

Image Based Real-Time Ice Load Prediction Tool for Ship and Offshore Platform in Managed Ice Field

By

Shamima Akter

A dissertation Submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, Newfoundland, Canada

May 2023

Abstract

The increased activities in arctic water warrant modelling of ice properties and ice-structure interaction forces to ensure safe operations of ships and offshore platforms. Several established analytical and numerical ice force estimation models can be found in the literature. Recently, researchers have been working on Machine Learning (ML) based, data-driven force predictors trained on experimental data and field measurement. Application of both traditional and ML-based image processing for extracting information from ice floe images has also been reported in recent literature; because extraction of ice features from real-time videos and images can significantly improve ice force prediction.

However, there exists room for improvement in those studies. For example, accurate extraction of ice floe information is still challenging because of their complex and varied shapes, colour similarities and reflection of light on them. Besides, real ice floes are often found in groups with overlapped and/or connected boundaries, making detecting even more challenging due to weaker edges in such situations. The development of an efficient coupled model, which will extract information from the ice floe images and train a force predictor based on the extracted dataset, is still an open problem.

This research presents two Hybrid force prediction models. Instead of using analytical or numerical approaches, the Hybrid models directly extract floe characteristics from the images and later train ML-based force predictors using those extracted floe parameters. The first model extracted ice features from images using traditional image processing techniques and then used SVM and FFNN to develop two separate force predictors. The improved ice image processing technique used here can extract useful ice properties from a closely connected, unevenly

illuminated floe field with various floe sizes and shapes. The second model extracted ice features from images using RCNN and then trained two separate force predictors using SVM and FFNN, similar to the first model.

The dataset for training SVM and FFNN force predictors involved variables extracted from the image (floe number, density, sizes, etc.) and variables taken from the experimental analysis results (ship speed, floe thickness, force etc.). The performance of both Hybrid models in terms of image segmentation and force prediction, are analyzed and compared to establish their validity and applicability.

Nevertheless, there exists room for further development of the proposed Hybrid models. For example, extend the current models to include more data and investigate other machine learning and deep learning-based network architectures to predict the ice force directly from the image as an input.

Acknowledgements

I am starting by the name of God for giving me the strength and patience to write this thesis. I express my immense gratitude to Him for enabling me to complete my thesis work successfully. I want to extend my profound and sincere thanks to my supervisor, Dr. Syed Imtiaz and Dr. Salim Ahmed, for their generous help, support, valuable guidance, advice, and careful supervision throughout the research and the successful completion of this thesis. They encouraged and supported me with appropriate directions, spending hours with me over various stages of the research work.

My sincere thanks to Dr. Mohammed Shameem Islam and Dr. Hasanat Zaman and for their valuable suggestions throughout the research period and enormous support in getting the experimental data from National Research Council (NRC), Canada. I gratefully acknowledge and thank the National Research Council of Canada for the financial support. My project was funded under the National Research Council of Canada Collaborative Research and Development program (CSTIP Grant # OCN-205-1).

I would like to thank my research colleagues, team members and friends at Memorial University who helped me through the journey with their knowledge and expertise. I would also appreciate the efforts of those who have helped me directly or indirectly during the different stages of this work. I would like to thank the staff of the Faculty of Engineering and Applied Science, Colleen Mahoney and Tina Dwyer, for their endless support throughout my Master's degree. I would also thank the Engineering Computing Service, Faculty of Engineering and Applied Science, for providing enormous assistance during my academic program at Memorial University.

Finally, I would like to convey my gratitude to my friends and family in St. John's and Bangladesh. I express my greatest appreciation to my mother, Rehana Soharab and dedicate this thesis to my father, Soharab Hossain Jamadder. I also thank my siblings, my daughter and my son for their encouragement and support in my life.

Co-authorship Statement

I, Shamima Akter, hold the primary author status for all the Chapters in this thesis. However, each manuscript is co-authored by my supervisors, co-supervisors, and external research collaborators. The contributions of each of the co-authors are listed below:

1. Shamima Akter, Mohammed Islam, Syed Imtiaz, Salim Ahmed, Hasanat Zaman, Robert Gash.
Image processing to extract ice features to aid modelling of ice force. Proceedings of the ASME 2022, 41st International conference on Ocean, Offshore and Arctic Engineering (OMAEE), 2022.

Statement: The research was conducted by Shamima Akter as the first author. She prepared the manuscript. Other authors supervised the student, reviewed the manuscript, and provided feedback.

2. Shamima Akter, Mohammed Islam, Syed Imtiaz, Salim Ahmed, Hasanat Zaman, Robert Gash.
Image Based Sea Ice-Field Characterization for Predicting Ice-Structure Interactions. Cold Regions Science and Technology (Submitted).

Statement: The research was conducted by Shamima Akter as the first author. She prepared the manuscript based on the research outcome. Mohammed Islam provided the experimental data. He and other authors supervised the student, reviewed the manuscript, and provided feedback.

3. Shamima Akter, Mohammed Islam, Syed Imtiaz, Salim Ahmed, Hasanat Zaman, Robert Gash.
Image based Ice load prediction on ship in a managed ice field using Machine learning. (Under preparation).

Statement: The research was conducted by Shamima Akter as the first author. She is preparing the manuscript. Mohammed Islam provided the experimental data. He and other authors supervised the student, reviewed the manuscript, and provided feedback.

Table of Contents

Abstract.....	ii
Acknowledgements	iv
Co-authorship Statement	v
List of Tables	viii
List of Figures.....	ix
1.0 Introduction.....	1
1.1 Background and motivation	1
1.2 Aim and Objectives.....	4
1.3 Research Tools and Expected Outcomes	5
1.4 State of the Thesis	7
2.0 Literature Review	10
2.1 Image processing for feature extraction	11
2.1.1 Unsupervised methods.....	13
2.1.2 Supervised methods	15
2.2 Image based ice force prediction.....	19
2.3 Summary	22
3.0 Image Based Sea Ice-Field Characterization for Predicting Ice-Structure Interactions	23
3.1 Introduction	24
3.2 Methodology	28
3.2.1 Image enhancement	28
3.2.2 Image segmentation	34
3.2.3 Morphological filtering.....	37
3.2.4 Region analysis	39
3.3 Results and discussion.....	39
3.3.1 Processing simulated ice images.....	39
3.3.2 Analysis of real ice tank images	42

3.3.3 Computational efficiency and real-time applicability.....	48
3.4 Conclusion.....	50
3.5 Acknowledgements.....	51
4.0 Image based Ice load prediction on ship in a managed ice field using machine learning	52
4.1 Introduction.....	52
4.2 Methodology.....	56
4.2.1 Image extraction from the input videos.....	58
4.2.2 Image processing tools.....	60
4.2.3 Preparing data (images) for training, validation, and testing.....	65
4.2.4 ML-based ice load prediction model.....	70
4.3 Results and discussion.....	74
4.3.1 Faster RCNN training for Ship and ice floe detection.....	74
4.3.2 Performance of the SVM and FFNN force predictors.....	78
4.4 Conclusion.....	88
4.5 Acknowledgements.....	89
5.0 Conclusions and Recommendations.....	90
5.1 Conclusions.....	90
5.2 Recommendations.....	92
References.....	94
Appendix A - Published Conference Paper in OMAE 2022.....	102
Appendix B - Source Code.....	110
B.1 Ice floe parameter extraction using image preprocessing and GVF SNAKE (Chapter 03)	110
B.2 Frame extraction from videos (Chapter 04).....	127
B.3 Faster RCNN code for ship and ice floe detection (Chapter 04).....	128
B.4 SVM and FFNN for force prediction (Chapter 04).....	136

List of Tables

Table 1-1: Research tools and expected outcomes.	6
Table 2-1: Summary of recent works on image segmentations.....	18
Table 3-1: Floe detection comparison among various models for simulated images.....	42
Table 3-2: Model parameters for ice floe image analysis.....	43
Table 3-3: Floe count and ice coverage comparison between Zhang & Skjetne and proposed models.....	45
Table 4-1: Test matrix for the Magne Viking model test in the ice basin	59
Table 4-2: Irrational data point removal. (a) Beginning of the test situation, (b) End of the test situation.....	81
Table 4-3: Definition of test dataset scenarios.....	82
Table 4-4: Variable definitions and sample values for force predictor input.	87

List of Figures

Figure. 1-1: Ship in a broken arctic ice field [1].	2
Figure 1-2: State of the thesis	8
Figure 3-1: Ship in complex, managed ice field [79]	25
Figure 3-2: Flowchart for the proposed ice image processing technique.	29
Figure 3-3: Steps of image enhancement: (a) original image, (b) default grayscale conversion, (c) grayscale conversion after normalizing the illumination, (d) colour inversion of ‘c,’ (e) histogram equalization, (f) improved grayscale after wavelet denoising.	31
Figure 3-4: Decomposition and reconstruction functions for bio-orthogonal 4.4 wavelet.	33
Figure 3-5: Image processed using Gaussian blurring (left) and wavelet (right).	35
Figure 3-6: Binary image I (a), morphological closing of I (b), morphological opening of I (c).	38
Figure 3-7: Binary image J and the structuring element (a), overlapping of structuring element on J to detect noises (b), removal of noises from J (c).	38
Figure 3-8: Simple (left) and complex (right) simulated ice floe images.	40
Figure 3-9: Simple, simulated ice floe image segmentation: (left) Zhang & Skjetne (2018), (middle) Marker Watershed (Turker et al., 2021), (right) proposed model.	41
Figure 3-10: Complex, simulated ice floe image segmentation: (left) Zhang & Skjetne (2018), (middle) Marker Watershed (Turker et al., 2021), (right) proposed model.	41
Figure 3-11: Ice floe mages extracted from Model Basin experimental test [62].	42
Figure 3-12: Segmentation results for ice floe images from model basin test – comparison among Zhang & Skjetne (2018), Marker Watershed (Turker et al., 2021) and proposed model.	44
Figure 3-13: Comparison of ice coverage prediction.	46
Figure 3-14: Processing time comparison among various models.	46

Figure 3-15: Histograms of floe count [no of floes (y) vs flow size in pixels (x). 1 mm ² =14.2 pixels]: (left column) outputs from the manually marked image, (middle column) outputs from Zhang & Skjetne (2018), (right column) outputs from the proposed method.	47
Figure 3-16: Evaluation of model performance after improving the computational efficiency by removing small floes below significant sizes (noises).....	49
Figure 3-17: Processing time comparison among various models after small floes removal.	50
Figure 4-1: Flow chart for the Hybrid ice force prediction models development	57
Figure 4-2: NRC-OCRE ice basin images [61]	58
Figure 4-3: Extracted frame from three different videos (25m, 50m and 100m floe size).....	60
Figure 4-4: Region based Convolutional Neural Network concept [86]	61
Figure 4-5: Faster RCNN object detection network	62
Figure 4-6: How the anchor box works in the faster RCNN object detector.....	65
Figure 4-7: Data preparation for ship detector training (a) Input table for the network, (b) Ship labelled in one training image.....	66
Figure 4-8: Data augmentation outcome.....	67
Figure 4-9: Cropped images for floe size (a) 25, (b) 50, (c) 100. Final input images after ship masking for floe size (d) 25, (e) 50, (f) 100.....	68
Figure 4-10: Ice floe pattern in the image: (a) ice floe labelling in small floe regions (b) labelling large floes.....	69
Figure 4-11: Example of a Feedforward Neural Network with two hidden layers	72
Figure 4-12: Training and validation summary (a), Test image with accuracy bounding box (b).	75
Figure 4-13: Precision-Recall curve (Average Precision 1.0) for ship detector training.....	76

Figure 4-14: Ice floe detection accuracy in larger ice floe images	77
Figure 4-15: Ice floe detection accuracy in smaller ice floe images (2X the size of images in Figure 4-14)	78
Figure 4-16: Initial input data for force predictor training	79
Figure 4-17: Reducing ice floe categories for better results (Scenario 1 test set)	80
Figure 4-18: Addition of ‘collision’ variable (Scenario 2 test set)	80
Figure 4-19: SVM force prediction performance for various input scenarios (Hybrid model 1)	83
Figure 4-20: SVM force prediction performance for various input scenarios, outliers removed (Hybrid model 1).....	84
Figure 4-21: FFNN force prediction performance for various input scenarios (Hybrid model 1)	85
Figure 4-22: SVM and FFNN force prediction performance comparison for Hybrid model 02.	87

1.0 Introduction

1.1 Background and motivation

The gradual melting of ice and exposure of the land in the Arctic region has increased the possibility of extensive commercial and scientific activities in that area. This subsequently boosted relevant research and development activities, particularly with respect to structural safety, including navigating technology in ice-infested areas to ensure structures are capable of withstanding extreme cold and ice forces. However, building proper technologies and operating them in the Arctic is more technically and physically challenging than any other water way due to low temperatures, remoteness, darkness, and the prevalence of ice. One of the major threats to the moving vessel and offshore installations is the interaction with multi-directional drifting sea ice with a wide variety of types and forms, ranging from isolated first-year floes to compacted multi-year ridges. The ice-ship interaction process is dynamic and depends on many complex and interconnected parameters and characteristics related to the icefield, the ship, and the surrounding environment. Therefore, developing analytical, numerical, data-driven and physical models to understand and analyze ice floe characteristics and their subsequent force on floating structures is essential in ensuring safe and reliable activities in the Arctic.

With the advancement of microwave satellite sensors, ice concentration data on a global scale has become available on a daily basis. The variability of sea ice extent on a global level can now be monitored easily due to this development. Various types of remote sensing technologies and corresponding image processing algorithms for analyzing sea-ice statistics and ice properties have been developed over the years. Satellite remote sensing has been widely used to extract ice

concentration [1], classify ice types [2], and analyze ice floes [3]. Digital visual image techniques are also applied to ice observation [4]–[6]. However, these technologies have various limitations in terms of applicability, area of operation, performance, and reliability concerns for real-time implementation [7]. It is still challenging to predict the sea ice behaviour at ship/ offshore structure level due to the lack of knowledge about the sub-grid scale information and the associated complexities of ice floe interactions at that level. To solve this, attention to understanding and extracting ice characteristics at ice floe levels has recently been increasing. Ships sailing or operating in polar regions most frequently travel/stay in Marginal Ice Zone (MIZ) or ice fields managed by an ice breaker (Figure 1-1). Understanding, analyzing and extracting ice floe and ice force characteristics at this sub-grid level are vital to ensure safe offshore operation.



Figure. 1-1: Ship in a broken arctic ice field [1].

Over the last few decades, understanding and modelling of ice forces on fixed, moored floating, dynamic positioning (DP) controlled platform/vessels; and on icebreaking or slowly maneuvering ships in managed or unmanaged broken ice fields are attempted on multiple frontiers – ranging from analytical formulation and numerical modelling to experimental and hybrid modelling. The analytical, empirical, and statistical methods often show high numerical efficiency and ease of integration; however, they do not accurately model all relevant physical processes. Hence these methods are not applicable to real-life applications. Most of the computational techniques require high computation resources. They often take a long calculation time, which is unsuitable for real-time simulations. To compensate for the deficiencies of a single method, often multiple methods are combined to obtain the desired result. Therefore, there is a recent increasing trend in using hybrid methods to model ice-structure interactions, for example, FEM and SPH coupling [8], combined FEM-DEM simulations [9], and CFD-DEM coupling numerical method [10]. However, regardless of the methods adopted, validations with quality measurements are paramount to the success of the ice-structure interaction models; and the lack of high-quality physical model tests and full-scale measurements for a thorough validation adversely impacts confidence in the modelling [11].

Though these methods achieve varying degrees of success in predicting ice forces for a specific ice condition and vessel type, almost in all cases, these techniques lack versatility and cannot be generalized. To address the above issue, many researchers have developed ML and deep learning techniques for ice force predictions in ship-ice interaction. However, all these Machine Learning based models involved the use of directly measured numerical data extracted from fields, model tests or numerical analyses, and no real-time processing of ice images is performed to extract those datasets.

Using real imagery and applying digital image processing techniques is one of the best ways to extract real-time ice characteristics in the oceans. With the advancement of computer visions, several advanced algorithms, for example, Convolutional Neural Networks (CNN), Regions with Convolutional Neural Networks (R-CNN), You Only Look Once (YOLO) are extensively applied in various fields for image analysis, the application of these advanced tools for ice image processing is still at its infancy. Some recent works involve Mask Region-based CNN architecture for detecting icebergs [12], CNN-based black ice detection platform to prevent accidents [13], Python-based open-source algorithm for detecting sea ice surface features using high-resolution optical imagery [14], SVM-based method to detect pancake ice and compute their size distribution [15]. However, none of these works used the extracted ice floe information to estimate ice forces on structures.

The motivation of this research work is to address these gaps in research and develop a tool that will extract the ice floe characteristics from experimental/field videos/images and use the information for predicting ice forces on the ship in real-time.

1.2 Aim and Objectives

The aim of this thesis is to develop a model for ice force estimation on moving ship/offshore structures in a managed ice field using supervised and unsupervised image processing and data analysis tools.

The objectives of this research work are as follows:

- ❖ Develop comprehensive image processing tools using both the traditional and ML-based image processing approaches that can separate unevenly illuminated and highly irregular ice floes with close connections and overlaps.
- ❖ Compare and validate the developed tools against existing methods in the literature.
- ❖ Extracting ice floe properties (i.e., segmenting ice regions from water, detecting floe boundaries, estimating ice concentration and floe sizes) from video images using the developed tools.
- ❖ Use the extracted features from the image processing tools to predict ice force on moving ships in real-time.

1.3 Research Tools and Expected Outcomes

The core research objectives, investigation tools, and expected outcomes are listed in Table 1-1.

These tools can be grouped into two categories:

Image processing Related: Tools to segment the images to extract icefield and vessel information to be used as input to ML modelling for ship-ice interaction loads. Both traditional image segmenting approaches and ML-based tools are used in this regard. Ice-field information may include concentration, floe size distribution, drift speed, etc. The vessel information may include vessel position, orientation, speed, etc. Validate the extracted data with manually collected and measured information. Comparison with other existing methods and justification of new developments are shown in the following Chapters. A combination of several modified MATLAB-based image processing tools ([16], [17]) is used for this section.

Table 1-1: Research tools and expected outcomes.

Core objectives	Features	Descriptions
Develop a comprehensive image processing tool using the traditional image processing approach.	Tool Used	MATLAB-based advanced GVF-SNAKE algorithm, including wavelet transform and morphological filtering.
	Expected Outcomes	Can separate unevenly illuminated and highly irregular ice floes with close connections and overlaps from both simulated and real ice images.
Comparisons and validations of the developed tool using simulated and experimental ice field images.	Tool Used	MATLAB-based Marker Watershed and typical GVF-Snake method
	Expected Outcomes	Improved computational efficiency based on the floe counting, ice concentration and time taken to run the algorithm.
Develop and validate another image processing tools using Machine Learning based image processing approach.	Tool Used	Region-based Convolutional Neural Network
	Expected Outcomes	A faster Machine Learning based ice floe image segmentor with reasonable accuracy.
Develop, train, validate and test two Machine Learning based force predictors using Support Vector Machine and Feed Forward Neural Network approaches.	Tool Used	MATLAB-based SVM and FFNN platforms
	Expected Outcomes	Realtime ice force prediction tools with reasonable accuracy.

Force Estimation Related: ML-based tools for ice force prediction using the extracted ice-field and vessel information from the previous step. Predicting the forces for certain ice conditions and validating with the corresponding measurements. The support vector machine and feed-forward neural network are used in the MATLAB environment to develop these tools [18].

1.4 State of the Thesis

The thesis is written in manuscript format. One conference manuscript has been published. One journal paper (an extended version of the conference paper) is at the review stage, and another in the preparation stage is included in the thesis. A co-authorship statement is provided at the beginning of the thesis. The overall state of the thesis is shown in Figure 1-2, and a brief overview of each chapter is described below:

Chapter 1 of the thesis describes the motivations and objectives of the research. This chapter also includes a brief review of the research problem addressed in this thesis.

Chapter 2 of the thesis represents the elaborative literature review of the research. This chapter also describes an extensive idea of previous works related to this thesis.

Chapter 3 presents image-based ice field parameter extraction for predicting ship-ice interactions. A detailed analysis of the performance of the proposed improved ice processing tool against other available tools is also described.

Chapter 4 describes another Machine Learning based image feature extraction tool. The chapter also includes the steps of developing, training, validating and testing two ice load prediction tools. These force predictors are trained using SVM and FFNN, respectively.

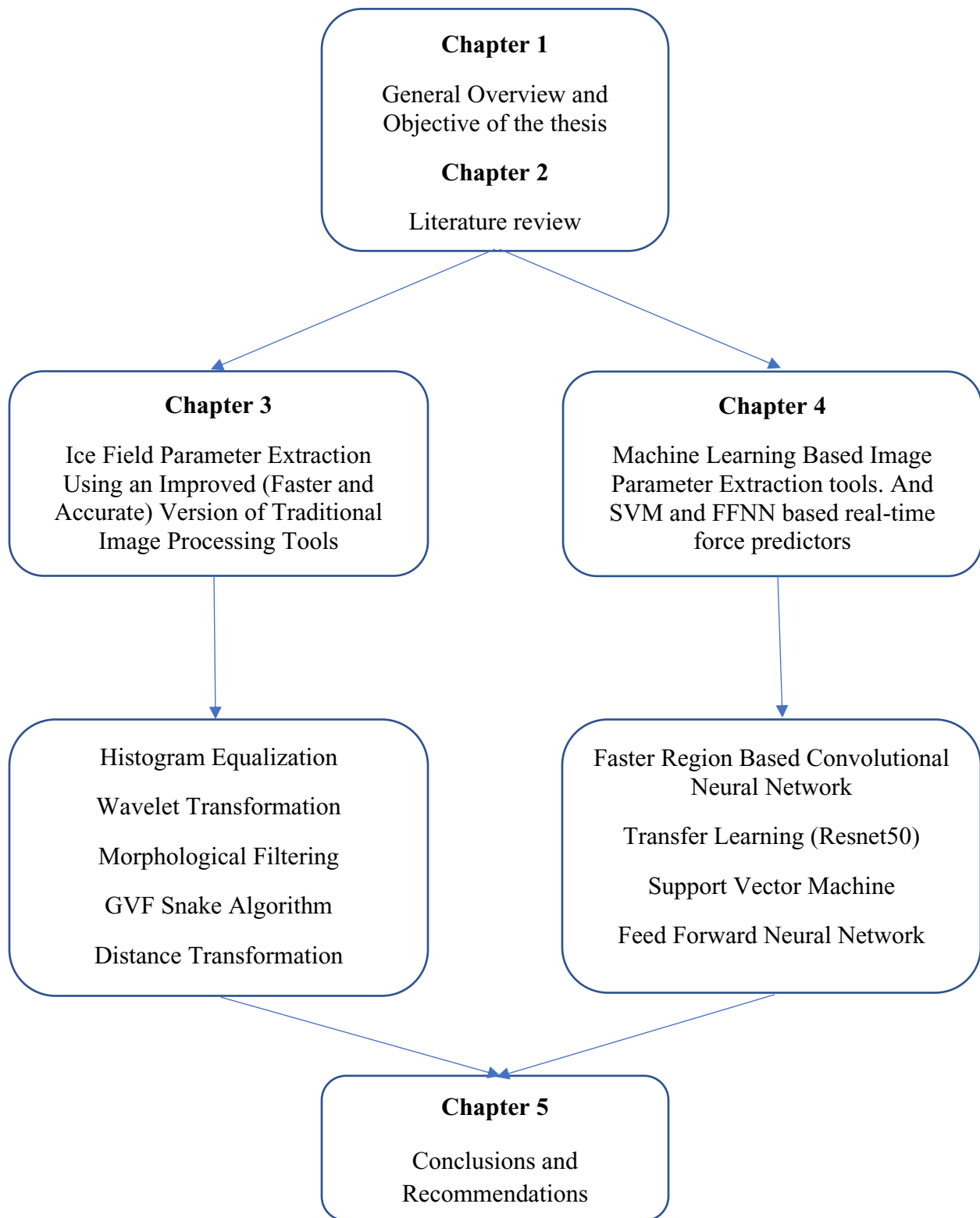


Figure 1-2: State of the thesis

Chapter 5 summarizes the key features of the thesis. The chapter also provides a few recommendations for future research.

Appendix A includes the conference paper which is presented in OMAE 2022.

Appendix B contains all the source codes modified, developed and used for this study.

2.0 Literature Review

Advancement in image processing through the development and application of novel theories and algorithms has been providing researchers with sophisticated tools to extract and process various information from an image or a set of sequential images. Such capabilities of image processing open the doors for a wide range of uses, for example, in visualization and recognition (detect objects of interest in an image), pattern recognition (for estimating various parameters from a sequential set of images), retrieval (search and locate similar images from a large database of digital images), object tracking, object segmentation, motion prediction, object reconstructions. Specific applications include traffic monitoring, autonomous driving, medical imaging, remote sensing, face detection, feature extraction, forecasting, augmented reality, security and monitoring, and many other fields [19]. However, unlike other engineering fields, the nature of solutions applied to process images largely varies depending on the area of applications. The information extracted from images is further used for decision-making, prediction or similar other applications. Advanced image analysis techniques in the medical field have enabled more efficient and accurate treatment plans. For example, in medical imaging, it might be more important to identify a particular object (e.g., a tumour) from an image. Video image processing systems for traffic sensing focus on detecting other vehicles, traffic signs etc., thus, bringing tremendous breakthrough in autonomous and safe manual driving, as well as better traffic management. For security and monitoring, on the other hand, one of the objectives is reconstructing and enhancing an otherwise blurry image (i.e., facial impression and fingerprint) to improve the quality of data extraction. Thus, aiding the surveillance and biometric authentication frontier. Image processing

in augmented reality and virtual reality fields focus on mimicking real-life environment to improve gaming and other virtual platform experiences.

This thesis focuses on sea ice image processing, where the objective is to identify and distinguish ice pieces of a complicated multi-domain ice field where floe sizes, concentrations, and shapes vary significantly (Figure 1-1). The extracted ice floe information can subsequently be used to predict ice forces on floating or fixed structures in the ice field. Therefore, in the context of the current research work, we performed a review of the literature in two areas: image processing for feature extraction and image-based ice force prediction.

2.1 Image processing for feature extraction

Extracting information from images through segmentation and other processing means is continually evolving, especially over the last few decades. There has been increasing interest in image recognition, image morphology, image data recognition and prediction through neural networks and knowledge-based image analysis. Processing an image means applying various operations to enhance its visual appearance or extract quantitative data for further technical applications and processing. In order to do so, the first task is digitizing the image to read it using the computer.

At its core, an image is a combination of many pixels. Pixels are the smallest controllable and representable building block of an image. An image can be disintegrated into a two-dimensional matrix of pixels. The number of pixels in the matrix depends on the resolution of the image; a higher resolution means the image contains a higher number of pixels. Each pixel in an image takes on a specific shade, opacity, or colour. When the pixels are modelled from the data extraction point of view, they are usually represented in one of the following forms:

Grayscale - A pixel is an integer with a value between 0 to 255 (0 is completely black, and 255 is completely white).

RGB - A pixel is made up of 3 integers between 0 to 255 (the integers represent the intensity of red, green, and blue).

RGBA - It is an extension of RGB with an added alpha field, which represents the opacity of the image.

Apart from acquiring and transforming an image into its digital form (the matrix form with pixel values), the other steps of image processing involve performing certain operations, depending on the field of applications, to enhance the image or extract useful information from it. Typical operations include:

Image enhancement – Manipulating the acquired image to meet the needs of the particular task for which this image processing is undertaken. The manipulations primarily aim to emphasize the hidden or important features in an image, for example, brightness and contrast adjustment. Therefore, this step is quite critical as it is highly subjective in nature.

Image restoration, colour, and filtering – Restoring missing features by applying certain probabilistic and mathematical models. Colour adjustment of RGB images for better feature extraction. Adjust the resolution and remove noise or blur from images using filtering techniques such as wavelet and Gaussian blurring.

Morphological processing and segmentation – Morphing the image (micro level crop, union, subtraction etc.) for better representation of the object shapes and then partitioning the image into

its constituent parts or objects. It is usually the most tricky and challenging step of image processing.

Representation, description, and recognition: Describing the segmented regions based on characteristics and regional properties, extracting quantitative information to differentiate one class of object from another and assigning labels to objects based on their description [20].

The literature on image processing to perform the above steps is vast and complex. A large number of algorithms and techniques are reported in the literature to complete each of these stages and achieve different results depending on the target to be achieved. The image processing and segmentation algorithms usually used for ice detection can be roughly divided into two categories: unsupervised methods and supervised methods.

2.1.1 Unsupervised methods

An unsupervised method for image segmentation is where the outcomes (groupings of pixels with common characteristics) are based on analyzing an image without the user providing sample classes. The applied algorithms determine which pixels are related and group them into classes. The user does not aid in the classification process, except by specifying the algorithm to be used and the number of expected output classes [21]. However, the user must have knowledge of the area being classified when the groupings of pixels with common characteristics produced by the computer have to be related to actual features on the ground (such as ice, water, etc.). Unsupervised methods are effective and widely used due to their simplicity and independency over training samples and labels. Over the last few decades, several unsupervised methods have been developed and applied for processing ice field images.

Some widely applied unsupervised methods are classical histogram thresholding, and clustering [22], [23], active contour model ([24], [25]) [26], graphcut [27], fuzzy entropy [28], watershed transform [29], wavelet transforms ([30],[31],[32]), gradient vector flow (GVF) and snake ([33], [7]), Markov random field, spectral clustering [7] and so on. The clustering algorithm itself can be further categorized into three sub-groups based on the inherent principles – decomposing density function [34], minimizing objective function [35] and graph theory [36].

Otsu thresholding and k-means clustering are two basic methods which are widely used for background and foreground separation of a grayscale image based on its histogram. Both methods work well for ice image segmentations when the floes are significantly brighter, well separated and have even illumination [37]. However, there is no room for pre- and post-processing of the images while using these techniques. Edge detection methods, for example, derivative and morphology edge detection techniques, usually perform better in this regard and can detect distinct boundaries with some non-uniform illumination. However, these methods fail to identify close boundaries and separate the ice floes that are tightly connected [38].

Watershed-based methods can be applied to separate connected floes with blurred edges in an ice field image. This technique has been successfully used in other applications, for example, cell nuclei image segmentations and grain separation [39], [40]. However, watershed-based techniques produce over- and under-segmentations when ice floes are unevenly illuminated, and floe sizes and shapes vary significantly [41]. An improved watershed method is proposed by [42], where a neighbouring region merging algorithm is added for better edge detection. Nevertheless, accurate identification of floe boundaries and floe numbers remains a challenge as watershed transformation operates on binary images, and a significant amount of real boundary information between connected floes can be lost while using this transformation. Recently, [15] and [43]

applied another modified version of the watershed method named the marker-controlled watershed (MCW) technique for pancake ice detection and agricultural field detection from satellite imagery. They incorporated non-linear support vector machine (SVM) analysis with traditional watershed to avoid over-segmentation. However, the problem of detecting irregular floes with uneven illumination and blur edging still remains.

Recently several researchers implemented a modified GVF method to tackle the issues of detecting complex shape floes [25], [4]. For example, the GVF snake algorithm, combined with distance transform-based automatic contour initialization, is adopted by [7] to separate seemingly connected floes. After that, ice floe shapes are enhanced using an ice shape enhancement algorithm, and individual ice floes are identified. This new technique is insensitive to proper initialization and can detect various complex floe shapes. However, connected ice floes may not be separated by this method because of the loss of the seeds when the ice floes are unevenly illuminated and closely packed without having clearly visible boundaries.

Therefore, this research work aims at developing a comprehensive image processing tool that can separate unevenly illuminated and highly irregular ice floes with close connections and overlaps. The tool developed in this research focuses on image enhancement to improve illumination, brightness, and edges of ice floes; and efficiently removing small floes to improve the processing time.

2.1.2 Supervised methods

The supervised method is based on the idea that the user specifies sample pixels in an image as representative of specific classes and then directs the image processing software to use these marked images as training and validation sets for the classification of all other pixels in the image

[21]. The training set is selected, processed, and labelled based on the experience and expertise of the user. The user also sets the bounds for how similar other pixels must be to group them together. These bounds are often set based on the spectral characteristics of the training area, plus or minus a certain increment (usually based on “brightness” or strength of reflection in specific spectral bands). The user also designates the number of classes into which the image is classified [21]. Therefore, supervised methods use feature learning to achieve image segmentation, and the benefit of using such approaches is that they provide faster detection results after successful training is completed. However, they also require significant training samples and label images to learn and predict accurate results.

The most common supervised methods used for image processing are Convolutional Neural Network (CNN) [44] and Fully Convolution Network (FCN) [45]. For example, [46] illustrates the CNN-based image recognition for autonomous driving, [47] describes another CNN-based method that identifies five different types of tissues to aid lung and colon cancer detection. [48] proposed a Deep CNN-based intelligent method that can automatically detect the presence of ice on wind turbines in regions affected by cold weather. An SVM-based platform to monitor sea ice using high-resolution synthetic aperture radar (SAR) images has been developed by [49]. There is also some dedicated machine learning (ML) based algorithms developed recently to process ice images, for example, river ice segmentation using deep learning [50], and segmentation of high-resolution satellite images of sea ice using weakly supervised CNNs [51]. [50] Presented the results of a comparative study using four state-of-the-art deep CNNs for segmenting images and videos of river surface into the water and two types of ice. These platforms are UNet, SegNet, Deeplab and DenseNet. While the overall detection performance is not quite impressive, the study demonstrated reasonable success in handling the lack of labelled images using data augmentation.

Then, [52] demonstrated the performance of two semantic segmentation-based SegNet and PSPNet101 neural network architectures for automated detection and classification of sea ice types using camera feeds onboard an ice breaker. However, these two nets only focused on detecting the ice classes as a whole, i.e., not separating the individual ice pieces. Recently, [53] developed another similar but more accurate novel CNN-based model for recognizing sea-ice images through semantic image segmentation. This approach provided better results in separating ice cluster from the background (i.e. water) compared to previous studies. [54] proposed a deep learning-based river surface ice quantification method using distant and oblique-viewed public cameras. Again, the focus in these analyses was detecting the overall ice extents and concentrations, rather than separating individual floes. Table 2-1 presents a succinct summary of the recent researches on image segmentations for an easy referencing.

As can be seen, although a significant number of algorithms for image processing have been proposed, image segmentation still remains one of the most challenging research topics, especially for situations similar to complex ice floe fields. It is because no existing method provides a unified framework for achieving fast and effective image segmentation. This can be attributed to two reasons: (a) image segmentation is a multiple-solution problem. It is possible to propose several best segmentation processes for a single image. (b) Complex images usually contain noise, background issues, nonuniform intensity and low signal-to-noise ratio. As a result, it is challenging to propose a generalized segmentation framework that will achieve satisfactory image segmentation performance and fit all application areas [55]. For example, the recent machine learning approach developed for line defect recognition in additive manufacturing based on DCNN [56], and another two models proposed for automatic detection of plant diseases based on SVC,

Table 2-1: Summary of recent works on image segmentations

Citation	Objectives	Methodology /Algorithms	Key Features
[40]	Automated segmentation and tracking of cancer cell nuclei	Watershed image segmentation method	Tracked cell and identified phases fairly well. Do not work well with uneven illumination and rapid variation in shapes.
[7]	Sea ice image processing and ice classifications	K-mean thresholding, Watershed segmentation, GVF SNAKE contour tracking	GVF SNAKE improves ice segmentation compared to traditional approaches. Managed to separate seemingly connected floes to some extent. Problem remains when floe size varies significantly, and noise exists.
[15]	Pancake ice detection in images	Marker Controlled Watershed and SVM	Uniform shaped pancake ice floes are effectively detected. Detection of irregular floes with blur edging and uneven illumination remains as a challenge.
[48]	Ice detection on wind turbine blades	Deep learning methods (Grad-CAM, U-Net, Resnet-50)	Boundaries of the icing regions are estimated with reasonable accuracy. Not applicable in separating different ice regions
[49]	Sea ice monitoring from satellites	SVM based ML	Capturing local and regional change in ice extents using time series image map. Not applicable for floe level ice detection.
[50]	River ice detection	CNN based deep learning (UNet, SegNet, Deeplab and DenseNet)	Computing surface ice concentration for two types of ice from images. Unable to handle noisy images. Unable to separate ice pieces.
[37]	Sea ice segmentation for climate change modelling	Weakly-Supervised CNNs	Detection, separation and segmentation of sea ice at continental level from high resolution satellite images. Not applied on ice floe level detection and classification.
[52]	Classifying sea ice using semantic segmentation	SegNet and PSPNet101 neural network	Automated detection and classification of se ice using camera feeds onboard. Focused on detecting ice classes as a whole, not separating the ice pieces.
[53]	Precise separation of ice clusters from water.	CNN based deep learning (Ice-Deeplab)	Aimed to segment the ice features from the image background. Not focused on separating ice floes
[54]	River surface ice quantification from oblique images	CNN based deep learning (UNet Marked watershed)	Quantify surface ice concentration and ice pan properties from oblique low-resolution images. Not focused on separating connected ice floes.

CNN [57]; and SVM, DT, RF [58], respectively, will not work for ice floe detection and feature extraction, as these models are focused on perfecting a particular aspect of the application.

This thesis focuses on proposing a Hybrid model to support an ice management system, at first, by providing useful ice information including ice concentration, density, and floe size distribution through image processing. Then, by training a dynamic ice force estimator, using those extracted information from the images to provide decision support.

2.2 Image based ice force prediction

The ship and offshore platforms operating in the Arctic region should be designed to have efficient performance in ice. So, with the increase in commercial activities, navigation and scientific investigation in polar regions, more attention has been paid to the structural design and maneuvering performance of ships in ice-covered waters. The determination of real-time ice loads on a ship hull, in this regard, is essential to analyze the ice-structure interaction and design appropriate structures against forces from ice.

Over the last few decades, understanding and modelling ice forces on fixed, moored floating, DP-controlled platforms/vessels; and on icebreaking or slowly maneuvering ships in managed or unmanaged broken ice fields have been attempted on multiple frontiers. For example, classical analytical formulation of single, large ice sheet load on fixed structure [59], analytical formulation of ice floes interaction with floater [60], empirical-statistical modelling of vessels and managed ice field interactions [61]. Physical model testing of vessels in managed ice [62], or platforms in shallow waters [63]. Ice force modelling on conical structures based on long-term field tests data [64]. Numerical modelling, such as Computational cohesive element model ([65], [66]), finite

element model ([67], [68], [69]), Discrete element modelling ([70], [71], [5], [72]), Smoothed Particle Hydrodynamics [73], Collision-Energy-Based Method [74], Particle-in-cell method [11], GPU event mechanics method [11]. Hybrid modelling, for example, model test and subsequent simulations of ice impacts on ship hulls in broken ice fields [75].

The analytical, empirical, and statistical methods often show high numerical efficiency and ease of integration; however, they do not accurately model the relevant physical processes. Hence these methods should not be considered where closeness to physical processes is the most important. Most of the computational methods require high computation resources. They often take a long calculation time, which is unsuitable for real-time simulations. To compensate for the deficiencies of one method by another, hybrid modelling methods are adopted to combine two or multiple computational methods. Therefore, there is an increasing trend in using hybrid methods to model ice-structure interactions. However, regardless of the methods adopted, validations with quality measurements are paramount to the success of the ice-structure interaction models; and the lack of high-quality physical model tests and full-scale measurements for a thorough validation adversely impacts confidence in the modelling [11]. Also, almost all these models lack versatility and are usually valid for certain ice conditions and certain vessels.

The applications of machine learning and deep learning techniques for ship performance predictions have been pursued by many researchers to tackle the above issue. Recently, [76] implemented Artificial Intelligence (AI) based Machine Learning (ML) and Deep Learning (DL) models to predict ship performance characteristics based on time-averaged and time-dependent data and prediction of forces on a dynamic positioning ship operating in a broken icefield. One modelling case involved developing an ML algorithm to predict time-averaged ice forces on DP-controlled ships at the given ranges of ice concentration, floe size, ice thickness, strength, density,

drift speeds and direction. The other modelling case involved predicting the time-dependent forces on a DP-controlled ship at specific operating conditions and ice-field parameters. The ML-based predictive models showed reasonable accuracy compared to the corresponding measurements and performed better than conventional regression-based models. Also, [77] reported a data-driven prediction model based on Artificial Neural Networks to estimate the ice resistance of ice-going ships in level ice, where the predictor was trained by various parameters, including ship geometries and test conditions. Later, [78] proposed an ML-based method to predict ice resistance for polar ships. Their methods include three ANN models, which are validated with full-scale and model-scale measurements. However, all these models involved the use of numerical data extracted from the field, model test or numerical analysis, which are, again, situation-specific, as real-time processing of ice images was not done to extract those datasets.

Using areal imagery and applying digital image processing techniques is one of the best ways to extract real-time ice characteristics in the oceans. Although with the advancement of computer visions, several advanced algorithms, for example, Convolutional Neural Networks (CNN), and Regions with Convolutional Neural Networks (R-CNN), You Only Look Once (YOLO) are developed and extensively applied in various fields for image analysis, the application of these advanced tools for ice image processing is still at its infancy. Some recent investigations include, iceberg detection using Masked-RCNN [12], detection of black ice to prevent accident using CNN-based platform [13], detecting sea ice surface features in high-resolution optical imagery using Python based opensource algorithm [14], detect pancake ice and compute their size distribution using SVM based method [15]. However, these works focused on extracting ice features only and did not investigate estimating ice forces.

2.3 Summary

The primary objective of this research work, therefore, is to address these gaps, and develop a tool that will extract the ice floe characteristics from experimental/field videos/images and use the information for ML-based real-time ice force prediction platform. The tools developed in this research work are faster compared to existing approaches, as it effectively focuses on image-based ice force prediction approach, mainly on moving vessel by reducing high computational time.

3.0 Image Based Sea Ice-Field Characterization for Predicting Ice-Structure Interactions

Shamima Akter¹, Syed Imtiaz², Salim Ahmed², Mohammed Islam³, Robert Gash³, Hasanat Zaman³

¹Department of Mechanical Engineering, Memorial University, St. John's, NL, Canada.

²Department of Process Engineering, Memorial University, St. John's, NL, Canada.

³National Research Council, Canada, St. John's, NL, Canada

Abstract

Accurate modelling of ice properties and ice-structure interaction forces is becoming increasingly crucial to ensure safe operations of ships and offshore platforms with the increased activities in arctic water. Extraction of ice features from real-time videos and images can significantly improve ice force prediction. However, accurate extraction of ice floe information is challenging due to several inherent complexities in ice images. This paper, in this regard, presents an improved ice image processing technique which can extract useful ice properties from a closely connected, unevenly illuminated floe field with various floe sizes and shapes. Several image processing features, including histogram equalization, wavelet denoising, gradient flow vector, snake algorithm, and distance transformation, were applied to develop the model. The effectiveness of the proposed method is demonstrated through the processing of simulated and model basin recorded ice field images, and its performance is compared with two other existing models. The potentiality of the model in real-time application is also described.

Keywords: Image Processing, Managed ice, Ice properties, Sea ice, Wavelet, Denoising

Nomenclature

x	Distance from origin in the horizontal axis
y	Distance from origin in the vertical axis
G	Gaussian function
σ	Standard deviation
R_0, G_0, B_0	Red, Green, Blue color matrices of the original image
R_n	Red color channel matrix for the normalized image
i, j	index for color matrices
u, v	The derivatives of the vector field in x and y directions.
μ	Parameters to control the balance between the integrands
f	Edge map which is larger near the edges
Φ	Scaling Function
Ψ	Wavelet Function

3.1 Introduction

Navigation in the Arctic is usually more challenging compared to other regions due to its remoteness, harsh weather and, most importantly, the presence of ice. Recently, increased commercial and scientific activities in ice-infested waters have driven further research on the accurate estimation of ice behaviors, including ice-structure interaction forces to ensure safe operations. Advancement in image processing has been providing researchers with sophisticated tools to extract and process various ice features and apply that knowledge to understand and forecast the behaviour of an ice field across multiple navigation scenarios.

Image processing through the development and application of novel theories and algorithms is gaining interest in recent years with a wide range of applications, for example, in computer vision,

medical imaging, remote sensing, face detection, feature extraction, forecasting, augmented reality, and many other fields [19]. The nature of solutions applied to process images largely varies depending on the field of applications. For example, in medical imaging, it might be more important to identify a particular object (e.g., a tumour) from an image. Whereas, for sea ice images, the objective is to identify and distinguish ice pieces of a complicated multi-domain ice field where floe sizes, concentrations, and shapes vary significantly. It becomes challenging to separate ice floes due to their close contact, overlaps and uneven illumination. Figure 3-1 shows a typical ice floe field with the usual complexities. Efficient and accurate extraction of relevant ice parameters from an image to facilitate ice force calculation and to understand its dynamic nature is a challenging problem.



Figure 3-1: Ship in complex, managed ice field [79]

Over the last few decades, several key technologies have been developed and applied for processing ice field images. For example, the classical histogram thresholding and clustering [23],

watershed transform [29], wavelet transforms ([30],[31],[32]), active contour model ([24], [25]), gradient vector flow (GVF) and snake ([33], [7]), Markov random field, spectral clustering, neural network [7] and so on. Otsu thresholding and k-means clustering are two basic methods widely used for background and foreground separation of a grayscale image based on its histogram. Though there is no room for pre and post-processing of the images while using these techniques, both methods work well for ice image segmentations when the floes are significantly brighter, well separated and have even illumination [37]. Edge detection methods, for example, derivative and morphology edge detection techniques, usually perform better in this regard and can detect distinct boundaries with some nonuniform illumination. However, these methods fail to identify close boundaries and separate the ice floes that are tightly connected [38].

Watershed-based methods can be applied in this regard to separate connected floes with blurred edges in an ice field image. This technique has been successfully used in other applications, for example, cell nuclei image segmentations and grain separation [39], [40]. However, watershed-based techniques produce over- and under-segmentations when ice floes are unevenly illuminated, and floe sizes and shapes vary significantly [41]. An improved watershed method is proposed by [42], where a neighbouring region merging algorithm is added for better edge detection. Nevertheless, accurate identification of floe boundaries and floe numbers remains a challenge as watershed transformation operates on binary images, and a significant amount of real boundary information between connected floes can be lost while using this transformation. Recently, [15] and [43] applied another modified version of the watershed method named marker-controlled watershed (MCW) technique for pancake ice detection and agricultural field detection from satellite imagery. They incorporated non-linear support vector machine (SVM) analysis with

traditional watershed to avoid over-segmentation. However, the problem of detecting irregular floes with uneven illumination and blur edging still remains.

Recently several researchers implemented a modified GVF method to tackle the issues of detecting complex shape floes [25], [4]. For example, the GVF snake algorithm, combined with distance transform-based automatic contour initialization, is adopted by [7] to separate seemingly connected floes. After that, ice floe shapes are enhanced using an ice shape enhancement algorithm and identification of individual ice floes is accomplished. This new technique is insensitive to proper initialization and can detect various complex floe shapes. However, connected ice floes may not be separated by this method because of the loss of the seeds when the ice floes are unevenly illuminated and closely packed without having clearly visible boundaries.

There are also some dedicated machine learning (ML) based algorithms developed recently to process ice images, for example, river ice segmentation using deep learning [50], and segmentation of high-resolution satellite images of sea ice using weakly supervised CNNs [51]. Though ML-based models are widely used in various fields, these techniques are known to be black-box approaches and do not offer physical insight. In this study, our focus is mainly on applying knowledge-based methods for image processing, primarily to use the knowledge to improve navigation performance.

The primary objective of this work is to develop a comprehensive image processing tool that can separate unevenly illuminated and highly irregular ice floes with close connections and overlaps. Thus, advancing the capabilities of ice image processing methods. The tool developed in this research focuses on image enhancement to improve illumination, brightness and edges of ice floes, and efficiently removing small floes to improve the processing time. The methodology is

implemented using MATLAB. Comparisons and validations of the developed tool are demonstrated using simulated and real-world ice field images.

The rest of the paper proceeds as follows: Section 3.2 describes the algorithms and techniques used for this study. In Section 3.3, the results obtained from the proposed method are described and compared with selected existing works. Finally, concluding remarks, including limitations and scopes for future work, is presented in Section 3.4.

3.2 Methodology

Extracting meaningful information from an image through image processing involved several tasks, namely, edge detection, noise removal, shape detection, and object count. A range of techniques are applied to complete these tasks that can be divided into the following four steps:

- Image enhancement - preparing images for analysis.
- Image segmentation - separating objects and regions of interest.
- Morphological filtering - removing noise.
- Region analysis - extracting statistical data.

A flow chart summarizing the methodology is shown in Figure 3-2, and the following subsections provide extended descriptions of the method.

3.2.1 Image enhancement

The purpose of this step is to acquire and preprocess the selected image through filtering, adjustment of contrast and brightness, sharpening etc., to make it easier to identify key features. It is a crucial step for successful image processing. In this study, a multistep preprocessing framework is proposed for ice image enhancement through the incorporation of a few novel features.

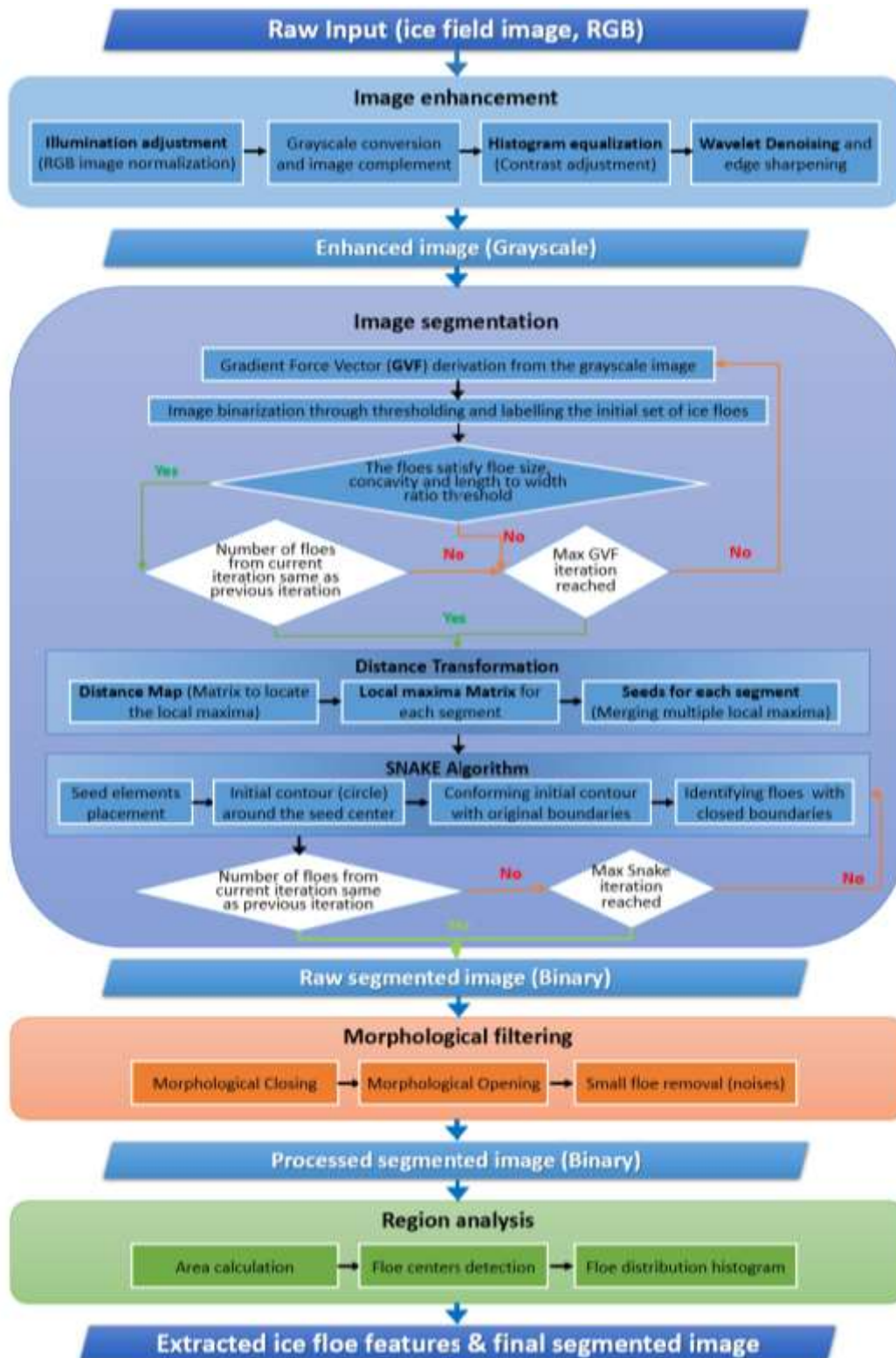


Figure 3-2: Flowchart for the proposed ice image processing technique.

3.2.1.1 Illumination adjustment

First, the uneven illumination is addressed through normalizing the colour image using a unique technique. The input colour image is separated into RGB channels, and each element of these colour matrices are normalized using all three corresponding colour values. Eq. 3-1 illustrates the method using normalization of the red colour as an example:

$$R_n(i, j) = R_0(i, j) / \sqrt{R_0^2(i, j) + G_0^2(i, j) + B_0^2(i, j)} \quad (3-1)$$

The three normalized matrices for red, green and blue colours are then combined back to obtain the improved image with better illumination, which is then converted to grayscale for further preprocessing.

The outcome of the image enhancement stage is demonstrated in Figure 3-3. A low-resolution segment of an ice field image recorded in an ice basin is shown in Figure 3-3(a) [62]. The default grayscale conversion of this image using the Otsu thresholding method is presented in Figure 3-3(b). Figure 3-3(c) is the grayscale conversion resulting from the improved colour image obtained from the brightness adjustment phase. As noticed, the ice floes became dark in the revised uniformly illuminated image. Hence, the complemented image is generated to reflect the darker background which is shown in Figure 3-3(d).

3.2.1.2 Histogram equalization

It is evident that the inverted image obtained in the previous stage, Figure 3-3(d), has very poor contrast. Thus, contrast adjustment is required to ensure a sharp difference between the foreground and background. As can be seen in the histogram, Figure 3-3(e), the intensity values are limited to the middle portion of the range, confined within 0.5-0.65 (left).

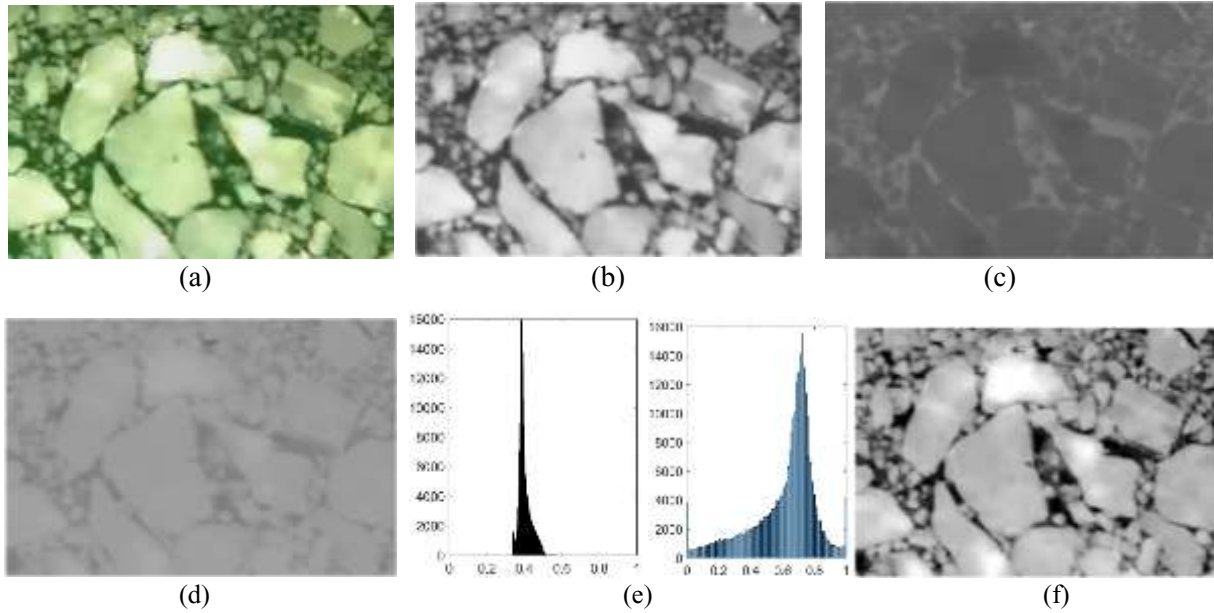


Figure 3-3: Steps of image enhancement: (a) original image, (b) default grayscale conversion, (c) grayscale conversion after normalizing the illumination, (d) colour inversion of ‘c,’ (e) histogram equalization, (f) improved grayscale after wavelet denoising.

Using Eq. 3-2, the image adjustment is made in such a way that the contrast fills the entire intensity range [0, 255], keeping the shape of the histogram similar, as seen on the right histogram in Figure 3-3(e).

$$I_n = [I_o \cdot (1/255)]^\gamma \cdot 255 \quad (3-2)$$

Here, I_n is the new image after equalization, I_o is the original image, and γ is the histogram correction factor. First, the original image is scaled, and then a correction factor is applied. Next, the image is scaled back to fit the [0 255] range through element-wise multiplication. γ can be any value between 0 and infinity. The mapping is linear when γ is 1. If γ is less than 1, the mapping is weighted toward higher (brighter) output values, and it is weighted toward lower (darker) output values if γ is greater than 1. For this study, $\gamma = 0.9$ is used.

A drawback of this process is that 1% of the colour data from the original histogram is saturated at low and high intensities in the new image (the two peaks in the histogram are near zero and 100). However, the resulting sharp differences between black and white, as compared to the default grayscale in Figure 3-3(b), prevent over-segmentations of the ice floes, as discussed later in section 3.

3.2.1.3 Denoising and edge sharpening

Following the histogram equalization, further enhancement is done to remove the noise from the image using wavelet based denoising technique. To detect the ice floes efficiently, it is important that the edges are preserved and sharpened as much as possible while denoising the image. Traditional lowpass filtering used for removing noises often smoothen the edges and adversely affects the image quality. The application of such technique creates a merging of multiple ice floes. Wavelet, on the other hand, can remove noises while preserving the important edge features on the image.

Wavelet transform concentrates signal and image features in a few large-magnitude wavelet coefficients. Appropriate thresholding is then used to remove those coefficients marked as noise without affecting the image quality. Discrete Wavelet transform (Eq. 3-3) is used in the transformation of image pixels to wavelets, and after thresholding, the image data is reconstructed using inverse wavelet transform (Eq. 3-4). In these equations, φ and ψ are the scaling function and wavelet function, respectively. More details on this can be found in [80].

$$W_{\varphi}(j_o, k) = \frac{1}{\sqrt{M}} \sum_x f(x) \varphi_{j_o, k}(x), \quad W_{\psi}(j, k) = \frac{1}{\sqrt{M}} \sum_x f(x) \psi_{j, k}(x) \quad (3-3)$$

$$f(x) = \frac{1}{\sqrt{M}} \sum_K W_{\varphi}(j_o, k) \varphi_{j_o, k}(x) + \frac{1}{\sqrt{M}} \sum_{j=j_o}^{\infty} \sum_K W_{\psi}(j, k) \psi_{j, k}(x) \quad (3-4)$$

In this study, biorthogonal wavelet (bior4.4) is used which is found to be effective for image denoising in the literature [81]. Figure 3-4 shows the shape of bior4.4 wavelet and scaling functions. The threshold is selected based on Stein’s unbiased estimate of risk -quadratic loss function [30]. The estimation of noise variance in the image is done based on the highest resolution wavelet coefficient. Further explanation of the setup is available in [82]. The final output of image enhancement, after wavelet denoising, is shown in Figure 3-3(f), which is evidently better than the original grayscale image in Figure 3-3(b) in terms of uniform illumination and image sharpness.

Following this appropriate enhancement comes the image segmentation step, which primarily involves separating the foreground from the background through edge detection and boundary separation.

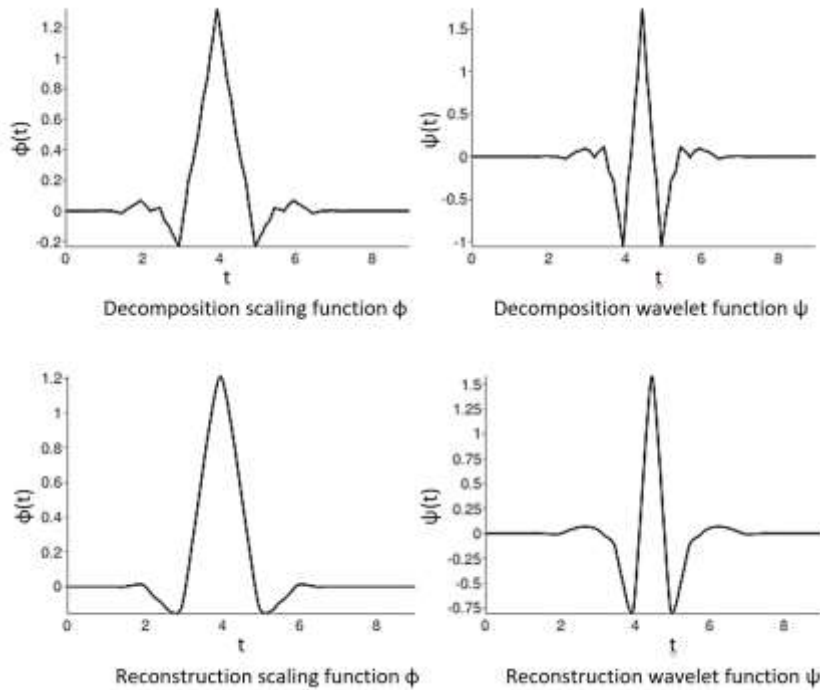


Figure 3-4: Decomposition and reconstruction functions for bio-orthogonal 4.4 wavelet.

3.2.2 Image segmentation

The core of image segmentation is to convert an image into a collection of regions of pixels that are separated by some criteria. A common criterion used is locating abrupt discontinuities in pixel values to identify those regions as edges. Some other approaches are clustering, thresholding, region growing, etc. However, as explained earlier, applying a single technique is inadequate to handle the complex scenario of separating ice floes.

Therefore, the enhanced image obtained from the previous sub-section is segmented by developing an improved version of the GVF snake model proposed by [7]. The model is a combination of thresholding, clustering and region growing where the GVF snake technique [83] is applied to efficiently track the boundaries of the ice floes starting from the initial contours around the seed element. The Gaussian blurring [33] is removed from the original approach as wavelet is now used to preprocess the image, resulting in better performance in terms of edge detection. Figure 3-5 shows the images processed using Gaussian blurring and wavelet. As can be seen, the latter produced a comparatively better preprocessed image with less noise and slightly sharper edges. Also, adaptive thresholding is used for binarizing the image instead of traditional global thresholding. It chooses the threshold based on the local mean intensity (first-order statistics) in the neighbourhood of each pixel, and such an approach is better suited for images where foreground and background are not clearly distinguishable throughout the image, as in the case of ice floe images. An improvement is added in iteration counting as well to maximize the computational efficiency, as shown in the flowchart in Figure 3-2. Previously, the snake algorithm was used to run until the set number of iterations was reached. However, it was found that the number of floes detected by the algorithm can reach saturation (not changing from the previous iteration) before the total set number of iterations is performed. The revised program will exit the

snake algorithm once the floe numbers no longer change, rather than completing the total number of iterations.

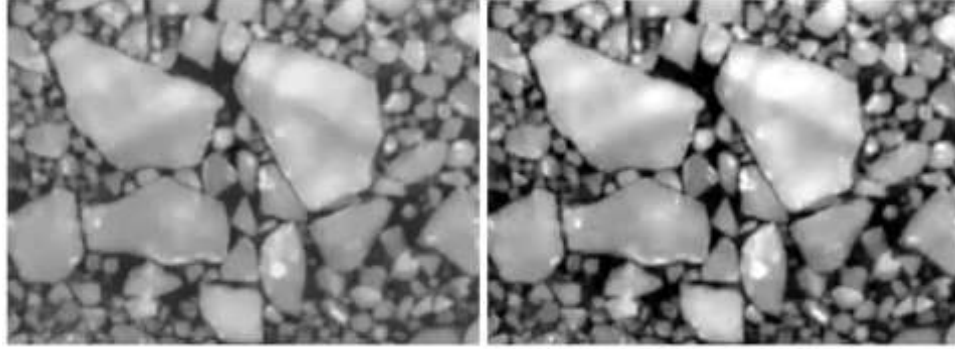


Figure 3-5: Image processed using Gaussian blurring (left) and wavelet (right).

The traditional snake algorithm [83] defines the boundaries as an active contour modelled as energy-minimizing spline. A typical snake boundary curve $S(l) = (x(l), Y(l))$ with normalized arc length $l \in [0,1]$ moves through the image's spatial domain in such a way that the summation of external and internal energy is minimized (Eq. 3-5).

$$E_{total} = \int_0^1 [E_{int}(S(l)) + E_{ext}(S(l))] dl \quad (3-5)$$

Here, the internal energy is defined using two parameters to control the snake's rigidity and tension. External energy, on the other hand, is calculated based on the image gradient. As explored in [7], the traditional snake algorithm is sensitive to the initial contour, which means if the initial guess of the boundary of the seed element is not near to actual boundary, the snake will not conform to the actual boundary. The snake also progresses unpredictably near concave regions. Such limitations result in significant over-segmentations for typical ice floe images. The modified method uses a GVF field $V(x,y) = [u(x,y), v(x,y)]$, derived from the spatial diffusion of the gradient of edges from the image to minimize the energy functional following this relationship:

$$\epsilon = \iint [\mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 [v - \nabla f]^2] dx dy \quad (3-6)$$

The improved method can track and get closer to the true boundary even if the initial contour is not close to the true boundary. However, if the initial contour is too far away from the true boundary, more iterations will be required to reach it. Also, the GVF algorithm operates on the grayscale image; thus, unlike in the watershed method, real boundary information is preserved.

Once the GVFs are calculated, ice floes are labelled after being separated from water using thresholding. Then, each floe is checked to ensure that the floe area is less than the given threshold, the ice floe has a convex shape (the ratio between the floe area and its minimum bounding polygon area is larger than the threshold), and the length-to-width ratio of the minimum bounding rectangle of the ice floe is less than the threshold [7].

The snake algorithm is then run for those floes which do not satisfy the criteria. Although GVF snake ensures that a detected boundary is a closed curve, initial contours for the seed elements are required for the successful implementation of this method. An automatic contour initialization used in [7] is applied in this regard with slight modification in iteration counting to improve the efficiency of the proposed ice image segmentation method. This automatic initialization is designed based on the distance transform [84] and the local maxima of the binary format of the input image. For a binary image I , the distance transform, $\mathbf{D}(x,y)$, is the minimum distance from each pixel in I (which belongs to the foreground, O) to the background \mathbf{B} , which is:

$$\mathbf{D}(x,y) = \begin{cases} 0 & \text{if } (x,y) \in B \\ \min_{b \in B} d[(x,y), b] & \text{if } (x,y) \in O \end{cases} \quad (3-7)$$

Where $d[(x,y), b]$ is the distance measure between pixel (x,y) and b [84]. More details related to these methods can be found in the referred publications.

The next step is morphological operation, where several techniques are used to clean the ice floes.

3.2.3 Morphological filtering

Once the ice floes are segmented, morphological closing and morphological opening are performed using a structuring element and through the process of erosion and dilation. It is a predefined shape used to probe an input image and check on how its element fits or misses the shapes in the image of interest [7]. The erosion operation is used to smoothen/trim an object by removing pixels from the boundaries, whereas the dilation operation adds pixels to the boundaries to fill/thickens the object.

Thus, morphological closing is used to fill holes and join narrow breaks, while the morphological opening is used to remove thin protrusions and break thin connections. To illustrate, for an image I (Figure 2-6(a)) and a 2 by 2 structuring element S , the results of morphological closing of I by S ($(I \blacksquare S)$) will be a dilation followed by an erosion, and the resultant output is shown in Figure 3-6(b). Similarly, the morphological opening of I by S ($(I \circ S)$) will be an erosion followed by a dilation, and the resultant output is shown in Figure 3-6(c).

Another noise removing filtering is used to detect and remove ice floes smaller than a particular size from the grayscale image before implementing the GVF snake algorithm, thus, improving the computational efficiency significantly (as described later in section 3.3). The process of noise removal from the grayscale image is illustrated in Figure 3-7.

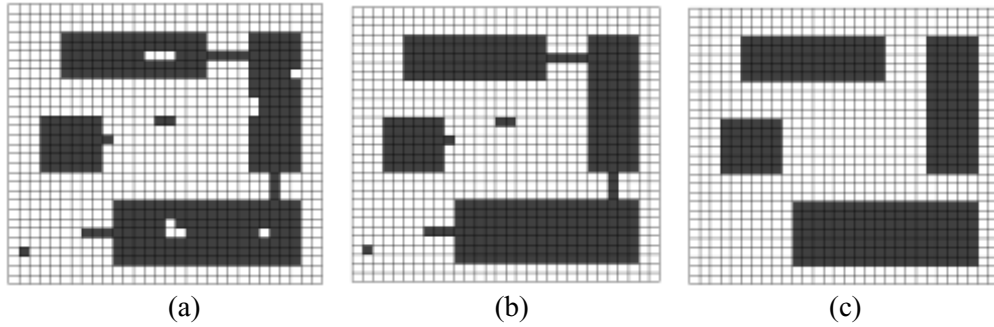


Figure 3-6: Binary image I (a), morphological closing of I (b), morphological opening of I (c).

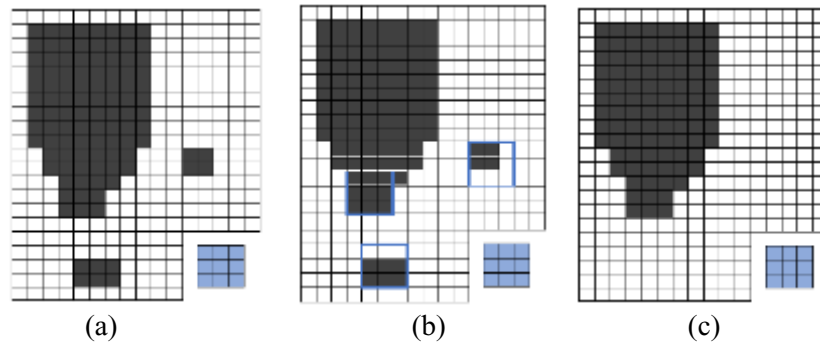


Figure 3-7: Binary image J and the structuring element (a), overlapping of structuring element on J to detect noises (b), removal of noises from J (c).

As can be seen, a structuring element (denoted in blue pixels) of size 3 (3 by 3) is chosen for this example. The noise removal algorithm is then run by moving the structuring element across the image **J** to check if there are any clusters of pixels that are unable to hold the structuring element entirely without exposing any of its pixels to the boundary. For illustration, the structuring element is overlapped in three locations on image **J** (Figure 3-7(b)). As noticed, the protruding segment of the big floe will be kept as the structuring element is fitting inside this big floe, and only those two small floes will be removed as they are unable to contain the structuring element entirely (Figure 3-7(c)). The reason for using another noise removal technique here apart from the morphological

opening and closing is that use of a bigger structural element for such opening, and closing will truncate the protruding portion of the big floes (such as in Figure 3-7). In contrast, this revised algorithm will only remove isolated small ice floes. As shown later in section 3.3, a noise size of 15 is used in this study to remove smaller floes.

3.2.4 Region analysis

Once the morphological operations are completed, the image is now ready for extraction of information about the ice floes in the image, for example, the number, size and centers of the ice floes, and the histogram of floe distribution. Several standard algorithms, such as regionprops, bwlevel, bwarea, are used for data collection in this regard. Accurate extraction of this floe related information is one of the crucial steps for ship-ice interaction force prediction modelling. The force prediction modeller requires reasonably precise estimation of ice concentrations, floe counts and floe size distributions as input to ensure a reliable ship-ice interaction force prediction. The next section describes how the information extracted from this region analysis phase is arranged, reported and compared with other existing methods.

3.3 Results and discussion

The performance of the proposed image processing model is evaluated in two stages - using simulated ice images with designed complexities and using images from ice tank tests.

3.3.1 Processing simulated ice images

One of the main objectives of this study is to develop a robust model that can detect ice floes of various sizes and irregular shapes with sharp concave and convex corners. Two simulated ice floe

images are prepared to test the efficiency of the developed tool in detecting ice floes with such complexities. These two images are shown in Figure 3-8.

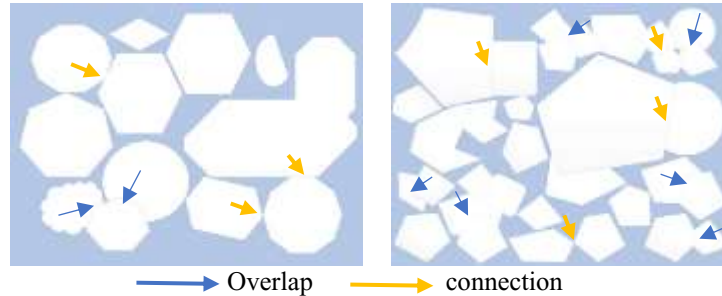


Figure 3-8: Simple (left) and complex (right) simulated ice floe images.

As can be seen, these images are carefully designed to include complex shapes with sharp corners, close contacts, and overlaps among adjacent floes. Both these images are analyzed using the model proposed by Zhang & Skjetne [33], MCW model by Turker et al. [43] and the improved model proposed in this study. The thresholding parameters mentioned in sub-section 3.2.2, and the iteration numbers are kept the same for both analyses. It should also be mentioned here that the RGB image normalization applied for image enhancement has no impact on these ideal images as the ice floes are evenly illuminated. As shown in Figure 3-9, for the simple ideal image, Zhang & Skjetne was unable to separate two closely touched floes (top left) and considered the three overlapped floes (bottom left) as one floe. The proposed model and MCW model detected all the floes with 100% accuracy (12 out of 12 floes).

Similarly, Figure 3-10 illustrates the performance comparison of the three models for the complex simulated image. As noticed, Zhang & Skjetne model combined all eight floes on the top right corner as one and failed to isolate them. The same goes for other overlaps near the bottom region of the image. MCW model, on the other hand, performed exceptionally well and detected all floes,

including overlaps, close touch and complex curves. There is just a small over-segmentation for the biggest floe.

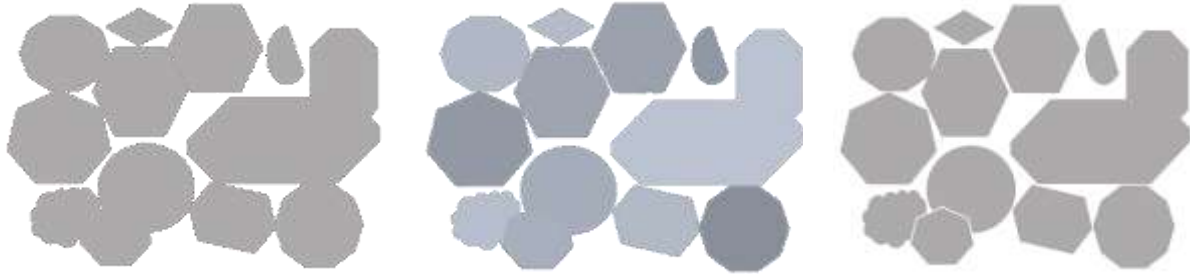


Figure 3-9: Simple, simulated ice floe image segmentation: (left) Zhang & Skjetne (2018), (middle) Marker Watershed (Turker et al., 2021), (right) proposed model.

The proposed method also detected all the overlap, and there was no under-segmentation. However, there appear to be a few over-segmentations, especially around the corners of the biggest floe. This happened because the initial contour for this big floe is too far from the true boundary, and the number of iterations set for this analysis is not enough to capture the actual boundary accurately. The results can be improved with the expense of computational efficiency by increasing the number of iterations for the GVF snake process. However, the level of improvement achieved is not justified compared to the additional computational time required. Table 3-1 compares the floe detection performances of the three models against manual counting.

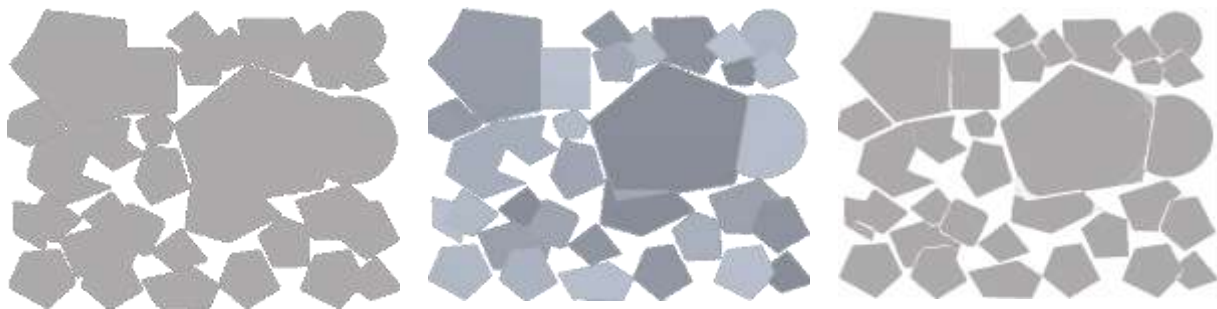


Figure 3-10: Complex, simulated ice floe image segmentation: (left) Zhang & Skjetne (2018), (middle) Marker Watershed (Turker et al., 2021), (right) proposed model.

Table 3-1: Floe detection comparison among various models for simulated images.

	Manual count	Zhang & Skjetne	Marker Watershed	Proposed model
Simple simulated image	12	7	12	12
Complex simulated image	33	16	33 (1 oversegmentation)	33 (6 oversegmentations)

3.3.2 Analysis of real ice tank images

After demonstrating the capability of the proposed model in processing ideal ice images, its performance in identifying closely packed ice floes with uneven illumination is tested in this subsection using five low-resolution images (350 by 230 pixels). These ice image segments are extracted from five different ice model testing reported in [62]. As can be seen in Figure 3-11, all these images contain noises, close contact and near overlapping situations, uneven illumination of floes, and a variety of floe sizes.

Similar to subsection 3.3.1, these five images extracted from the ice model test are processed using Zhang & Skjetne, MCW and the proposed model. To make a fair comparison, various model parameters are kept constant across models, as shown in Table 3-2.

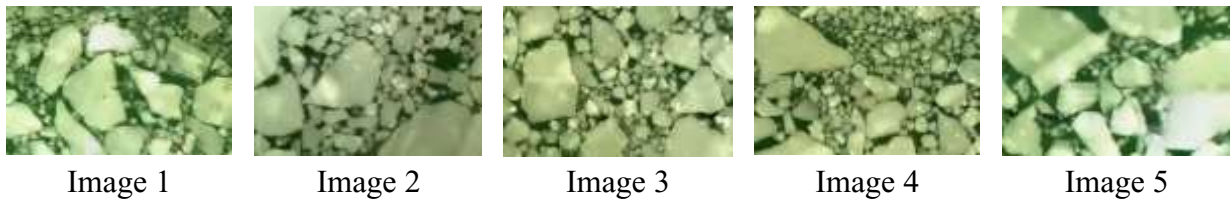


Figure 3-11: Ice floe mages extracted from Model Basin experimental test [62].

Table 3-2: Model parameters for ice floe image analysis

Parameter	Value
Min ice piece area for segmentation	580
Max ice piece area for segmentation	8000
Convexity threshold	0.85
Length to width ratio	2.0
Strel size	1.0
GVF iteration	800
Snake iteration	200
Min detected ice floe area	20

Figure 3-12 compares the processed images obtained from the three methods with the original version. The MCW performed poorly for all five images and produced significant over-segmentations. This is because the MCW method is unable to handle nonuniform illumination. The method proposed by Zhang & Skjetne did well in detecting large and medium size floes, those with prominent boundaries. However, it over-segmented many big floes (as highlighted using black circles) due to uneven illumination issues. This model also missed many small floes and part of bigger floes due to blurry edges, weak boundaries, and colour shading issues.

On the other hand, the proposed improved method performed much better in detecting and segmenting both the small and the large floes. It does produce under-segmentations, especially in images 2, 3 and 4, which can be argued in favour of a conservative estimation. Also, both Zhang & Skjetne and the proposed method failed to detect the large floe near the lower right corner in image 2 and ended up over-segmenting it.

Nevertheless, the floe counting accuracy and detection of the overall amount of ice concentrations improved significantly when the proposed model was used. Improved prediction of floe numbers

and ice concentrations is crucial to ensure a better estimation of ship-ice interaction. Table 3-3 represents a floe count comparison chart for both methods against floe counted via manual observation. The proposed method is doing much better in terms of floe identification in complex situations with nonuniform illuminations, complex floe shapes and weak boundaries. For images

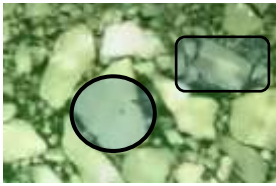
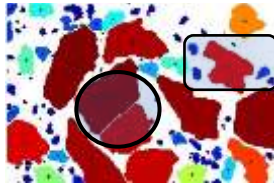

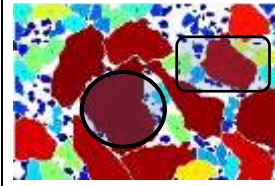

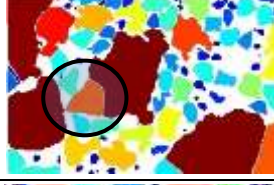



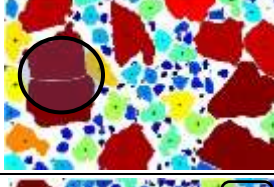



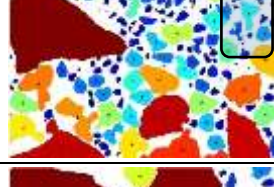






	Original image from experiment	Output from Zhang & Skjetne (2018)	Output from Marker Watershed (2021)	Output from proposed model
Image 1				
Image 2				
Image 3				
Image 4				
Image 5				

Figure 3-12: Segmentation results for ice floe images from model basin test – comparison among Zhang & Skjetne (2018), Marker Watershed (Turker et al., 2021) and proposed model.

1 and 5, the floe counting from the proposed method is nearly the same as the manual observation. As for images 2 to 4, both the proposed method and Zhang & Skjetne missed a significant number

of small floes. However, Zhang & Skjetne's model could not capture those floes, while the proposed method captured those small floes but combined them on many occasions, thus reducing the floe counts. This fact can be clearly understood from the ice coverage prediction chart in Table 3-3. As noticed, the ice coverage prediction for the proposed method is consistently close to the actual percentage. Therefore, it can be said that floe information extracted from the proposed method will help achieve a better ship-ice interaction force prediction. In contrast, the Zhang & Skjetne model performs poorly in capturing the coverage with accuracy. Also, as seen in images 2 to 4, the ice coverage percentage for the proposed model is slightly higher than the actual coverage, which occurred as the proposed model combined small floes on several occasions, thus increasing the percentage. It should be mentioned here that the boundaries of all the ice floes are marked manually on an image so that none of them are missed when the actual ice coverage percentage is calculated.

Table 3-3: Floe count and ice coverage comparison between Zhang & Skjetne and proposed models

		Image 1	Image 2	Image 3	Image 4	Image 5
No of floes	Manual Observation	105	114	119	157	60
	Zhang & Skjetne	74 (-31)	81 (-33)	90 (-29)	123 (-34)	50 (-10)
	Proposed model	108 (+3)	80 (-34)	91(-28)	127(-30)	63 (+3)
% of Ice coverage	Actual Coverage	76.56	77.59	79.90	74.91	81.50
	Zhang & Skjetne	57.02	71.13	70.10	63.60	69.93
	Proposed model	75.68	78.47	81.41	75.91	80.89

Figures 3-13 and 3-14 depict the variation in ice coverage percentage prediction and time taken by various methods. A laptop with a 10th generation core i7 CPU@2.30 GHz with 16 GB of ram is used to run the simulations for this study. As already mentioned, the proposed model predicts the

ice coverage more precisely compared to Zhang & Skjetne. As for processing time, MCW is the most efficient; however, very poor in terms of accuracy, as described earlier. Of the other two models, the proposed one is fairly better in terms of efficiency as compared to Zhang & Skjetne. It is also observed that processing time is more dependent on the complexities of the image rather than concentration because image 2 (which is more complex) took the longest time, although image 5 has the highest ice density.

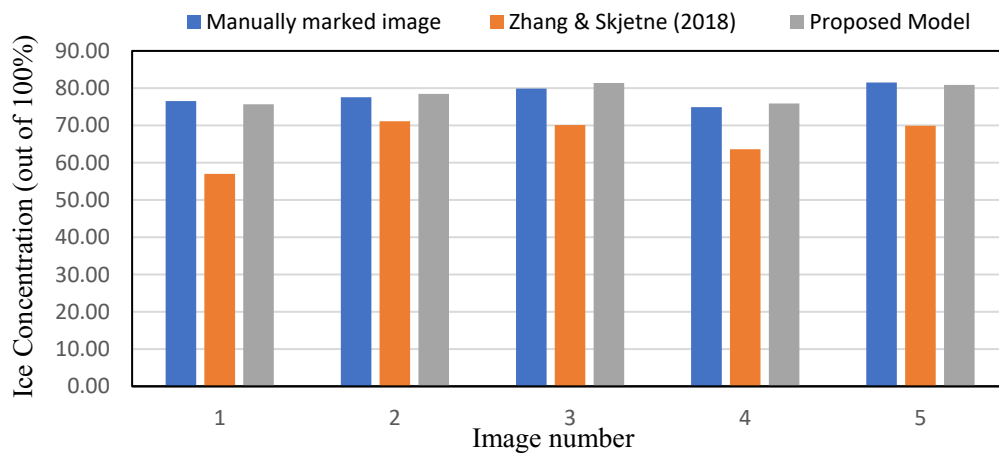


Figure 3-13: Comparison of ice coverage prediction.

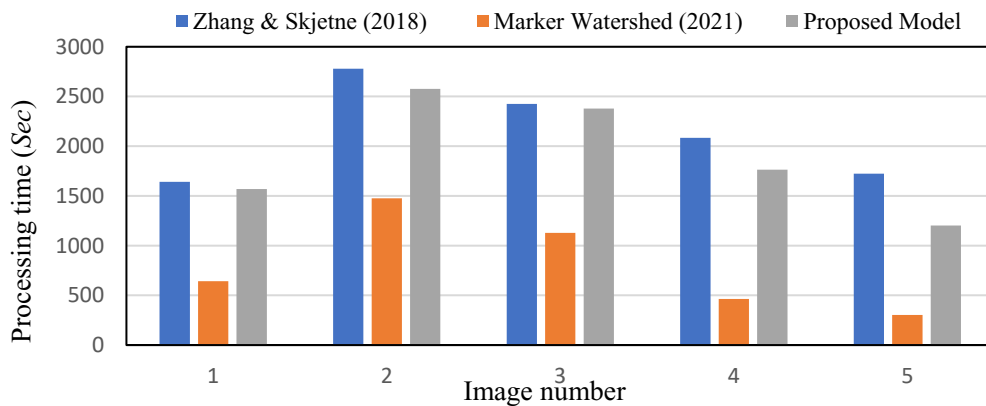


Figure 3-14: Processing time comparison among various models.

Figure 3-15 compares the histograms of floe counts for all five test images obtained from the Zhang & Skjetne and proposed methods with the manually marked image results. As can be seen, Zhang & Skjetne missed smaller floes as compared to the proposed method, especially in images 1 and 5. The counting of larger floes also varies for Zhang & Skjetne produced a few over-segmentations (images 1,2,3, and 5).

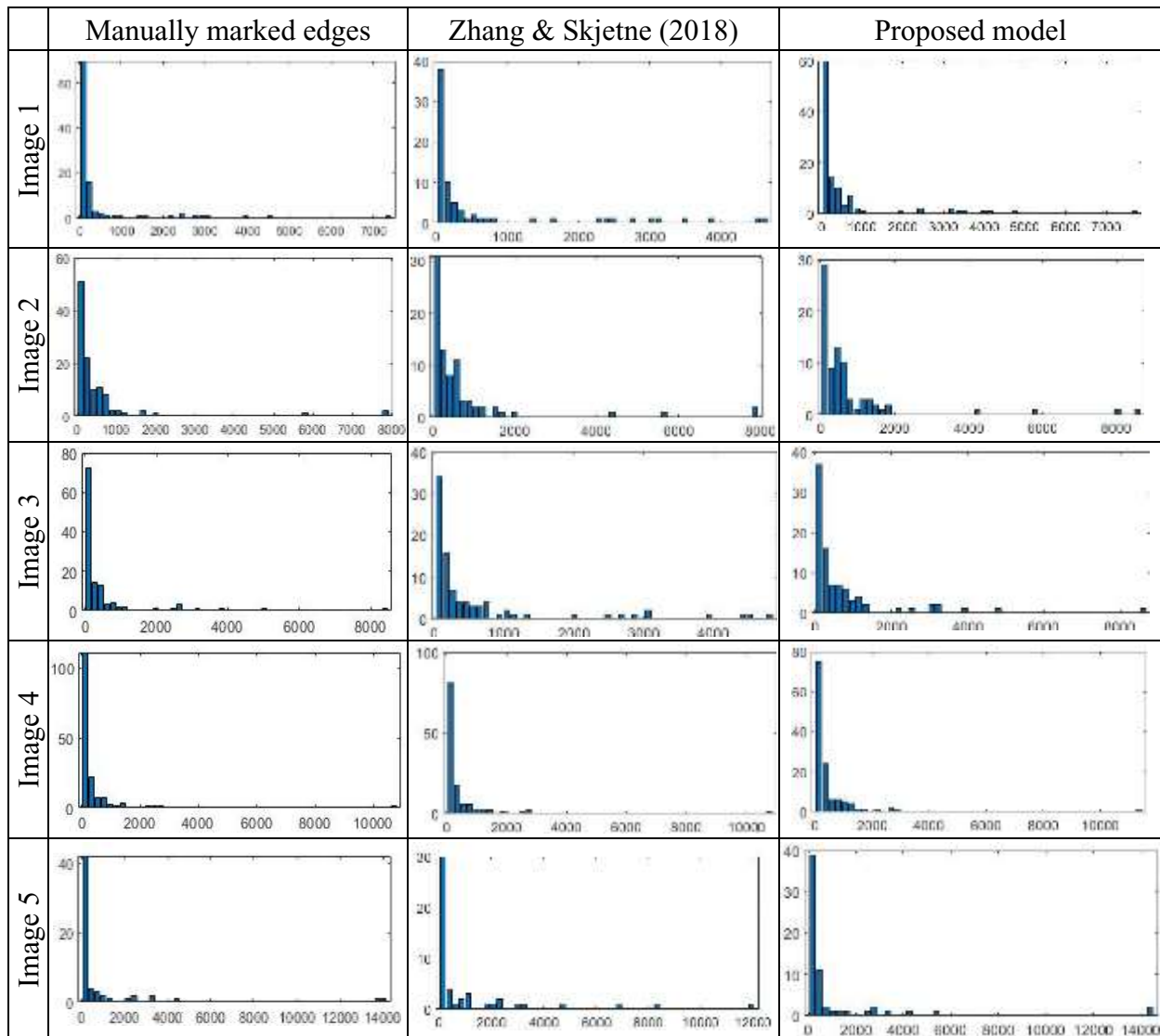


Figure 3-15: Histograms of floe count [no of floes (y) vs flow size in pixels (x). 1 mm² =14.2 pixels]: (left column) outputs from the manually marked image, (middle column) outputs from Zhang & Skjetne (2018), (right column) outputs from the proposed method.

Histogram comparisons also confirmed that both methods miss small floes on various occasions, as already described earlier, especially for images 2 to 4. Apart from that, both methods captured the overall floe distribution pattern reasonably well.

3.3.3 Computational efficiency and real-time applicability

As can be seen from the analysis results presented in subsection 3.3.2 above, the accuracy of the proposed method is quite acceptable; however, the processing time taken is not feasible for real-time ship-ice interaction image processing applications. The floe information should be extracted from the image within a minute, and fed into the ship-ice interaction force modelling platform for a real-time prediction. Therefore, as described in Methodology subsection 3.2.3, a technique for removing small ice floes during the image enhancement step is proposed. As small ice floes are removed as noises before running the actual segmentation algorithm, the computational load in detecting the number of ice floes and floe edges reduces drastically. Thus, making the model significantly faster, improving the overall efficiency and feasibility of its application in real-time ship-ice interaction force prediction.

Two full-size model test images are used to test the performance of the proposed model in extracting ice features from full-size experimental images by removing noises (small floes below a certain size). As shown in Figure 3-16, both images A and B are of size 4573 by 1117 pixels and contain a significant amount of noise. Floe sizes below 30 pixels are marked as noise and removed before the analysis is run. The floes in image A are more uniform in terms of size and shape as compared to image B. As noticed in the segmented images, the proposed method detected most of the floes accurately in image A, and for image B several under-segmentations occurred. However, from the practical point of view, those under-segmentations can be compensated under a

conservative estimation approach, keeping in mind the significant improvement achieved in computational time.

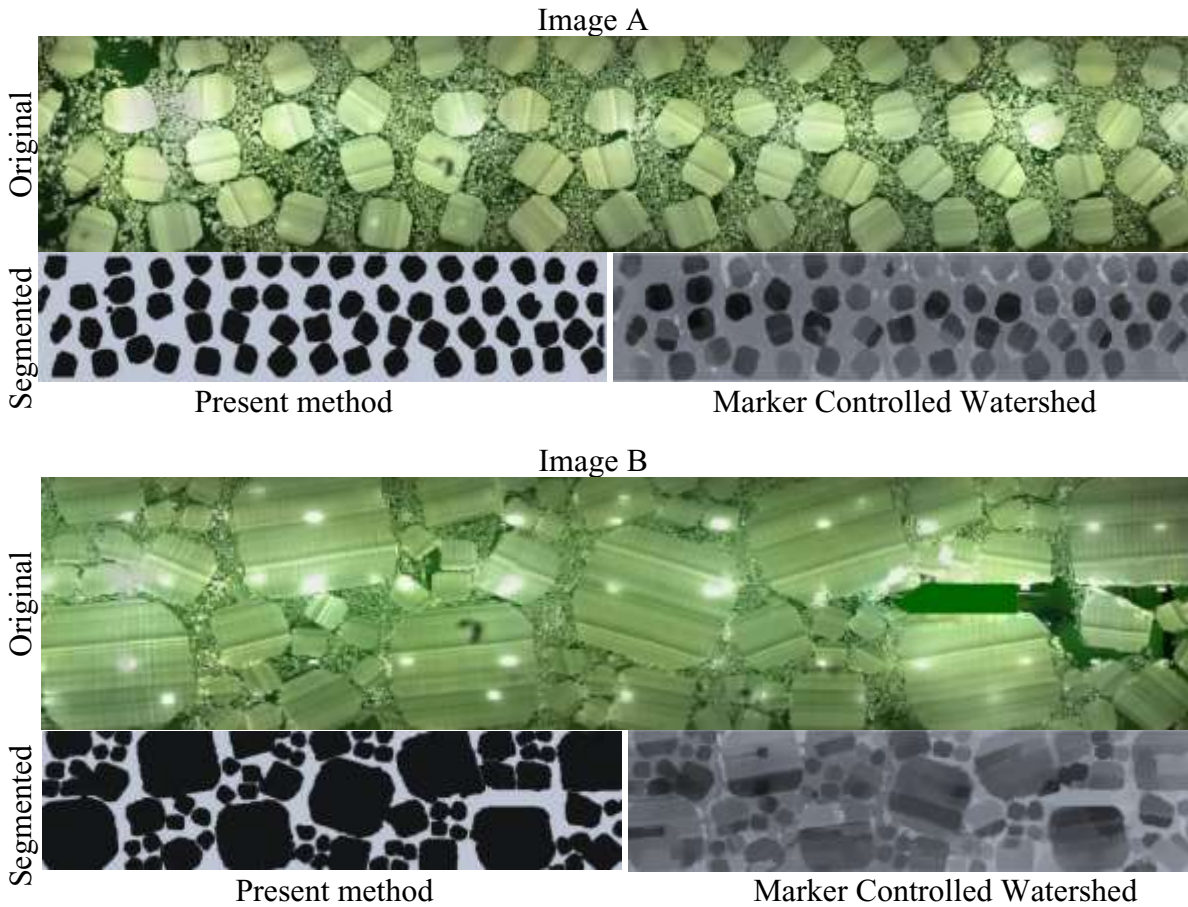


Figure 3-16: Evaluation of model performance after improving the computational efficiency by removing small floes below significant sizes (noises).

The same noise removal technique is applied in Zhang & Skjetne (2018) and the Marker watershed model (2021). Both these images, A and B, are then processed to compare the improvement in computational performances achieved among these three methods. As can be seen in Figure 3-17, the processing time reduces drastically for all three methods. The marker watershed stills take the lowest time. However, it produces over-segmentation for more than 30% of the floes (Figure 3-

16). Therefore, the slightly longer time taken by the proposed method can be justified considering the accuracy achieved compared to the Marker Watershed. The superiority of the proposed method over the other two approaches in detecting the number of floes and ice concentrations is already discussed in detail in section 3.3.2; thus, not repeated here.

The lowest time taken by the proposed method for processing the five images in Figure 3-11 was for image 5, which was 1202 sec. The size of those images was 350 by 230 pixels. Now, images A and B in Figure 3-14 are 63 times larger than image 5 in Figure 3-11. However, the processing time for images A and B was 62 sec and 76 sec, respectively, which became nearly 16 times faster. Therefore, it can be said that the proposed method has good potential for real-time ship-ice interaction force prediction application, and further improvement can be made in this regard in future studies.

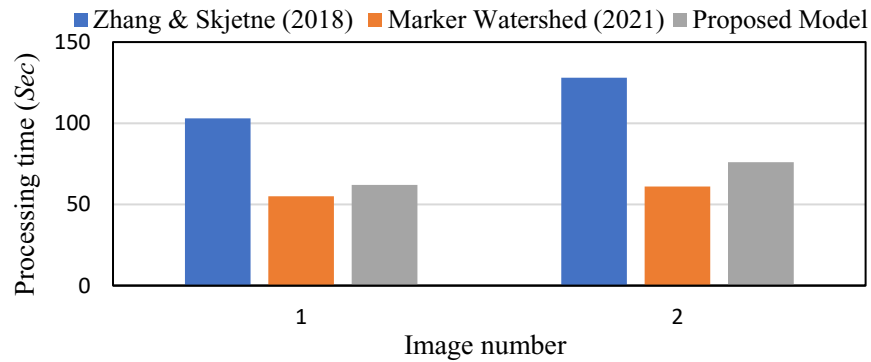


Figure 3-17: Processing time comparison among various models after small floes removal.

3.4 Conclusion

Detection of ice floes from images is challenging because of their complex and varied shapes, colour similarities and reflection of light on them. Besides, real ice floes are often found in groups with overlapped and/or connected boundaries, making detecting even more challenging due to

weaker edges in such situations. This paper presents the development and implementation of an improved ice segmentation approach, and the key features are summarized below:

- Wavelet filtering and histogram equalization are used to enhance the input images through denoising and contrast adjustment.
- Image segmentation is carried out using an efficient model developed by combining GVF, snake algorithm and distance transformation. Various morphological filtering and region analysis techniques are then applied to extract floe features.
- The capability of the proposed model to separate complex floes with weaker and connected boundaries in noisy and nonuniform illumination environments more efficiently compared to existing models in the literature is demonstrated.
- Overall, the model can detect the total number of floes with more than 85% accuracy and ice concentration at 95% and above accuracy.
- It is nearly 50% faster compared to the previous model. It can be used for real-time ice floe detection due to its faster processing time with the expense of accuracy to a certain acceptable limit, as described in section 2.3.
- However, the current model manually sets some parameters related to ice properties. These parameters could be automatically tuned as the texture and size of ice fields varies significantly. Thus, there exists room for improvement, especially in terms of automation.

3.5 Acknowledgements

The authors acknowledge the financial support from the National Research Council (NRC) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

4.0 Image based Ice load prediction on ship in a managed ice field using machine learning

Shamima Akter¹, Syed Imtiaz², Salim Ahmed², Mohammed Islam³, Robert Gash³, Hasanat Zaman³

¹Department of Mechanical Engineering, Memorial University, St. John's, NL, Canada.

²Department of Process Engineering, Memorial University, St. John's, NL, Canada.

³National Research Council, Canada, St. John's, NL, Canada

Abstract

The navigable ice for a longer summer period, coupled with technological advancements, is making the Northwest passage an attractive, commercially viable shipping route. Increasing commercial and scientific activities in this ice-infested region are driving the research in the precise estimation of ice forces on the ship and other offshore structures. Two Hybrid models for force prediction are proposed. The first model extracts ice features from images using traditional image processing techniques and then uses SVM and FFNN to develop two separate force predictors. The second model, on the other hand, extracts ice features from images using RCNN, and then trains two separate force predictors using SVM and FFNN. The performance of both Hybrid models is demonstrated using experimental data collected from a large state-of-the-art wave tank.

Keywords: Machine learning; Transfer learning; Region Based Convolutional Neural Networks (RCNN); Support Vector Machine (SVM); Feed-Forward Neural Network (FFNN)

4.1 Introduction

Explorers in the mid-19th century attempted to map the Northwest passage as a shortcut between North Pacific and North Atlantic by navigating the Arctic Ocean. At that time, in addition to the

lack of technology, the route was mostly blocked by impassable ice, even in the summer; thus, leading to the loss of many great explorers and resources. Nearly one and a half centuries later, today, the arctic route is increasingly accessible for a few months during summer due to the melting of ice in the Arctic. This navigable ice, coupled with the advancement in technology, is making the Northwest passage a commercially viable shipping route. However, the ship and offshore platforms operating in the Arctic region should be designed to have efficient performance in ice. So, with the increase in commercial activities, navigation and scientific investigation in polar regions, more attention is being paid to the structural design and maneuvering performance of ships in ice-covered waters. The determination of real-time ice loads on a ship hull, in this regard, is essential to analyze the ice-structure interaction and design appropriate structures against forces from ice.

Over the last few decades, understanding and modelling of ice forces on fixed, moored floating, DP-controlled platforms/vessels; and on icebreaking or slowly maneuvering ships in managed or unmanaged broken ice fields have been attempted on multiple frontiers. For example, classical analytical formulation of single, large ice sheet load on fixed structure [59], analytical formulation of ice floes interaction with floater [60], empirical-statistical modelling of vessels and managed ice field interactions [61]. Physical model testing of vessels in managed ice [62], or platforms in shallow waters [63]. Ice force modelling on conical structures based on long-term field test data [64]. Numerical modelling, such as Computational cohesive element model ([65], [66]) finite element model ([67], [68], [69]), Discrete element modelling ([70], [71], [5], [72]), Smoothed Particle Hydrodynamics [73], Collision-Energy-Based Method [74], Particle-in-cell method [11], GPU event mechanics method [11]. Hybrid modelling, for example, model test and subsequent simulations of ice impacts on ship hulls in broken ice fields [75].

The analytical, empirical, and statistical methods often show high numerical efficiency and ease of integration; however, they do not accurately model the relevant physical processes. Hence these methods should not be considered where closeness to physical processes is the desired objective. Computational methods require high computation resources, and calculation time is high, which is unsuitable for real-time simulations. In order to compensate for the deficiencies of a single technique, often empirical and computational methods are combined to achieve the goal. Regardless of the methods adopted, validations with quality measurements are paramount to the success of the ice-structure interaction models and, the lack of high-quality physical model tests and full-scale measurements for a thorough validation adversely impacts confidence in the modelling [11]. Also, almost all these models lack versatility, and are usually valid for certain ice conditions and certain vessels.

The applications of machine learning and deep learning techniques for ship performance predictions have been pursued by many researchers for ice characterization and ice-ship interaction analysis. Recently, [76] implemented Artificial Intelligence (AI) based Machine Learning (ML) and Deep Learning (DL) models to predict ship performance characteristics based on time-averaged and time-dependent data and prediction of forces on a dynamic positioning ship operating in a broken ice field. One modelling case involved developing an ML algorithm to predict time-averaged ice forces on DP-controlled ships at the given ranges of ice concentration, floe size, ice thickness, strength, density, drift speeds and direction. The other modelling case involved predicting the time-dependent forces on a DP-controlled ship at specific operating conditions and ice-field parameters. The ML-based predictive models showed reasonable prediction accuracy and performed better than conventional regression-based models. In [77] researchers reported a data-driven prediction model based on Artificial Neural Networks to estimate the ice resistance on ships

in level ice fields, where the predictor was trained by various parameters, including ship geometries and test conditions. Later, [78] proposed an ML-based method to predict ice resistance on polar ships. Their methods include three ANN models, which are validated with full-scale and model-scale measurements. However, all these models involved the use of data extracted at a particular time from the field, model test or numerical analysis. They did not consider how the parameters will vary in that exact location with the changes of time; that is, real-time processing of ice images was not done to extract those datasets.

Apart from onboard camera captured images, using areal imagery and applying digital image processing techniques is also an efficient way to extract real-time ice characteristics on the macro level in the oceans. Although with the advancement of computer visions, several advanced algorithms, for example, Convolutional Neural Networks (CNN), and Regions with Convolutional Neural Networks (R-CNN), You Only Look Once (YOLO) are developed and extensively applied in various fields for image analysis, the application of these advanced tools for ice image processing is still at its infancy. Among the existing works, [4], [85] proposed image processing models to identify and characterize sea ice floes. [14] introduced another open-source algorithm for detecting sea ice surface features using high-resolution optical imagery. SVM was used by [15] to detect pancake ice and compute their size distribution. However, none of these works used this extracted ice floe information to estimate ice forces on structures.

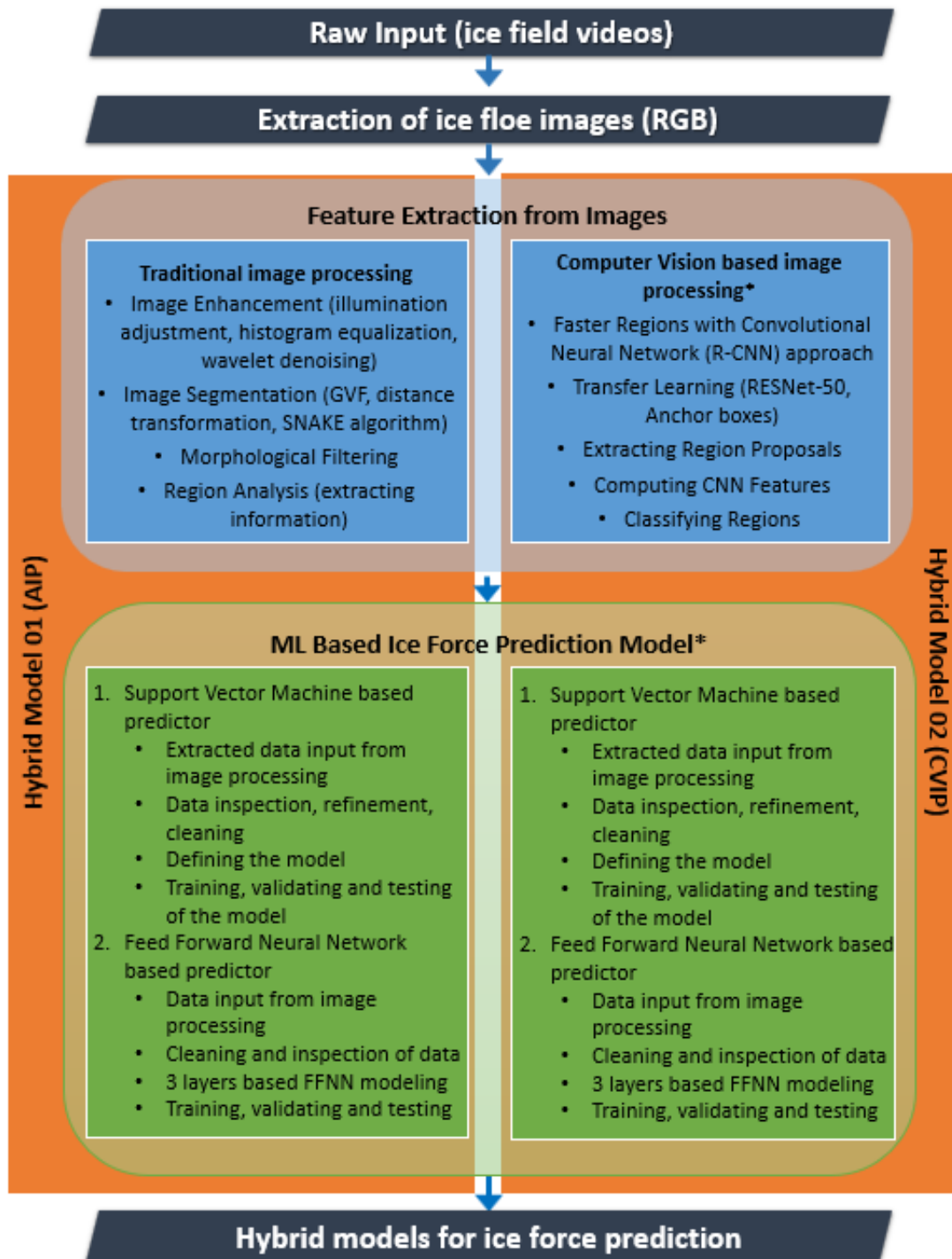
The primary objective of this research work, therefore, is to develop a tool which will extract the ice floe characteristics from experimental/field videos/images and use the information for real-time ice force prediction. Two Hybrid approaches are developed in this study: (1) extraction of ice floe characteristics using traditional edge detection-based image processing tool (as presented in chapter 2) and feed the extracted information to develop a Feed-Forward Neural Network (FFNN)

and a Support vector machine (SVM) based predictors, which will be able to estimate ice force on ships in ice-covered water; (2) use of ML based RCNN to train an image processing detector which will extract the ice floe information directly from the videos from ice tank test and field images; then, feed the extracted information to FFNN and SVM, as described in approach 1. Ice tank test images from NRC [62] were used to develop these Hybrid tools. The models are developed on the MATLAB platform.

This paper is structured as follows: after the introduction in Section 4.1, Section 4.2 presents the methodology and techniques used for ice load prediction models, including an introduction to the RCNN, SVM and FFNN models. Section 4.3 reports the results obtained from the RCNN model for image feature extraction. Following that, force prediction results were obtained from the SVM and FFNN models and comparisons of the models' performances. Finally, Section 4.4 gives some concluding remarks, limitations of the model and scopes for future work.

4.2 Methodology

A flowchart outlining the key steps of the Hybrid models developed in this study is shown in Figure 4-1. The key tasks can be grouped under two main categories – feature extraction from images, and development of subsequent force prediction based on the extracted features. The Analytical Image Processing (AIP) Based first Hybrid model extracts the image features using traditional image processing tools as described in Chapter 03. The Computer Vision Based Image Processing (CVIP), 2nd Hybrid model uses ML based R-CNN approach to extract ice floe information from the images. After that, both models use SVM and FFNN based force predictors for ice load estimation.



* Development of all the ML based predictors includes data processing, training, validating and testing of the models.

Figure 4-1: Flow chart for the Hybrid ice force prediction models development

The ship-ice interaction is a complex phenomenon. Therefore, similar to the traditional analytical and numerical approaches, the accuracy and reliability of the image-based modelling method depends on how accurately the most relevant ice-structure interaction information are extracted, and then formulated through the prediction model. The following subsections describe the methodology applied to achieve force estimation from the input images. Methodology for extracting image features using traditional image segmentation techniques is excluded, as those are described in Chapter 03.

4.2.1 Image extraction from the input videos

This study uses a set of videos captured during the experimental analysis performed at NRC-OCRE to evaluate the effects of various managed ice-field characteristics on the thruster forces [62]. Various video sensors are used to acquire a dataset of ice-field and vessel interactions in various managed ice conditions. Ice-1A classed anchor handling tug supply (AHTS) vessel called the Magne Viking, equipped with full-scale representative propulsion arrangement, was used for the model testing programs which is shown in figure 4-2 [61].



Figure 4-2: NRC-OCRE ice basin images [61]

Table 4-1: Test matrix for the Magne Viking model test in the ice basin

Label for the video	Thickness (<i>m</i>)	Floe Size (<i>m</i>)	Concentration (<i>ice/water ratio</i>)	Drift Speed (<i>knots</i>)	Drift angle (<i>deg</i>)
12p5m_7ths_0p5kts_0p6m_0deg_001	0.6	12.5	7/10 th	0.5	0°
12p5m_9ths_0p5kts_0p6m_0deg_001	0.6	12.5	9/10 th	0.5	0°
12p5m_9ths_1p2kts_0p6m_0deg_001	0.6	12.5	9/10 th	1.2	0°
12p5m_9ths_1p2kts_1m_0deg_001	1	12.5	9/10 th	1.2	0°
25m_7ths_0p5kts_0p6m_0deg_001	0.6	25.0	7/10 th	0.5	0°
25m_7ths_0p5kts_0p6m_10deg_001	0.6	25.0	7/10 th	0.5	10°
25m_7ths_1p2kts_0p6m_0deg_001	0.6	25.0	7/10 th	1.2	0°
25m_7ths_1p2kts_0p6m_25deg_001	0.6	25.0	7/10 th	1.2	25°
25m_8ths_1p2kts_0p6m_0deg_001	0.6	25.0	8/10 th	1.2	0°
25m_9ths_0p2kts_0p6m_0deg_001	0.6	25.0	9/10 th	0.2	0°
25m_9ths_0p2kts_1m_0deg_001	1	25.0	9/10 th	0.2	0°
25m_9ths_0p5kts_0p6m_0deg_001	0.6	25.0	9/10 th	0.5	0°
25m_9ths_0p5kts_0p6m_0deg_002	0.6	25.0	9/10 th	0.5	0°
25m_9ths_0p5kts_1m_0deg_001	1	25.0	9/10 th	0.5	0°
25m_9ths_0p5kts_1m_10deg_001	1	25.0	9/10 th	0.5	10°
25m_9ths_1p2kts_0p6m_0deg_001	0.6	25.0	9/10 th	1.2	0°
50m_7ths_0p5kts_0p6m_0deg_001	0.6	50.0	7/10 th	0.5	0°
50m_7ths_1p2kts_0p6m_0deg_001	0.6	50.0	7/10 th	1.2	0°
50m_8ths_0p5kts_0p8m_10deg_001	0.8	50.0	8/10 th	0.5	10°
50m_9ths_0p2kts_0p4m_0deg_001	0.4	50.0	9/10 th	0.2	0°
50m_9ths_0p2kts_1m_10deg_001	1	50.0	9/10 th	0.2	10°
50m_9ths_0p5kts_0p4m_30deg_002	0.4	50.0	9/10 th	0.5	30°
50m_9ths_0p5kts_0p8m_0deg_002	0.8	50.0	9/10 th	0.5	0°
50m_9ths_0p5kts_1m_0deg_001	1	50.0	9/10 th	0.5	0°
50m_9ths_1p2kts_0p6m_0deg_001	0.6	50.0	9/10 th	1.2	0°
50m_9ths_1p2kts_1m_10deg_001	1	50.0	9/10 th	1.2	10°
50m_100m_9ths_1p2kts_0p4m_0deg001	0.4	50/100	9/10 th	1.2	0°
100m_9ths_1p0kts_0p4m_0deg_002	0.4	100.0	9/10 th	1.0	0°
100m_dist_9ths_1p2kts_0p4m_0deg_001	0.4	100.0	9/10 th	1.2	0°

Tests were performed for various parametric ranges, including variables such as, ice thickness, floe size, ice concentration, model ship drift speed and angle. Among the videos for all the test

cases, 29 videos were used for this study. Videos are labelled in a *'floe size_concentration_drift speed_thickness_drift_angle_test run number'* format, as can be seen in Table 4-1.

A multimedia reader object is then created in MATLAB to read and extract frames from these videos. Frames are extracted at an interval of 1 second. Figure 4-3 shows three random frames extracted from three experimental videos with 25, 50 and 100 floe sizes.

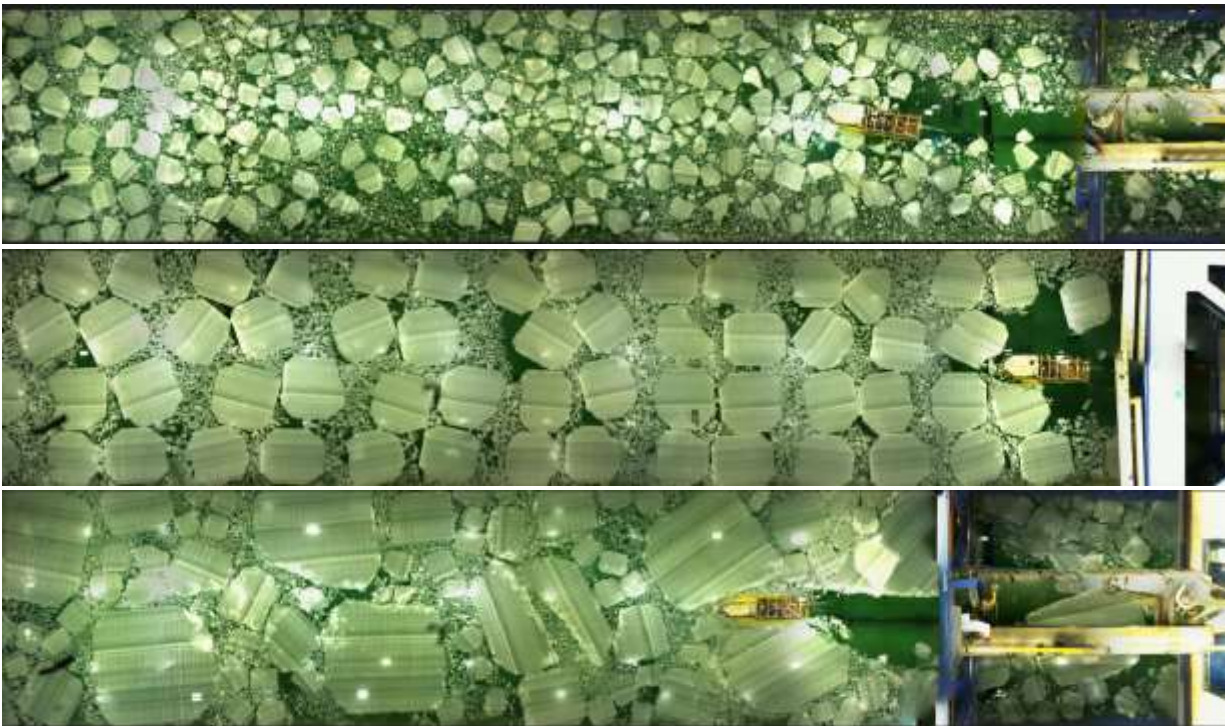


Figure 4-3: Extracted frame from three different videos (25m, 50m and 100m floe size)

The next steps are detecting and masking the ship, followed by ice floe detection and feature extraction.

4.2.2 Image processing tools

Faster R-CNN, or Region-based Convolutional Neural Network, is a deep learning object detection framework that uses a convolutional neural network (CNN) for detection. The key concept behind

the R-CNN series is region proposals, which are used to localize objects within an image. Then, the information is passed through a deep neural network which is trained for image classification using annotated training images. Essentially, RCNN combines rectangular region proposals with convolutional neural network features. R-CNN is a two-stage detection algorithm. The first stage identifies a subset of regions in an image that might contain an object. The second stage classifies the object in each region. As can be seen in Figure 4-4, given an input image (Step 1), all possible region proposals or regions of interest (ROI) are extracted using an algorithm like Edge Boxes (Step 2). Then, the algorithm resizes (wrap) all the extracted crops and pass them through the trained network for detection (Step 3). The classifier portion of the network then takes all those ROIs as input and gives a label and confidence to each ROI as an output (Step 4).

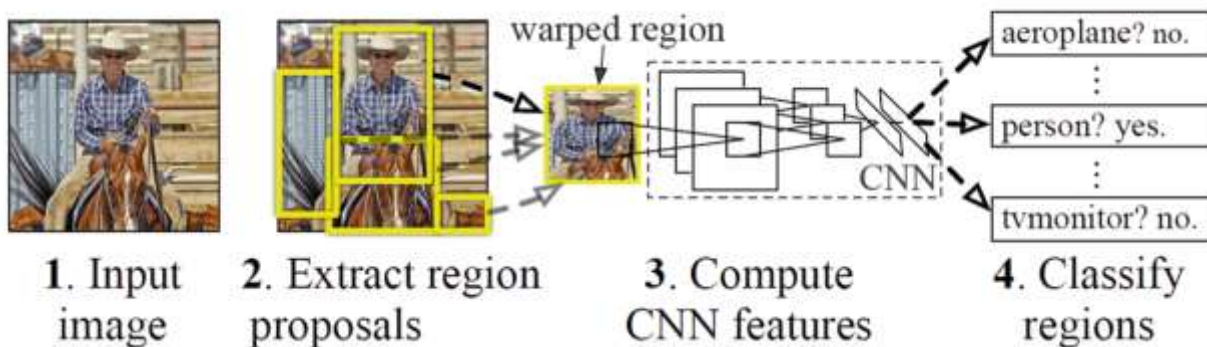


Figure 4-4: Region based Convolutional Neural Network concept [86]

RCNN provides quite reasonable results, but it is a computationally expensive process as the Neural Network has to be evaluated for each ROI. The faster RCNN [87], tackles this problem by adding a region proposal network (RPN) to generate ROIs directly in the network instead of using an external algorithm like the Edge Boxes. The RPN uses Anchor Boxes (as described later) for object detection, and generating ROI inside the network makes it faster compared to the usual RCNN. Faster RCNN has achieved great success for generic object detection and has been

successfully used in various applications, including car detection, facial recognition, animal identification ([88], [89], [90], [91]). Another benefit is that the faster RCNN is composed of a feature extraction network followed by two subnetworks. The feature extraction network is usually a pre-trained network, as shown in Figure 4-5. For this study, the ‘Resnet50’ is used as the feature extraction network. ResNet-50 is a deep residual neural network that has been trained on a large dataset called ImageNet for image classification tasks. This pretrained model is used as a starting point to train a new model for a different image classification task by "transferring" the knowledge it has learned from the ImageNet dataset to the new task. This is done by modifying the last layer of the pre-trained model to fit the new task, and then training this layer using the new dataset. The rest of the network weights are kept fixed, since they have already learned meaningful features from the large dataset. This is much faster and requires much less data compared to training the entire network from scratch. Transfer learning in the present study is applied by configuring only the last four layers to train this ‘Resnet50’ network to perform the new recognition task of detecting the ship and ice floes, using the ice floe images acquired from the experimental videos.

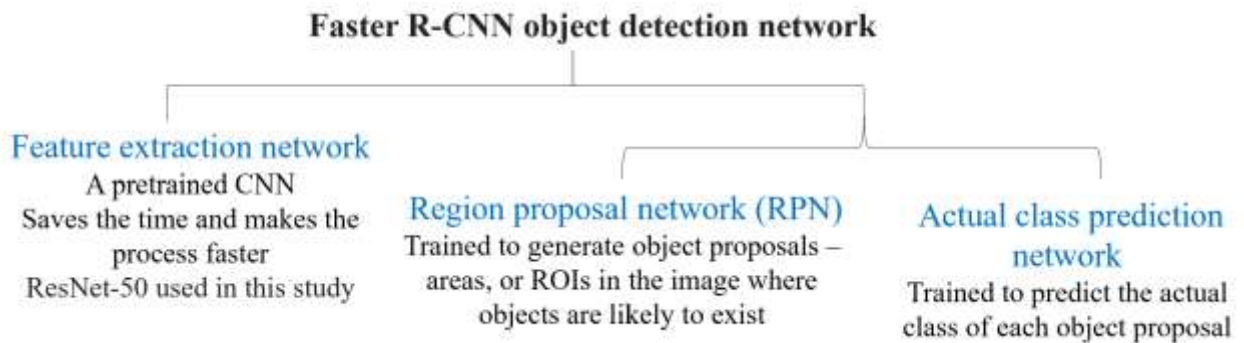


Figure 4-5: Faster RCNN object detection network

The equation for the residual block in ResNet-50 is:

$$y = F(x, \{W_i\}) + x \tag{4-1}$$

where \mathbf{x} is the input to the residual block, \mathbf{F} is the non-linear mapping learned by the block, and $\{\mathbf{W}_i\}$ are the parameters of the block. \mathbf{y} is the output of the block, and \mathbf{x} is added back to the output to ensure that the residual block learns a residual mapping.

A loss function is used to train the final layer is the cross-entropy loss between the predicted probabilities and the true labels. The equation for the loss function in transfer learning is:

$$L = -\sum_{i=1}^{\mathbf{N}} y_i \log(\hat{y}_i) \quad (4-2)$$

where \mathbf{N} is the number of samples in the dataset, y_i is the true label for the i -th sample, and \hat{y}_i is the predicted probability for the i -th sample. The loss function is used to update the parameters of the final layer in order to minimize the error between the predicted and true labels.

The first subnetwork following this feature extraction network is a region proposal network (RPN), which is trained to generate object proposals or ROI, that means areas in the image where objects are likely to exist. The second subnetwork is trained to predict the actual class of each object proposal, which are ship and ice floes in this study.

The following three parameters are defined at first to initialize the network:

The network input size: To train this network, it is essential to standardize the sizes of all the input images to avoid scaling issues in the detection results. The minimum size required for this network is [224 224 3], where 224 by 224 is the image pixel size in y and x , and 3 refers to the three-colour channels, RGB. The computational cost incurred by processing data increases as the size increases. Nevertheless, for this study, the actual resolution of the images from the video output is used to achieve a better result for ship and ice piece detection. [1134 6096 3] is used when the network is trained for ship detection, and [1134 1024 3] is used for ice floe detection training (as the input images for ice floe detection are cropped along the x direction to reduce the size). Although full-size images took longer for training, the detection accuracy improved significantly.

Anchor boxes: Anchor boxes are a set of predefined bounding boxes of a certain height and width. These boxes are defined to capture the scale and aspect ratio of specific object classes which wants to detect and are typically chosen based on object sizes in the training datasets. During detection, the predefined anchor boxes are tiled across the image. The network predicts the probability and other attributes, such as background, intersection over union (IoU) and offsets for every tiled anchor box. The predictions are used to refine each individual anchor box. Several anchor boxes can be defined, each for a different object size. The network does not directly predict bounding boxes, but rather predicts the probabilities and refinements that correspond to the tiled anchor boxes. The use of anchor boxes enables a network to detect multiple objects, objects of different scales, and overlapping objects. The embedded region proposal network generates the size of the anchor boxes based on the input training images. This study used three anchor boxes with the following sizes: 140 498, 140 458, and 159 496. The position of an anchor box is determined by mapping the location of the network output back to the input image. The process is replicated for every network output. The result produces a set of tiled anchor boxes across the entire image. Each anchor box represents a specific prediction of a class. For example, there are two anchor boxes to make two predictions per location in the image, as shown in Figure 4-6 [18].

Feature extraction network: As described earlier in this section, a pre-trained CNN, 'Resnet50,' is used as the feature extraction network. The 'activation_40_relu' is used as the feature extraction layer for configuring and re-training through the transfer learning process. This feature extraction layer outputs feature maps that are down-sampled by a factor of 16. This amount of downsampling offers a good trade-off between spatial resolution and the strength of the extracted features, as features extracted further down the network encode stronger image features at the cost of spatial resolution.



Figure 4-6: How the anchor box works in the faster RCNN object detector

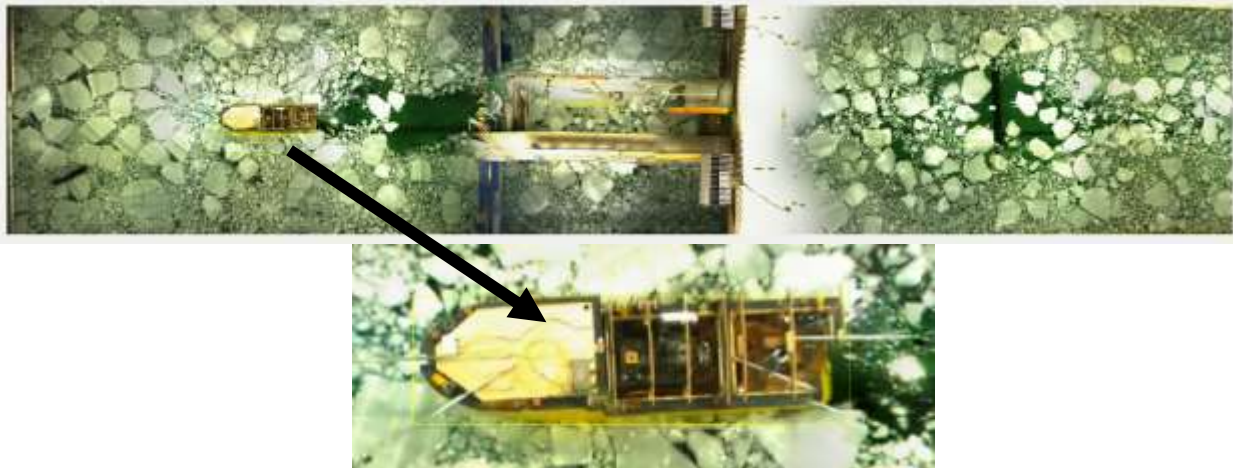
4.2.3 Preparing data (images) for training, validation, and testing

Full-size images are used for the ship detection network. Once ships are detected and masked, the images are cropped using the midship as the center point and keeping 2 ship lengths at the front and 0.5 ship lengths at the back of the ship. Experimental analysis [62] showed that ice floes beyond this limit have no significant effect on the ice force exerted on the ship. The same image sizes are also used in Chapter 03, thus, ensuring a valid comparison of force prediction between traditionally processed images and images processed through ML.

Data for ship detection network: The video frames extracted as images in section 4.2.1 are prepared for training through labelling the ship in all the images using the MATLAB image labeller function. The ship data is then stored in a two-column table, where the first column contains the image file paths and the second column contains the ship bounding boxes, as shown in Figure 4-7(a). It will be an input for training the detector network. The bounding boxes are also shown in one of the randomly selected images in Figure 4-7(b). A total of 107 images were used. 60% of the images were used for training, 10% for validation, and the rest for testing the trained detector.

	VARIABLE	SELECTION	EDIT
ShipLabel			
107x2 table			
	1	2	
	imageFilename	Ship	
1	'shipimage\Image0001.jpg'	[4005,496,511,128]	
2	'shipimage\Image0011.jpg'	[4018,492,494,136]	
3	'shipimage\Image0021.jpg'	[4016,495,488,133]	
4	'shipimage\Image0031.jpg'	[4010,495,490,132]	
5	'shipimage\Image0041.jpg'	[3994,486,462,138]	
6	'shipimage\Image0051.jpg'	[3971,479,465,141]	
7	'shipimage\Image0061.jpg'	[3942,471,457,140]	
8	'shipimage\Image0071.jpg'	[3910,481,459,130]	
9	'shipimage\Image0081.jpg'	[3897,474,443,141]	
10	'shipimage\Image0091.jpg'	[3871,475,442,134]	
11	'shipimage\Image0101.jpg'	[3834,478,448,133]	
12	'shipimage\Image0111.jpg'	[3800,481,453,129]	
13	'shipimage\Image0121.jpg'	[3777,467,447,143]	
14	'shipimage\Image0131.jpg'	[3748,480,442,133]	
15	'shipimage\Image0141.jpg'	[3701,481,452,134]	
16	'shipimage\Image0151.jpg'	[3674,483,453,137]	

(a)



(b)

Figure 4-7: Data preparation for ship detector training (a) Input table for the network, (b) Ship labelled in one training image

While training, the network will use data augmentation to improve the accuracy by generating more variety in the training data through randomly transforming the original data without having to increase the number of labeled training samples. Transformation is used to augment the training

data by randomly flipping the image and associated box labels horizontally. The equation for horizontal flipping is:

$$x'_{i,j} = x_{\{n-i,j\}} \quad (4-3)$$

where \mathbf{x} is the original image, \mathbf{x}' is the flipped image, \mathbf{n} is the number of rows in the image, and \mathbf{i}, \mathbf{j} are the indices of the pixel in the image.

However, data augmentation is not applied to test and validation data. Ideally, test and validation data are representative of the original data and are left unmodified for unbiased evaluation. One data augmentation sample is illustrated in Figure 4-8, where three augmented images are generated from one training sample.

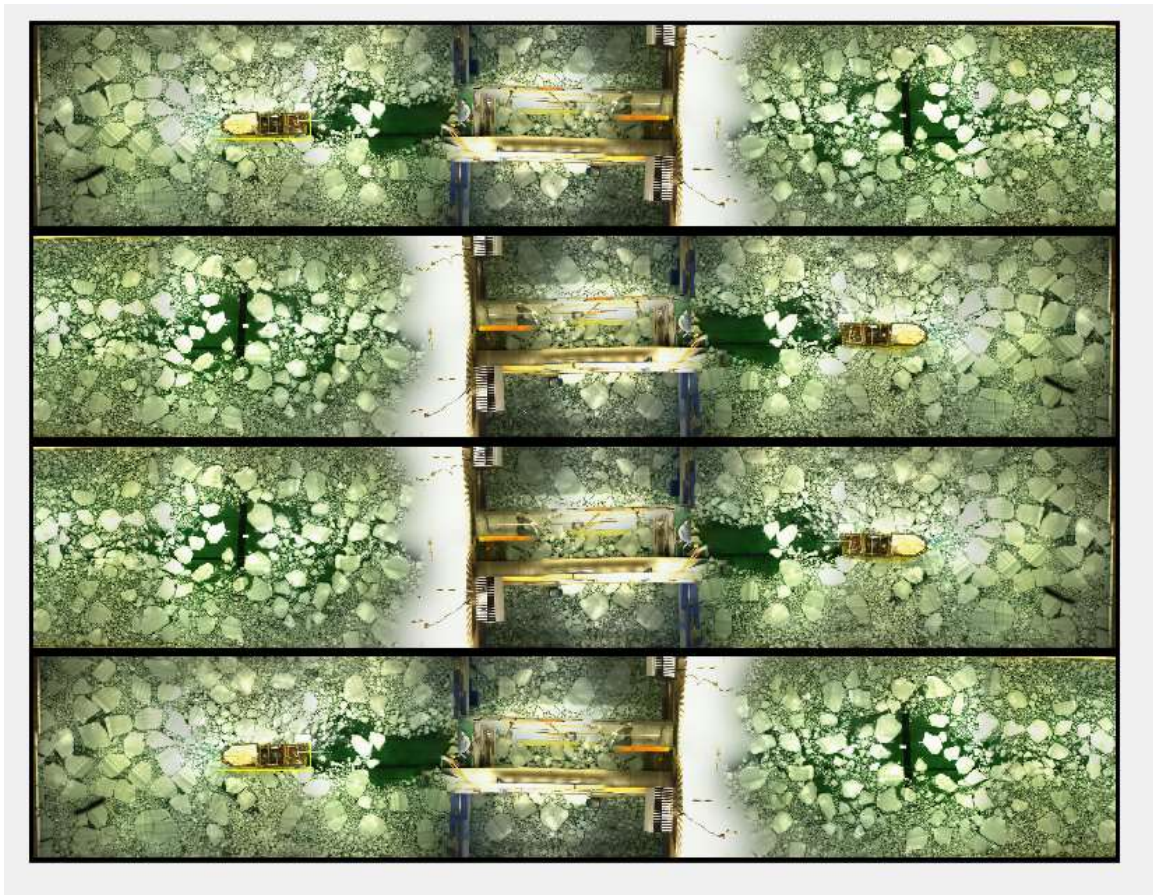


Figure 4-8: Data augmentation outcome

Data for ice floe detection network: After the detection of the ship in the images, it is now cropped manually (using the midship as the center point and keeping 2 ship length at the front and 0.5 ship length at the back of the ship) to ensure the proper input size for the ice floe detector training. The dimensions of the cropped images ranged from 1020-1100 pixels for width, and 1120-1140 pixels for height. However, as mentioned earlier, the network will resize the images to default [1134 1024] size to ensure uniform measurement of the floe sizes while extracting the features. Followed by this, a polygonal mask, with a fill colour similar to the deep blue water background colour in the training images, is applied to absorb the ship in the background. It will ensure better efficiency of the ice floe detector. Figure 4-9 illustrates three cropped images with different floe sizes, and another three images after the ships are masked.

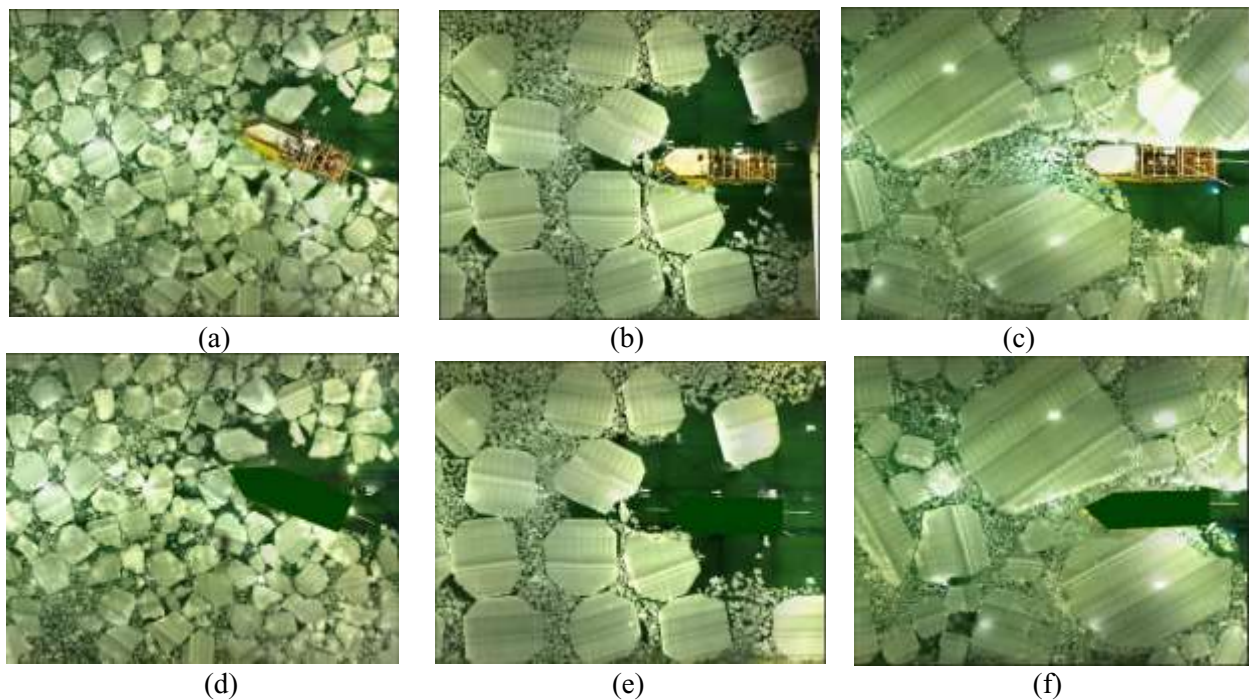


Figure 4-9: Cropped images for floe size (a) 25, (b) 50, (c) 100. Final input images after ship masking for floe size (d) 25, (e) 50, (f) 100

Once the ships are masked, the images are imported to the image labeller for labelling the ice floes, similar to the step performed for image preparation for ship detection. Next, the data are stored in a two columns table, which is later used for training, validation and testing of the detector. The detection of ice floes in the image is challenging as floe size, shapes, and illuminations vary significantly. There is no fixed/ nearly the same aspect ratio for the ice floe shapes, compared to other objects, for example, human shape, faces, cars etc. In addition to this, faster RCNN uses rectangular bounding boxes, as shown in Figure 4-10. Due to this, it was difficult to label individual ice floes as completely separated objects without overlapping with adjacent floes or missing some portion of them. This becomes more difficult when floe sizes are smaller, as seen in Figure 4-10.

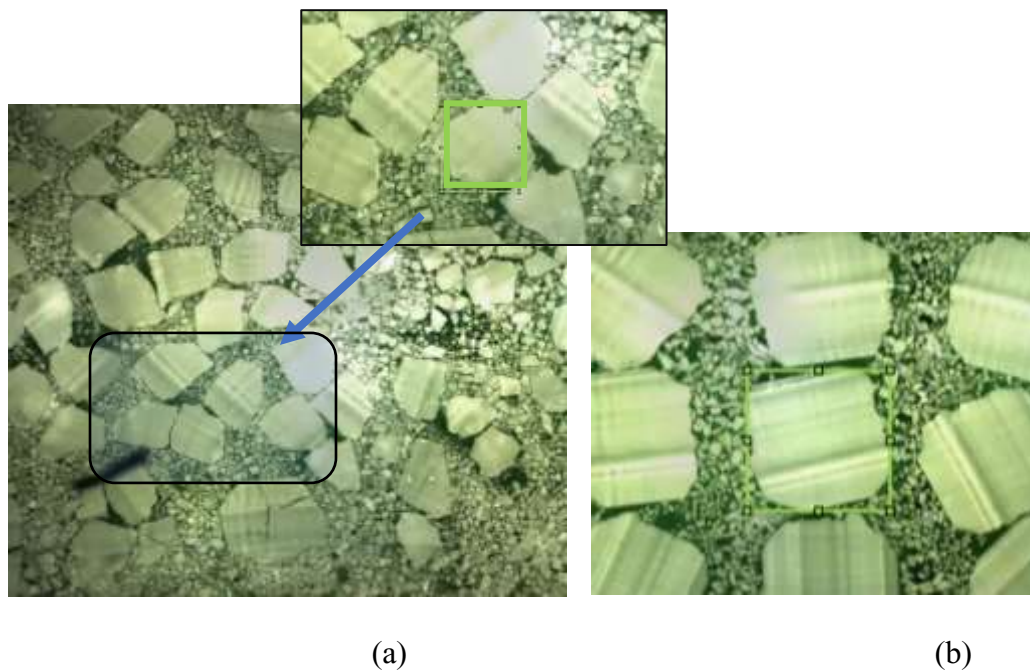


Figure 4-10: Ice floe pattern in the image: (a) ice floe labelling in small floe regions (b) labelling large floes

In this study, 520 images were used to train the ice floe detector. Among these, 60% of images were used for training, 20% for validation and 20% for testing. The validated and tested network is then used for ice floe detection in new images, i.e., images which were not used for training, validation, or testing.

4.2.4 ML-based ice load prediction model

Support Vector Machine (SVM) and Feed-Forward Neural Network (FFNN) were used to develop the ice load prediction models.

4.2.4.1 Support Vector Machine (SVM) regression model

Support Vector Machine (SVM) is a supervised machine learning algorithm. Depending on the objective, it can be used as a classification (discrete target variable) or a regression (continuous target variable) model. Prediction is made with a mapping function which maps independent variables to the dependent variable. The mapping function for SVM is a decision boundary which makes the distinction between two or more classes [92]. The SVM algorithm performs a classification by constructing a multidimensional hyperplane that optimally discriminates between two classes by maximizing the margin between two data clusters. This algorithm achieves high discriminative power by using special nonlinear functions called kernels to transform the input space into a multidimensional space [93].

The basic idea behind the SVM technique is to construct an $n-1$ dimensional separating hyperplane to discriminate two classes in an n -dimensional space. A data point is viewed as an n -dimensional vector. For example, two variables in a dataset will create a two-dimensional space; the separating hyperplane would be a straight line (one-dimensional) dividing the space in half. When more

dimensions are involved, SVM searches for an optimal separating hyperplane called the maximum-margin separating hyperplane. Kernel functions help separate the classes by adding more dimensions to the low-dimensional space so classes can be separable in the high-dimensional space.

The governing equation for an SVM classifier is:

$$f(x) = \text{sign}(w^T x + b) \quad (4-4)$$

where w is the weight vector that defines the normal direction of the hyperplane, x is the feature vector of a sample, b is the bias term, and **sign** is the sign function that returns 1 for positive values and -1 for negative values. The feature vector x is replaced by the output of the kernel function, when it is used for non-linear SVM.

The Radial Basis Function (RBF) kernel, as shown in Eq. 4-1 is used for the SVM model used in this study. It maps the original feature space into a higher-dimensional feature space where a linear hyperplane can separate the classes. The RBF kernel is defined as:

$$K(x, x') = e^{(-\gamma * \|x - x'\|^2)} \quad (4-5)$$

where $K(x, x')$ is the kernel function that calculates the similarity between two samples x and x' , $\|x - x'\|^2$ is the squared Euclidean distance between the samples, and γ is a parameter that controls the shape of the RBF. With the RBF, the quadratic objective functions with linear constraints solver from the MATLAB optimization toolbox is used as the optimization routine [18].

4.2.4.2 Feed Forward Neural Network (FFNN) model

Feedforward Neural Network consists of a series of layers. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer

produces the network's output. The purpose of feedforward neural networks is to approximate function and can be used for any kind of input to output mapping [94]. It has been successfully applied to pattern classification, clustering, regression, association, optimization, control, and forecasting problems.

Figure 4-11 shows an example of a multilayer feedforward Neural Network with 8 input variables. As mentioned above, it is a directed acyclic graph which means that there are no feedback connections or loops in the network. It has an input layer, an output layer, and one or more hidden layers. Each node in the layer is a Neuron, which can be thought of as the basic processing unit of a Neural Network. Neurons in the input layer receive input signals to the network. Neurons in each hidden layer may have connections to and hence receive signals from some or all neurons from the immediately preceding layer. A feedforward network applies a series of functions to the input. By having multiple hidden layers, it can compute complex functions by cascading simpler functions. Neurons in the output layer provide output signals computed by the network to the environment external to the network. This is the layer which gives out the predictions.

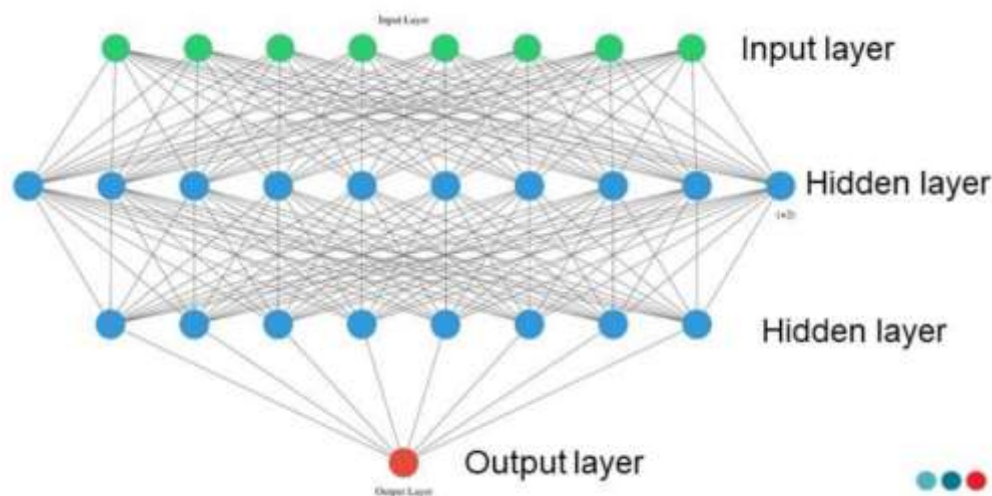


Figure 4-11: Example of a Feedforward Neural Network with two hidden layers

While initiating the training, an activation function is used in the output layer as a decision-making body at the output of a neuron. The activation function used in this layer is different for different problems. The neuron learns Linear or Non-linear decision boundaries based on the activation function. It also has a normalizing effect on the neuron output, which prevents the output of neurons after several layers once become very large, due to the cascading effect. There are three most widely used activation functions: Sigmoid, Tanh, and Rectified Linear Unit (ReLU). The latter is used for this study as it is found to be effective for similar regression analysis [94]. The equation for a ReLU activation function is:

$$f(x) = \max(0, x) \quad (4-6)$$

where $f(x)$ is the output of the activation function and x is the input to the activation function. The ReLU activation function returns the input x if it is positive, and 0 if it is negative. ReLU is used for this feedforward neural networks for several reasons:

Simplicity: The ReLU activation function is simple to implement and computationally efficient, making it a popular choice for deep learning architectures.

Introducing non-linearity: ReLU activation functions introduce non-linearity into the network, allowing the network to learn complex relationships between inputs and outputs. This is important for many real-world applications, as the underlying relationship between inputs and outputs is often non-linear.

Solving the vanishing gradient problem: ReLU activation functions do not suffer from the vanishing gradient problem that can occur in other activation functions, such as sigmoid, where the gradient becomes very small for large input values. This allows the network to converge faster and learn more effectively.

4.3 Results and discussion

The training performance of the RCNN for ship and ice floe detection is discussed first in this section. Followed by that, the outcome of the SVM and FFNN based force predictors will be analyzed and compared for the proposed Hybrid models.

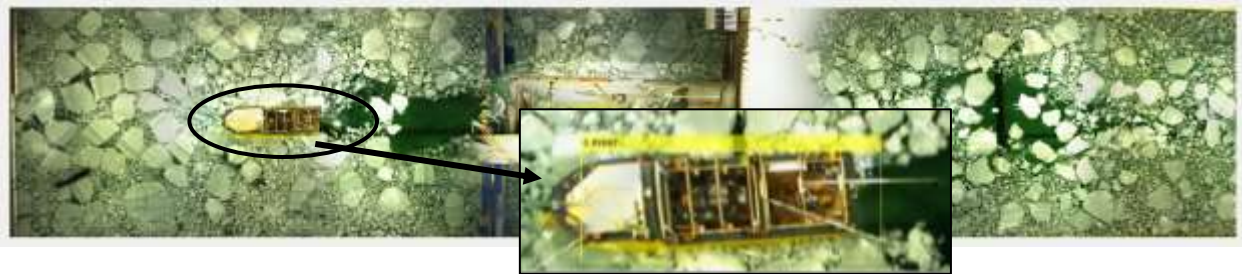
4.3.1 Faster RCNN training for Ship and ice floe detection

Ship detector training performance: The dataset of 107 images prepared in section 4.2 for ship detector training was provided as an input to the RCNN based detector. The training is run using 10 Epochs with a mini-batch size of 2. The negative and positive overlap ranges for detection were set to [0 0.3] and [0.6 1.0], respectively. A 10th generation core i7 CPU@2.30 GHz with 16 GB of ram was used for the training.

Figure 4-12(a) summarizes the training information. As can be seen, 10 Epochs took 320 iterations and nearly 28 hours to complete the training. The main reasons for high training time are the use of high-resolution training images and the use of PC with limited computational power. Nevertheless, 100% validation accuracy is achieved. Figure 4-12(b) shows the detection results for one test image with a detection score of 99.98%. The precision/recall (PR) curve for the training is also depicted in Figure 4-13. The ideal precision is 1 at all recall levels. The average precision provides a single number that incorporates the ability of the detector to make correct classifications, which is known as ‘precision.’ And the ability of the detector to find all relevant objects is known as ‘recall. As can be seen, a perfect precision score is achieved for all recalls in this training.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Loss	Mini-batch Accuracy	Mini-batch RMSE	RPN Mini-batch Accuracy	RPN Mini-batch RMSE	Validation Loss	Validation Accuracy	Validation RMSE	Base Learning Rate
1	1	00:15:42	2.6020	31.41%	0.10	58.43%	1.15	1.6323	99.67%	0.10	0.0010
2	50	00:54:09	0.8422	99.70%	0.08	98.05%	0.96	0.3954	99.96%	0.09	0.0010
4	100	02:40:20	0.4581	100.00%	0.10	99.22%	0.66	0.4099	100.00%	0.18	0.0010
5	150	04:34:01	0.1886	100.00%	0.05	99.61%	0.41	0.3087	99.99%	0.06	0.0010
7	200	08:14:22	0.2348	100.00%	0.09	100.00%	0.46	0.4660	100.00%	0.07	0.0010
8	250	13:39:01	0.1222	99.92%	0.05	99.22%	0.33	0.2540	100.00%	0.08	0.0010
10	300	22:22:17	0.1685	100.00%	0.07	99.61%	0.38	0.2748	100.00%	0.10	0.0010
10	320	27:42:55	0.0985	100.00%	0.06	99.61%	0.30	0.2239	100.00%	0.08	0.0010

(a)



(b)

Figure 4-12: Training and validation summary (a), Test image with accuracy bounding box (b).

Ice floe detector training performance: Similar to the ship detector training process, the ice floe dataset of 520 images was used as the RCNN input for the ice floe detector training. The network parameters are kept the same as ship detection training, except that the negative and positive overlap ranges are changed to $[0 \ 0.3]$ and $[0.4 \ 1.0]$, respectively. The training took around 32 hours to complete. As discussed in section 4.2.3, for ice floe labelling, the use of rectangular bounding boxes creates difficulty in processing, as it is challenging to isolate every ice floe without losing some part of it overlapping with neighbouring ice floes.

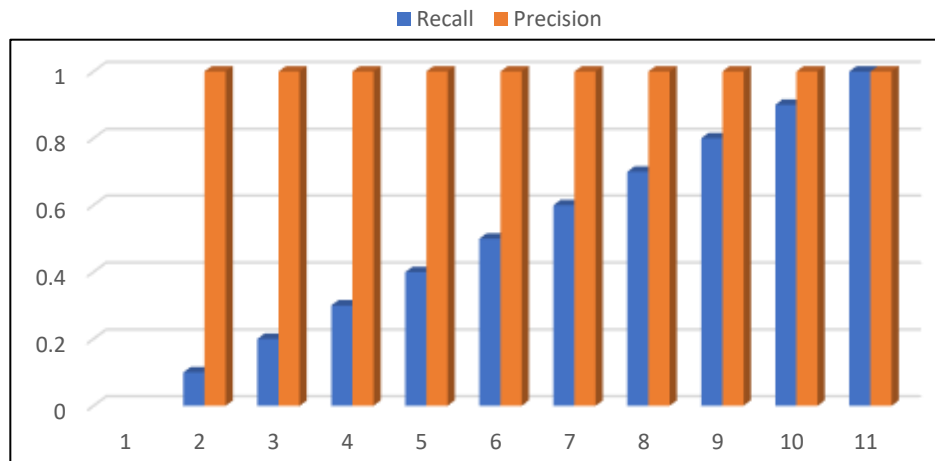


Figure 4-13: Precision-Recall curve (Average Precision 1.0) for ship detector training

As a result, the trained network managed to detect nearly all the ice pieces in ice regions where the floes are comparatively larger and uniform in size, or floes are reasonably apart from each other, as illustrated in Figure 4-14. As can be seen, nearly all floes are detected with an average detection score of 66%, except a few small floes are missed in images 01 and 03 (highlighted using blue circles). The reason for the lower detection score compared to ship detection is the presence of granular ice pieces, which are not considered for detection (the similar approach used in Chapter 03).

However, for small ice floe sizes with more random shapes and orientations, the trained detector performed rather poorly, as shown in Figure 4-15. In this case, using rectangular bounding boxes, it is difficult to label individual ice floes in complete isolation without overlying on the neighbouring floes. Also, the shapes, illumination and brightness varied quite significantly in these images. As a result, the detector missed a significant number of floes. Therefore, while extracting the floe information from the images for the force predictor training for Hybrid model 02, 75

images with comparatively better illumination and less variation in sizes are used. Images with comparatively bigger floes are used to ensure a more accurate force prediction model.

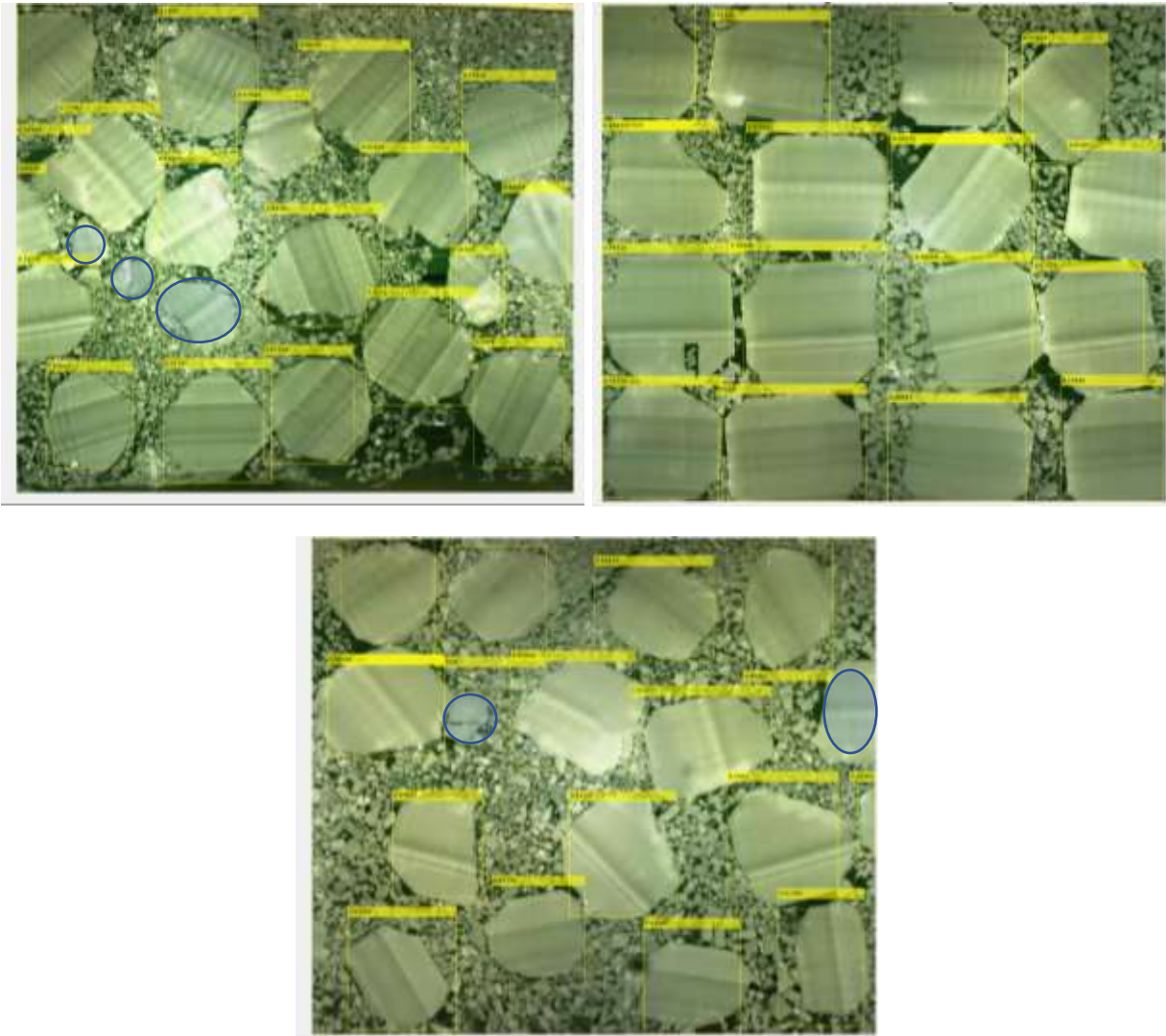


Figure 4-14: Ice floe detection accuracy in larger ice floe images

It should be noted here that the trained network for ship and ice floe detection works quite fast. It takes around 2 seconds to detect the ship using the trained detector. As for ice floe images, it takes only around 1 second to detect the floes and extract the information on number of floes and floe sizes.

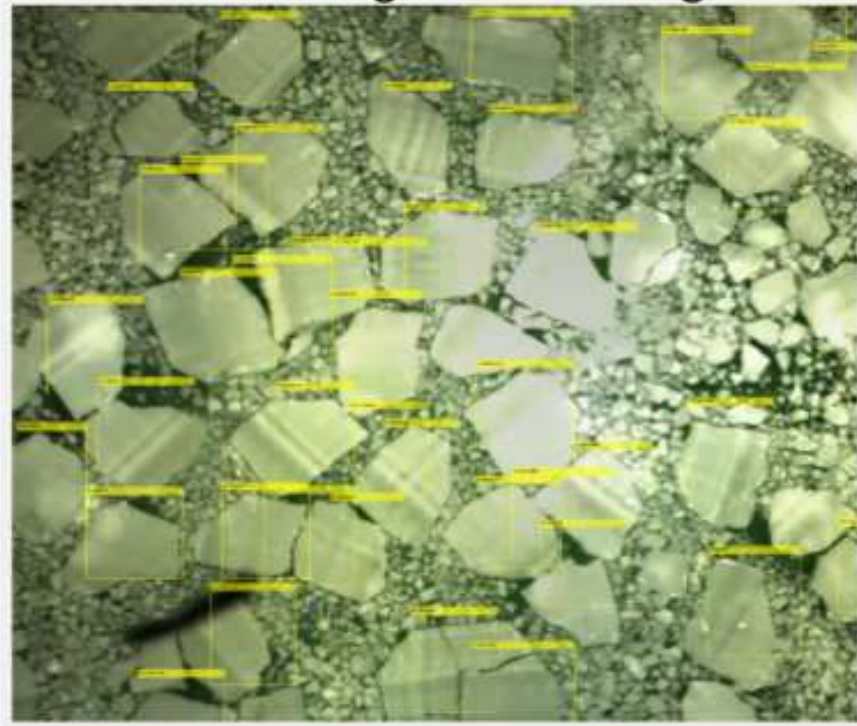


Figure 4-15: Ice floe detection accuracy in smaller ice floe images (2X the size of images in Figure 4-14)

4.3.2 Performance of the SVM and FFNN force predictors

The SVM and FFNN force predictors are developed for the Hybrid models. For Hybrid model 01, the input data related to ice floe characteristics for force predictors come from the traditional image processing, where unsupervised image pre-processing (including wavelet denoising), image segmentation using GVF and SNAKE are used instead of supervised machine learning approach. As for Hybrid model 02, force predictors input data related to ice floe characteristics are extracted from the ML-based RCNN predictor for ice floe images (as described in the previous section 4.3.1). More details on the development, analysis and performance comparisons of the force predictors are discussed below:

4.3.2.1 Force predictors for Hybrid model 01

The inputs for the Hybrid model 01 force predictor are extracted from images captured from the test videos at an interval of 20 seconds. A total of 627 images are used from 29 experimental videos. Figure 4-16 shows a snapshot of the input variables extracted from different frames of a video. Initially, 14 variables are proposed. The thickness, ship velocity and force are taken from the experimental records (highlighted in blue). The remaining data comes from image processing. N is the number of nonzero pixels in the image. R and C are the numbers of pixels in the x and y direction. Using N, R and C, the ice concentration is calculated as follows:

$$\text{Ice concentration} = \frac{N}{(R \times C)} \times 100 \quad (4-2)$$

The number of ice floes is extracted based on 10 size categories, as can be seen in Figure 4-16. Type 1, contains the number of floes between 0 to 500 pixel sizes, Type 5, for floe sizes between 30,000 to 40,000 pixels, and so on. The size categories are defined based on the maximum and minimum floe size in the images.

Frame	N	R	C	Concentration	Thickness	Ship Velocity	Floe size										Force
							0-5000	5000-10000	10000-20000	20000-30000	30000-40000	40000-50000	50000-60000	60000-70000	70000-80000	80000-90000	
							Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9	Type10	
101	1002164	1126	1223	72.774	0.6	0.000	5	12	2	0	0	0	0	0	0	0	0.368
121	1070009	1129	1285	73.755	0.6	0.056	6	9	2	1	0	0	0	0	0	0	31.092
141	1205442	1129	1414	75.510	0.6	0.059	7	15	1	1	0	0	0	0	0	0	4.647
161	1213286	1129	1418	75.787	0.6	0.059	6	15	2	0	0	0	0	0	0	0	1.313
181	1226951	1129	1429	76.050	0.6	0.059	9	14	1	1	0	0	0	0	0	0	1.752
201	1294483	1133	1491	76.628	0.6	0.058	9	15	2	1	0	0	0	0	0	0	1.756
221	1248520	1124	1437	77.299	0.6	0.058	8	15	2	1	0	0	0	0	0	0	2.242
241	1299958	1134	1448	79.168	0.6	0.058	8	15	1	2	0	0	0	0	0	0	1.730
261	1269560	1134	1381	81.067	0.6	0.059	10	15	2	0	0	1	0	0	0	0	2.101
281	1314715	1134	1404	82.576	0.6	0.059	10	18	1	2	0	0	0	0	0	0	0.966
301	1332787	1134	1381	85.105	0.6	0.059	8	19	2	0	0	0	1	0	0	0	3.176
321	1311369	1134	1384	83.556	0.6	0.059	8	18	0	0	0	0	1	0	0	0	4.281
341	1277812	1133	1310	86.093	0.6	0.059	7	17	2	0	0	0	0	1	0	0	9.495
361	1342971	1134	1369	86.507	0.6	0.058	9	19	2	0	0	0	0	0	1	1	5.136

Figure 4-16: Initial input data for force predictor training

Once the data extraction is completed, inspection, refinement and cleaning are carried out to ensure no errors, missing values, or irrational data.


After some trial and error, ice floe size categories were reduced to 3 by combining type 1 to 3 as the first category, type 4 to 7 as the second category, and type 8 to 10 as the third category. The process is illustrated in Figure 4-17 for a better understanding.

Con	Thick	Vel	Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9	Type10	Force
72.77361524	0.6	0.5	5	12	2	0	0	0	0	0	0	0	0.368229
73.75481212	0.6	0.5	6	9	2	1	0	0	0	0	0	0	31.09194
75.50973875	0.6	0.5	7	15	1	1	0	0	0	0	0	0	4.646596

Con	Thick	Vel	type1	Type2	Type3	Force
72.77361524	0.6	0.5	19	0	0	0.368229
73.75481212	0.6	0.5	17	1	0	31.09194
75.50973875	0.6	0.5	23	1	0	4.646596

Figure 4-17: Reducing ice floe categories for better results (Scenario 1 test set)

In order to accommodate the large range for the force variables, a logarithmic transformation was done on the force values. The training data provided a better force predictor when the logarithmic force was used as a prediction variable rather than actual force values. This transformed output with the 7 input variables is marked as the ‘scenario 1’ test set.



Concentration	Thickness	Velocity	Floe	Type1	Type2	Type3	Collision	Force (log)	Force actual
88.830	1	0.02	41	3	1	0	1.7505	56.3	
88.408	1	0.02	40	4	1	1	2.0007	100.2	
88.578	1	0.02	39	6	0	0	1.6814	48.0	

Figure 4-18: Addition of ‘collision’ variable (Scenario 2 test set)

As explained later in this section, it was observed that the prediction results could be further improved by incorporating another new variable to quantify the status of collision between the ship and ice floes. As demonstrated in Figure 4-18, the new ‘collision’ variable is defined as such it is ‘1’ when the ship hits the ice and ‘0’ when no ship-ice collision occurs. This revised dataset of 8 variables (including the new collision variable) is named the ‘scenario 2’ test set.

While analyzing the training performance using the scenario 2 test data set, it was noticed that the results become even better if some irrational data points are removed from the test set. For example, when the vessel just started to move and the ship velocity suddenly increased from zero, the force sensor picked up a large value compared to the previous and next values after this point. as can be seen in the force in the second row of Table 4-2 (a). This might be happening due to the initial momentum of the experimental system. Similarly, when the ship stopped at the end of the experiment, it produced another impact and, consequently, a sudden rise in the force value (Table 4-2(b), the force in the third row). As a result, those initial and ending data points for each video are omitted, and the total samples are reduced to 583 after performing this data cleaning operation. This cleaned dataset is named the ‘scenario 3’ test set.

Table 4-2: Irrational data point removal. (a) Beginning of the test situation, (b) End of the test situation.

(a)

Concentration	Thickness	Velocity	Floe Type1	Type2	Type3	Collision	Force (log)
72.77361524	0.6	0.00	19	0	0	0	-0.43388
73.75481212	0.6	0.06	17	1	0	0	1.492648
75.50973875	0.6	0.06	23	1	0	0	0.667135
83.555	0.6	0.06	22	1	0	0	0.631533

(b)

Concentration	Thickness	Velocity	Floe Type1	Type2	Type3	Collision	Force (log)
84.532	0.6	0.14	30	1	0	0	1.106064
86.578	0.6	0.14	35	3	0	0	1.11479
86.581	0.6	0.03	25	5	0	0	1.711258

Table 4-3 summarizes how these three test datasets are prepared.

Table 4-3: Definition of test dataset scenarios

Scenario 1 (Original Scenario, 7 variables)	Scenario 2 (8 variables, addition of collision variable with scenario 1)	Scenario 3 (Irrational data points removed from scenario 2)
<ul style="list-style-type: none"> • 627 dataset form images extracted from 29 videos. • 7 variables: concentration, thickness, velocity, 3 types of ice floe and force. 	<ul style="list-style-type: none"> • 627 dataset form images extracted from 29 videos. • 8 variables, additional collision variable to scenario 1. • Collision variable definition: 0 - no collision, 1 - ship hit the ice. 	<ul style="list-style-type: none"> • 583 dataset form 29 videos after deleting irrational data points. • 8 variables, same as Scenario 2

The SVM regression model is trained for each of these three scenarios. The training takes around a minute to complete, with 10 iterations for all scenarios. 80% data is used to train the model, 10% for validation and 10% for testing. In addition to comparing the originally-observed and model-predicted values of forces for the test dataset, the RMSE values (Eq. 4-3) are compared for the different dataset scenarios.

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (x_i - \hat{x}_i)^2}{N}} \quad (4-3)$$

Overall, the prediction performance improved from scenario 1 towards scenario 3 (Figure 4-19). However, the RMES value is quite large, even for scenario 3. There are some prominent outliers which are contributing towards these higher values of RMES. Most of these noticeable mismatches occur when the force values are larger, especially more than 150. A possible reason behind this is that the dataset has a very limited number of force values higher than 150 (only 3% of the total data sample). Thus, making it difficult for the predictor to have a reasonable estimation. Moreover, some of these higher force values occurred during the start and end of the experiment (due to

momentum), as explained earlier. Those values were removed in scenario 3. Therefore, only one force value higher than 150 is observed in that case.

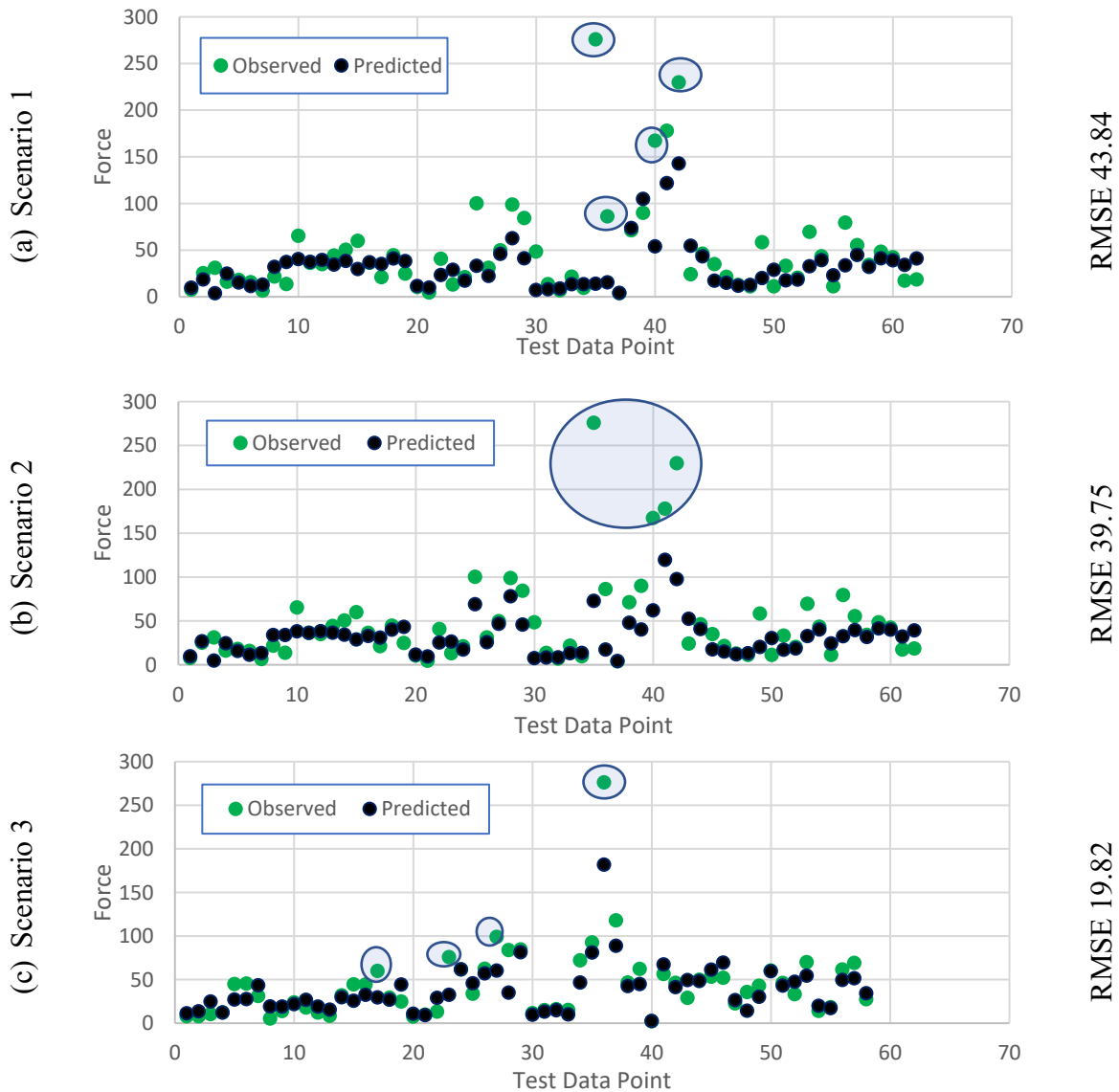


Figure 4-19: SVM force prediction performance for various input scenarios (Hybrid model 1)

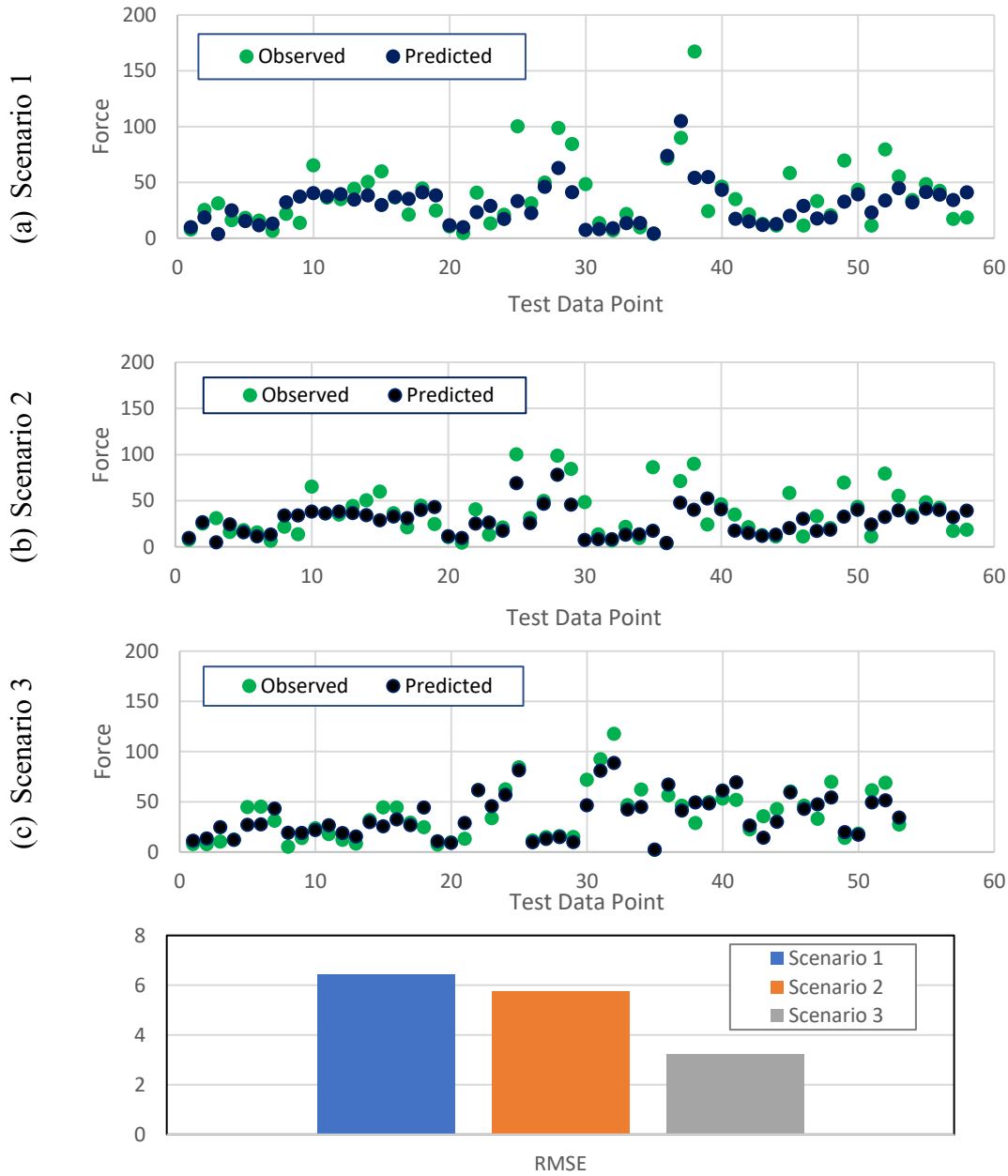


Figure 4-20: SVM force prediction performance for various input scenarios, outliers removed (Hybrid model 1)

Nevertheless, the top 4 outlier points, in terms of the difference between observed and predicted values, are removed from the test to better assess the force predictor’s performance. The outliers are circled in Figure 4-19, and the revised results are presented in Figure 4-20. As noticed, the RMSE now reduced significantly, and for scenario 3, the reduction is more than 50%.

At this point, the FFNN force predictor is trained using the same datasets as reported above for the SVM. Three hidden layers are used with 15, 10 and 5 neurons, respectively, to define the FFNN. However, unlike the SVM training, only two scenarios for the datasets are used to train the network.

Scenario 1: The same as SVM Scenario 3 (583 datasets from 29 videos after deleting irrational data points. 8 variables, including collision variable).

Scenario 2: The same as SVM Scenario 3, except that no collision variable is used (583 datasets from 29 videos after deleting irrational data points. 7 variables).

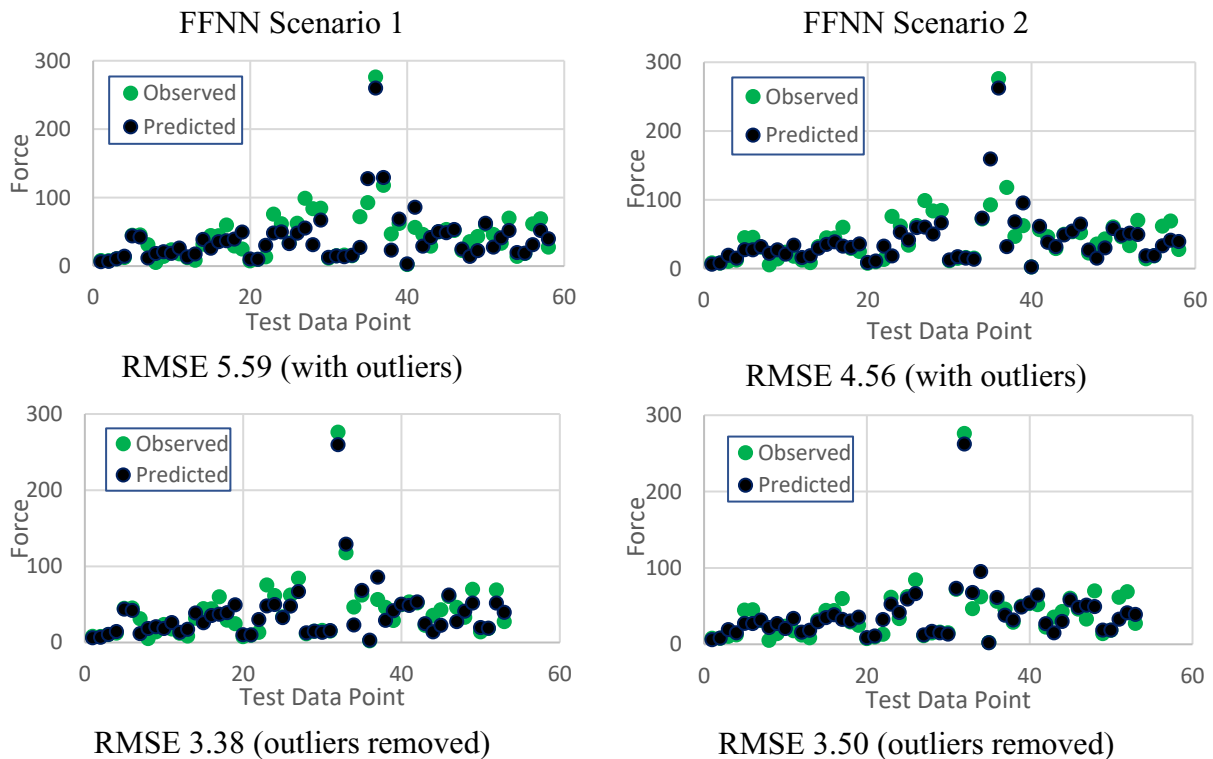


Figure 4-21: FFNN force prediction performance for various input scenarios (Hybrid model 1)

Figure 4-21 shows the observed and predicted force value comparisons for the two FFNN predictors. As can be seen, the FFNN results appeared to be much better compared to the SVM,

especially since FFNN prediction for higher force values is quite accurate. The RMSE values also reduced significantly. Most importantly, the results with the collision variable (Scenario 1) and without it (Scenario 2) don't differ too much, thus, eliminating the need for this extra variable while using FFNN. There were no significant outliers as well in FFNN prediction, as compared to the SVM analysis. Therefore, after removing 5 points based on the highest differences between the observed and predicted forces, the RMSE values only improved slightly.

4.3.2.2 Force predictors for Hybrid model 02

Unlike Hybrid model 01, the ice floe information for Hybrid model 02 is extracted through ML-based image processing using RCNN, as described earlier. A total of 520 images were used for training the ice floe detector. However, as described in section 4.2.1, not all the ice floe images produced reasonable prediction results. Therefore, only images with comparatively larger and uniform ice floes (Figure 4-13) are identified to be used for force prediction. As a result, the number of samples reduced significantly to 74. Nevertheless, a total of 148 samples are prepared using the data augmentation process described earlier. Also, compared to the number of variables for Hybrid model 01, only 6 variables are used here. Instead of using three categories for separating the floes, here, the total number of floes is counted under one heading. Also, the average floe area is added as a new variable. Table 4-4 shows the variable names and a few sample values.

The training process for SVM and FFNN for Hybrid model 02 is the same as for Hybrid model 01, as described earlier. The only difference is that the number of datasets has now reduced to 148. Therefore only 14 data are used for testing the force predictor models.

Table 4-4: Variable definitions and sample values for force predictor input.

Concentration	Thickness	Velocity	Floe No	Floe Area	Force (log)
81.278	0.8	0.0004	15	63147.3	1.5138
80.873	0.8	0.0005	15	61106.7	1.4673
82.029	0.8	0.0004	16	60619.4	1.2440
83.990	0.8	0.0487	15	57164.5	1.5112
84.842	0.8	0.5880	15	63614.0	2.3861
85.125	0.8	0.5880	15	62991.2	2.4409
84.839	0.8	0.4940	12	67579.0	2.4406
79.736	0.8	0.0589	13	65071.9	1.9353

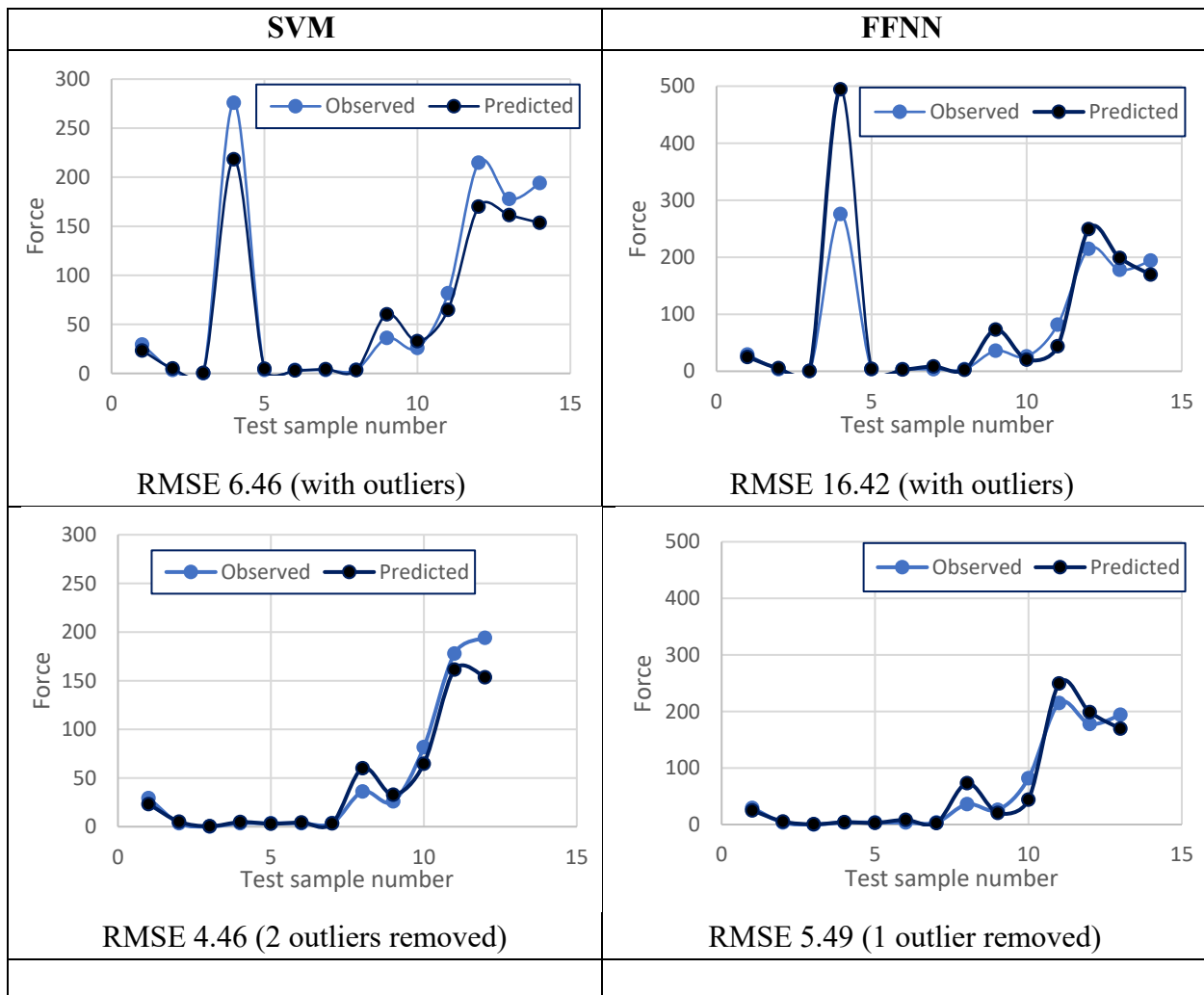


Figure 4-22: SVM and FFNN force prediction performance comparison for Hybrid model 02.

As depicted in Figure 4-22, the results from both the SVM and FFNN predictor for Hybrid model 02 are superior compared to Hybrid model 01, keeping in mind that the number of samples for Hybrid model 02 is nearly 5 times lower than that for Hybrid model 01. Nevertheless, as can be seen, the prediction for most of the force values is quite reasonable, except for higher values, due to the limited number of high force value data samples. Also, the detection of ice floes and subsequent force prediction using the trained model only takes a few seconds (5 to 10 secs) in Hybrid model 02, whereas it takes around 80 seconds (70 to 90 secs) in Hybrid model 01. Therefore, considering the computational efficiency, it can be said that Hybrid model 02 has a higher potential for being developed as a real-time force prediction model as compared to Hybrid model 01, which relies on traditional image processing for data extraction.

4.4 Conclusion

This study proposed two Hybrid models for ice force prediction using ice floe images. Instead of using analytical or numerical approaches, the Hybrid models directly extract floe characteristics from the images and later train ML-based force predictors using those extracted floe parameters. Hybrid model 01 used the traditional image processing technique for information extraction, whereas Hybrid model 02 applied the RCNN approach for the same task.

Compared to the existing analytical and numerical tools for ice force estimation on floating platforms, the prediction of forces from ice floe images around a ship has a greater potential for application as a real-time prediction. However, estimating ice forces from images is challenging because it requires accurate extraction of ice floe information. Various factors, for example, the complex and varied shapes and sizes of the ice floes, ice thickness, ice strength, the concentration

of the floe field, colour similarities and light reflection on images, make that extraction task challenging.

Both the SVM and FFNN also performed well across the board while detecting the force in the lower range. However, the SVM predictor for Hybrid model 01 failed to predict higher force values accurately. In comparison, FFNN predictor performance is much more reasonable as it managed to predict force values in the upper range with some degree of accuracy. As for Hybrid model 02, both the SVM and FFNN performed nearly at the same levels. They estimated the higher range force values reasonably, compared to Hybrid model 01, despite the number of datasets with higher range force values for Hybrid model 02 being quite limited (just 2% of all data points).

More importantly, the tools developed in this research work are faster than existing approaches. It effectively focuses on an image-based ice force prediction approach, mainly on moving vessels, by reducing high computational time. Nevertheless, more extensive validation and testing are required to confirm the effectiveness and significance of these tools before considering them for real-time application. Future modification of the algorithms and use of a wider range of data will improve confidence in these models and their applicability as force prediction tools with reasonable accuracy in the near future.

4.5 Acknowledgements

The authors acknowledge the financial support from the National Research Council (NRC) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

5.0 Conclusions and Recommendations

5.1 Conclusions

This thesis deals with developing image-based ice force prediction models for ship / offshore structures in managed ice fields. The first step towards creating the model is extracting useful features from the images. These images are complex in nature and characterized by indistinct boundaries, overlapping ice floe, and unevenly illuminated ice floe fields with various floe sizes and shapes. Two different tools were developed to achieve the goal:

- (a) An improved version of an existing traditional image segmentation platform was developed by combining several image processing features, including histogram equalization, wavelet denoising, gradient flow vector, snake algorithm, and distance transformation.
- (b) A Machine Learning based image processing platform was developed using ‘Region Based Convolutional Neural Network.’

Ice floe characteristics were extracted through both these tools, and a few additional parameters from model tank experimental analysis were used to train SVM and FFNN based force predictors. Thus, two Hybrid models are developed. The first Hybrid model, called AIP, combines image processing tool (a) and the corresponding SVM and FFNN force predictors; the second Hybrid model, called CVIP, combines image processing tool (b) with the corresponding SVM and FFNN force predictors.

The main contributions and outcomes of the thesis are summarized below:

- i. The traditional image processing method was improved by incorporating wavelet denoising, histogram equalization, and illumination normalization with the GVF SNAKE segmentation technique to develop the high-accuracy ice image segmentation tool (a). The updated method was able to separate floes from images with complex and varied shapes ice floes with overlapped and/or connected boundaries and weaker edges in an unevenly illuminated environment. The capability of the proposed model in separating complex floes with weaker and connected boundaries in noisy and nonuniform illumination environments is compared with existing models in the literature. On average, the new model is 40% faster and provides a 10% better prediction of ice concentrations.
- ii. Wavelet filtering and histogram equalization are used to enhance the input images through denoising and contrast adjustment in this improved model, which improved the floe separation accuracy.
- iii. Various morphological filtering and region analysis techniques are then applied to extract floe features. Overall, the model can detect the total number of floes with more than 85% accuracy and ice concentration accuracy of 95% and above. It can also be used for real-time ice floe detection due to its faster processing time with the expense of accuracy to a certain acceptable limit, as described in section 3.3.3.
- v. The other image segmentation tool developed using the faster RCNN (b) is more efficient than the traditional alternative. The trained model only takes a few seconds to detect floes in an image, compared to more than 1 minute taken by the traditional approach. However, its accuracy is considerably lower, especially for images where the floe sizes vary more prominently and floes are in close proximity.

vi. The SVM based force predictors trained for both the image segmentation tools demonstrated reasonable estimation accuracy with acceptable RMSE values after removing the outliers. The SVM predictor for the tool (a) achieved an RMSE of 3.21. However, it could not predict the higher force value with expected accuracy. This can be attributed to the limited number of data samples with high force values. SVM models' performance also varied between the 'with' and 'without' collision variable scenarios (RMSE decreased by 0.7 with the collision variable).

vii. The FFNN based force predictors, on the other hand, demonstrated noticeable estimation performance for higher force values, along with reasonable estimation for regular force values. The three-layered (FFNN) predictors attained RMSE values of 3.5 and 5.49 for the tool (a) and tool (b), respectively. Also, the models' performed nearly similarly for both 'with' and 'without' collision variable scenarios (RMSE differs by 0.12 only). Therefore, it can be concluded that FFNN prediction performances are much better (able to detect high force values with limited training samples and one less variable) as compared to the SVM counterpart.

viii. Both the Hybrid model, 1 and 2, demonstrated promising performances in terms of accuracy and efficiency, offering potential as a real-time ice force prediction platform.

5.2 Recommendations

The proposed Hybrid models for ice force estimation offer reasonable performance. However, they should be further improved to increase the possibility of real-time applications. A few areas that can be considered for further research are:

- i. The current image processing model manually sets some parameters related to ice properties, which need to be tuned as ice field texture, illumination and size vary significantly. Thus, there exists room for improvement, especially in terms of automation.
- ii. Due to the shortage of training data, further study should be carried out to extend the current models to include diverse range of experimental and augmented training data. Also, performance of other machine learning and deep learning-based network architectures in predicting the ice force directly from the input images should be explored.
- iii. The proposed models are developed for images taking from bird's eye view. The model can be extended to consider images taken using front/rear, onboard cameras.
- iv. The difficulty of developing a simple global ice force model is emphasized by many variables and the fact that they are often interconnected and not accurately measurable. For instance, Ship performance such as ship in straight running, zigzag maneuver and turning motion also affects the global ice forces which cannot be measured accurately in some cases. The proposed model does not account for these factors, which can be explored in future research.

References

- [1] C. Gignac, Y. Gauthier, J. S. Bédard, M. Bernier, and D. A. Clausi, “High resolution RADARSAT-2 SAR data for sea-ice classification in the neighborhood of Nunavik’s marine infrastructures,” in *21st International Conference on Port and Ocean Engineering under Arctic Conditions 2011, POAC 2011*, 2011, pp. 1181–1190.
- [2] A. v Bogdanov, S. Sandven, O. M. Johannessen, V. Y. Alexandrov, and L. P. Bobylev, “Multi-sensor approach to automated classification of sea ice image data,” in *Signal and Image Processing for Remote Sensing*, CRC Press, 2006, pp. 576–607.
- [3] T. Toyota and H. Enomoto, “Analysis of sea ice floes in the Sea of Okhotsk using ADEOS/AVNIR images. paper presented at 16th International Symposium on Ice,” *Int. Assoc. for Hydraul Res. Dunedin, New Zealand*, 2002.
- [4] Q. Zhang, “Image processing for ice parameter identification in ice management,” 2015.
- [5] J. Tuhkuri and A. Polojärvi, “A review of discrete element simulation of ice–structure interaction,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 376, no. 2129, p. 20170335, 2018.
- [6] P. Lu and Z. Li, “A method of obtaining ice concentration and floe size from shipboard oblique sea ice images,” *IEEE Transactions on geoscience and remote sensing*, vol. 48, no. 7, pp. 2771–2780, 2010.
- [7] Q. Zhang and R. Skjetne, *Sea Ice Image Processing with MATLAB®*. CRC Press, 2018.
- [8] N. Düchting, “Combined FEM-SPH simulations for ice in compression,” 2018.
- [9] I. Lemström, A. Polojärvi, and J. Tuhkuri, “Numerical experiments on ice-structure interaction in shallow water,” *Cold Reg Sci Technol*, vol. 176, p. 103088, 2020.
- [10] W. Luo *et al.*, “Numerical simulation of an ice-strengthened bulk carrier in brash ice channel,” *Ocean Engineering*, vol. 196, p. 106830, 2020.
- [11] M. Islam, J. Mills, R. Gash, and W. Pearson, “A literature survey of broken ice-structure interaction modelling methods for ships and offshore platforms,” *Ocean Engineering*, vol. 221, p. 108527, 2021.
- [12] M. S. Sivapriya and P. Mohamed Fathimal, “Ice Berg Detection in SAR Images Using Mask R-CNN,” in *Emergent Converging Technologies and Biomedical Systems*, Springer, 2022, pp. 625–633.

- [13] H. Lee, K. Hwang, M. Kang, and J. Song, "Black ice detection using CNN for the Prevention of Accidents in Automated Vehicle," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2020, pp. 1189–1192.
- [14] N. C. Wright and C. M. Polashenski, "Open-source algorithm for detecting sea ice surface features in high-resolution optical imagery," *Cryosphere*, vol. 12, no. 4, pp. 1307–1329, 2018.
- [15] F. Parmiggiani, M. Moctezuma-Flores, P. Wadhams, and G. Aulicino, "Image processing for pancake ice detection and size distribution computation," *Int J Remote Sens*, vol. 40, no. 9, pp. 3368–3383, May 2019, doi: 10.1080/01431161.2018.1541367.
- [16] Mathworks Inc, "Computer Vision Toolbox (r2022b)," 2022.
- [17] Mathworks Inc, "Image Processing Toolbox (r2022b)," 2022.
- [18] MathWorks Inc., "Deep Learning Toolbox: User's Guide (r2022a)," 2022.
- [19] S. K. Dewangan, "Importance & Applications of Digital Image Processing," *International Journal of Computer Science & Engineering Technology (IJCSET)*, vol. 7, no. 7, pp. 316–320, 2016.
- [20] Simplilearn, "What Is Image Processing : Overview, Applications, Benefits, and Who Should Learn It," Jun. 2022.
- [21] Mapasyst, "What's the difference between a supervised and unsupervised image classification?," 2019.
- [22] B. Abu-Jamous, R. Fa, and A. K. Nandi, "Integrative cluster analysis in bioinformatics," 2015.
- [23] T. Lei, X. Jia, Y. Zhang, L. He, H. Meng, and A. K. Nandi, "Significantly fast and robust fuzzy c-means clustering algorithm based on morphological reconstruction and membership filtering," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 3027–3041, 2018.
- [24] M. Baştan, S. S. Bukhari, and T. Breuel, "Active Canny: edge detection and recovery with open active contour models," *IET Image Process*, vol. 11, no. 12, pp. 1325–1332, 2017.
- [25] Z. Zhang, C. Duan, T. Lin, S. Zhou, Y. Wang, and X. Gao, "GVFOM: a novel external force for active contour based image segmentation," *Inf Sci (N Y)*, vol. 506, pp. 1–18, Jan. 2020, doi: 10.1016/j.ins.2019.08.003.

- [26] M. Gong, H. Li, X. Zhang, Q. Zhao, and B. Wang, “Nonparametric statistical active contour based on inclusion degree of fuzzy sets,” *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 5, pp. 1176–1192, 2015.
- [27] J. Ma, S. Li, H. Qin, and A. Hao, “Unsupervised multi-class co-segmentation via joint-cut over L_1 -manifold hyper-graph of discriminative image regions,” *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1216–1230, 2016.
- [28] S. Yin, Y. Qian, and M. Gong, “Unsupervised hierarchical image segmentation through fuzzy entropy maximization,” *Pattern Recognit*, vol. 68, pp. 245–259, 2017.
- [29] C. R. Jung and J. Scharcanski, “Robust watershed segmentation using wavelets,” *Image Vis Comput*, vol. 23, no. 7, pp. 661–669, Jul. 2005, doi: 10.1016/j.imavis.2005.03.001.
- [30] D. D. N. de Silva, S. Fernando, I. T. S. Piyatilake, and A. V. S. Karunaratne, “Wavelet based edge feature enhancement for convolutional neural networks,” in *Eleventh International Conference on Machine Vision (ICMV 2018)*, 2019, vol. 11041, p. 110412R.
- [31] S. S. Ganesh, K. Mohanaprasad, and Y. Karuna, “Object identification using wavelet transform,” *Indian J Sci Technol*, vol. 9, no. 5, pp. 1–7, 2016.
- [32] A. Kumar, S. Saha, and R. Bhattacharya, “Wavelet transform based novel edge detection algorithms for wideband spectrum sensing in CRNs,” *AEU-International Journal of Electronics and Communications*, vol. 84, pp. 100–110, 2018.
- [33] Q. Zhang and R. Skjetne, “Image processing for identification of sea-ice floes and the floe size distributions,” *IEEE Transactions on geoscience and remote sensing*, vol. 53, no. 5, pp. 2913–2924, 2014.
- [34] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Trans Pattern Anal Mach Intell*, vol. 24, no. 5, pp. 603–619, 2002.
- [35] N. R. Pal and J. C. Bezdek, “On cluster validity for the fuzzy c-means model,” *IEEE Transactions on Fuzzy systems*, vol. 3, no. 3, pp. 370–379, 1995.
- [36] A. Ng, M. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” *Adv Neural Inf Process Syst*, vol. 14, 2001.
- [37] B. C. Gonçalves and H. J. Lynch, “Fine-Scale Sea Ice Segmentation for High-Resolution Satellite Imagery with Weakly-Supervised CNNs,” *Remote Sens (Basel)*, vol. 13, no. 18, p. 3562, 2021.
- [38] Q. Zhang, S. van der Werff, I. Metrikin, S. Løset, and R. Skjetne, “Image processing for the analysis of an evolving broken-ice field in model testing,” in *International Conference on Offshore Mechanics and Arctic Engineering*, 2012, vol. 44939, pp. 597–605.

- [39] A. Talukder, D. P. Casasent, H.-W. Lee, P. M. Keagy, and T. F. Schatzki, “Modified binary watershed transform for segmentation of agricultural products,” in *Precision Agriculture and Biological Quality*, 1999, vol. 3543, pp. 53–64.
- [40] X. Chen, X. Zhou, and S. T. C. Wong, “Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy,” *IEEE Trans Biomed Eng*, vol. 53, no. 4, pp. 762–766, 2006.
- [41] J. D. Blunt, V. Y. Garas, D. G. Matskevitch, J. M. Hamilton, and K. Kumaran, “Image analysis techniques for high Arctic, deepwater operation support,” in *OTC Arctic Technology Conference*, 2012.
- [42] Q. Zhang, R. Skjetne, and B. Su, “Automatic image segmentation for boundary detection of apparently connected sea-ice floes,” in *The proceedings of the 22nd International Conference on Port and Ocean Engineering under Arctic Conditions*, 2013.
- [43] M. Turker and A. Rahimzadeganasl, “Agricultural Field Detection from Satellite Imagery Using the Combined Otsu’s Thresholding Algorithm and Marker-Controlled Watershed-Based Transform,” *Journal of the Indian Society of Remote Sensing*, vol. 49, no. 5, pp. 1035–1050, May 2021, doi: 10.1007/s12524-020-01276-4.
- [44] A. Safonova, E. Guirado, Y. Maglinets, D. Alcaraz-Segura, and S. Tabik, “Olive tree biovolume from UAV multi-resolution image segmentation with mask R-CNN,” *Sensors*, vol. 21, no. 5, p. 1617, 2021.
- [45] Z. Wang, X. Gao, R. Wu, J. Kang, and Y. Zhang, “Fully automatic image segmentation based on FCN and graph cuts,” *Multimed Syst*, pp. 1–13, 2022.
- [46] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, “Deep learning-based image recognition for autonomous driving,” *IATSS research*, vol. 43, no. 4, pp. 244–252, 2019.
- [47] M. Masud, N. Sikder, A.-A. Nahid, A. K. Bairagi, and M. A. AlZain, “A machine learning approach to diagnosing lung and colon cancer using a deep learning-based classification framework,” *Sensors*, vol. 21, no. 3, p. 748, 2021.
- [48] K. Hacıefendioğlu, H. B. Başağa, Z. Yavuz, and M. T. Karimi, “Intelligent ice detection on wind turbine blades using semantic segmentation and class activation map approaches based on deep learning method,” *Renew Energy*, vol. 182, pp. 1–16, 2022.
- [49] C. O. Dumitru, V. Andrei, G. Schwarz, and M. Datcu, “Machine learning for sea ice monitoring from satellites,” *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 83–89, 2019.

- [50] A. Singh, H. Kalke, M. Loewen, and N. Ray, “River ice segmentation with deep learning,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 11, pp. 7570–7579, 2020.
- [51] B. C. Gonçalves and H. J. Lynch, “Fine-Scale Sea Ice Segmentation for High-Resolution Satellite Imagery with Weakly-Supervised CNNs,” *Remote Sens (Basel)*, vol. 13, no. 18, p. 3562, 2021.
- [52] B. Dowden, O. de Silva, W. Huang, and D. Oldford, “Sea ice classification via deep neural network semantic segmentation,” *IEEE Sens J*, vol. 21, no. 10, pp. 11879–11888, 2020.
- [53] C. Zhang, X. Chen, and S. Ji, “Semantic image segmentation for sea ice parameters recognition using deep convolutional neural networks,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 112, p. 102885, 2022, doi: <https://doi.org/10.1016/j.jag.2022.102885>.
- [54] C. Pei, Y. She, and M. Loewen, “Deep learning based river surface ice quantification using a distant and oblique-viewed public camera,” *Cold Reg Sci Technol*, vol. 206, p. 103736, 2023, doi: <https://doi.org/10.1016/j.coldregions.2022.103736>.
- [55] K. Chen, H. Che, M.-F. Leung, and Y. Wang, “An Improved Superpixel-based Fuzzy C-Means Method for Complex Picture Segmentation Tasks,” in *2022 14th International Conference on Advanced Computational Intelligence (ICACI)*, 2022, pp. 231–238.
- [56] A. Caggiano, J. Zhang, V. Alfieri, F. Caiazzo, R. Gao, and R. Teti, “Machine learning-based image processing for on-line defect recognition in additive manufacturing,” *CIRP annals*, vol. 68, no. 1, pp. 451–454, 2019.
- [57] B. S. Kusumo, A. Heryana, O. Mahendra, and H. F. Pardede, “Machine learning-based for automatic detection of corn-plant diseases using image processing,” in *2018 International conference on computer, control, informatics and its applications (IC3INA)*, 2018, pp. 93–97.
- [58] M. A. Islam, M. S. Islam, M. S. Hossen, M. U. Emon, M. S. Keya, and A. Habib, “Machine learning based image classification of papaya disease recognition,” in *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2020, pp. 1353–1360.
- [59] R. Frederking, “Dynamic Ice Forces on an Inclined Structure,” in *Physics and Mechanics of Ice*, 1980, pp. 104–116.
- [60] A. Ahmed, M. A. al Maruf, A. Kr. Dev, and M. A. Hannan, “Preliminary Analytical Formulation of Ice-Floater Interactions Including the Effect of Wave Load,” in *Volume 8:*

Polar and Arctic Sciences and Technology; Petroleum Technology, Jun. 2018. doi: 10.1115/OMAE2018-78340.

- [61] M. Islam, J. Mills, R. Gash, and W. Pearson, “Modelling of Dynamically Positioned vessels and managed ice-field interactions using multiple regression techniques,” *Ocean Engineering*, vol. 243, p. 110248, 2022.
- [62] S. Islam *et al.*, “Physical Model Testing for Supporting Ice Force Model Development of DP Vessels in Managed Ice,” in *OTC Arctic Technology Conference*, 2018.
- [63] K.-U. Evers, W. Kuehnlein, P. Jochmann, W. Spring, and J. Foulkes, “Ice model testing of an exploration platform for shallow waters in the North Caspian Sea,” 2001.
- [64] Qu Yan, Yue Qianjin, Bi Xiangjun, and Kärnä Tuomo, “A random ice force model for narrow conical structures,” *Cold Reg Sci Technol*, vol. 45, no. 3, pp. 148–157, 2006, doi: <https://doi.org/10.1016/j.coldregions.2006.05.008>.
- [65] A. Gürtner, “Experimental and numerical investigations of ice-structure interaction,” 2009.
- [66] D. Hilding, J. Forsberg, and A. Gürtner, “Simulation of ice action loads on offshore structures,” in *Proceedings of the 8th European LS-DYNA Users Conference, Strasbourg*, 2011.
- [67] C. Wang, X. Hu, T. Tian, C. Guo, and C. Wang, “Numerical simulation of ice loads on a ship in broken ice fields using an elastic ice model,” *International Journal of Naval Architecture and Ocean Engineering*, vol. 12, pp. 414–427, 2020.
- [68] M.-C. Kim, S.-K. Lee, W.-J. Lee, and J. Wang, “Numerical and experimental investigation of the resistance performance of an icebreaking cargo vessel in pack ice conditions,” *International Journal of Naval Architecture and Ocean Engineering*, vol. 5, no. 1, pp. 116–131, 2013.
- [69] F. Li, M. Körgesaar, P. Kujala, and F. Goerlandt, “Finite element based meta-modeling of ship-ice interaction at shoulder and midship areas for ship performance simulation,” *Marine Structures*, vol. 71, p. 102736, 2020.
- [70] S. Ji, Z. Li, C. Li, and J. Shang, “Discrete element modeling of ice loads on ship hulls in broken ice fields,” *Acta Oceanologica Sinica*, vol. 32, no. 11, pp. 50–58, 2013.
- [71] X. Long and S. Ji, “Numerical Simulations of Ice Loads on Fixed and Floating Offshore Structures using the Discrete Element Method,” 2016.

- [72] O. Jou, M. A. Celigueta, S. Latorre, F. Arrufat, and E. Oñate, “A bonded discrete element method for modeling ship–ice interactions in broken and unbroken sea ice fields,” *Comput Part Mech*, vol. 6, no. 4, pp. 739–765, 2019.
- [73] N. Zhang, X. Zheng, Q. Ma, and Z. Hu, “A numerical study on ice failure process and ice-ship interactions by Smoothed Particle Hydrodynamics,” *International Journal of Naval Architecture and Ocean Engineering*, vol. 11, no. 2, pp. 796–808, 2019.
- [74] S. Idrissova, M. Bergström, S. E. Hirdaris, and P. Kujala, “Analysis of a collision-energy-based method for the prediction of Ice loading on ships,” *Applied Sciences*, vol. 9, no. 21, p. 4546, 2019.
- [75] J.-H. Kim, Y. Kim, H.-S. Kim, and S.-Y. Jeong, “Numerical simulation of ice impacts on ship hulls in broken ice fields,” *Ocean Engineering*, vol. 182, pp. 211–221, 2019, doi: <https://doi.org/10.1016/j.oceaneng.2019.04.040>.
- [76] M. Islam, “Machine Learning Techniques for Ship Performance Predictions in Open Water and Ice,” in *The 31st International Ocean and Polar Engineering Conference*, 2021.
- [77] J.-H. Kim, Y. Kim, and W. Lu, “Prediction of ice resistance for ice-going ships in level ice using artificial neural network technique,” *Ocean Engineering*, vol. 217, p. 108031, 2020, doi: <https://doi.org/10.1016/j.oceaneng.2020.108031>.
- [78] Q. Sun, M. Zhang, L. Zhou, K. Garne, and M. Burman, “A machine learning-based method for prediction of ship performance in ice: Part I. ice resistance,” *Marine Structures*, vol. 83, p. 103181, 2022, doi: <https://doi.org/10.1016/j.marstruc.2022.103181>.
- [79] J. Hamilton, H. Curtis, B. Joshua, M. Douglas, and K. Ted, “Ice management for support of arctic floating operations.,” in *OTC Arctic Technology Conference.*, 2011.
- [80] H. Dong, L. Zhao, Y. Shu, and N. N. Xiong, “X-ray image denoising based on wavelet transform and median filter,” *Applied Mathematics and Nonlinear Sciences*, vol. 5, no. 2, pp. 435–442, 2020, doi: [doi:10.2478/amns.2020.2.00062](https://doi.org/10.2478/amns.2020.2.00062).
- [81] B. Han, “Framelets and wavelets,” *Algorithms, Analysis, and Applications, Applied and Numerical Harmonic Analysis. Birkhäuser xxxiii Cham*, 2017.
- [82] MathWorks, “Wavelet image denoising,” *Matlab 2021.b documentation*, 2021.
- [83] C. Xu and J. L. Prince, “Snakes, shapes, and gradient vector flow,” *IEEE Transactions on image processing*, vol. 7, no. 3, pp. 359–369, 1998.
- [84] A. Rosenfeld and J. L. Pfaltz, “Distance functions on digital pictures,” *Pattern Recognit*, vol. 1, no. 1, pp. 33–61, 1968.

- [85] S. Akter, M.S. Islam, H. Zaman, S. Ahmed, S. Imtiaz, and R. Gash, “Image Processing to Extract Ice Features to Aid Modelling of Ice Force,” in *Proceedings of the ASME 41st International Conference on Ocean, Offshore and Arctic Engineering*, Jun. 2022.
- [86] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [87] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Adv Neural Inf Process Syst*, vol. 28, 2015.
- [88] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. van Gool, “Domain adaptive faster r-cnn for object detection in the wild,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3339–3348.
- [89] C. Zhang, X. Xu, and D. Tu, “Face detection using improved faster rcnn,” *arXiv preprint arXiv:1802.02142*, 2018.
- [90] X. Sun, P. Wu, and S. C. H. Hoi, “Face detection using deep learning: An improved faster RCNN approach,” *Neurocomputing*, vol. 299, pp. 42–50, 2018.
- [91] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Adv Neural Inf Process Syst*, vol. 28, 2015.
- [92] A. Tzotso, “A support vector machine approach for object based image analysis,” *Proceedings of OBIA*, 2006.
- [93] W. Yu, T. Liu, R. Valdez, M. Gwinn, and M. J. Khoury, “Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes,” *BMC Med Inform Decis Mak*, vol. 10, no. 1, p. 16, 2010, doi: 10.1186/1472-6947-10-16.
- [94] MathWorks Inc., “Deep Learning Toolbox, Function approximation and nonlinear regression (r2022b),” 2022.

Appendix A - Published Conference Paper in OMAE 2022

Proceedings of the ASME 2022 41st International
Conference on Ocean, Offshore and Arctic Engineering
OMAE2022
June 5-10, 2022, Hamburg, Germany

OMAE2022-79255

IMAGE PROCESSING TO EXTRACT ICE FEATURES TO AID MODELLING OF ICE FORCE

Shamima Akter

Memorial University of
Newfoundland,
St. John's, NL, Canada

Mohammed Islam

National Research Council Canada
St. John's, NL, Canada

Syed Imtiaz

Memorial University of Newfoundland,
St. John's, NL, Canada

Hasanat Zaman

National Research Council
Canada
St. John's, NL, Canada

Salim Ahmed

Memorial University of Newfoundland
St. John's, NL, Canada

Robert Gash

National Research Council Canada
St. John's, NL, Canada

ABSTRACT

Sea-ice observation and estimation of ice forces are becoming increasingly important with the increased activities in arctic water. Proper modelling of ice properties and forces is crucial in such cases to ensure safe operations, for example, Dynamic Positioning (DP). This work, therefore, aims at developing algorithms for image processing to extract useful ice properties and subsequently aid the modelling of ice load exerted on a floating platform or a ship. A robust algorithm capable of detecting closely connected and overlapped ice floes with various sizes and shapes is presented, which is the first step for accurate modelling of ice forces. To demonstrate the effectiveness of this approach, image frames processed from videos produced during an experiment using a model ship performed at the National Research Council's Ocean Coastal and River Engineering Research Centre (NRC-OCRE) in Canada are used. Simulated ice floe images are also used to show the efficiency of the proposed model and compare it with other published work. The model will be extended further to extract and correlate other ice properties with the ice forces and develop a machine learning based ice force prediction model. The predicted force from that model will then be used as a feedforward to Dynamic Positioning (DP) controller with an aim to improve the performance of the controller.

Keywords: Image Processing, Managed ice, Ice properties, Sea ice, Gaussian blurring, Image sharpening

NOMENCLATURE

x	Distance from origin in the horizontal axis
y	Distance from origin in the vertical axis
G	Gaussian function
σ	Standard deviation
R_0, G_0, B_0	Red, Green, Blue colour matrices of the original image
R_n	Red colour channel matrix for the normalized image
i, j	index for colour matrices
u, v	The derivatives of the vector field in x and y directions.
μ	Parameters to control the balance between the integrands
f	Edge map which is larger near the edges

1. INTRODUCTION

Recent increase in scientific and commercial activities in Arctic has led to more research on improving sea ice property extraction and dynamic analysis of ice forces. Compared to other sea regions, exploration in the Arctic is more challenging due to its harsh weather, remoteness, and most importantly, the presence of ice. Advancement in image capturing technologies is providing researchers with additional tools to capture important information about the ice infested waters. Efficient image processing can help to extract and process various ice parameters

of interest and apply that knowledge to understand and predict the behavior of an ice field.

Development and application of novel theories and algorithms in image processing is gaining interests in recent years with wide range of applications, for example, in computer vision, remote sensing, feature extraction, face detection, forecasting, augmented reality, medical imaging and many other fields [19]. However, sea ice is a complicated multi-domain system; floe sizes, shapes, concentrations vary significantly on a relatively small scale. Moreover, the ice floes typically touch each other as well as overlap, making it difficult to detect the junctions from their images. Therefore, developing an automated computer algorithm for understanding of the dynamical properties of sea ice through accurate extraction of relevant ice parameters from the images is still quite challenging.

Over the last two decades, some key technologies used for image segmentation involve thresholding and clustering [23], wavelet transform ([30],[31],[32]), watershed transform [29], active contour model ([24], [25]), Gradient Vector Flow (GVF) and snake ([7], [33]), spectral clustering, Markov random field, neural network [51]. Otsu thresholding and k-means clustering are two basic methods which are widely used to separate background and foreground of a grayscale image based on its histogram. Both methods work well for ice image segmentations when the floes are well separated, significantly brighter and have even illumination [38]. A better approach is using an edge detection methods, for example, derivative and morphology edge detection technique, which can detect distinct boundaries. However, these methods fail to identify closed boundaries and separate the ice floes that are tightly connected [38]. Application of watershed-based method can help in this regard to separate connected floes for which the junctions are blurred in the image. This method has been successfully applied in various fields, including grain and cell nuclei image segmentations [39], [40]. However, it can produce over- and under-segmentations of an ice image, as ice floe sizes and shapes vary significantly [41]. An improved watershed method, through inclusion of neighboring region merging algorithm is proposed by [42]. Yet, the problem with accurate identification of floe boundaries and floe numbers remains as watershed transformation operates on binary images and significant amount of real boundary information between connected floes can be lost while using the watershed transformation.

The Gradient Flow Vector (GVF) method and several of its improved variations have been implemented recently by some researchers to tackle the detection of irregular shapes, as it is comparatively faster and less sensitive to initialization [4], [25]. The GVF snake algorithm, combined with automatic contour initialization based on the distance transform is adopted by [7] to separate seemingly connected floes. After that, an ice shape enhancement algorithm is applied to enhance ice floe shapes and accomplish the identification of individual ice floes. However, still some connected ice floes may not be separated by this

method because of the loss of the seeds when the ice floes are unevenly illuminated and closely packed without having clearly visible boundaries.

In summary, edge detection of real field ice images is still challenging, even though numerous approaches of image segmentation have been proposed. None of them are sufficiently robust and efficient, especially with the presence of overlapped and closely connected ice floes and uneven brightness. The primary objective of this work, therefore, is to propose an image processing algorithm that can separate unevenly illuminated and highly irregular ice floes with close connections and overlaps; thus, improving the capabilities of the existing ice image processing methods.

The rest of paper proceeds as follows: Section 2 describes the algorithms and techniques used for this study. In Section 3, the results obtained from the proposed method are described and compared with selected existing works. Finally, concluding remarks including limitations and scopes for future work are presented in Section 4.

2. METHODOLOGY

A typical image processing technique can be broadly subdivided into three major steps. The first step is image acquisition and enhancement, where the digital image is acquired and preprocessed through filtering, adjustment of contrast and brightness, sharpening etc. to improve the image quality. It is a crucial step for successful image processing. The selection of proper preprocessing tools and methodologies also varies greatly based on applications. Preprocessing techniques applied for ice image might not necessarily be applicable for other fields. Following appropriate preprocessing, comes the image analysis which involves edge detection, boundary separation, segmentations etc. The final step is morphological representation, where the relevant important details are extracted from the analyzed image, such as, number and types of separated objects, their sizes and other geometrical features.

In this study, significant effort is given on preprocessing in order to overcome the problem with uneven illumination as well as to ensure better visualization of the connected and overlapped edges during image analysis.

2.1 Image Preprocessing

A multistep preprocessing is used to improve the image for this study.

2.1.1 Brightness adjustment

The first step is improving the illumination of the image. In order to do so, the input colour image is first separated into RGB channels, and each element of these colour matrices are then normalized using all the three corresponding colour values as shown in Eq. 1 as an example for red channel normalization:

$$R_n(i, j) = R_0(i, j) / \sqrt{R_0^2(i, j) + G_0^2(i, j) + B_0^2(i, j)} \quad (1)$$

The three normalized matrices are then combined back to get the colour image with improved illumination, which is then converted to grayscale for further preprocessing.

Images of ice floes at different preprocessing steps are depicted in Figure 1. Figure 1(a) represents a low-resolution segment of ice tank image extracted from one of the experimental analyses performed in [62]. Figure 1(b) shows the default grayscale conversion using the Otsu thresholding method. Figure 1(c) is the grayscale conversion resulting from the improved colour image obtained from normalization. As can be seen, the ice floes are darker here compared to the background. Therefore, the complemented image is generated as shown in Figure 1(d) for further processing.

2.1.2 Contrast Mapping

The contrast of the inverted image is quite poor. Thus, further preprocessing is done to improve the contrast through mapping the histogram. The histogram for Figure 1(d) is confined within 0.3-0.65. The image adjustment is done in such a way so that the shape of the histogram remains similar, however, 1% of the colour data from the original histogram is saturated at low and high intensities in the new image. Figure 1(e) represents the improved version of the resulting grayscale image. This new image is evidently more uniformly illuminated as compared to the default grayscale in Figure 1(b), which prevents over-segmentations of the ice floes as discussed in section 3.

2.1.3 Edge Sharpening

In the final step of preprocessing, the improved grayscale image is processed further to increase the sharpness of the edges. An unsharp masking method is used in this regard, where a seven-by-seven Matrix is used for Gaussian lowpass filter with '3' standard deviation (Equation 2) is used at first to reduce the image's high-frequency components (noises).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

Following that image gradient is calculated using a threshold of 0.1. Threshold specifies the minimum contrast for a pixel to be considered as edge pixel and be sharpened by the unsharp masking. Values closer to 1 allow sharpening only at high-contrast regions, and lower values near to zero helps sharpening relatively smoother regions of the images, thus avoiding sharpening noise in the output image. The final version of the preprocessed image thus obtained, is shown in Figure 1(f).

2.2 Image Analysis

Once the preprocessing is complete, the image is then analyzed by developing an improved version of the model proposed by [7]. GVF snake [83] is applied to efficiently track the boundaries of the ice floes starting from the initial contours

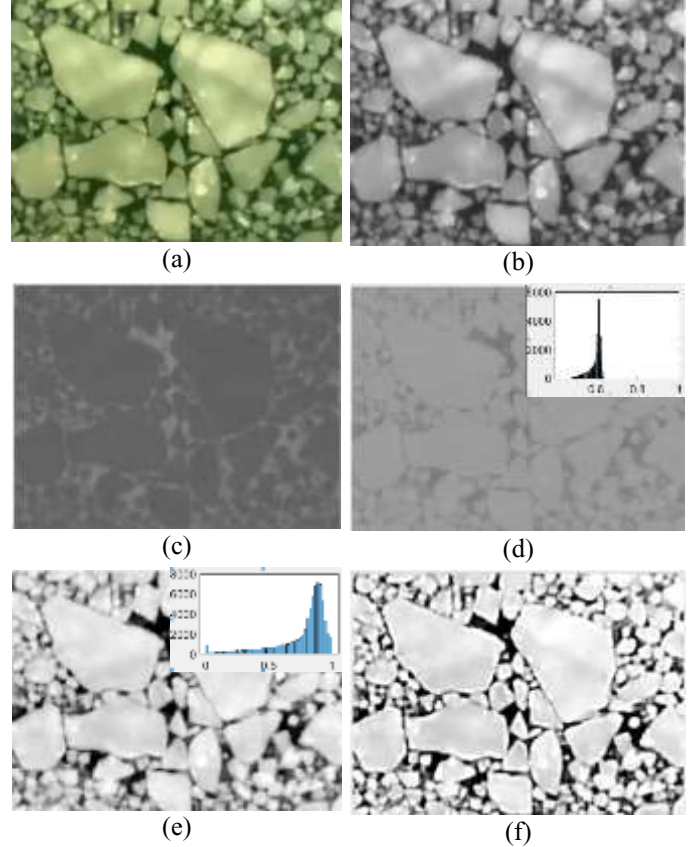


FIGURE 1: Preprocessing of the input ice image: (a) original image, (b) default grayscale conversion, (c) grayscale conversion after normalizing the illumination, (d) colour inversion of 'c', (e) contrast improvement through histogram adjustment, (f) improved grayscale after sharpening of 'e'.

around the seed element. Also, an improvement is added in iteration counting to maximize the computational efficiency. In this method the vector field for image pixels are computed in such a way that minimizes the energy functional:

$$\epsilon = \iint [\mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 [v - \nabla f]^2] dx dy \quad (3)$$

One significant advantage of this method is that it can track and get closer to the true boundary even if the initial contour is not close to the true boundary. However, more iterations will be required to reach the true floe boundary if initial contour is too far away. Another benefit of GVF is that this algorithm operates on grayscale image, thus real boundary information is preserved, unlike in the watershed method.

Once the GVF are calculated, ice floes are labelled after being separated from water using thresholding. Then, each floe is checked whether - the floe area is less than the given threshold, the ice floe has a convex shape (the ratio between the floe area and its minimum bounding polygon area is larger than the

threshold), and the length-to-width ratio of the minimum bounding rectangle of the ice floe is less than the threshold [7].

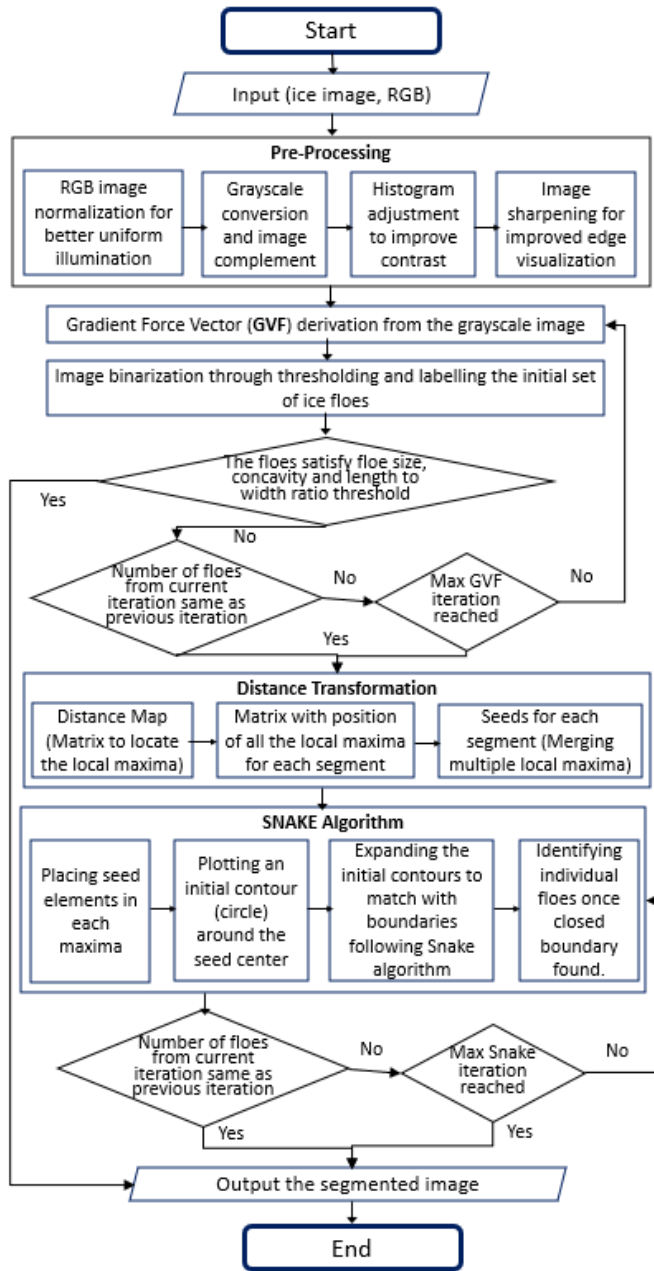


FIGURE 2: Algorithm for the proposed ice image processing technique.

The snake algorithm is then run for those floes which do not satisfy the criteria. Although, GVF snake ensures that a detected boundary is a closed curve, initial contours for the seed elements are required for successful implementation of this method. An automatic contour initialization used in [7] is applied in this

regard with slight modification in iteration counting to improve the efficiency of the proposed ice image segmentation method. This automatic initialization is designed based on distance transform [84] and local maxima of the binary format of the input image. For a binary image I , the distance transform, $D(x,y)$, is the minimum distance from each pixel in I to the background B , which is:

$$D(x,y) = \begin{cases} 0 & \text{if } (x,y) \in B \\ \min_{b \in B} d[(x,y), b] & \text{if } (x,y) \in O \end{cases} \quad (4)$$

Where $d[(x,y), b]$ is the distance measure between pixel (x,y) and b [84]. More details related to these methods can be found in the referred publications.

2.3 Feature Extractions

Once the analysis is completed, the necessary features, for examples, number, size and centers of the ice floes are extracted using histogram and other standard postprocessing tools (bwlable, regionprops, imopen, imclose) available in MATLAB. A flow chart summarizing the methodologies described in sub-sections 2.1 to 2.3 is shown in Figure 2.

3. RESULTS AND DISCUSSION

The proposed image processing model is tested using two types of images, ideal images with desired complexities and real images from ice tank experimental tests.

3.1 Analysis of ideal images

One of the main objectives of this study is to develop a technique which can detect ice floes with various sizes and irregular shapes with sharp concave and convex corners. Two ideal ice floe images are prepared (Figure 3) in this regard to test the efficiency of the developed model.

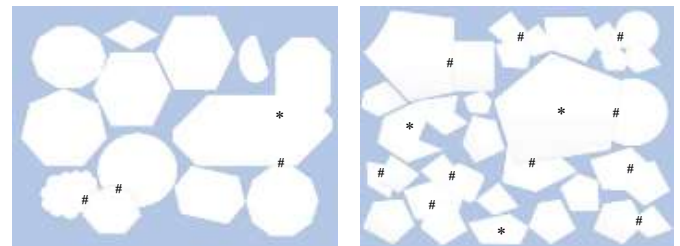


FIGURE 3: Simple (left) and complex (right) ideal ice floe images.

These images are carefully designed to include complex shapes with sharp corners (highlighted using '*'), overlaps and close contacts (highlighted using '#') among adjacent floes. Both these images are processed using the original model proposed by [10] and the improved model presented in this study. The thresholding parameters mentioned in sub-section 2.2, and the iteration numbers are kept the same for both analyses. It should

also be mentioned here that the RGB image normalization has no impact for these ideal images as ice floes are evenly illuminated. As can be seen in Figure 4, for the simple ideal image, the original model was unable to predict one of the closely touched floe sets (top left) and considered the three overlapped floes (bottom left) as one floe. The proposed model on the other hand, detected all the floes with 100% accuracy (12 out of 12 floes).

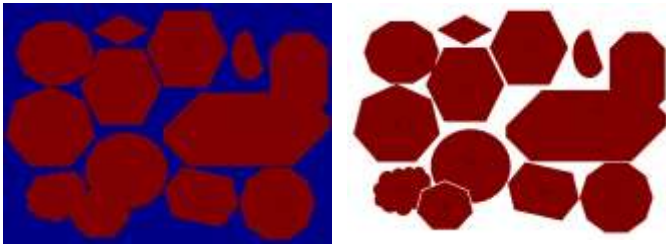


FIGURE 4: Simple ice floe image segmentation: (left) original method [7], (right) proposed model. [identified floes are marked with ‘*’]

Similarly, Figure 5 shows the performance comparison of both methods for the complex ideal image. As noticed, the original method failed to isolate the overlapped floes and combined all the eight floes on the top right corner as one. The same goes for other overlaps near the bottom region of the image. As for the proposed method, there is no under-segmentation, and all the overlaps are detected. However, there appears a few over-segmentations, especially around the big floes at the center. This happened because, the initial contour for this big floe is too far from the true boundary and the set number of iterations are not enough to capture it accurately. The results can be improved with the expense of computational efficiency by increasing number of Snake iteration. However, the level of improvement achieved is not justified compared to the required additional computational time.

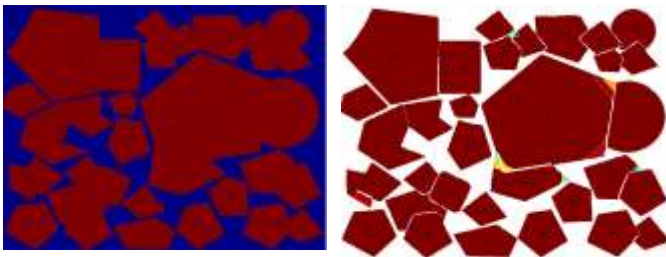


FIGURE 5: Complex ice floe image segmentation: (left) original method [7], (right) proposed model. [identified floes are marked with ‘*’]

3.2 Analysis of real ice tank images

Next, the capability of the proposed model in identifying closely packed ice floes with uneven illumination is tested using two ice image segments extracted from two different ice model testing reported in [62]. These images are shown in Figure 6.

Both images contain noises, uneven illumination of floes, close contact and near overlapping situations, and a variety of floe sizes.

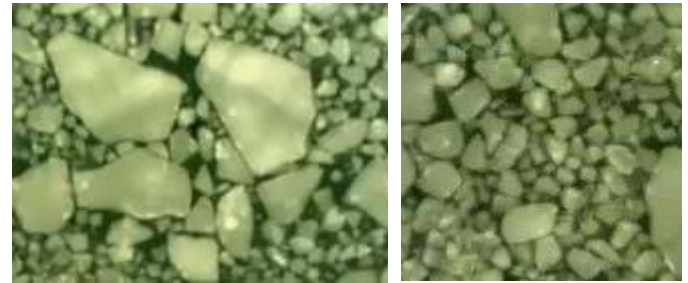


FIGURE 6: Ice floe images from ice tank model tests: (left) image from test 1, (right) image from test 2.

Similar to the analysis for the simple ice floe images, these two images extracted from ice model test are processed using both the original method [10] and the proposed improved model. To make a fair comparison, various model parameters are set as shown in Table 1 for both models.

Table 1: Model parameters for ice floe image analysis

Parameter	Value
Min ice piece area for Snake segmentation	580
Max ice piece area for Snake segmentation	8000
Convexity threshold	0.85
Length to width ratio	2.0
Steeple size	1.0
GVF iteration	800
Snake iteration	200
Min detected ice floe area	20

Figure 7 shows the comparison of the segmented images obtained from the original and the proposed methods. As noticed, especially the bottom right area of the images (highlighted by a rectangle), the original method did well in detecting larger floes with much prominent boundaries. However, it missed many small floes due to colour shading issues. It also contains several over- and under-segmentations due to the presence of noises and weak/connected boundaries. On the other hand, the proposed improved method performed much better in detecting and segmenting both the small and the large floes.

However, it creates an under-segmentation region (in yellow) near the bottom where the resolution and illumination are extremely poor, also over-segmentation of one big floe due to uneven illumination. Nevertheless, the floe counting accuracy

and detection of overall amount of ice concentrations improved significantly when the proposed model is used.

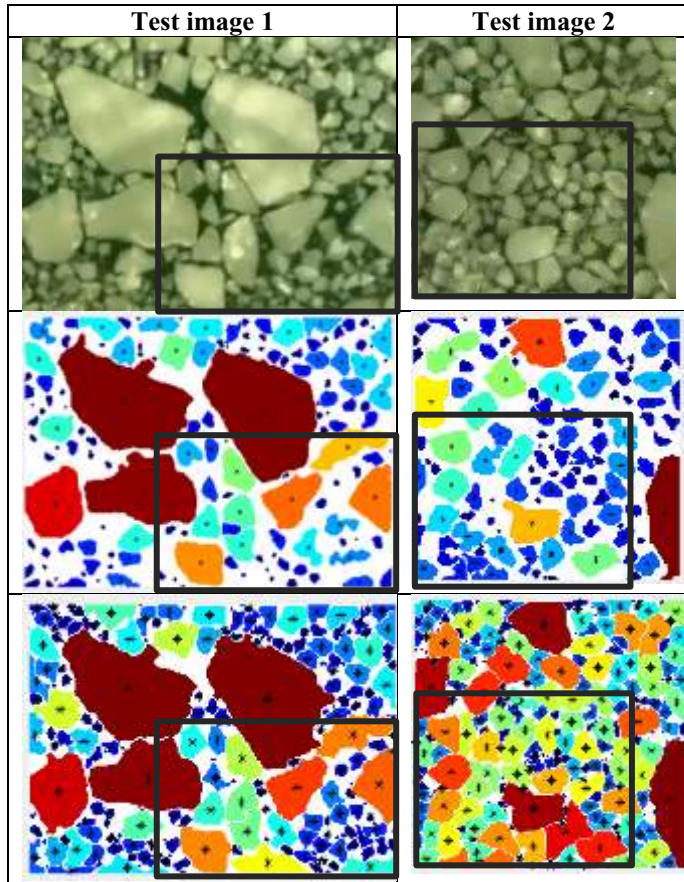


FIGURE 7: Ice floe images segmentation: (top row) images from ice tank test, (middle row) outputs from original method, (bottom row) outputs from proposed method.

Figure 8 compare the histograms of floe counts for both the test images obtained from the original and the proposed method, with the manually marked image results. As can be seen, the original method missed nearly 40 smaller floes (0-500 pixels) as compared to proposed improved method for test image 1. Though, the counting for larger floes is fairly similar for both methods.

For test image 2, on the other hand, the original method was unable to detect many smaller ice floes as compared to the proposed method. This is because, test image 2 is much noisier and contains weaker boundaries as compared to test image 1. The improvement obtained using the proposed method, therefore, is more prominent in this case. For example, a closer look near the lower left corner (highlighted by the rectangle) reveals that density of ice floes detected by the proposed method is higher compared to the original approach. Though, a few under segmentation still exists.

As for histogram comparison for test image 2, it is evident in Figure 8 that the proposed method performed much better in detecting small floes, and in overall floe distribution counting, as compared to the original method. However, as mentioned earlier, presence of over-segmentations for the proposed method is clearly visible in the histogram as well.

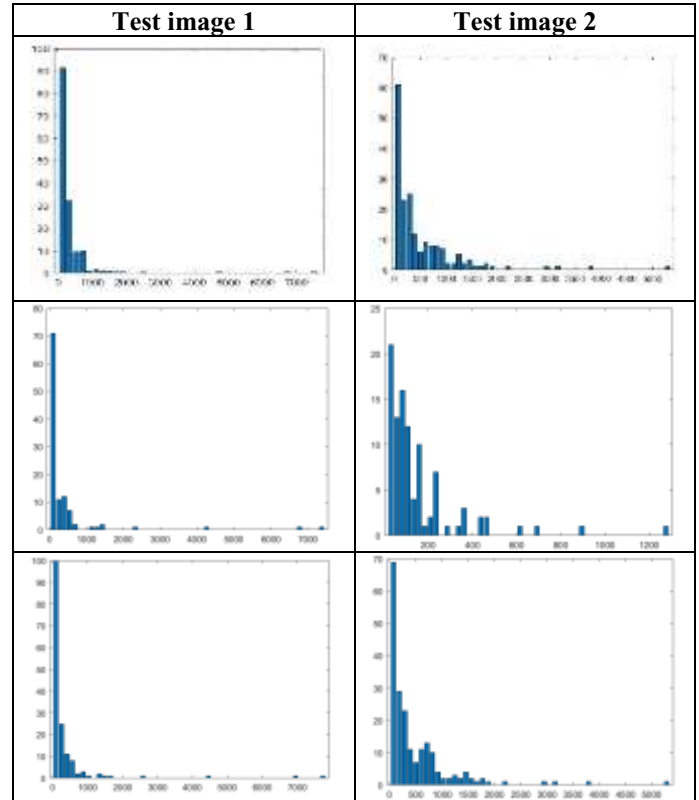


FIGURE 8: Histograms of floe count [no of floes (y axis) vs flow size in pixels (x axis)]: (top row) outputs from the actual image (manually marked edges), (middle row) outputs from original method, (bottom row) outputs from proposed method.

Table 2: Floe count and ice coverage comparison between original and improved models

	Manual observation	Original method	Proposed method
Test image 1			
No of floes	160	111	158
Variation	-	-30.60%	-1.25%
Ice coverage	80.54%	60.21%	76.08%
Test image 2			
No of floes	180	99	201
Variation	-	-45.00%	11.60%
Ice coverage	86.75%	55.63%	78.67%

Table 2 represents a floe count comparison chart for both the methods against floe counted via manual observation. It evidently shows that the new method is performing much better as compared to original method in terms of floe identification in complex situations with non-uniform illuminations, complex floe shapes and weak boundaries. Besides, as can be seen, the ice coverage prediction by the proposed method is also much closer to the actual coverage percentage as compared to the original method.

4. CONCLUSION

Ice detection and segmentation is a complex image processing problem as real ice are most often found in groups with connected and overlapped boundaries. Ice edge detection from images is quite challenging because of their colour similarities, complex and varied shapes, and light reflection on them. Separating every single connected/overlapped ice floe is even more difficult as the edges become weaker in such situations. In this paper, an improved approach for ice segmentation is proposed which can separate connected and overlapped floes in noisy and non-uniform illumination environment more efficiently compared to existing models in the literature. However, there exists significant room for improvements, especially in terms of automation, as some parameters related to ice properties are manually set for the analysis reported here. These parameters need to be automatically tuned as the size and texture of ice fields varies significantly.

ACKNOWLEDGEMENTS

This project was funded under the National Research Council of Canada Collaborative Research and Development program (CSTIP Grant # OCN-205-1). The authors would like to thank National Research Council of Canada for the financial support.

REFERENCES

[1] S. K. Dewangan, "Importance & Applications of Digital Image Processing," *International Journal of Computer Science & Engineering Technology (IJCSET)*, vol. 7, no. 7, pp. 316–320, 2016.

[2] T. Lei, X. Jia, Y. Zhang, L. He, H. Meng, and A. K. Nandi, "Significantly fast and robust fuzzy c-means clustering algorithm based on morphological reconstruction and membership filtering," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 3027–3041, 2018.

[3] D. D. N. de Silva, S. Fernando, I. T. S. Piyatilake, and A. V. S. Karunarathne, "Wavelet based edge feature

enhancement for convolutional neural networks," in *Eleventh International Conference on Machine Vision (ICMV 2018)*, 2019, vol. 11041, p. 110412R.

[4] S. S. Ganesh, K. Mohanaprasad, and Y. Karuna, "Object identification using wavelet transform," *Indian Journal of Science and Technology*, vol. 9, no. 5, pp. 1–7, 2016.

[5] A. Kumar, S. Saha, and R. Bhattacharya, "Wavelet transform based novel edge detection algorithms for wideband spectrum sensing in CRNs," *AEU-International Journal of Electronics and Communications*, vol. 84, pp. 100–110, 2018.

[6] C. R. Jung and J. Scharcanski, "Robust watershed segmentation using wavelets," *Image and Vision Computing*, vol. 23, no. 7, pp. 661–669, Jul. 2005, doi: 10.1016/j.imavis.2005.03.001.

[7] M. Baştan, S. S. Bukhari, and T. Breuel, "Active Canny: edge detection and recovery with open active contour models," *IET Image Processing*, vol. 11, no. 12, pp. 1325–1332, 2017.

[8] Z. Zhang, C. Duan, T. Lin, S. Zhou, Y. Wang, and X. Gao, "GVFOM: a novel external force for active contour based image segmentation," *Information Sciences*, vol. 506, pp. 1–18, Jan. 2020, doi: 10.1016/j.ins.2019.08.003.

[9] Q. Zhang and R. Skjetne, "Image processing for identification of sea-ice floes and the floe size distributions," *IEEE Transactions on geoscience and remote sensing*, vol. 53, no. 5, pp. 2913–2924, 2014.

[10] Q. Zhang and R. Skjetne, *Sea Ice Image Processing with MATLAB®*. CRC Press, 2018.

[11] B. C. Gonçalves and H. J. Lynch, "Fine-Scale Sea Ice Segmentation for High-Resolution Satellite Imagery with Weakly-Supervised CNNs," *Remote Sensing*, vol. 13, no. 18, p. 3562, 2021.

[12] Q. Zhang, S. van der Werff, I. Metrikin, S. Løset, and R. Skjetne, "Image processing for the analysis of an evolving broken-ice field in model testing," in *International Conference on Offshore Mechanics and Arctic Engineering*, 2012, vol. 44939, pp. 597–605.

[13] A. Talukder, D. P. Casasent, H.-W. Lee, P. M. Keagy, and T. F. Schatzki, "Modified binary watershed transform for segmentation of agricultural products," in *Precision Agriculture and Biological Quality*, 1999, vol. 3543, pp. 53–64.

- [14] X. Chen, X. Zhou, and S. T. C. Wong, "Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 4, pp. 762–766, 2006.
- [15] J. D. Blunt, V. Y. Garas, D. G. Matskevitch, J. M. Hamilton, and K. Kumaran, "Image analysis techniques for high Arctic, deepwater operation support," 2012.
- [16] Q. Zhang, R. Skjetne, and B. Su, "Automatic image segmentation for boundary detection of apparently connected sea-ice floes," 2013.
- [17] Q. Zhang, "Image processing for ice parameter identification in ice management," 2015.
- [18] S. Islam et al., "Physical Model Testing for Supporting Ice Force Model Development of DP Vessels in Managed Ice," 2018.
- [19] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on image processing*, vol. 7, no. 3, pp. 359–369, 1998.
- [20] A. Rosenfeld and J. L. Pfaltz, "Distance functions on digital pictures," *Pattern recognition*, vol. 1, no. 1, pp. 33–61, 1968.

Appendix B - Source Code

B.1 Ice floe parameter extraction using image preprocessing and GVF SNAKE (Chapter 03)

MAIN CODE

```
%% Extracting Ice Floe Information Through Traditioanl Image Processing %%  
% This code is an improved version of the work of Zhang, Q. and R. Skjetne,  
% "Image processing for identification of  
% sea-ice floes and the floe size distributions." IEEE Transactions  
% on Geoscience and Remote Sensing, Vol. 53, No. 5, pp. 2913-2924, 2015.
```

```
clc;  
clear all;  
%% parameter setting. The following should be tuned as needed based on the  
% properties of the image set under consideration
```

```
Iin= imread('I.jpg'); % input image
```

```
kms0 = 2; % kmeans
```

```
Ra_min =5000; % minimum ice piece area  
Ra = 400000; % maximum ice piece area  
Rc = 0.8; % convexity threshlod  
Rl = 1.5; % threshold ratio between length and width
```

```
noise=30; % ice piece size in bw image considered as noise  
se = strel('disk', 30); % morphology structure element
```

```
% parameters for GVF Snake
```

```
Num = 150; % number of GVF iterations  
iter = 50; % number of Snake iterations  
% N = 50; % iter of deformation
```

```
% parameters for shape enhancement
```

```
se_th = 200; % threshold for adaptive morphology structure element  
min_floe = 5000; % minimum floe area  
min_brash = 200; % minimum brash area
```

```
% other parameters for GVF Snake, Tune with care (after reading the details  
% from the thesis).
```

```
sigma = 0; % gaussianBlur, not used and CODE REMOVED  
GradientOn = 1; % 1 : Gradient on 0 : Gradient off
```

```
GVFOn = 1; % 1 : GVF 0 : SVF  
mu = 0.5; % parameter for GVF (more noise, increase mu)
```

```
alpha = 0.05; % internal weight that control snake's tension  
beta = 0; % internal weight that control snake's rigidity  
gamma = 1; % step size in one iteration
```

```

kappa = 0.5; % external force weight
Dmin = 0; % min raster of the snake
Dmax = 1; % max raster of the snake

timer = 1;

% Time record parameter
SegTime=[];
SegName=[];

%% End Parameter Setting Section

%% Preprocessing
tic; % START Timer
[I] = normillum(Iin);
toc; %END Timer
SegTime=[SegTime,toc];
SegName=[SegName,"Pre-Processing"];

%% End Preprocessing

%% Image Segmentation Using GVF and SNAKE
[seg, bk,tt,nn] = seaice_kmean_GVF_foreenhancement( I, kms0, sigma, GradientOn, GVFOn,
Num, mu,...
    iter, alpha, beta, gamma, kappa, Dmin, Dmax, Ra_min, Ra, Rc, Rl, se,
timer,noise,SegName,SegTime);
SegTime=[SegTime,tt];
SegName=[SegName,nn];
%% End Segmentation

%% Post-processing - Extracting Floe information
tic;
[out, index_floe, ice_floe, index_brash, brash_ice, index_slush, index_water,
index_residue, coverage] = ...
    ice_shape_enhancement(bk, seg, min_floe, min_brash, se_th);

toc; %END Timer
SegTime=[SegTime,toc];
SegName=[SegName,"Post-Processing"];
%% End Post-processing

ComNamenTimme=[SegName;SegTime]; %% Saving code running time

```

ASSOCIATED FUNCTIONS

normillum

%%% Pre-processing of the input image %%%%

function [I] = normillum(Iin)

Image=Iin;

Image=imresize(Image,2,"bicubic");

Image_rgb =Image;

Image_rgb = double(Image_rgb);

Image_red = Image_rgb(:,:,1);

Image_green = Image_rgb(:,:,2);

Image_blue = Image_rgb(:,:,3);

[row,col] = size(Image_rgb(:,:,1));

for y = 1:row %-->numberof rows in image

for x = 1:col %-->number of columns in the image

Red = Image_red(y,x);

Green = Image_green(y,x);

Blue = Image_blue(y,x);

NormalizedRed = Red/sqrt(Red^2 + Green^2 + Blue^2);

NormalizedGreen = Green/sqrt(Red^2 + Green^2 + Blue^2);

NormalizedBlue = Blue/sqrt(Red^2 + Green^2 + Blue^2);

Image_red(y,x) = NormalizedRed;

Image_green(y,x) = NormalizedGreen;

Image_blue(y,x) = NormalizedBlue;

end

end

Image_rgb(:,:,1) = Image_red;

Image_rgb(:,:,2) = Image_green;

Image_rgb(:,:,3) = Image_blue;

Image_rgb = Image_rgb .* Image_rgb;

Ig=rgb2gray(Image_rgb);

% figure;imshow(Ig);

Ig = imcomplement(Ig);

Ig=imadjust(Ig);

%% Wavelet for Denoising

%%Ref: <https://www.mathworks.com/help/wavelet/ref/wdenoise2.html>

Ig =

wdenoise2(double(Ig),'Wavelet','bior4.4','DenoisingMethod','SURE','NoiseEstimate','LevelDependent','NoiseDirection',['h',

"v","D'],'ThresholdRule','hard');%,'CycleSpinning',4);

% figure

% imshow(I2);

I=Ig;

figure

```

imshow(Ig);
-----

# seaice_kmean_GVF_foreenhancement

%% Snake GVF with automatic initial contour based on local minima derive from
distance transform to detect the floes %%

function [out, bk,tt,nn] = seaice_kmean_GVF_foreenhancement(I, kms0, sigma,
GradientOn, GVFOn, Num, mu,...
    iter, alpha, beta, gamma, kappa, Dmin, Dmax, Ra_min, Ra, Rc, Rl, se,
timer,noise,SegTime,SegName)

% Input data:
% I:          input colour image
% kms0:       k-means clutster number
% sigma = 0:  gaussianBlur is removed from the original code
% GradientOn = 1: 1 : Gradient on  0 : Gradient off

% GVFOn = 1:    1 : GVF  0 : SVF
% Num:          number of GVF iterations
% mu = 0.1:     parameter for GVF (more noise, increase mu)

% iter:        number of Snake iterations
% alpha = 0.05: internal weight that control snake's tension
% beta = 0:    internal weight that control snake's rigidity
% gamma = 1:   step size in one iteration
% kappa = 0.6: external force weight
% Dmin = 0:    min raster of the snake
% Dmax = 1:    max raster of the snake

% Ra_min = 20; minmum area
% Ra:         maximum area
% Rc:         convexity threshlod, <= 1
% Rl:         threshold ratio between length and width, >= 1

% se:        morphology structure element
% timer:     iteration time

% Output data:
% out:       sea ice segmentation image
% bk:        binary ice image by kmeans method

tt=[];
nn=[];

tic;
bw = imbinarize(I,graythresh(I));

figure; subplot(1,1,1); imshow(bw); title('Binarized using Autothresholding');

f=I;
[s1, s2] = size(bw);

```

```

s_1 = [0, s1, s1, 0];
s_2 = [0, 0, s2, s2]; % vertex of image boundary

%% GVF
[ss1, ss2] = size(bw);
out = zeros(ss1, ss2);
[s1, s2] = size(bw);
s_1 = [0, s1, s1, 0];
s_2 = [0, 0, s2, s2]; % vertex of image boundary

% graient on the image in order to get edges
if GradientOn
f2 = abs(gradient2(double(f)));
else
    f2 = f;
end
figure; subplot(1,1,1); imshow(f2,[]); title('Gradient of Pre-processed image');

% vector field
if GVFOn
    % GVF field
    [u, v] = GVF(f2, mu, Num); % gradient vector force
else
    % standard vector field
    [u, v] = gradient2(f2); % calculates standard external force vector
    % field using gradient
end

% normalize vectors in vector field
mag = sqrt(u.*u + v.*v);
px = u ./ (mag + 1e-10);
py = v ./ (mag + 1e-10);

toc; %END Timer
tt=[tt,toc];
nn=[nn,"GVF"];

%% Snake
tic;

bw1 = bw;
bw1=imopen(bw1,strel('disk',noise));
figure; subplot(1,1,1); imshow(bw1); title('After removing small floes');

for time = 1 : timer
    [label, num] = bwlabel(bw1, 4);
    a = zeros(num, 1);
    rc = zeros(num, 1);
    l = zeros(num, 1);
    w = zeros(num, 1);
    for n = 1 : num
        aa = regionprops(label == n, 'Area'); % area
        a(n) = cat(1, aa.Area);
    end
end

```

```

    rcc = regionprops(label == n, 'Solidity'); % convex area
    rc(n) = cat(1, rcc.Solidity);
    ll = regionprops(label == n, 'MajorAxisLength'); % length
    l(n) = cat(1, ll.MajorAxisLength);
    ww = regionprops(label == n, 'MinorAxisLength'); % width
    w(n) = cat(1, ww.MinorAxisLength);
end
r1 = 1 ./ w; % ratio between length and width

% find the component which need to be segmented
k1 = find(a > Ra);

k2 = find(rc < Rc);
k3 = find(r1 > R1);
k = [k1;k2;k3];
k = unique(k);

if length(k) ~= 0

% initial contour
bw2 = zeros(s1, s2);
for m = 1 : length(k)
    pp = find(label == k(m));
    bw2(pp) = 1;
end
bw2 = bwareaopen(bw2, Ra_min);
figure; imshow(bw2,[]);
img_Dist = bwdist(~bw2,'cityblock'); % distance transform
figure; imshow(img_Dist,[]);
imgDist = -img_Dist;
imgDist(~bw2)=-inf;
Dis_img = imregionalmin(imgDist);
dis = Dis_img.*bw2; % local minuma
dis = imdilate(dis, se);
figure; imshow(dis,[]);

[label1, num1] = bwlabel(dis, 8);
t = 0:0.05:6.28;
for n1 = 1 : num1
    cen = regionprops(label1 == n1, 'centroid'); % center
    cen = cat(1, cen.Centroid); % cen(:,1): horizontal, cen(:,2): vertical
    r = abs(img_Dist(round(cen(2))), round(cen(1))) / sqrt(2));
    if r == 0
        r = 2;
    end
    x = double(cen(1) + r * cos(t));
    y = double(cen(2) + r * sin(t));
    [x, y] = snakeinterp(x, y, Dmax, Dmin);
    [x, y] = polybool('intersection', s_2, s_1, x, y);

%     PolyA = polyshape(s_2, s_1,'Simplify',false);
%     PolyB = polyshape(x, y,'Simplify',false);
%     CPoly = intersect(PolyA,PolyB);
%     [x,y] = boundary(CPoly);

```

```

                                % in case polygon vertex is outside of image
                                % x: horizontal, y: vertical

    x = x';
    y = y';

    % snake deformation
    for i=1 : ceil(iter/5)
        if i <= floor(iter/5)
            [x, y] = snakedeform(x, y, alpha, beta, gamma, kappa, px, py, 5);
        else
            [x, y] = snakedeform(x, y, alpha, beta, gamma, kappa, px, py,
iter-floor(iter/5)*5);
        end
        [x, y] = snakeinterp(x, y, Dmax, Dmin);
    end
    xx = ceil(x); yy = ceil(y);
    len = length(xx);
    for i = 1:len
        if xx(i) <= s2 & yy(i) <= s1
            bw1(yy(i),xx(i)) = 0;
        end
    end

    end

    else
        break;
    end

end

toc; %END Timer
tt=[tt,toc];
nn=[nn,"Snake"];

%% kmeans + GVF
% kmeans
si = size(I);
ima = double(I(:));
map0 = kmeans(ima, kms0, 'EmptyAction', 'singleton');

for i = 1 : kms0
    pp = zeros(size(map0));
    pp(map0 == i) = 1;
    s0(i) = sum(ima.*pp)/sum(pp);
end
[A0, ind0] = sort(s0);

bw_kmeans = ones(size(map0));
p = find(map0 == ind0(1));
bw_kmeans(p) = 0;
bw_kmeans = double(reshape(bw_kmeans, si(1), si(2)));

```

```

bk = bw_kmeans;

bk=imopen(bk,strel('disk',noise));

out = bw1;

figure
subplot(1,2,1), imshow(out);
title('Segmented ice floes');
subplot(1,2,2), imshow(bk);
title('Binary ice image by kmeans');

end

-----

# grad

%% GRADIENT Approximate gradient %%
function [xx,yy] = grad(a,xax,yax)
%   [PX,PY] = GRADIENT(Z,DX,DY) returns the numerical partial derivatives
%   of matrix Z in matrices PX = dZ/dx and PY = dZ/dy.  DX and DY
%   may be scalars containing the sample spacing in the X and Y
%   directions, or they may be vectors containing all the explicit
%   locations.
%
%   [PX,PY] = GRADIENT(Z) assumes DX = DY = 1.
%
%   If Y is a vector, GRADIENT(Y) and GRADIENT(Y,DX) return the one
%   dimensional numerical derivative dY/dX.
%
%   For example, try
%       [x,y] = meshgrid(-2:.2:2, -2:.2:2);
%       z = x .* exp(-x.^2 - y.^2);
%       [px,py] = gradient(z,.2,.2);
%       contour(z),hold on, quiver(px,py), hold off
%
%   See also DIFF, DEL2, QUIVER, CONTOUR.

%   Charles R. Denham, MathWorks 3-20-89
%   Copyright (c) 1984-94 by The MathWorks, Inc.

[m,n] = size(a);
if nargin == 1, xax = 1; yax = 1; end
if nargin == 2, yax = xax; end
if length(xax) == 1, xax = xax .* (0:n-1); end
if length(yax) == 1, yax = yax .* (0:m-1); end

y = [];
ax = xax(:).';
for i = 1:2
    x = y;
    [m,n] = size(a);
    y = zeros(m, n);

```



```

j = 1:m;
if n > 1
    d = ax(2) - ax(1);
    y(j, 1) = (a(j, 2) - a(j, 1)) ./ d;    % Left edge.
    d = ax(n) - ax(n-1);
    y(j, n) = (a(j, n) - a(j, n-1)) ./ d; % Right edge.
end
if n > 2
    k = 1:n-2;
    d = ones(m, 1) * (ax(k+2) - ax(k));
    y(j, k+1) = (a(j, k+2) - a(j, k)) ./ d; % Middle.
end
a = a.';
ax = yax(:).';
end
z = (x + sqrt(-1) .* y. ');
if nargout < 2
    xx = z;
else
    xx = real(z); yy = imag(z);
end

```

#GVF

%% GVF Compute gradient vector flow %%

```

function [u,v] = GVF(f, mu, ITER)
% [u,v] = GVF(f, mu, ITER) computes the
% GVF of an edge map f. mu is the GVF regularization coefficient
% and ITER is the number of iterations that will be computed.

```

```

[m,n] = size(f);
fmin = min(f(:));
fmax = max(f(:));
f = (f-fmin)/(fmax-fmin); % Normalize f to the range [0,1]

```

```

f = BoundMirrorExpand(f); % Take care of boundary condition
[fx,fy] = gradient(f); % Calculate the gradient of the edge map
u = fx; v = fy; % Initialize GVF to the gradient
SqrMagf = fx.*fx + fy.*fy; % Squared magnitude of the gradient field

```

```

% Iteratively solve for the GVF u,v
for i=1:ITER,
    u = BoundMirrorEnsure(u);
    v = BoundMirrorEnsure(v);
    u = u + mu*4*del2(u) - SqrMagf.*(u-fx);
    v = v + mu*4*del2(v) - SqrMagf.*(v-fy);
end

```

```

u = BoundMirrorShrink(u);
v = BoundMirrorShrink(v);

```

```

end

```

#snakedeform

```
% SNAKEDEFORM Deform snake in the given external force field
function [x,y] = snakedeform(x,y,alpha,beta,gamma,kappa,fx,fy,ITER)
```

```
% [x,y] = snakedeform(x,y,alpha,beta,gamma,kappa,fx,fy,ITER)
%
% alpha: elasticity parameter
% beta: rigidity parameter
% gamma: viscosity parameter
% kappa: external force weight
% fx,fy: external force field
```

```
% generates the parameters for snake
```

```
N = length(x);
```

```
alpha = alpha* ones(1,N);
beta = beta*ones(1,N);
```

```
% produce the five diagonal vectors
alpham1 = [alpha(2:N) alpha(1)];
alphap1 = [alpha(N) alpha(1:N-1)];
betam1 = [beta(2:N) beta(1)];
betap1 = [beta(N) beta(1:N-1)];
```

```
a = betam1;
b = -alpha - 2*beta - 2*betam1;
c = alpha + alphap1 +betam1 + 4*beta + betap1;
d = -alphap1 - 2*beta - 2*betap1;
e = betap1;
```

```
% generate the parameters matrix
A = diag(a(1:N-2),-2) + diag(a(N-1:N),N-2);
A = A + diag(b(1:N-1),-1) + diag(b(N), N-1);
A = A + diag(c);
A = A + diag(d(1:N-1),1) + diag(d(N),-(N-1));
A = A + diag(e(1:N-2),2) + diag(e(N-1:N),-(N-2));
```

```
invAI = inv(A + gamma * diag(ones(1,N)));
```

```
for count = 1:ITER
    vfx = interp2(fx,x,y,'*linear',0);
    vfy = interp2(fy,x,y,'*linear',0);
```

```
    % deform snake
    x = invAI * (gamma* x + kappa*vfx);
    y = invAI * (gamma* y + kappa*vfy);
end
```

```
-----
```

snakedisp

```
%% SNAKEDISP Initialize the snake %%
function snakedisp(x,y,style)

%     snakedisp(x,y,line)
%     style is same as the string for plot

hold on

% convert to column data
x = x(:); y = y(:);

if nargin == 3
    plot([x;x(1,1)],[y;y(1,1)],style);
    hold off
else
    disp('snakedisp.m: The input parameter is not correct!');
end
```

snakeindex

```
%% SNAKEINDEX Create index for adaptive interpolating the snake %%
function y = snakeindex(IDX)
%     y = snakeindex(IDX)

N = length(IDX);
y=1:0.5:N+0.5;
x=1:N;
y(2*x(IDX==0))=[];
```

snakeinterp

```
%% SNAKEINTERP Interpolate the snake adaptively %%
function [xi,yi] = snakeinterp(x,y,dmax,dmin)

%     [xi,yi] = snakeinterp(x,y,dmax,dmin)
%
%     dmax: the maximum distance between two snake points
%     dmin: the maximum distance between two snake points
%     d(i,i+1)>dmax, then a new point is added between i and i+1
%     d(i,i+1)<dmin, then either i or i+1 is removed
%
%     NOTE: the spacing of original curve must be close to the
%           range defined by dmax and dmin. For arbitrary spacing,
%           try snakeinterp1.
%
%     See also SNAKEINTERP1

% convert to column vector
x = x(:); y = y(:);
```

```

N = length(x);

d = abs(x([2:N 1])- x(:)) + abs(y([2:N 1])- y(:));

% remove the points which distance to neighbor points is shorter than dmin
IDX = (d<dmin);

idx = find(IDX==0);
x = x(idx);
y = y(idx);

N = length(x);
d = abs(x([2:N 1])- x(:)) + abs(y([2:N 1])- y(:));

IDX = (d>dmax);

z = snakeindex(IDX);

p = 1:N+1;

xi = interp1(p,[x;x(1)],z');
yi = interp1(p,[y;y(1)],z');

N = length(xi);
d = abs(xi([2:N 1])- xi(:)) + abs(yi([2:N 1])- yi(:));

while (max(d)>dmax),

    IDX = (d>dmax);
    z = snakeindex(IDX);

    p = 1:N+1;

    xi = interp1(p,[xi;xi(1)],z');
    yi = interp1(p,[yi;yi(1)],z');

    N = length(xi);
    d = abs(xi([2:N 1])- xi(:)) + abs(yi([2:N 1])- yi(:));
end

-----

# xconv2

function Y = xconv2(I,G)
% function Y = xconv2(I,G)
% I: the original image
% G: the mask to be convoluted
% Y: the convoluted result (by taking fft2, multiply and ifft2)
%
% a similar version of the MATLAB conv2(I,G,'same'), 7/10/95
% implemented by fft instead of doing direct convolution as in conv2
% the result is almost same , differences are under 1e-10.
% However, the speed of xconv2 is much faster than conv2 when

```

```

% gaussian kernel has large standard variation.

% Chenyang Xu and Jerry L. Prince, 7/10/95, 6/17/97
% Copyright (c) 1995-97 by Chenyang Xu and Jerry L. Prince
% Image Analysis and Communications Lab, Johns Hopkins University
%

[n,m] = size(I);
[n1,m1] = size(G);
FI = fft2(I,n+n1-1,m+m1-1); % avoid aliasing
FG = fft2(G,n+n1-1,m+m1-1);
FY = FI.*FG;
YT = real(ifft2(FY));
n1 = floor(n1/2);
m1 = floor(m1/2);
Y = YT(1+n1:n+n1,1+m1:m+m1);

-----

# ice_shape_enhancement

%% Post processing and Extract Floe Information %%

function [out, index_floe, ice_floe, index_brash, brash_ice, index_slush, ...
    index_water, index_residue, coverage] = ice_shape_enhancement(bk, seg, min_floe,
min_brash, se_th)

% Input data:
% bk:          binary ice image by kmeans method
% seg:        sea ice segmentation image, the output of function
%             seaice_kmean_GVF
% min_floe:   threshold of minimum size of ice floe
% min_brash:  threshold of minimum size of brash ice
% se_th:      threshold of element structure for morphology operator

% Output data:
% out:        morphology cleaned ice pieces (ice floe & brash ice)
% index_floe: layer of ice floe
% ice_floe:   ice floe information (areas, centers, perimeter, and
%             pixel positions)
% index_brash: layer of brash ice
% brash_ice:  brash ice information (areas, centers, perimeter, and
%             pixel positions)
% index_slush: layer of slush ice
% index_water: layer of water
% index_residue: layer of residue
% coverage:   percentage of floe, brash, slush and water

bw = seg == 1;      % ice segmentation
bw = double(bw);
% k = seg == 0.5;   % dark ice segmenation
% k = double(k);

```

```

l = zeros(size(bw)); % labeled sea ice image
ice_area = [];
[label_bw, nn_bw] = bwlabel(bw, 4);
for i = 1 : nn_bw
    p = find(label_bw == i);
    l(p) = i;
    area0 = length(p);
    ice_area = [ice_area, area0];
end

% k = k - k.*bw;
% [label_k, nn_k] = bwlabel(k, 4);
% for i = 1 : nn_k
%     p = find(label_k == i);
%     l(p) = i + nn_bw;
%     area0 = length(p);
%     ice_area = [ice_area, area0];
% end

[A, ind] = sort(ice_area);

out = zeros(size(bw)); % filled hole, morphology, labeled sea ice image
fill = zeros(size(bw)); % filled hole, labeled sea ice image
t = 0;
for i = 1 : max(max(l))
    p = find(l == ind(i));
    b = zeros(size(out));
    b(p) = 1;

    b = imfill(b, 'hole'); % fill hole

    [ll, kk] = bwlabel(b, 4);
    for j = 1 : kk
        pp = find(ll == j);
        if length(pp) > 0
            fill(pp) = 1;
        end
    end
end

r = length(p);
if r < se_th
    r = 1;
else
    r = 2;
end
se = strel('disk', r);

b = imclose(b, se); % morphology
b = imopen(b, se);
b = imfill(b, 'hole'); % fill hole

[ll, kk] = bwlabel(b, 4);
for j = 1 : kk
    pp = find(ll == j);

```

```

        if length(pp) > 0
            t = t + 1;
            out(pp) = t;
        end
    end
end

%% floe + brash
ice_area = [];           % ice area
floe_area = [];         % floe area
floe_cen = [];          % floe center
colour_floe = [];
brash_area = [];        % brash area
brash_cen = [];         % brash center
colour_brash = [];
index_floe = zeros(size(out)); % floe map
index_brash = zeros(size(out)); % brash map
ice_floe = [];          % structure of ice floe information
brash_ice = [];         % sturcture of brash ice information

for i = 1: max(max(out))
    p = find(out == i);
    area0 = length(p); % area
    [r, c] = find(out == i); % pixel position
    pixels = [c, r];

    if area0 ~= 0
        ice_area = [ice_area, area0];
        colour_label = fix( (1 - exp(-area0/1000)) * 10000 );

        cen = regionprops(out == i, 'centroid'); % center
        cen = cat(1, cen.Centroid);

        per = regionprops(out == i, 'perimeter'); % perimeter
        per = cat(1, per.Perimeter);

        s0 = struct('Center', cen, 'Area', area0, 'Perimeter', per,...
            'PixelsPosition', pixels);

        if area0 > min_floe % ice floe
            index_floe(p) = colour_label;
            floe_area = [floe_area, area0];
            colour_floe = [colour_floe, colour_label];
            floe_cen = [floe_cen; cen];
            ice_floe = [ice_floe; s0];

        elseif area0 > min_brash % brash ice
            index_brash(p) = colour_label;
            brash_area = [brash_area, area0];
            colour_brash = [colour_brash, colour_label];
            brash_cen = [brash_cen; cen];
            brash_ice = [brash_ice; s0];
        end
    end
end
end

```

```

end

index = index_floe + index_brash; % floe + brash
p = find(index ~= 0);

%% slush + water
bk0 = bk;
bk0(p) = 1;

index_slush = bk0;
index_slush(p) = 0; % slush map

index_water = 1 - bk0; % water map

index_residue = ones(size(fill));
index_residue(find(fill ~= 0)) = 0;
index_residue = index_residue.*index_slush; % residue map

%% percentage
floe = sum(floe_area) / (size(out, 1)*size(out, 2));
brash = sum(brash_area) / (size(out, 1)*size(out, 2));

sp = find(index_slush == 1);
slush = length(sp) / (size(out, 1)*size(out, 2));

wp = find(index_water == 1);
water = length(wp) / (size(out, 1)*size(out, 2));

coverage = struct('IceFloe', floe, 'BrashIce', brash, 'Slush', slush, 'Water',
water);
%% rgb
rgb = label2rgb(index, @jet, [1,1,1]);
figure; subplot(1,1,1); imshow(rgb); title('Segmented image (Center highlighted for
bigger floes)');
hold on
for i = 1 : length(floe_cen)
    plot(floe_cen(i, 1), floe_cen(i, 2), 'k*')
end
for i = 1 : length(brash_cen)
    plot(brash_cen(i, 1), brash_cen(i, 2), 'k*')
end
axis off

area_ice = [colour_floe, colour_brash];
colourmap(jet);
n = 6;
d = fix((max(area_ice)-min(area_ice))/n);
ysh = min(area_ice) : d : max(area_ice);
hcb = colourbar;
ytic = get(colourbar, 'Ytick');
set(colourbar, 'YTick', linspace(min(ytic), max(ytic), length(ysh)));
YT = [];
for i = 1 : length(ysh)
    YT{1, i} = -round(1000 * log(1 - ysh(i)/10000));
end

```



```

set(colourbar, 'YTickLabel', YT)

%% histogram
figure,
nbins = 25;
[z, n] = hist(floe_area, nbins);
h = bar(n(1 : nbins), z(1 : nbins));
ch = get(h, 'Children');
fvd = get(ch, 'Faces');
fvcd = get(ch, 'FaceVertexCData');
[zs, ize] = sortrows(z, 1);
k = 255;
colourmap(jet(k));
% for i = 1 : nbins
%     colour(i) = fix( (1 - exp(-n(i)/1000)) * 10000 );
%     fvcd(fvd(i,:)) = colour(i);
% end
% set(ch, 'FaceVertexCData', fvcd)
% colourmap(jet);
% nn = 8;
% d = fix((max(colour_floe)-min(colour_floe))/nn);
% ysh = min(colour_floe) : d : max(colour_floe);
% hcb = colourbar;
% ytic = get(colourbar, 'Ytick');
% set(colourbar, 'YTick', linspace(min(ytic), max(ytic), length(ysh)));
% YT = [];
% for i = 1 : length(ysh)
%     YT{1, i} = -round(1000 * log(1 - ysh(i)/10000));
% end
% set(colourbar, 'YTickLabel', YT)

end

-----

```

B.2 Frame extraction from videos (Chapter 04)

```
%% Extracting Frame from Videos %%

clc; % Clear the command window.
close all; % Close all figures (except those of imtool.)
imtool close all; % Close all imtool figures.
clear; % Erase all existing variables.
workspace; % Make sure the workspace panel is showing.
fontSize = 22;

% The name of the video to be manually changed below.
ReadObj = VideoReader('25m_9ths_1p2kts_0p6m_0deg_001_c_overhead.mkv');
get(ReadObj); %Video properties
numberOfFrames = ReadObj.NumFrames;

Frm = [];
for frame = 1 :20:numberOfFrames % extracting frames at an interval of 20 seconds.
    Frm=[Frm,frame];
end
GetFrame=Frm;

% Reading and saving the frames as jpg images.
CurFrame=0;
while hasFrame(ReadObj)
    CurImage = readFrame(ReadObj);
    CurFrame = CurFrame+1;
    if ismember(CurFrame, GetFrame)
        imwrite(CurImage, sprintf('Image%.4d.jpg', CurFrame));
    end
end

-----
```

B.3 Faster RCNN code for ship and ice floe detection (Chapter 04)

```
% Ship and Ice detection Using Faster R-CNN (regions with convolutional neural
networks) Deep Learning %%

% The detector is developed and trained based on "trainFasterRCNNObjectDetector"
function of Matlab.
% These input parameters should be tuned depending on image properties to
% control the performance of the detector: inputSize, numAnchors,
% 'NegativeOverlapRange','PositiveOverlapRange', trainingOptions.

%% Download Pretrained Detector or Train Own Detector
% |doTraining| variable set to true, as custom detector will be trained.
clear all;
clc;
doTraining = true;
if ~doTraining && ~exist('fasterRCNNResNet50EndToEndVehicleExample.mat','file')
    disp('Downloading pretrained detector (118 MB)...');
    pretrainedURL =
'https://www.mathworks.com/supportfiles/vision/data/fasterRCNNResNet50EndToEndVehicle
Example.mat';
    websave('fasterRCNNResNet50EndToEndVehicleExample.mat',pretrainedURL);
end
%% End Section

%% Load Data Set and Separate to Training, Validation and Test sets
% Datasets are labelled using matlab image labeler app. Labelling should be
% done using rectangular bounding box. More details on dataset is reported
% in the thesis.

data = load('vehicleDatasetGroundTruth.mat');
vehicleDataset = data.data.ShipLabel;
% The vehicle data is stored in a two-column table, where the first column contains
% the image file paths and the second column contains the bounding boxes for ship/ice
floe.

% Split the dataset into training, validation, and test sets. Select 60% of
% the data for training, 10% for validation, and the rest for testing the trained
% detector.
rng(0)
shuffledIndices = randperm(height(vehicleDataset));
idx = floor(0.6 * height(vehicleDataset));

trainingIdx = 1:idx;
trainingDataTbl = vehicleDataset(shuffledIndices(trainingIdx),:);

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = vehicleDataset(shuffledIndices(validationIdx),:);

testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = vehicleDataset(shuffledIndices(testIdx),:);

% Use |imageDatastore| and |boxLabelDatastore| to create datastores for loading
```

```

% the image and label data during training and evaluation.
imdsTrain = imageDatastore(trainingDataTbl{:, 'imageFilename'});
blsTrain = boxLabelDatastore(trainingDataTbl(:, 'vehicle'));
blsTrain2 = boxLabelDatastore(trainingDataTbl(:, 'ice'));

imdsValidation = imageDatastore(validationDataTbl{:, 'imageFilename'});
blsValidation = boxLabelDatastore(validationDataTbl(:, 'vehicle'));
blsValidation2 = boxLabelDatastore(validationDataTbl(:, 'ice'));

imdsTest = imageDatastore(testDataTbl{:, 'imageFilename'});
blsTest = boxLabelDatastore(testDataTbl(:, 'vehicle'));
blsTest2 = boxLabelDatastore(testDataTbl(:, 'ice'));

% Combine image and box label datastores.
trainingData = combine(imdsTrain, blsTrain, blsTrain2);
validationData = combine(imdsValidation, blsValidation, blsValidation2);
testData = combine(imdsTest, blsTest, blsTest2);

% Display one of the training images and box labels.
data = read(trainingData);
I = data{1};
bbox = data{2};
bbox2 = data{4};
annotatedImage = insertShape(I, 'Rectangle', bbox);
annotatedImage = insertShape(annotatedImage, 'Rectangle', bbox2);
annotatedImage = imresize(annotatedImage, 2);
figure
imshow(annotatedImage)
title('Preview one training image');
%% End Section

%% Create Faster R-CNN Detection Network
% A Faster R-CNN object detection network is composed of a feature extraction
% network followed by two subnetworks. The feature extraction network is typically
% a pretrained CNN, such as ResNet-50 or Inception v3. The first subnetwork following
% the feature extraction network is a region proposal network (RPN) trained to
% generate object proposals - areas in the image where objects are likely to exist.
% The second subnetwork is trained to predict the actual class of each object
% proposal.
%
% The feature extraction network is typically a pretrained CNN. This thesis
% uses ResNet-50 for feature extraction. |fasterRCNNLayers| is used to create a
% Faster R-CNN network automatically given
% a pretrained feature extraction network. |fasterRCNNLayers| requires
% specifying several inputs that parameterize a Faster R-CNN network:
% * Network input size
% * Anchor boxes
% * Feature extraction network
%
% When feasible, choose a network input size that is close to the
% size of the training image and larger than the minimum input size [224 224 3]
% required for the network.
% Larger size will increase the computational time.

inputSize = [1134 6096 3]; % This is the input size for ship detection images

```

```

% All images must be resized to this input size in a preprocessing step prior to
training.
%
% Next, use |estimateAnchorBoxes| to estimate anchor boxes based on the size
% of objects in the training data. To account for the resizing of the images prior
% to training, resize the training data for estimating anchor boxes. Use |transform|
% to preprocess the training data, then define the number of anchor boxes and
% estimate the anchor boxes.
preprocessedTrainingData = transform(trainingData,
@(data)preprocessData(data,inputSize));
numAnchors = 4;
anchorBoxes = estimateAnchorBoxes(preprocessedTrainingData,numAnchors)

% Now, use |resnet50| to load a pretrained ResNet-50 model.
featureExtractionNetwork = resnet50;

% Select '|activation_40_relu|' as the feature extraction layer. This feature
% extraction layer outputs feature maps that are downsampled by a factor of 16.
% This amount of downsampling is a good trade-off between spatial resolution and
% the strength of the extracted features, as features extracted further down the
% network encode stronger image features at the cost of spatial resolution.
featureLayer = 'activation_40_relu';

% Define the number of classes to detect.
numClasses = width(vehicleDataset)-1; % For this thesis work, number of class was 1,
either icefloe or ship.

% Create the Faster R-CNN object detection network.
lgraph =
fasterRCNNLayers(inputSize,numClasses,anchorBoxes,featureExtractionNetwork,featureLayer);

% You can visualize the network using |analyzeNetwork| or Deep Network Designer
% from Deep Learning Toolbox™.
%
% If more control is required over the Faster R-CNN network architecture, use
% Deep Network Designer to design the Faster R-CNN detection network manually.

%% End Section

%% Data Augmentation
% Data augmentation is used to improve network accuracy by randomly transforming
% the original data during training. By using data augmentation, you can add more
% variety to the training data without actually having to increase the number
% of labeled training samples.
%
% Use |transform| to augment the training data by randomly flipping the image
% and associated box labels horizontally. Data augmentation is not applied
% to test and validation data. Ideally, test and validation data are representative
% of the original data and are left unmodified for unbiased evaluation.

augmentedTrainingData = transform(trainingData,@augmentData);

```

```

% Read the same image multiple times and display the augmented training data.
augmentedData = cell(4,1);
for k = 1:4
    data = read(augmentedTrainingData);
    augmentedData{k} = insertShape(data{1},'Rectangle',data{2},'Rectangle',data{4});
    reset(augmentedTrainingData);
end
figure
montage(augmentedData,'BorderSize',10)

%% End Section

%% Preprocess Training Data
% Preprocess the augmented training data, and the validation data to prepare
% for training.
trainingData =
transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));
validationData = transform(validationData,@(data)preprocessData(data,inputSize));

% Read the preprocessed data.
data = read(trainingData);

% Display the image and box bounding boxes.
I = data{1};
bbox = data{2};
bbox2=data{3};
annotatedImage = insertShape(I,'Rectangle',bbox);
annotatedImage = insertShape(annotatedImage,'Rectangle',bbox2);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)
title('Viewing one preprocessed training data');

%% End section

%% Train Faster R-CNN
% Use |trainingOptions| to specify network training options. Set |'ValidationData'|
% to the preprocessed validation data. Set |'CheckpointPath'| to a temporary
location.
% This enables the saving of partially trained detectors during the training process.
% If training is interrupted, such as by a power outage or system failure, you
% can resume training from the saved checkpoint.

options = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize',2,...
    'InitialLearnRate',1e-3,...
    'CheckpointPath',tempdir,...
    'ValidationData',validationData);

% Use |trainFasterRCNNObjectDetector| to train Faster R-CNN object detector
% if |doTraining| is true. Otherwise, load the pretrained network.

if doTraining
    % Train the Faster R-CNN detector.

```

```

% * Adjust NegativeOverlapRange and PositiveOverlapRange to ensure
% that training samples tightly overlap with ground truth.
[detector, info] = trainFasterRCNNObjectDetector(trainingData,lgraph,options, ...
    'NegativeOverlapRange',[0 0.3], ...
    'PositiveOverlapRange',[0.6 1]);
else
    % Load pretrained detector for the example.
    pretrained = load('fasterRCNNResNet50EndToEndVehicleExample.mat');
    detector = pretrained.detector;
end

% Detail on training time is provided in the thesis.
% As a quick check, run the detector on one test image. Make sure you resize
% the image to the same size as the training images.

I = imread(testDataTbl.imageFilename{3});
I = imresize(I,inputSize(1:2));
[bboxes,scores] = detect(detector,I);

% Display The Results.
scores; % The confidence score for detection results.
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)
title('Running detector on one test image to check image size and box location');

savenet = detector; % Saving the trained detector for later use.

%% End Section

%% Evaluate Detector Using Test Set
% Evaluate the trained object detector on a large set of images to measure the
% performance. Computer Vision Toolbox™ provides object detector evaluation functions
% to measure common metrics such as average precision (|evaluateDetectionPrecision|)
% and log-average miss rates (|evaluateDetectionMissRate|). For this example,
% use the average precision metric to evaluate performance. The average precision
% provides a single number that incorporates the ability of the detector to make
% correct classifications (precision) and the ability of the detector to find
% all relevant objects (recall).
% Apply the same preprocessing transform to the test data as for the training
% data.

testData = transform(testData,@(data)preprocessData(data,inputSize));
% Run the detector on all the test images.
detectionResults = detect(detector,testData,'MinibatchSize',4);

% Evaluate the object detector using the average precision metric.
[ap, recall, precision] = evaluateDetectionPrecision(detectionResults,testData);

% The precision/recall (PR) curve highlights how precise a detector is at varying
% levels of recall. The ideal precision is 1 at all recall levels. The use of
% more data can help improve the average precision but might require more training
% time. Plot the PR curve.

figure

```

```

plot(recall,precision)
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision = %.2f', ap))

%% End Section

%% Supporting Functions

function data = augmentData(data)
% Randomly flip images and bounding boxes horizontally.
tform = randomAffine2d('XReflection',true);
sz = size(data{1});
rout = affineOutputView(sz,tform);
data{1} = imwarp(data{1},tform,'OutputView',rout);

% Sanitize box data, if needed.
data{2} = helperSanitizeBoxes(data{2}, sz);

% Warp boxes.
data{2} = bboxwarp(data{2},tform,rout);
end

function data = preprocessData(data,targetSize)
% Resize image and bounding boxes to targetSize.
sz = size(data{1},[1 2]);
scale = targetSize(1:2)./sz;
data{1} = imresize(data{1},targetSize(1:2));

% Sanitize box data, if needed.
data{2} = helperSanitizeBoxes(data{2}, sz);

% Resize boxes.
data{2} = bboxresize(data{2},scale);
end

%% References
% [1] Ren, S., K. He, R. Gershick, and J. Sun. "Faster R-CNN: Towards Real-Time
% Object Detection with Region Proposal Networks." _IEEE Transactions of Pattern
% Analysis and Machine Intelligence_. Vol. 39, Issue 6, June 2017, pp. 1137-1149.
%
% [2] Girshick, R., J. Donahue, T. Darrell, and J. Malik. "Rich Feature Hierarchies
% for Accurate Object Detection and Semantic Segmentation." _Proceedings of the
% 2014 IEEE Conference on Computer Vision and Pattern Recognition_. Columbus,
% OH, June 2014, pp. 580-587.
%
% [3] Girshick, R. "Fast R-CNN." _Proceedings of the 2015 IEEE International
% Conference on Computer Vision_. Santiago, Chile, Dec. 2015, pp. 1440-1448.
%
% [4] Zitnick, C. L., and P. Dollar. "Edge Boxes: Locating Object Proposals
% from Edges." _European Conference on Computer Vision_. Zurich, Switzerland,
% Sept. 2014, pp. 391-405.
%
% [5] Uijlings, J. R. R., K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders.

```



```
% "Selective Search for Object Recognition." _International Journal of Computer  
% Vision_. Vol. 104, Number 2, Sept. 2013, pp. 154-171.
```

```
-----
```

ASSOCIATED FUNCTIONS

```
%% Sanitize bounding box data. Called in FasterRCNN %%
```

```
% If none of the boxes are valid, this function passes the data through to  
% enable downstream processing to issue proper errors.
```

```
function boxes = helperSanitizeBoxes(boxes, imageSize)  
persistent hasInvalidBoxes  
valid = all(boxes > 0, 2);  
if any(valid)  
    if ~all(valid) && isempty(hasInvalidBoxes)  
        % Issue one-time warning about removing invalid boxes.  
        hasInvalidBoxes = true;  
        warning('Removing ground truth bounding box data with values <= 0.')    end  
    boxes = boxes(valid,:);  
    boxes = roundFractionalBoxes(boxes, imageSize);  
end
```

```
end
```

```
function boxes = roundFractionalBoxes(boxes, imageSize)  
% If fractional data is present, issue one-time warning and round data and  
% clip to image size.  
persistent hasIssuedWarning
```

```
allPixelCoordinates = isequal(floor(boxes), boxes);  
if ~allPixelCoordinates  
  
    if isempty(hasIssuedWarning)  
        hasIssuedWarning = true;  
        warning('Rounding ground truth bounding box data to integer values.')    end
```

```
    boxes = round(boxes);  
    boxes(:,1:2) = max(boxes(:,1:2), 1);  
    boxes(:,3:4) = min(boxes(:,3:4), imageSize([2 1]));  
end  
end
```

```
-----
```

```
%% Code to Read the Trained Detector for Ship and Ice Detection on New images %%
```

```
clc;  
clear all;
```

```
load savenet.mat;
inputSize = [1430 1130 3]; % Define the average input size of the images, should be
the same with the training data to avoid bad detection performance.
I = imread('1.jpg');
I = imresize(I,inputSize(1:2));
[bboxes,scores] = detect(detector,I); % Applying the trained detector
scores; % The confidence scores for the detected bounding boxes.

I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)
title('Visualization of detected ship/ice floes with bounding boxes and associated
scores');
```

B.4 SVM and FFNN for force prediction (Chapter 04)

SVM

```
%% Predict Test Sample Response for SVM Regression Model %%

clear all;
clc;
load tbl % loading the input data
N = size(tbl,1);

% Partition the data into training and test sets. Hold out 10% of the data for
% testing.

rng(10); % For reproducibility
cvp = cvpartition(N,'Holdout',0.1);
idxTrn = training(cvp); % Training set indices
idxTest = test(cvp); % Test set indices

% Train a linear SVM regression model. Standardize the data.
Mdl =
fitrsvm(tbl(idxTrn,:), 'Force', 'KernelFunction', 'gaussian', 'KernelScale', 'auto', 'Standardize', true, 'solver', 'L1QP')
conv=Mdl.ConvergenceInfo.Converged
iter=Mdl.NumIterations

% |Mdl| is a |RegressionSVM| model.

% Predict responses for the test set.

YFit = predict(Mdl,tbl(idxTest,:));

% Create a table containing the observed response values and the predicted response
% values side by side.
table(tbl.Force(idxTest),YFit,'VariableNames',...
      {'ObservedValue','PredictedValue'})
```

FFNN

```
%% Calling the FFNN for Training the Force Predictor %%

clear all;
clc;
load tbl % load the training data
load tblo % Load the testing data

inputs = [tbl.Con'; tbl.Thick'; tbl.Vel'; tbl.FloeNo'; tbl.FloeArea'];
targets=[tbl.Force']; % Defining the target variables from the training data.
```

```

net=feedforwardnet([20,15,10,5]); % Defining the net 4 layers, input, output and 2
hidden layers.
[net,tr] = train(net,inputs,targets); % Training the net

test=[tblo.Con'; tblo.Thick'; tblo.Vel'; tblo.FloeNo'; tblo.FloeArea']; % Testing the
net.
output=net(test); % Saving the predicted forces for the test set.
predct=output';

savenet=net;
save savenet % Saving the trained net for later use.

```

ASSOCIATED FUNCTIONS

feedforwardnet

% FEEDFORWARDNET Feedforward neural network %%

function out1 = feedforwardnet(varargin)

```

% Two (or more) layer feedforward networks can implement any finite
% input-output function arbitrarily well given enough hidden neurons.
% It takes a 1xN vector of N hidden layer sizes, and a backpropagation training
function, and returns
% a feed-forward neural network with N+1 layers.
%
% Input, output and output layers sizes are set to 0. These sizes will
% automatically be configured to match particular data Or the
% user can manually configure inputs and outputs with configure option.
%

```

```

if nargin > 0
    [varargin{:}] = convertStringsToChars(varargin{:});
end

```

```

persistent INFO;
if isempty(INFO), INFO = get_info; end
if (nargin > 0) && ischar(varargin{1}) ...
    && ~strcmpi(varargin{1},'hardlim') && ~strcmpi(varargin{1},'hardlims')
    code = varargin{1};
    switch code
        case 'info'
            out1 = INFO;
        case 'check_param'
            err = check_param(varargin{2});
            if ~isempty(err), nerr.throw('Args',err); end
            out1 = err;
        case 'create'

```

```

        if nargin < 2, error(message('nnet:Args:NotEnough')); end
        param = varargin{2};
        err = nntest.param(INFO.parameters,param);
        if ~isempty(err), nnerr.throw('Args',err); end
        out1 = create_network(param);
        out1.name = INFO.name;
    otherwise
        % Quick info field access
        try
            out1 = eval(['INFO.' code]);
        catch %#ok<CTCH>
            nnerr.throw(['Unrecognized argument: '' code '''])
        end
    end
end
else
    [args,param] = nnparam.extract_param(varargin,INFO.defaultParam);
    [param,err] = INFO.overrideStructure(param,args);
    if ~isempty(err), nnerr.throw('Args',err,'Parameters'); end
    net = create_network(param);
    net.name = INFO.name;
    out1 = init(net);
end
end

function v = fcversion
    v = 7;
end

function info = get_info
    info = nnfcnNetwork(mfilename,'Feed-Forward Neural Network',fcversion, ...
        [ ...
            nnetParamInfo('hiddenSizes','Hidden Layer
            Sizes','nntype.strict_pos_int_row',10,...
            'Sizes of 0 or more hidden layers. '), ...
            nnetParamInfo('trainFcn','Training Function','nntype.training_fcn','trainlm',...
            'Function to train the network. '), ...
        ]);

    % TODO - hiddenSizes => hiddenSizes
end

function err = check_param(~)
    err = '';
end

function net = create_network(param)

    % Layers
    net = network;
    N1 = length(param.hiddenSizes)+1;
    net.numLayers = N1;
    net.biasConnect = true(N1,1);
    [j,i] = meshgrid(1:N1,1:N1);
    net.layerConnect = (j == (i-1));

```

```

for i=1:Nl
    if i == Nl
        net.layers{i}.name = 'Output';
    else
        if (Nl == 2)
            net.layers{i}.name = 'Hidden';
        else
            net.layers{i}.name = ['Hidden ' num2str(i)];
        end
        net.layers{i}.size = param.hiddenSizes(i);
        net.layers{i}.transferFcn = 'tansig';
    end
    net.layers{i}.initFcn = 'initnw';
end

% Inputs
net.numInputs = 1;
net.inputConnect(1,1) = true;
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};

% Outputs
net.outputConnect(Nl) = true;
net.outputs{Nl}.processFcns = {'removeconstantrows','mapminmax'};

% Training
net.divideFcn = 'dividerand';
net.trainFcn = param.trainFcn;
net.performFcn = 'mse';

% Adaption
net.adaptFcn = 'adaptwb';
net.inputWeights{1,1}.learnFcn = 'learngdm';
net.layerWeights{find(net.layerConnect)'.learnFcn = 'learngdm';
net.biases{:}.learnFcn = 'learngdm';

% Plots
net.plotFcns = iPlotFcns();
end

function plotFcns = iPlotFcns()
if isdeployed
    plotFcns = {};
else
    plotFcns = {'plotperform','plottrainstate','ploterrhist','plotregression'};
end
end

```

--- End of Appendix B ---