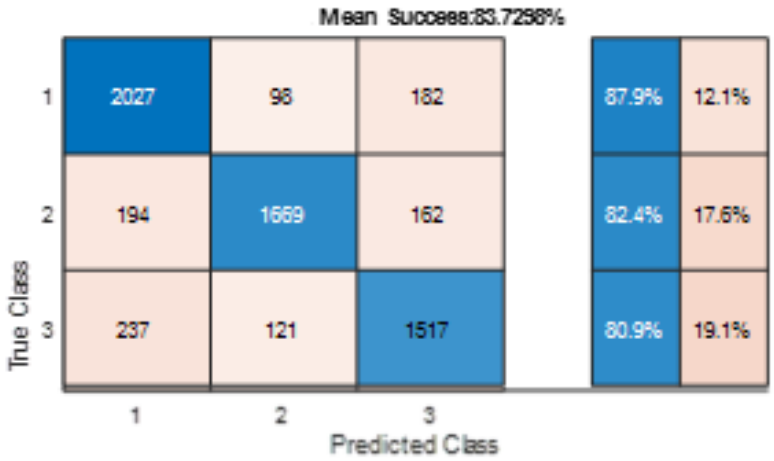


Figure 19. Confusion matrix and success rate example

B. RANDOM FORESTS CLASSIFICATION

Random forests (RF) is a machine learning algorithm which uses ensemble (or forest) of decision trees for classification and regression [38]. In random forests, the collection of predictor variables is randomly restricted in each split to form diverse trees (i.e., the forests). The maximum number of random splits, termed *NumMaxSplits*, and the total number of trees, termed *NumTrees*, are parameters of random forests that affect classification performance [39]. We thus examine classification rates obtained for various maximum numbers of random splits and various total numbers of trees to derive the optimum parameters for final testing. The mathematical representation of the RF classification is provided in Appendix B.

1. Optimum Parameters

For training, we used the MATLAB function *fitcensemble.m* to create the *Baseline* and *Augmented* models. These two models were then used with the function *predict.m* to compute ship classification rates for the test set. We compare performance rates of predictions returned by the trained models based on the three different configurations of class features. We varied the maximum number of splits *NumMaxSplits* using the values in the sequence $\{3^0,3^1,...,3^m\}$. m is such that 3^m is no greater than $n - 1$ [40], where n is equal to the number of observations in the augmented training set; that is, n equals to

(6,680+16,700) 23,380. The values of *NumMaxSplits* hence consists of *{1,3,9,27,81,243,729,2187,6561,19683}*.

Due to time constraint, only three runs were computed for each configuration. The average classifications rates versus *NumMaxSplits* is shown in Figure 20, and the detailed performance results are provided in Appendix C. We observed that the classification rates remained relatively unchanged at values of *NumMaxSplits* greater than 2,000 and 6,000 for the *Baseline* model and *Augmented* model, respectively. We also noted that the *4x3 Augmented* model has the best performance among the rest, with an average classification rate of 69.05%.

The total number of trees also affects the performance of the random forests algorithm, thus we also varied the total numbers of trees *NumTrees* to examine the models performances of the three different image partition configurations. A typical total number of trees to be generated in random forests algorithm ranges under 300 [41]. We varied the *NumTrees* to range from 10 to 500 in this study. The classification rates of correct classifications versus *NumTrees* is shown in Figure 21. We observed that classification rates remained relatively unchanged at values of *NumTrees* greater than 200 for all cases.

The *4x3 Augmented* model was also observed to emerge as the best performer among the rest, consistent with the performance seen in Figure 20. Note the performance of the *Baseline* models are largely similar for the three configurations (i.e., *4x3, 6x8, 8x8*). From the findings, we chose the *4x3* configuration with the maximum number of splits *NumMaxSplits* equals to 6,561, and total number of trees *NumTrees* equals to 200, as the optimum parameters for another test composing 10 independent runs.

In addition, we observed that the image partition configuration which has the least segments in our tests, provided the best classification rates. We hypothesized that fewer segments in the image partitions may be less sensitive to orientation changes of images, hence producing better classification rates. Considering this behavior, we investigated the *2x2* image partition configuration, which has even fewer segments than the *4x3* configuration, to determine if the classification performance peaks somewhere near our selected *4x3* configuration.

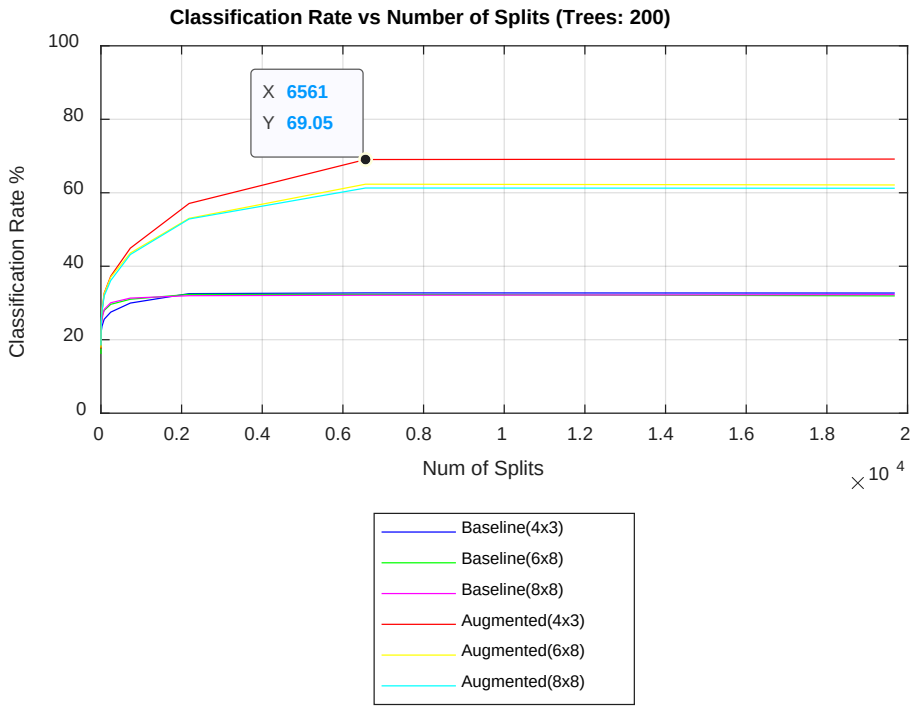


Figure 20. Classification rates vs. number of maximum splits

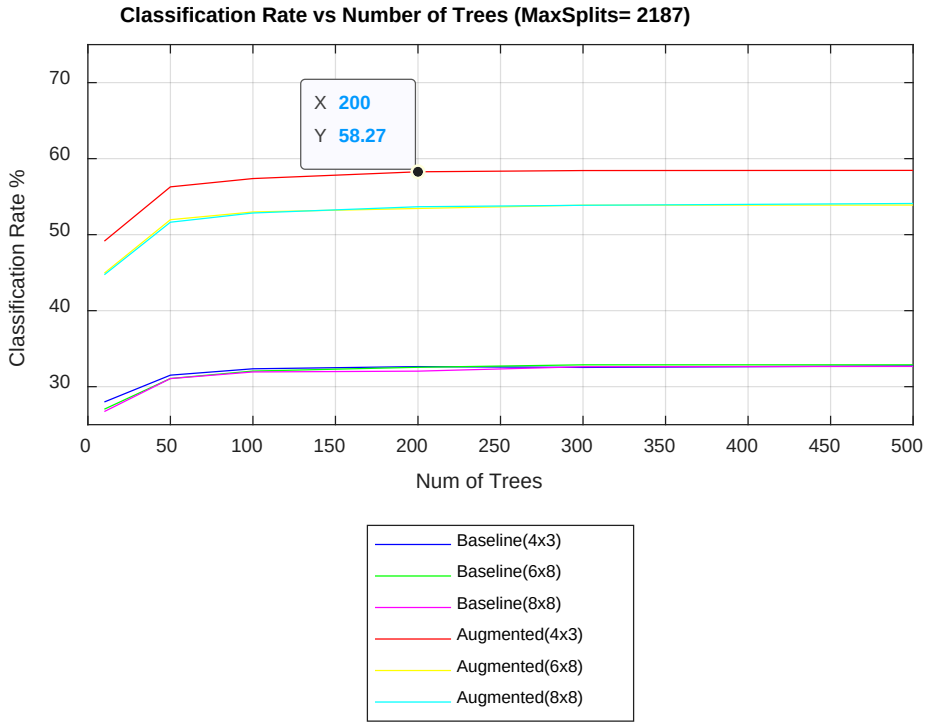


Figure 21. Classification rates vs. number of trees

2. Test Results

A separate test consisting of 10 independent runs was conducted using the *4x3* and *2x2* configurations with optimum parameters listed in Table 4.

Table 4. Optimum parameters

Parameters	**Configuration**	
	4x3	***2x2***
NumMaxSplits	6,561	6,561
NumTrees	200	200

The performance of the two configurations were computed over 10 independent runs (i.e., training set and test set are independently generated for each run), and we observed that the *Augmented* model for the *4x3* and *2x2* configurations achieved an average classification rates at about 70% and 65%, respectively, and the *Baseline* model only achieved at about 32% and 27%, respectively, listed in Table 5. The detailed results are provided in Appendix C. Note the close to 40% improvement in classification rate performance between the *Baseline* model and the *Augmented* model for both the *4x3* and *2x2* configurations, tabulated in Table 5. This improvement demonstrated the ability of using augmented synthetic images to improve ship classification rate performance

Since both the *4x3 Augmented* and *2x2 Augmented* attained better classification rate as compared with the *6x8 Augmented* and *8x8 Augmented*, we can thus infer that fewer segments used in the image partitioning stage improves classification rates. Collective results from Figure 20, Figure 21 and Table 5 showed that the *4x3* configuration is still a better performer than the *2x2* configuration. These results indicate that the optimal number of segments in the image partitions is closer to the *4x3* configuration.

Table 5. Classification rates results

Run	**Classification Rate**			
	4x3 Baseline	***4x3 Augmented***	***2x2 Baseline***	***2x2 Augmented***
Average	32.4097	69.7295	27.4707	65.3488
95% C.I. Lower	32.3124	69.6268	27.3832	65.2369
95% C.I. Upper	32.5190	69.8147	27.5597	65.4738

3. Classifier Strength

We observed that Classes 1, 3, and 6 (i.e., container ships, cargo ships, and sailing vessels) have the highest classification rates for the *4x3* and *2x2 Augmented* models, as illustrated by the confusion matrices example shown in Figure 22. The confusion matrices generated from all tests are provided in Appendix C. Ten independent runs of testing for these three ship classes using the *4x3 Augmented* (best performing) model attained an average classification rate of about 82%, as listed in Table 6.

Figure 22. Confusion matrices example

Table 6. Classification rates for classes 1,3 and 6 (10 runs)

Classification Rate (Classes: 1, 3, 6 Only) ***4x3 Augmented***			
Class	6	3	1
Average Per Class	82.5340	82.0235	81.9181
Overall Average	82.1586		
95% C.I. Lower	82.0702		
95% C.I. Upper	82.2506		

C. RESULTS SUMMARY

The *4x3 Augmented* configuration was identified to be the best performing configuration in our test, with classification rates performance at close to 70% for eight ship classes, and at over 80% accuracy for the three ships classes (i.e., container ships, cargo ships, and sailing vessels).

VI. CONCLUSIONS AND FUTURE WORK

This study considered the ship classification problem under hazy conditions. The lack of readily available real-life data calls for the use of synthetic hazy images to examine the impact of haze on ship classification. This thesis implements the atmospheric scattering model to generate a dataset of synthetic hazy images to augment the original dataset comprising haze-free ship images. Next, it proposes using (1) the count of CPs located within the image partitions; and (2) the higher order color moments information, as class features for ship classification. Finally, it demonstrates the ability of using a dataset augmented with synthetic hazy ship images to improve ship classification rate performance using RF classification.

Results show that (1) the average ship classification rate is about 40% higher when the model is trained using a dataset augmented with synthetic hazy ship images; and (2) the classifier can classify for ship classes such as container ships, cargo ships, and sailing vessels, with an 80% average accuracy rates.

Considerations for future work could include collecting ship images under real hazy conditions and validating results obtained in this study using the collected field data. Future improvements could also include different class features such as characterization of the ship images into its two-dimensional power spectrum representation, which could potentially improve classification rates for other ship classes.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB CODES

A. TRANSMISSION.M

```
% Codes in this function are adapted from Pseudo Codes found in
% [15] F. Guo, J. Tang, and H. Peng, “A Markov Random Field Model
% for the Restoration of Foggy Images,” International Journal of
% Advanced Robotic Systems, vol. 11, no. 6, p. 92–,
% Jun. 2014, doi: 10.5772/58674.

% NOTE: This function requires the GCO-v3.0 library,
% Developed by O.Veksler and A. Delong, and
% Source at https://vision.cs.uwaterloo.ca/code/

% Input: RGB image x
% Output: Returns a column vector of all labels.

function  Label = transmission(x)

    X = rgb2gray(x);
    g = 1;

    [M,N] = size(X);
    NumSites = N*M;
    NumLabels = 32;

%     figure(3*(cnt-1)+1);
%     imshow(x);
%     text = strcat(‘Original Clear Image (‘,num2str(M),’x’, num2str(N),’)’);
%     title(text);

    Data = reshape(X,[],1);

    h = GCO_Create(NumSites,NumLabels);

    % Compute Data Term
    for i=1:NumLabels
        for j = 1:NumSites
            L(i)= (i-1)/(NumLabels-1)*255;
            DataTerm(i,j) = abs(Data(j) - L(i));
        end
    end

    GCO_SetDataCost(h, DataTerm);

    % Compute Smooth Term
    for i=1:NumLabels
        for j = 1:NumLabels
            SmoothTerm(i,j) = g*abs((i)-j);
        end
```

```
    end

    GCO_SetSmoothCost(h,SmoothTerm);

    GCO_Expansion(h);

    Label = GCO_GetLabeling(h);

    % Show Initial Transmission map
%     t = 255 - (Label - 1)*8;
%     t = uint8(t);
%     I = reshape(t,M,N);

%     imshow(I);
%     text = strcat('Generated Depth Map (',num2str(M),'x', num2str(N),')');
%     title(text);
end
```

B. SETHAZE.M

```
%                       Date:26-08-2020(DD-MM-YYYY)                    %
% This function is written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Using Atmospheric Scattering Model to Set Haze in Synthetic Image

function [HazyImage, J] = setHaze(x, tx, beta, Lamda, h, nx)

% Input:
% Haze-free RGB image, x
% Label values in transmission map, tx
% Extinction coefficient, beta
% Wavelength, Lamda, equals 1 for hazy and foggy scenes
% Flag, h equals 0 for homogeneous, h = 1 for heterogeneous
% FBM noise image, nx

% Output:
% Rendered RGB synthetic hazy image, HazyImage
% Single dimensional vector of the reshaped RGB image, J

    Airlight = 255; % Global airlight coefficients
    lamda = Lamda;

    hetereogenous = h; % 0 for homogenous, 1 for hetereogenous

    X = double(reshape(x,[],3)); % reshape into 3 vectors, into R,G,B
    tx = exp(-tx*(beta)*Lamda);

    % Constant
    t0 = 0.1;
```

```
    if hetereogenous == 1
        %tx_lamda = exp(-tx*beta*lamda);
        % Heterogeneous Hazy Image
        max_tx = max(tx,t0);
        J = (X - Airlight).*(max_tx)+ Airlight;

        H = J + nx;
    else
    % Homogeneous Hazy Image
        J = X.*tx;
        H = J + Airlight*(1-tx);
    end

    H = uint8(round(H,0));
    [M,N,D] = size(x);

    % Reshape the RGB vectors into a RGB image of original dimension
    HazyImage = reshape(H,M,N,D);

    % Show Hazy Image
%     imshow(HazyImage);
%
%     if hetereogenous == 1
%         text = strcat('Heterogenous Synthetic Hazy Image: Beta=',num2str(beta),...
%             ', Lamda=', num2str(lamda));
%     else
%         text = strcat('Homogenous Synthetic Hazy Image: Beta=',num2str(beta),...
%             ', Lamda=', num2str(lamda));
%     end
%     title(text);
end
```

C. FBM.M

```
%                       Date:26-08-2020(DD-MM-YYYY)                    %
% This function is written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Generate Fractional Brown Motion (FBM) Noise Image

function image = FBM(N,M)

% Input: Required dimensions of FBM noise image, NxM
% Output: Returns FBM noise image of NxM dimension, image

% Dimenision of each FBM noise tiles: 256x256
n = 256;
m = 256;
im = zeros(n, m);
```

```
% Generate FBM tiles with 256 x 256 dimension
im1 = FBM_noise(im);
im2 = FBM_noise(im);
im3 = FBM_noise(im);
im4 = FBM_noise(im);
im5 = FBM_noise(im);
im6 = FBM_noise(im);
im7 = FBM_noise(im);
im8 = FBM_noise(im);
im9 = FBM_noise(im);
im10 = FBM_noise(im);

% Concatenate into 512 x 256
image1a = cat(1,im1,im2);
image2a = cat(1,im3,im4);
image1b = cat(1,im5,im6);
image2b = cat(1,im7,im8);
image3a = cat(1,im1',im10');
image3b = cat(1,im3',im8');
image4a = cat(1,im2',im7');
image4b = cat(1,im4',im6');
image5a = cat(1,im9,im10);
image5b = cat(1,im5',im9');

% Concatenate into 1024 x 256
image1 = cat(1,image2a,image1b);
image2 = cat(1,image1a,image2b);
image3 = cat(1,image3a,image4b);
image4 = cat(1,image4a,image5b);
image5 = cat(1,image2a,image5a);
image6 = cat(1,image1a,image4b);
image7 = cat(1,image5a,image5b);
image8 = cat(1,image2b,image3b);

% Concatenate into 1024 x 2048 baseline FBM noise image
I = cat(2,image1,image2,image3,image4,image5,image6,image7,image8);
image = imgaussfilt(I, 5);

% Crop FBM baseline noise image to required dimension of NxM
image = image(1:N,1:M); % Output FBM Noise image to required dimensions
end
```

D. FBM_NOISE.M

```
% Codes in this function are adapted from
% [16] "Noise Fractals and Clouds « null program."
% https://nullprogram.com/blog/2007/11/20/ (accessed Jul. 25, 2020).

% Generate FBM image
```

```
function im = FBM_noise(im)

    [n, m] = size(im);
    i = 0;
    w = sqrt(n*m);

    while w > 3
        i = i + 1;
        d = interp2(randn(n, m), i-1, 'spline');
        im = im + i * d(1:n, 1:m);
        w = w - ceil(w/2 - 1);
    end
end
```

E. GENERATEHAZYIMAGE.M

```
%                       Date:26-08-2020(DD-MM-YYYY)                    %
% This function is written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Generate Homogeneous and Heterogeneous Hazy Images

clc;
FBM_image = FBM(1024,2048); % Generate Baseline FBM Noise Image once

for class_num = 1:8
close all;
clearvars -except class_num FBM_image;

folder = 'C:\Users\admin\Documents\MATLAB\Thesis\Random Forest\Ship Images Database
Flipped & Hazy\';

    switch class_num
        case 1
            subfolder = 'W0_1';
            class_name = 'Container Ship';
        case 2
            subfolder = 'W2_1';
            class_name = 'Cruise';
        case 3
            subfolder = 'W4_1';
            class_name = 'Roro Cargo';
        case 4
            subfolder = 'W5_1';
            class_name = 'Tug';
        case 5
            subfolder = 'W8_1';
            class_name = 'Yacht';
        case 6
            subfolder = 'W9_1';
```

```
            class_name = 'Sailing Vessel';
        case 7
            subfolder = 'W17_1';
            class_name = 'Warship';
        case 8
            subfolder = 'W22_1';
            class_name = 'Fishing Vessel';
    end

    folder = strcat(folder,subfolder,'\');

    listing = dir (folder);
    ll = length(listing);
    start = 1;
    limit = ll;

    for ii = start:limit
        f = listing(ii).name;
        [~,filename,ext] = fileparts(f);
        if ext == '.jpg'

            file_str = strcat(folder,f);
            x = imread(file_str);
            if ndims(x) == 3 % check if image is RGB, omit if its not

                % Generate FBM Noise Image
                [n,m,~] = size(x);
                FBM_noise = FBM_image(1:m,1:n);
                k = 0.15;
                nx = k* double(reshape(FBM_noise,[],1));

                % Generate transmission map & labels
                Label = transmission(x);
                tx = (double(Label)) / 32;

                beta = 1.96; % extinction coefficient = 1.96 for visibility distance ~2-
4km
                lamda = 1;

                % Generate homogenenous image
                hetereogenous = 0; % 0 for homogenous
                [HazyImage, ~] = setHaze(x,tx,beta,lamda,cnt,hetereogenous, nx);
                savefilename = strcat(folder,filename,'_homo',ext);
                imwrite(HazyImage,savefilename); % Saving hazy images

                % Generate hetereogenous image
                hetereogenous = 1; %1 for hetereogenous
                [HazyImage, ~] = setHaze(x,tx,beta,lamda,cnt,hetereogenous, nx);

                savefilename = strcat(folder,filename,'_heter',ext);
                imwrite(HazyImage,savefilename); % Saving hazy images
```

```
                clear x HazyImage tx Label FBM_noise nx;
                clc;
            end
        end
    end
    fprintf('Ended, Class_Num:%d\n',class_num);
end
```

F. DETECTSHIPANDPARTITION.M

```
%                       Date:26-08-2020(DD-MM-YYYY)                    %
% This function is written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Detect Ship Structure Boundaries & Perform Image Partitioning

% Inputs:
% Original RGB Image = Original_Image
% Intended Number of Horizontal Segments = NumRowSegments
% Intended Number of Vertical Segments = NumColSegments

% Outputs:
% Cropped Image with borders & text at bottom of image = I
% Binary Image of Cropped Image = binaryImage
% Points depicting area of detected ship = FourCorners
% FourCorners = [topRow, bottomRow, leftColumn, rightColumn]
% row_lines = partition horizontal lines indices
% col_lines = partition vertical lines indices

function [I, binaryImage, FourCorners, row_lines, col_lines] = ...
    DetectShipAndPartition (Original_Image, NumRowSegments, NumColSegments)

    conn = 80; % 80 pixels, for exclusion of perimeters with less than 80 pixels of info
    N = NumRowSegments; % Number of Horizontal partition, 3
    M = NumColSegments; % Number of Vertical partition, 4
    crop = 10; % pixels from border to crop, to remove unnecessary borders

    [h,w,~]=size(Original_Image); % w = width, h = height
    crop_rect = [crop crop w-2*crop h-2*crop];
    I = imcrop(Original_Image,crop_rect);

    results = ocr(I);
    %wording = results.WordBoundingBoxes(2:end,:);
    wording = results.WordBoundingBoxes;
    % Cropping words portion at bottom of image
    if ~isnan(wording)
       %fprintf('Wordings in Image\n');
       [h1,w1,~]=size(I); % w = width, h = height

       wording_list = [];
```

```
        [t,~] = size(wording);
        for k = 1:t
            % Omit cropping if the wording is at first row & wording occurs
            % above 20% from image bottom
            twentypercent = h1 - floor(0.2*h1);
            if ((wording(k,2) ~= 1) && (wording(k,2)> twentypercent))
                wording_list = [wording_list;wording(k,:)];
            end
        end
        if ~isnan(wording_list) % check if there are wordings for cropping
            word_vert = min(wording_list(:,2));
            word_crop_rect = [1 1 w1 word_vert];
            I = imcrop(I,word_crop_rect);
        end
    end

    img = rgb2gray(I);
    BW = edge(img,'Sobel'); % locating edges / lines in gray image
    BW_out = bwareaopen(imfill(BW,'holes'),conn); % fill areas with more than "Conn" of
pixels

    % Finding all the points in images in 'white', i.e., lines/edges
    binaryImage = BW_out;
    [rows, columns] = find(binaryImage);
    topRow = min(rows); % Min X-value
    bottomRow = max(rows); % Max X-value
    leftColumn = min(columns);% Min Y-value
    rightColumn = max(columns); % Max Y-value

    % Output
    FourCorners = [topRow, bottomRow, leftColumn, rightColumn];

    % Partition Detected Area in Image into N rows x M coloumns
    col_dimension = bottomRow - topRow;
    row_dimension = rightColumn - leftColumn;
    horizontal_part_width = col_dimension / N;
    vertical_part_width = row_dimension / M;
    for r = 1:N-1
       row_lines(r) = topRow +  r*horizontal_part_width;
    end

    for c = 1:M-1
       col_lines(c) = leftColumn +  c*vertical_part_width;
    end

end
```

G. GENERATECORNERPOINTS.M

```
%                       Date:26-08-2020(DD-MM-YYYY)                     %
% This function is written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Generate corner points in image.

% Input: RGB Image, x
% Output: Image with Corner Points(CPs) inserted, Image_CP
% Output: Computed Corner Points (CPs) in X-Y coordinates.

function [Image_CP,points] = GenerateCornerPoints(x)

    I = rgb2gray(x);

    hx = [-1 0 1;
          -2 0 2;
          -1 0 1];

    hy = [-1 -2 -1;
           0  0  0;
           1  2  1];

    [N,M] = size(I);

    Ix = conv2(I,hx);
    Ix = Ix(2:N+1,2:M+1);

    Iy = conv2(I,hy);
    Iy = Iy(2:N+1,2:M+1);

    W = [1 1 1 1 1;
         1 1 1 1 1;
         1 1 1 1 1;
         1 1 1 1 1;
         1 1 1 1 1];

    Gx = conv2(Ix.*Ix, W);
    Gy = conv2(Iy.*Iy, W);
    Gxy = conv2(Ix.*Iy, W);

    Gx = Gx(3:N+2, 3:M+2);
    Gy = Gy(3:N+2, 3:M+2);
    Gxy = Gxy(3:N+2, 3:M+2);

    for n1 = 1:N
        for n2 = 1:M
            G = [Gx(n1,n2)  Gxy(n1,n2);
                 Gxy(n1,n2) Gy(n1,n2)];
            lamda(n1,n2) = min(eig(G));
```

```
        end
    end

    % Set threshold to be 15% of the maximum value from the min eigenvalues
    max_lamda = max(max(lamda));
    threshold = max_lamda*0.15;

    % Locate Corner Points
    points = [];
    for n1 = 1:N
        for n2 = 1:M
            if (lamda(n1,n2)>threshold) && (lamda(n1,n2)<=max_lamda)
                CP = [n2, n1];
                points = [points; CP];
            end
        end
    end

    % Show Images
    Image_CP = insertMarker(x, points,'x','color','m','size',1);

end
```

H. COUNTINGCPS.M

```
%                       Date:26-08-2020(DD-MM-YYYY)                    %
% This function is written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Count CPs located within partitions

%Inputs:
% CPs = points
% Detected area = topRow, bottomRow, leftColumn, rightColumn
% Partition Rows = row_lines
% Partition columns = col_lines

%Output: Struct data
% data.CPs_partition = [topleft(x,y),bottomright(x,y)] of
% each partition (in sequence from left-to-right, then top-to-bottom)
% data.CPs_index = all indices of CPs within each corresponding partition
% data.CPs_count = number of CPs in each corresponding partition

function data = CountingCPs...
    (points,topRow,bottomRow,leftColumn,rightColumn,...
    row_lines,col_lines)

    % Counting CPs
    row_rng = [topRow, row_lines, bottomRow]; % Y-values
    col_rng = [leftColumn, col_lines, rightColumn]; % X-values
```

```
    partition_array = [];

    % Building from top to bottom
    for n = 1:length(row_rng)-1
        % Building from left to right
        for m = 1:length(col_rng)-1
            % Coordinates of topleft(x,y) & bottomright(x,y) of partition
            partition = [col_rng(m),row_rng(n),col_rng(m+1),row_rng(n+1)];
            partition_array = [partition_array;partition];
        end
    end

    [K,~] = size(partition_array);
    [P,~] = size(points);

    for k = 1:K % check for all partitions
        CPs_index = [];
        data(k).CPs_partition = partition_array(k,:);
        for p = 1:P % running through all CPs
            % current CP
            coord = [points(p,1),points(p,2)];
            topleft = [partition_array(k,1),partition_array(k,2)];
            bottomright = [partition_array(k,3),partition_array(k,4)];
            % checking if current CP is within current partition
            if ( (coord(1)>=topleft(1)) && (coord(2)>=topleft(2))...
                 && (coord(1)<bottomright(1)) && (coord(2)<bottomright(2)) )
                CPs_index = [CPs_index,p];
            end
        end
        data(k).CPs_index = CPs_index;
        data(k).CPs_count = length(CPs_index);
    end
end
```

I. EXTRACTCOLORMOMENTS.M

```
%                       Date:26-08-2020(DD-MM-YYYY)                    %
% This function is written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Generate corner points in image.
% Extract Higher Order Color Moments of Ship Structure

%Inputs:
% Original Image
% Detected area = topRow, bottomRow, leftColumn, rightColumn
% Partition Rows = row_lines
% Partition columns = col_lines

%Output: Struct colorMoments
```

```
% colorMoments.mean = [Mean_R, Mean_G, Mean_B]
% colorMoments.var = [Variance_R, Variance_G, Variance_B]
% colorMoments.skewness = [Skewness_R, Skewness_G, Skewness_B]
% colorMoments.kurtosis = [Kurtosis_R, Kurtosis_G, Kurtosis_B]

function colorMoments = ExtractColorMoments (Original_Image, topRow, bottomRow,
leftColumn, rightColumn, row_lines, col_lines)

    I = Original_Image;

    r1 = topRow;
    r2 = bottomRow;
    c1 = leftColumn;
    c2 = rightColumn;

    partition_image = I(r1:r2,c1:c2,:);
%     figure;
%     imshow(partition_image);

    block = reshape(partition_image,[],3);

    block_double = double(block);

    % Mean values for RGB
    currentMean = mean(block_double);

    % Variance values for RGB
    currentVar = std(block_double);

    % Skewness values for RBG
    currentSkewness = skewness(block_double,1);

    % Bias corrected values for RGB
    currentKurtosis = kurtosis(block_double,0);

    colorMoments.mean = currentMean;
    colorMoments.var = currentVar;
    colorMoments.skewness = currentSkewness;
    colorMoments.kurtosis = currentKurtosis;

end
```

J. RF50MEAN.M

```
%                      Date:26-08-2020(DD-MM-YYYY)                    %
% This codes are written by
% TANG JUN WEN (Naval Postgraduate School. CA, Monterey, USA)
% and this is free to use. email: junwen.tang.sn@nps.edu
% Compute classification rates for 10 independent runs and
% calculates the average classification rates and 95% confidence level
```

```
% for 4x3 configuration models

clear all;
clc;

Baseline_Success = [];
Augmented_Success = [];

% Collect success rates for 10 runs
for rr = 1:10
    [meanSuccess, meanSuccess2] = RF50Mean();
    Baseline_Success = [Baseline_Success;meanSuccess];
    Augmented_Success = [Augmented_Success; meanSuccess2];
end

% Average of 10 runs
meanBaseline_Success = mean(Baseline_Success)
meanAugmented_Success = mean(Augmented_Success)

% 95% Confidence Level
ci_aug = bootci(2000,@mean,Augmented_Success)
ci_base = bootci(2000,@mean,Baseline_Success)

%
function [meanSuccess, meanSuccess2] = RF50Mean()

Percent='50%: ';
N = 4; M = 3;
load('C:\Users\admin\Documents\MATLAB\Thesis\Random
Forest\Workspace\N4M3_Baseline_Only_6680.mat')
load('C:\Users\admin\Documents\MATLAB\Thesis\Random
Forest\Workspace\N4M3_Flipped_Hazy_Only_33400.mat')

%
Xtrain_Baseline = A;
Ytrain_Baseline = B;

cvpart50 = cvpartition(HB,'holdout',0.5);
Xtrain_50 = HA(training(cvpart50),:);
Ytrain_50 = HB(training(cvpart50),:);

Xtrain_New = [Xtrain_Baseline;Xtrain_50];
Ytrain_New = [Ytrain_Baseline;Ytrain_50];

Xtest_50 = HA(test(cvpart50),:);
Ytest_50 = HB(test(cvpart50),:);

% N4M3 Predictor Estimate

rng(1); % For reproducibility
t = templateTree('MaxNumSplits',6561);
Mdl_50 = fitcensemble(Xtrain_New,Ytrain_New,'Method','Bag',...
```

```
                    'NumLearningCycles',200,'Learners',t);

Mdl_Baseline = fitcensemble(Xtrain_Baseline,Ytrain_Baseline,'Method','Bag',...
                    'NumLearningCycles',200,'Learners',t);

% N4M3 Confusion Chart for Baseline Model
figure();
Percent='50%';
Ytest_pred_Baseline = predict(Mdl_Baseline,Xtest_50);
cm_Baseline = confusionchart(Ytest_50,Ytest_pred_Baseline,'RowSummary','row-
normalized','ColumnSummary','column-normalized');
cm =cm_Baseline.NormalizedValues;
diagM = diag(cm);
rowSum = sum(cm,2);
rowpercent = diagM./rowSum;
meanSuccess = mean(rowpercent)*100;
disp = strcat(Percent,'RF Baseline Model: Partitions
('',num2str(N),'x',num2str(M),''),' ,'MaxSplits:6561, Mean
Success:',num2str(meanSuccess),'%');
title(disp);

% Confusion Chart for Augment Model
figure();
Ytest_pred_New = predict(Mdl_50,Xtest_50);
cm_New = confusionchart(Ytest_50,Ytest_pred_New,'RowSummary','row-
normalized','ColumnSummary','column-normalized');
cm2 =cm_New.NormalizedValues;
diagM2 = diag(cm2);
rowSum2 = sum(cm2,2);
rowpercent2 = diagM2./rowSum2;
meanSuccess2 = mean(rowpercent2)*100;
disp = strcat(Percent,'RF Augmented Model: Partitions ('',num2str(N),'x',num2str(M),''), '',
''MaxSplits:6561, Mean Success:',num2str(meanSuccess2),'%');
title(disp);


% Sort Classification Rates in Confusion Matrix in Decending Order
cm_Baseline.Normalization = 'row-normalized';
sortClasses(cm_Baseline,'descending-diagonal');
cm_Baseline.Normalization = 'absolute';

cm_New.Normalization = 'row-normalized';
sortClasses(cm_New,'descending-diagonal');
cm_New.Normalization = 'absolute';

% Clear Unwanted Variables for Storage
clearvars -except meanSuccess meanSuccess2;

end
```

APPENDIX B. RANDOM FORESTS REPRESENTATION

This appendix shows the mathematical representation of the random forests classification. For further details on random forests algorithm, refer to [42].

A. VARIABLES

The following variables are used:

- D_n *is the training set, where* $D_n = \{(X_1,Y_1),\ldots,(X_n,Y_n)\}$,
- *n is the size of the training set,*
- X is the input random vector,
- Y is the labels vector,
- m_n is the classifier trained with D_n,
- M is the number of trees,
- Θ is a distribution of random vector
- θ_j is a vector of randomly drawn components from Θ with replacement,
- j is an integer, where $j=1,2,\ldots,M$,

B. CLASSIFICATION

The random forests classifier is acquired through a majority vote, and is given by [42, p. 9]

$$m_{M,n}(x;\theta_1,\ldots,\theta_M,D_n) = \begin{cases} 1 & if \ \frac{1}{M}\sum_{j=1}^{M} m_n(x;\theta_j,D_n) > 0.5 \\ 0 & otherwise \end{cases}.$$

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. TEST RESULTS

This appendix shows the details of our test results.

A. CLASSIFICATION RATES VS. MAXIMUM NUMBER OF SPLITS

Classification rates results for various values of *NumMaxSplits* for the *Baseline* and *Augmented* models using the *4x3*, *6x8*, and *8x8* configurations, as listed in Table 7 to Table 12.

Table 7. Classification rates vs. *NumMaxSplits* (*4x3 Baseline* model)

NumMaxSplits	**Classification Rates (*4x3 Baseline*)**			
	Run 1	**Run 2**	**Run 3**	**Average**
1	17.9038	17.8386	17.8038	17.8488
3	19.4039	19.8431	19.4401	19.5624
9	22.5356	22.0322	22.4715	22.3464
27	23.6090	23.4906	23.9835	23.6944
81	25.5124	25.5258	25.6887	25.5757
243	27.3503	27.6172	27.5549	27.5075
729	29.9892	29.9884	29.9508	29.9761
2187	32.4356	32.5673	32.6431	32.5487
6561	32.8655	32.4347	32.9201	32.7401
19683	32.7567	32.7224	32.6397	32.7063

Table 8. Classification rates vs. *NumMaxSplits* (*4x3 Augmented* model)

NumMaxSplits	**Classification Rates *(4x3 Augmented)***			
	Run 1	**Run 2**	**Run 3**	**Average**
1	17.4520	17.6417	17.5999	17.5645
3	18.9990	19.3592	19.1342	19.1641
9	22.8990	24.8437	23.7484	23.8304
27	27.0144	27.5420	28.1623	27.5729
81	32.0938	32.6388	33.2653	32.6660
243	36.3662	37.6441	37.8738	37.2947
729	44.2054	45.1723	45.3909	44.9229
2187	56.7003	56.8573	57.7129	57.0902
6561	68.6399	68.6896	69.8142	69.0479
19683	68.8283	68.7895	69.8956	69.1711

Table 9. Classification rates vs. *NumMaxSplits* (*6x8 Baseline* model)

NumMaxSplits	**Classification Rates *(6x8 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
1	16.1316	16.1991	16.0408	16.1238
3	19.8715	20.3914	20.3969	20.2199
9	24.0942	24.3974	24.6820	24.3912
27	26.0017	26.7582	26.7978	26.5192
81	27.6440	28.3202	27.9686	27.9776
243	29.4225	30.0610	29.4055	29.6297
729	30.9486	31.1928	30.9102	31.0172
2187	31.9850	32.5427	32.5136	32.3471
6561	32.2083	32.7195	32.0936	32.3405
19683	32.2778	32.1721	31.3497	31.9332

Table 10. Classification rates vs. *NumMaxSplits* (*6x8 Augmented* model)

NumMaxSplits	**Classification Rates *(6x8 Augmented)***			
	Run 1	**Run 2**	**Run 3**	**Average**
1	17.7628	18.1385	17.6641	17.8551
3	21.4157	22.0273	21.4153	21.6194
9	24.6488	25.2990	24.6395	24.8624
27	28.3785	28.8549	28.1689	28.4674
81	33.0632	32.3330	32.3103	32.5689
243	37.0746	36.9196	37.0594	37.0179
729	43.7041	43.3386	43.7269	43.5899
2187	53.7327	52.5803	52.8151	53.0427
6561	62.6622	62.1359	62.2069	62.3350
19683	62.6058	61.8908	61.8771	62.1246

Table 11. Classification rates vs. *NumMaxSplits* (*8x8 Baseline* model)

NumMaxSplits	**Classification Rates *(8x8 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
1	19.9920	19.7624	19.6965	19.8170
3	21.4902	21.6605	21.8687	21.6731
9	24.3705	23.4309	23.6748	23.8254
27	25.9564	25.2579	25.6076	25.6073
81	28.3662	27.8897	28.4754	28.2438
243	30.3184	29.6362	30.2005	30.0517
729	31.8571	30.7517	31.3772	31.3287
2187	32.2151	31.6541	32.0718	31.9803
6561	31.9828	32.1526	32.1991	32.1115
19683	32.5617	32.0077	32.2345	32.2680

Table 12. Classification rates vs. *NumMaxSplits* (*8x8 Augmented* model)

NumMaxSplits	**Classification Rates *(8x8 Augmented)***			
	Run 1	**Run 2**	**Run 3**	**Average**
1	18.4834	18.4939	18.3451	18.4408
3	22.9282	22.9436	20.6255	22.1658
9	25.1384	24.5535	25.4667	25.0529
27	27.8411	27.8149	27.8919	27.8493
81	31.7881	31.6682	32.3194	31.9253
243	35.9558	35.9247	36.4730	36.1178
729	43.1387	42.7503	43.6679	43.1856
2187	53.0255	52.2526	53.3108	52.8630
6561	61.4001	60.7457	61.7346	61.2935
19683	60.8829	60.9646	61.8318	61.2264

B. CLASSIFICATION RATES VS. NUMBER OF TREES

Classification rates results for various values of *NumTrees* for the *Baseline* and *Augmented* models using the *4x3*, *6x8*, and *8x8* configurations, as listed in Table 13 through Table 18.

Table 13. Classification rates vs. *NumTrees* (*4x3 Baseline* model)

NumTrees	**Classification Rates *(4x3 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
10	27.8791	28.3209	27.7637	27.9879
50	30.9544	31.8784	31.7095	31.5141
100	32.5096	32.4481	32.0976	32.3518
200	32.1923	33.3772	32.3464	32.6386
300	32.5486	32.7084	32.3349	32.5307
500	32.6009	32.8256	32.7659	32.7308

Table 14. Classification rates vs. *NumTrees* (*4x3 Augmented* model)

NumTrees	**Classification Rates *(4x3 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
10	48.0348	49.6127	49.7942	49.1472
50	55.7382	56.7186	56.3996	56.2855
100	56.7868	57.5240	57.8216	57.3775
200	58.0039	58.5686	58.2314	58.2680
300	58.1648	58.6688	58.4602	58.4313
500	58.1308	58.6174	58.6090	58.4524

Table 15. Classification rates vs. *NumTrees* (*6x8 Baseline* model)

NumTrees	**Classification Rates *(6x8 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
10	26.5235	27.6011	26.9870	27.0372
50	31.3486	31.1355	30.7362	31.0734
100	32.2062	32.1085	31.8396	32.0514
200	32.9926	32.4979	32.1584	32.5496
300	33.0560	32.9341	32.6272	32.8724
500	32.7817	32.8806	32.9632	32.8752

Table 16. Classification rates vs. *NumTrees* (*6x8 Augmented* model)

NumTrees	**Classification Rates *(6x8 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
10	44.5495	44.6688	45.5895	44.9359
50	51.2984	52.3137	52.2870	51.9664
100	52.3712	53.6523	52.9231	52.9822
200	52.8980	53.5848	53.8476	53.4435
300	53.2469	54.1889	54.1865	53.8741
500	53.3081	53.9665	54.3984	53.8910

Table 17. Classification rates vs. *NumTrees* (*8x8 Baseline* model)

NumTrees	**Classification Rates *(8x8 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
10	26.5034	27.2312	26.4505	26.7284
50	31.4125	30.9257	30.8509	31.0630
100	32.0586	31.4402	32.3097	31.9362
200	32.5453	31.4770	32.1007	32.0410
300	32.6526	32.0549	33.1651	32.6242
500	32.9423	32.3113	32.7355	32.6630

Table 18. Classification rates vs. *NumTrees* (*8x8 Augmented* model)

NumTrees	**Classification Rates *(8x8 Baseline)***			
	Run 1	**Run 2**	**Run 3**	**Average**
10	44.1869	45.4754	44.5196	44.7273
50	51.5701	51.9440	51.3877	51.6339
100	52.4426	53.4265	52.6742	52.8478
200	53.3491	54.1040	53.5580	53.6704
300	53.5372	54.3673	53.6775	53.8607
500	53.4567	54.7766	54.0649	54.0994

C. TEST RESULTS FOR *4X3* AND *2X2* CONFIGURATION

Classification rates results for all ship classes using the *4x3* and *2x2* configurations, as listed in Table 19. The results for classes 1, 3, and 6 only are tabulated in Table 20 and Table 21. The confusion matrices generated from the iterative tests using the *4x3* and *2x2* configurations, are shown in Figure 23 and Figure 24, respectively. The confusion matrices (for Classes 1,3,6) are shown in Figure 25.

Table 19. Classification rates for all classes (10 runs)

Run	**Classification Rate**			
	4x3 Baseline	***4x3 Augmented***	***2x2 Baseline***	***2x2 Augmented***
1	32.5372	69.8378	27.5839	65.7286
2	32.4090	69.6105	27.4449	65.2593
3	32.6949	69.6065	27.1784	65.5681
4	32.2558	69.7419	27.3217	65.5215
5	32.4407	69.9280	27.4733	65.4268
6	32.1784	69.8746	27.4180	65.2767
7	32.4178	69.6488	27.4726	65.0384
8	32.6690	69.4171	27.7607	65.2097
9	32.1993	69.8169	27.5007	65.2363
10	32.2947	69.8124	27.5524	65.2227
Average	32.4097	69.7295	27.4707	65.3488
95% C.I. Lower	32.3124	69.6268	27.3832	65.2369
95% C.I. Upper	32.5190	69.8147	27.5597	65.4738

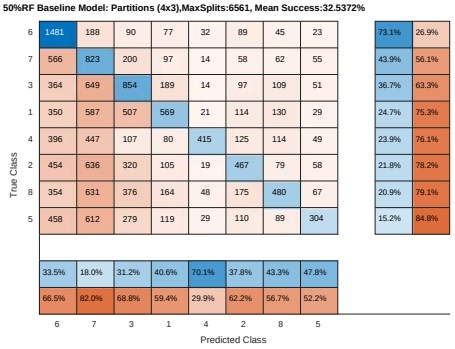
Table 20. Average classification rates for classes 1,3,6 only (10 runs)

Run	**Classification Rate (Classes: 1, 3, 6 Only)**	
	4x3 Baseline	***4x3 Augmented***
1	58.5228	82.0092
2	59.0838	82.0429
3	59.2173	82.2922
4	59.5443	82.3168
5	59.0809	82.2672
6	58.8555	82.0279
7	59.1970	82.0118
8	59.3180	82.2837
9	58.9767	82.3409
10	58.9974	81.9926
Average	59.0880	82.1586
95% C.I. Lower	58.9337	82.0702
95% C.I. Upper	59.2380	82.2506

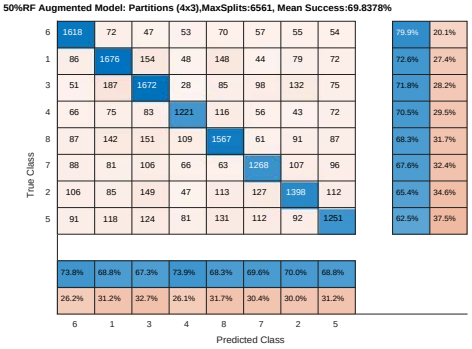
Table 21. Classification rates for classes 1,3,6 only

Run	**Classification Rates *4x3 Augmented***		
	Class 6	**Class 3**	**Class 1**
1	85.4886	80.9196	80.2774
2	79.8872	85.8835	81.7010
3	80.8336	80.0606	84.0493
4	86.1303	81.0055	79.7572
5	79.8439	85.7354	80.8336
6	81.0485	80.4508	85.5380
7	86.1303	80.4899	80.2774
8	80.1473	85.1431	80.9626
9	80.4899	79.6272	85.6367
10	85.3405	80.9196	80.1473
Average	82.5340	82.0235	81.9181

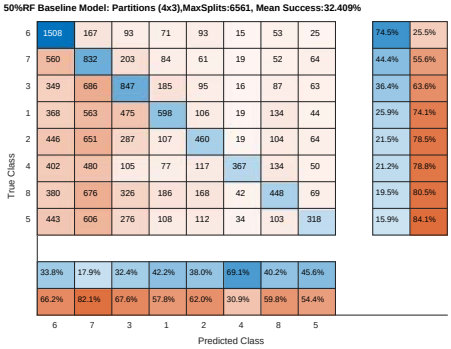
(a) *4x3 Baseline* model – Run 1



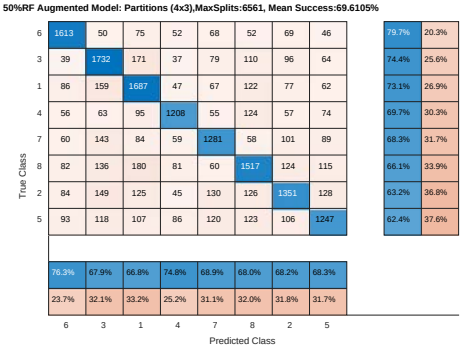
(b) *4x3 Augmented* model – Run 1



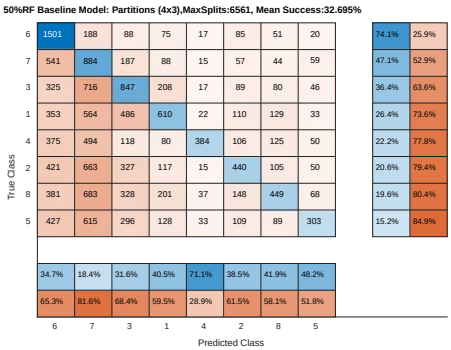
(c) *4x3 Baseline* model – Run 2



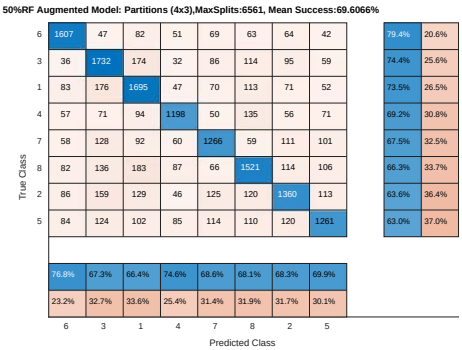
(d) *4x3 Augmented* model – Run 2



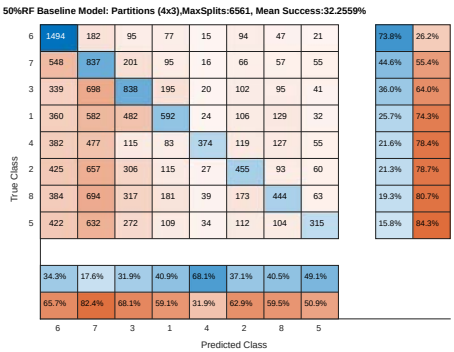
(e) *4x3 Baseline* model – Run 3



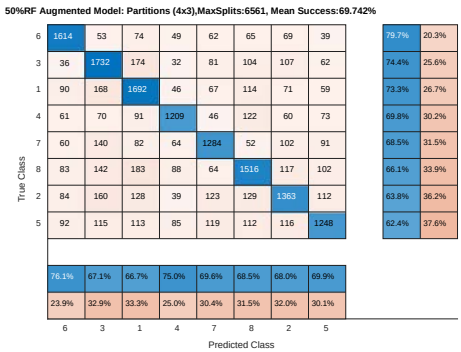
(f) *4x3 Augmented* model – Run 3



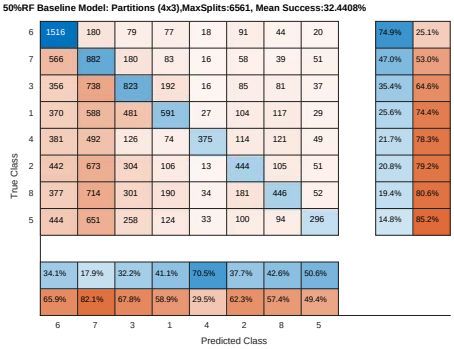
(g) *4x3 Baseline* model – Run 4



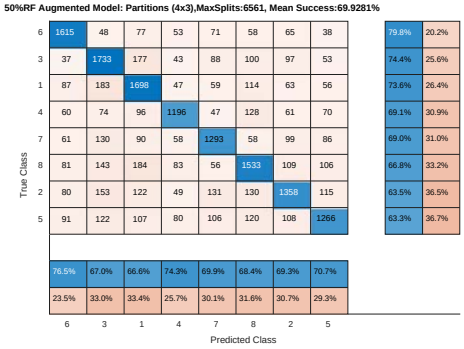
(h) *4x3 Augmented* model – Run 4



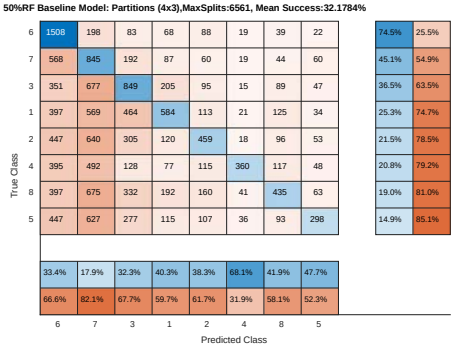
(i) *4x3 Baseline* model – Run 5



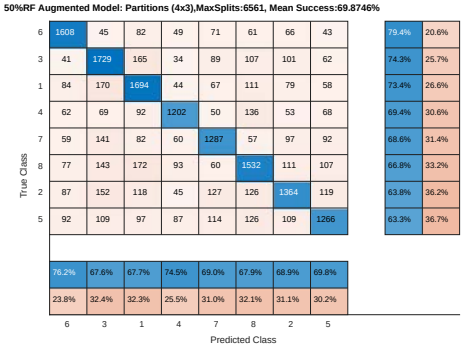
(j) *4x3 Augmented* model – Run 5



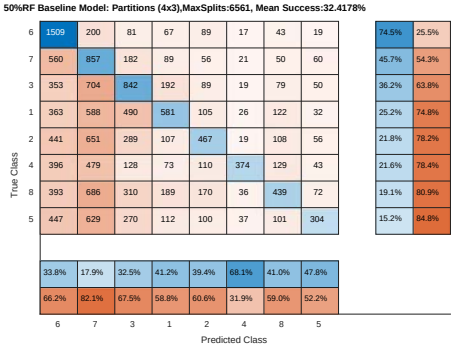
(k) *4x3 Baseline* model – Run 6



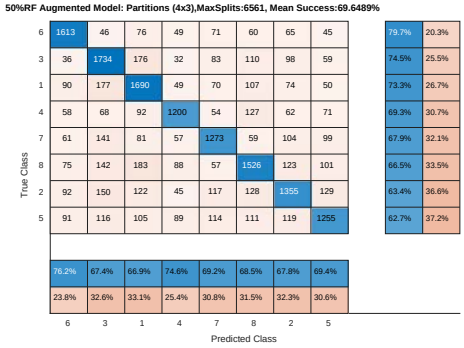
(l) *4x3 Augmented* model – Run 6



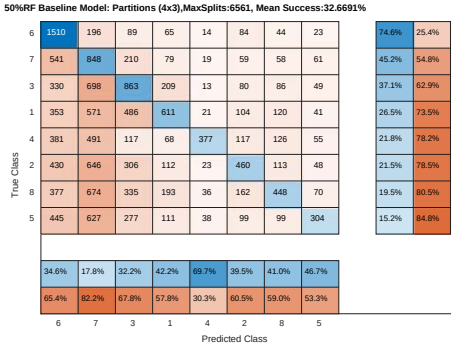
(m) *4x3 Baseline* model – Run 7



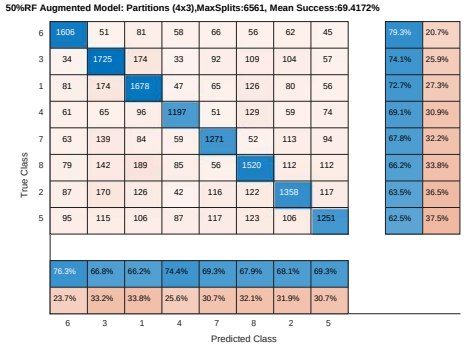
(n) *4x3 Augmented* model – Run 7



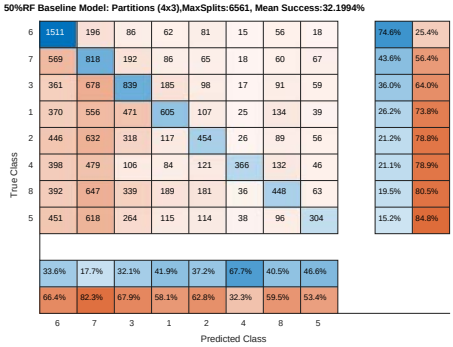
(o) *4x3 Baseline* model – Run 8



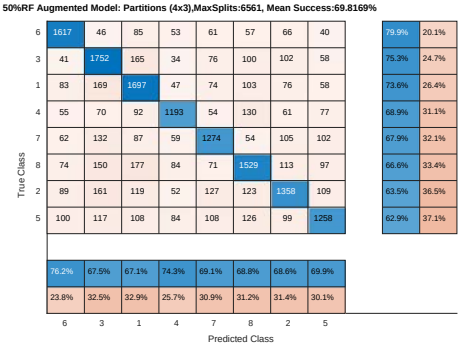
(p) *4x3 Augmented* model – Run 8



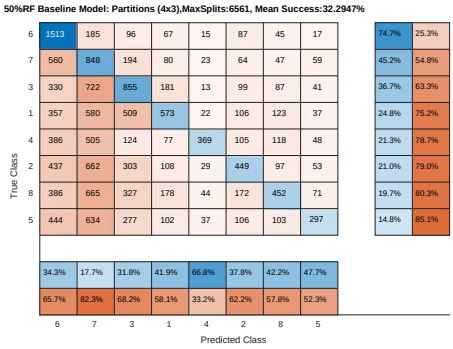
(q) *4x3 Baseline* model – Run 9



(r) *4x3 Augmented* model – Run 9



(s) *4x3 Baseline* model – Run 10



(t) *4x3 Augmented* model – Run 10

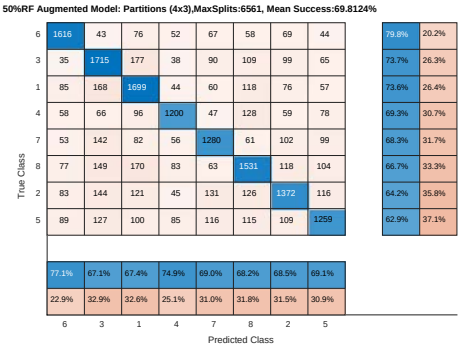
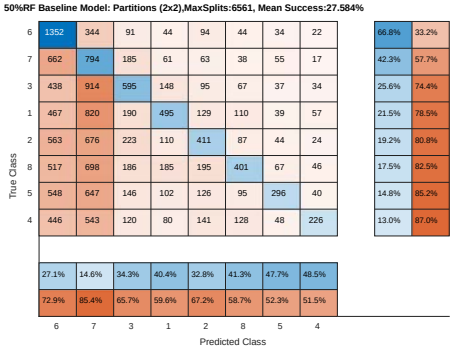
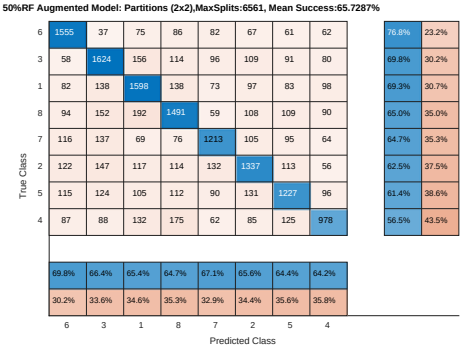


Figure 23. *4x3* configuration results (10 runs)

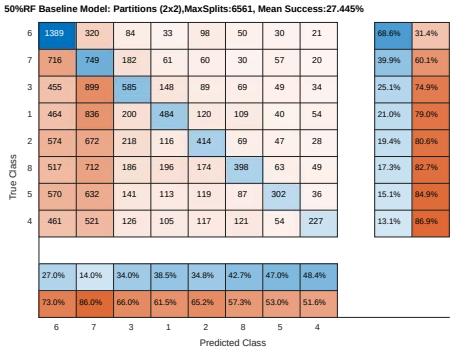
(a) *2x2 Baseline* model – Run 1



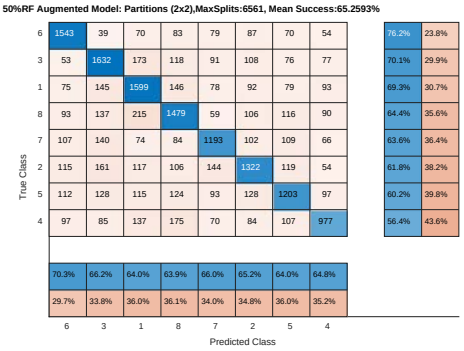
(b) *2x2 Augmented* model – Run 1



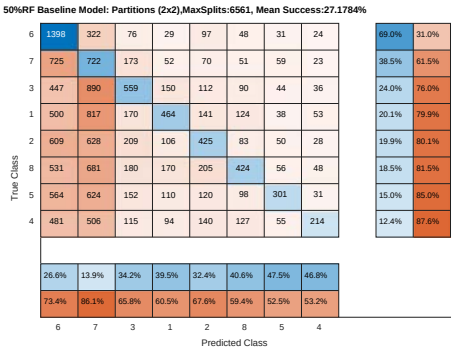
(c) *2x2 Baseline* model – Run 2



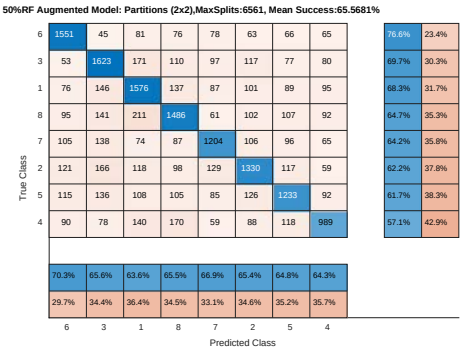
(d) *2x2 Augmented* model – Run 2



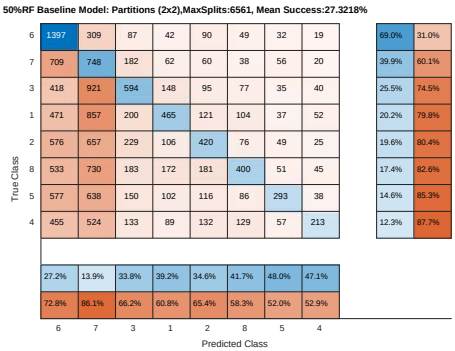
(e) *2x2 Baseline* model – Run 3



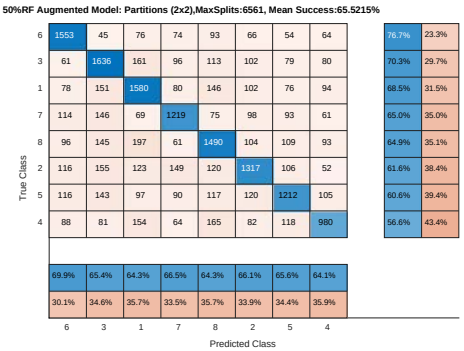
(f) *2x2 Augmented* model – Run 3



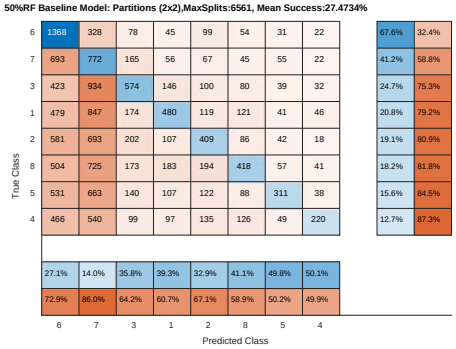
(g) *2x2 Baseline* model – Run 4



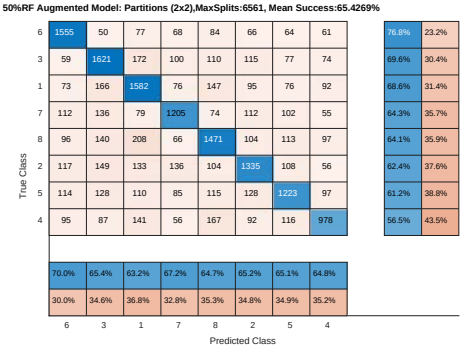
(h) *2x2 Augmented* model – Run 4



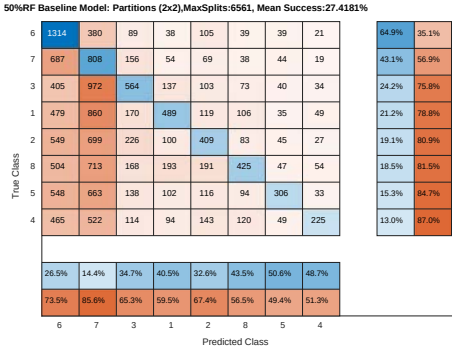
(i) *2x2 Baseline* model – Run 5



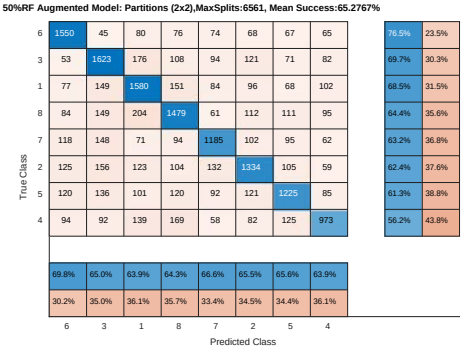
(j) *2x2 Augmented* model – Run 5



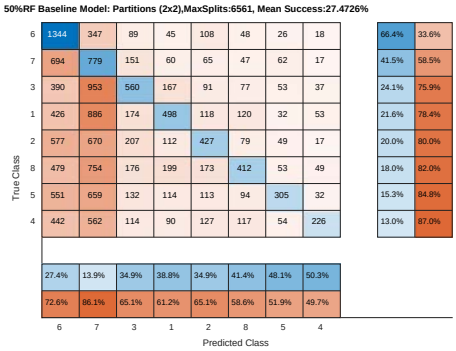
(k) *2x2 Baseline* model – Run 6



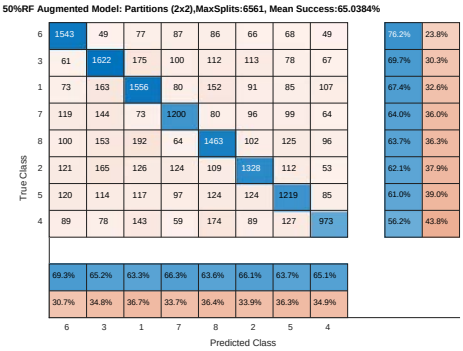
(l) *2x2 Augmented* model – Run 6



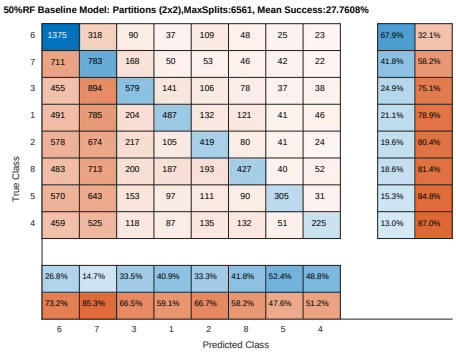
(m) *2x2 Baseline* model – Run 7



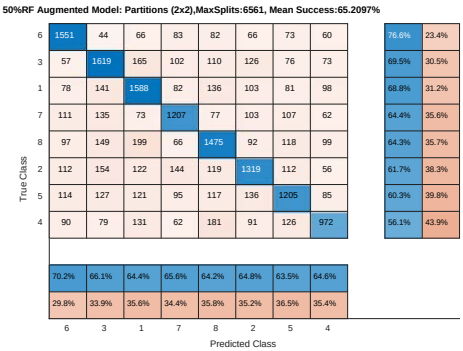
(n) *2x2 Augmented* model – Run 7



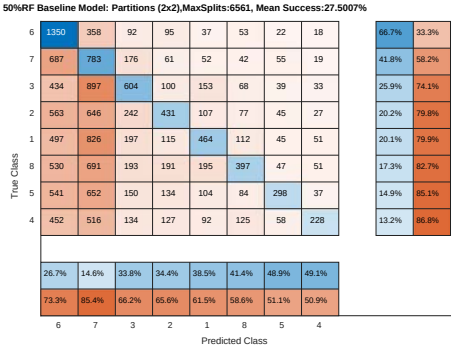
(o) *2x2 Baseline* model – Run 8



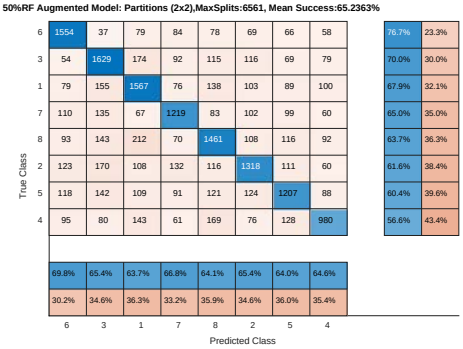
(p) *2x2 Augmented* model – Run 8



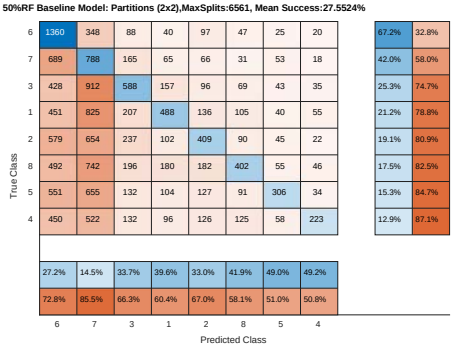
(q) *2x2 Baseline* model – Run 9



(r) *2x2 Augmented* model – Run 9



(s) *2x2 Baseline* model – Run 10



(t) *2x2 Augmented* model – Run 10

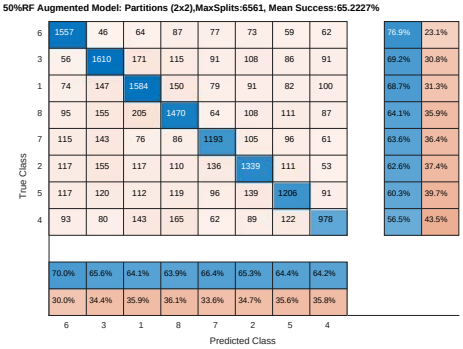
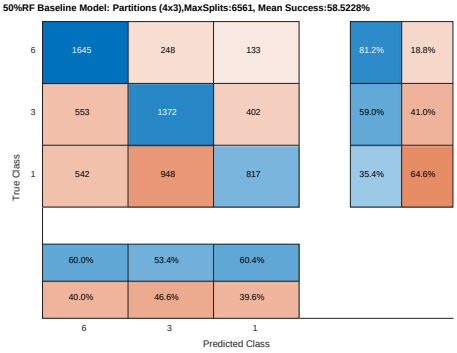
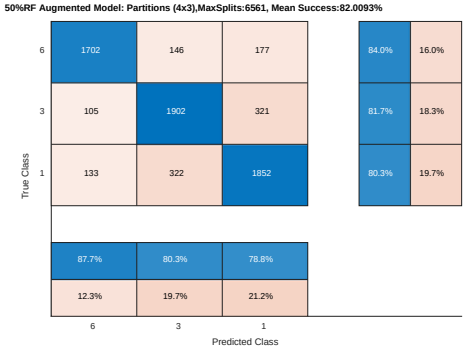


Figure 24. *2x2* configuration results (10 runs)

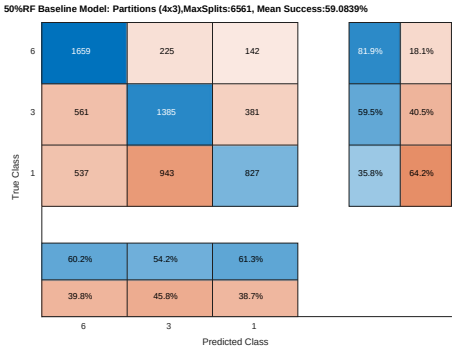
(a) *4x3 Baseline* model – Run 1



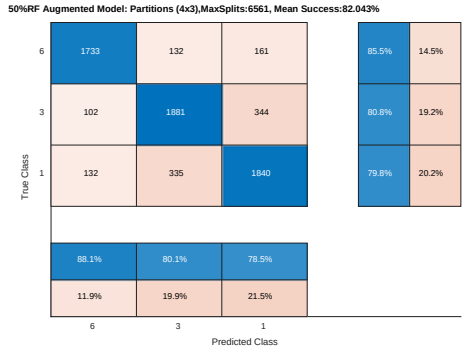
(b) *4x3 Augmented* model – Run 1



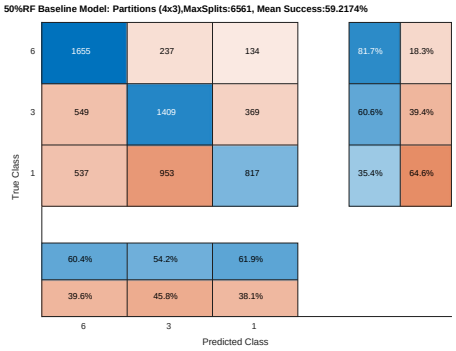
(c) *4x3 Baseline* model – Run 2



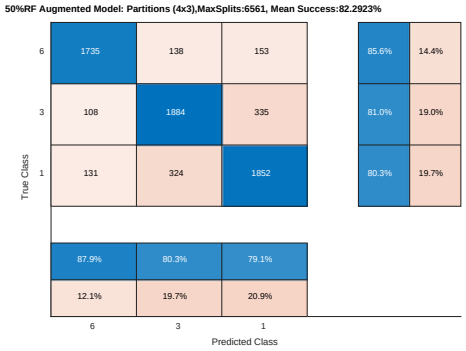
(d) *4x3 Augmented* model – Run 2



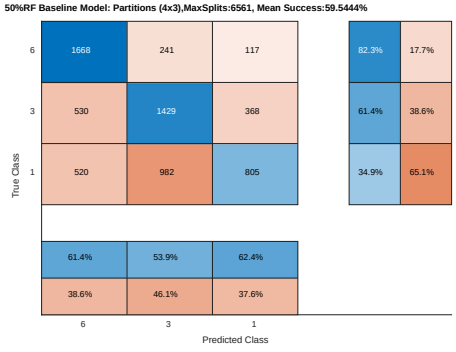
(e) *4x3 Baseline* model – Run 3



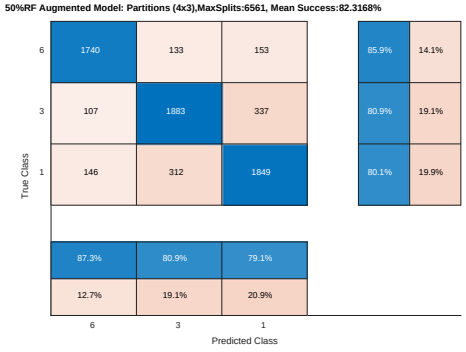
(f) *4x3 Augmented* model – Run 3



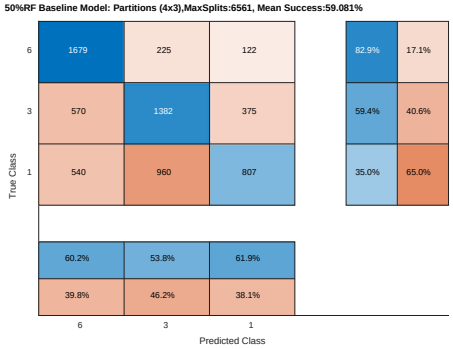
(g) *4x3 Baseline* model – Run 4



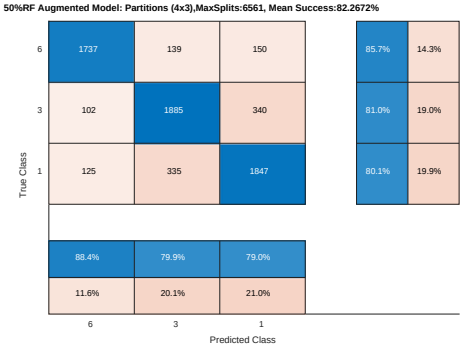
(h) *4x3 Augmented* model – Run 4



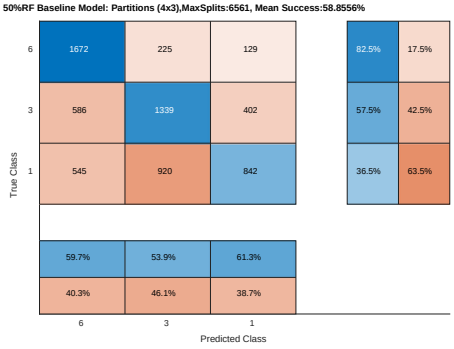
(i) *4x3 Baseline* model – Run 5



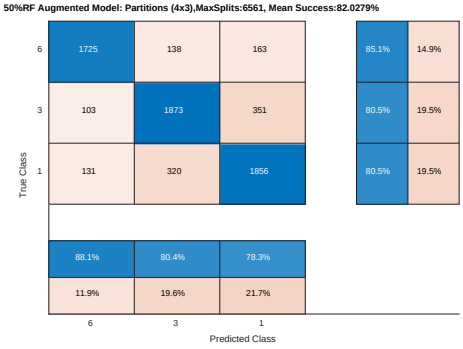
(j) *4x3 Augmented* model – Run 5



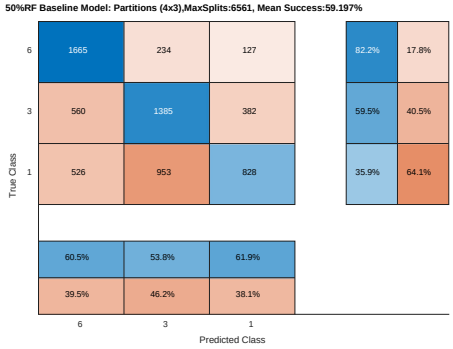
(k) *4x3 Baseline* model – Run 6



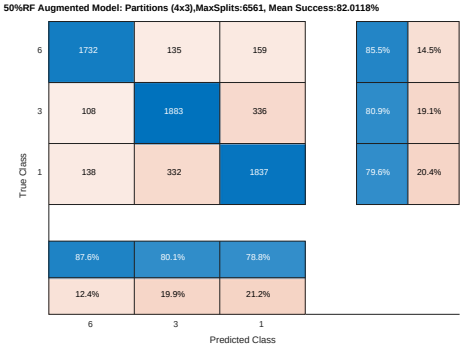
(l) *4x3 Augmented* model – Run 6



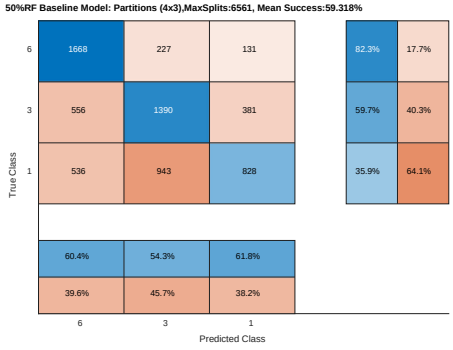
(m) *4x3 Baseline* model – Run 7



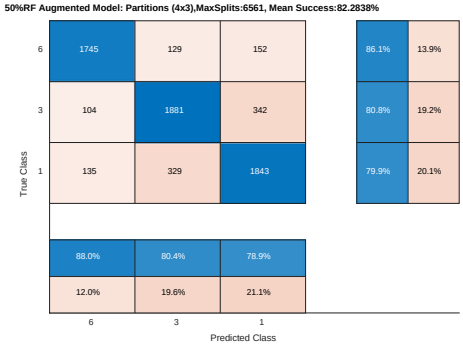
(n) *4x3 Augmented* model – Run 7



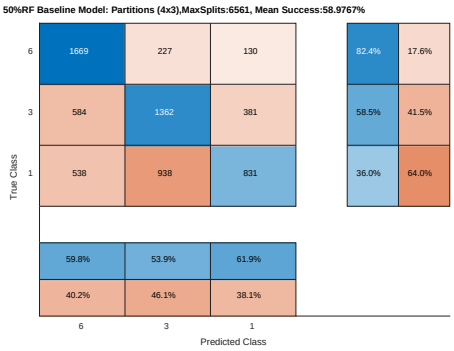
(o) *4x3 Baseline* model – Run 8



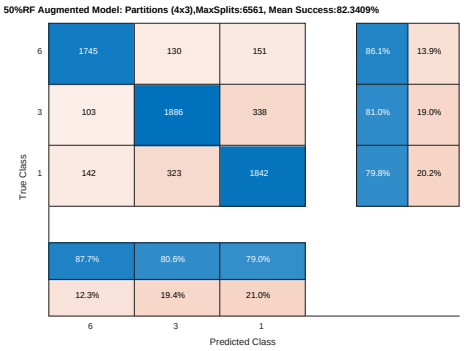
(p) *4x3 Augmented* model – Run 8



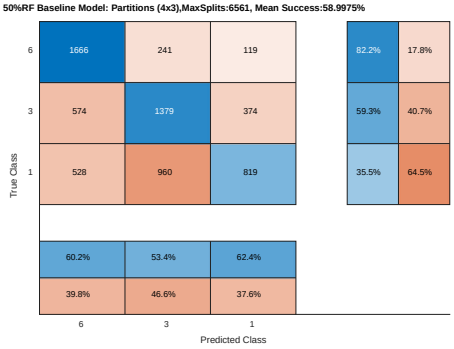
(q) *4x3 Baseline* model – Run 9



(r) *4x3 Augmented* model – Run 9



(s) *4x3 Baseline* model – Run 10



(t) *4x3 Augmented* model – Run 10

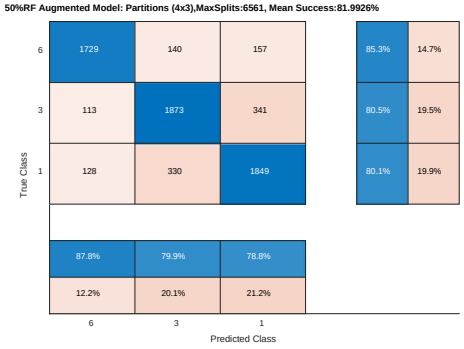


Figure 25. *4x3* configuration results for classes 1,3,6 only (10 runs)

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Committee on Autonomous Vehicles in Support of Naval Operations, National Research Council, Division on Engineering and Physical Sciences, Naval Studies Board, and National Academy of Sciences, *Autonomous Vehicles in Support of Naval Operations*. Washington, DC, United States: National Academies Press, 2005.

[2] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017, doi: 10.1109/MITS.2016.2583491

[3] N. Kalra, "Challenges and approaches to realizing autonomous vehicle safety," 2017. https://www.rand.org/pubs/testimonies/CT463.html (accessed Aug. 06, 2020)

[4] "Visibility in Singapore drops to as low as 3km on Monday amid hazy conditions: NEA," *CNA*. https://www.channelnewsasia.com/news/singapore/haze-singapore-visibility-reduced-air-quailty-monday-nea-11934886 (accessed Aug. 06, 2020)

[5] hermes, "Singapore tops list of leading maritime capitals for fourth time," *The Straits Times*, Apr. 11, 2019. https://www.straitstimes.com/singapore/transport/singapore-tops-list-of-leading-maritime-capitals-for-fourth-time (accessed Aug. 07, 2020)

[6] F. Guo, J. Tang, and X. Xiao, "Foggy scene rendering based on transmission map estimation," *International Journal of Computer Games Technology*, vol. 2014, no. 2014, p. 13, 2014, doi: 10.1155/2014/308629

[7] E. Ullah, R. Nawaz, and J. Iqbal, "Single image haze removal using improved dark channel prior," 2013, pp. 245–248.

[8] R. T. Tan, "Visibility in bad weather from a single image," 2008, pp. 1–8, doi: 10.1109/CVPR.2008.4587643

[9] Y. Li, S. You, M. S. Brown, and R. T. Tan, "Haze visibility enhancement: A Survey and quantitative benchmarking," *Computer Vision and Image Understanding*, vol. 165, no. C, pp. 1–16, 2017, doi: 10.1016/j.cviu.2017.09.003

[10] Kaiming He, Jian Sun, and Xiaoou Tang, "Single image haze removal using dark channel prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 12, pp. 2341–2353, 2011, doi: 10.1109/TPAMI.2010.168

[11] S. Narasimhan and S. Nayar, “Vision and the atmosphere,” *International Journal of Computer Vision*, vol. 48, no. 3, pp. 233–254, 2002, doi: 10.1023/A:1016328200723

[12] S. G. Narasimhan and S. K. Nayar, “Contrast restoration of weather degraded images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 713–724, 2003, doi: 10.1109/TPAMI.2003.1201821

[13] “Table 1 : International visibility grades with their medium extinction...,” *ResearchGate*. https://www.researchgate.net/figure/International-visibility-grades-with-their-medium-extinction-coefficients_tbl1_287395953 (accessed Aug. 22, 2020)

[14] C. O. Ancuti, C. Ancuti, and R. Timofte, “NH-HAZE: An image dehazing benchmark with non-homogeneous hazy and haze-free images,” 20200507, Accessed: Aug. 07, 2020. [Online]. Available: http://arxiv.org/abs/2005.03560

[15] F. Guo, J. Tang, and H. Peng, “A Markov random field model for the restoration of foggy images,” *International Journal of Advanced Robotic Systems*, vol. 11, no. 6, 2014, doi: 10.5772/58674

[16] “Noise fractals and clouds « null program.” https://nullprogram.com/blog/2007/11/20/ (accessed Jul. 25, 2020)

[17] P. Kohli, M. P. Kumar, and P. H. S. Torr, “P3 & beyond: Solving energies with higher order cliques,” 2007, pp. 1–8, doi: 10.1109/CVPR.2007.383204

[18] P. Carr and R. Hartley, “Improved single image dehazing using geometry,” 2009, pp. 103–110, doi: 10.1109/DICTA.2009.25

[19] “Computer vision at Waterloo - Code.” Accessed: Jul. 23, 2020. [Online]. Available: https://vision.cs.uwaterloo.ca/code/

[20] “Hazy start to the weekend in Singapore after air quality nears unhealthy levels,” *AsiaOne*, Sep. 14, 2019. https://www.asiaone.com/singapore/hazy-start-weekend-singapore-after-air-quality-nears-unhealthy-levels (accessed Aug. 15, 2020)

[21] “Home - ShipSpotting.com - Ship Photos and Ship Tracker.” http://www.shipspotting.com/ (accessed Jul. 27, 2020)

[22] “Flip order of elements - MATLAB flip.” https://www.mathworks.com/help/matlab/ref/flip.html (accessed Jul. 27, 2020)

[23] “Find edges in intensity image - MATLAB edge.” https://www.mathworks.com/help/images/ref/edge.html (accessed Jul. 27, 2020)

[24] C. Tomasi and T. Kanade, “Detection and tracking of point features,” *International Journal of Computer Vision*, 1991.

[25] M. Mahesh and M. Subramanyam, “Feature based image mosaic using steerable filters and Harris corner detector,” *International Journal of Image, Graphics and Signal Processing*, vol. 5, no. 6, pp. 9–15, 2013, doi: 10.5815/ijigsp.2013.06.02

[26] J. K. Suhr, “Kanade-Lucas-Tomasi (KLT) feature tracker,” *Computer Vision*, p. 36, 2009.

[27] “Edge detection in images: how to derive the Sobel operator – Najam R Syed.” https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/ (accessed Aug. 11, 2020)

[28] I. Sobel, “An isotropic 3x3 image gradient operator,” *Presentation at Stanford A.I. Project 1968*, Feb. 2014.

[29] B. H. Taher, M. A. H. Hasab, and E. W. Abood, “Corner detection using gradient and topological properties of digital images,” vol. 6, no. 5, p. 6, 2016.

[30] R. Misra and K. S. Ray, “Dual approach for object tracking based on optical flow and swarm intelligence,” 20180815, Accessed: Aug. 11, 2020. [Online]. Available: http://arxiv.org/abs/1808.08186

[31] J.-Y. Bouguet, “Pyramidal Implementation of the Lucas Kanade feature tracker description of the algorithm,” p. 9.

[32] “(PDF) An automatic optimum kernel-size selection technique for edge enhancement,” *ResearchGate*. https://www.researchgate.net/publication/223377841_An_automatic_optimum_kernel-size_selection_technique_for_edge_enhancement (accessed Aug. 15, 2020)

[33] “What is the advantage of a 5x5 gradient mask edge detector over a 3x3 detector?,” *ResearchGate*. https://www.researchgate.net/post/What_is_the_advantage_of_a_5x5_gradient_mask_edge_detector_over_a_3x3_detector (accessed Aug. 15, 2020)

[34] “Average or mean value of array - MATLAB mean.” https://www.mathworks.com/help/matlab/ref/mean.html (accessed Aug. 11, 2020)

[35] “Standard deviation - MATLAB std.” https://www.mathworks.com/help/matlab/ref/std.html?s_tid=srchtitle (accessed Aug. 11, 2020)

[36] “Skewness - MATLAB skewness.” https://www.mathworks.com/help/stats/skewness.html?s_tid=srchtitle (accessed Aug. 11, 2020)

[37] “Kurtosis - MATLAB kurtosis.” https://www.mathworks.com/help/stats/kurtosis.html?s_tid=srchtitle (accessed Aug. 11, 2020)

[38] G. Louppe, *Understanding Random Forests: From Theory to Practice*, 2014.

[39] C. Strobl, J. Malley, and G. Tutz, “An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests,” *Psychological Methods*, vol. 14, no. 4, pp. 323–348, 2009, doi: 10.1037/a0016973

[40] “Fit ensemble of learners for classification - MATLAB fitcensemble.” https://www.mathworks.com/help/stats/fitcensemble.html (accessed Aug. 18, 2020)

[41] “How to determine the number of trees to be generated in Random Forest algorithm?,” *ResearchGate*. https://www.researchgate.net/post/How_to_determine_the_number_of_trees_to_be_generated_in_Random_Forest_algorithm (accessed Aug. 18, 2020)

[42] G. Biau and E. Scornet, “A random forest guided tour,” 2015, Accessed: Aug. 20, 2020. [Online]. Available: https://arxiv.org/abs/1511.05741

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California