Faster R-CNN based CubeSat Close Proximity Detection and Attitude Estimation

in the Department of Aerospace Engineering:

By

N. G. I. Sujeewa Samarawickrama

A Dissertation
Submitted to the Faculty of Mississippi State University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Aerospace Engineering in the Department of Aerospace Engineering

Mississippi State, Mississippi

August 2019
Faster R-CNN based CubeSat Close Proximity Detection and Attitude Estimation
in the Department of Aerospace Engineering:

By

N. G. I. Sujeewa Samarawickrama

Approved:

Yang Cheng
(Major Professor)

Donghoon Kim
(Co-Advisor)

Keith Koenig
(Committee Member)

Yu Lv
(Committee Member)

David S. Thompson
(Graduate Coordinator)

Jason M. Keith
Dean
Bagley College of Engineering
Automatic detection of space objects in optical images is important to close proximity operations, relative navigation, and situational awareness. To better protect space assets, it is very important not only to know where a space object is, but also what the object is. In this dissertation, a method for detecting multiple 1U, 2U, 3U, and 6U CubeSats based on the faster region-based convolutional neural network (Faster R-CNN) is described. CubeSats detection models are developed using Web-searched and computer-aided design images. In addition, a two-step method is presented for detecting a rotating CubeSat in close proximity from a sequence of images without the use of intrinsic or external camera parameters. First, a Faster R-CNN trained on synthetic images of 1U, 2U, 3U, and 6U CubeSats locates the CubeSat in each image and assigns a weight to each CubeSat class. Then, these classification results are combined using Dempster’s rule. The method is tested on simulated scenarios where the rotating 3U and 6U CubeSats are in unfavorable views or in dark environments.
Faster R-CNN detection results contain useful information for tracking, navigation, pose estimation, and simultaneous localization and mapping. A coarse single-point attitude estimation method is proposed utilizing the centroids of the bounding boxes surrounding the CubeSats in the image. The centroids define the line-of-sight (LOS) vectors to the detected CubeSats in the camera frame, and the LOS vectors in the reference frame are assumed to be obtained from global positioning system (GPS). The three-axis attitude is determined from the vector observations by solving Wahba’s problem. The attitude estimation concept is tested on simulated scenarios using Autodesk Maya.

Key words: Faster R-CNN, CubeSats, Close Proximity Detection, Attitude Estimation
DEDICATION

To my parents for their endless love, support, and encouragement.
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Yang Cheng, for his guidance on my research work. I would also like to thank my parents for their support towards pursuing higher education. Also, a special thanks to my committee for their comments on my work.

Permission to reproduce the MSU logo was given by Mississippi State University.
TABLE OF CONTENTS

DEDICATION ......................................................... ii
ACKNOWLEDGEMENTS ................................................... iii
LIST OF TABLES .......................................................... viii
LIST OF FIGURES ........................................................ xi
LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE ....... xv

CHAPTER

1. INTRODUCTION .................................................... 1
   1.1 Background .................................................. 1
   1.2 Object Detection Using Web-Searched Images and CAD-based Im-
       ages .............................................................. 4
       1.2.1 Synthetic Images for Training .......................... 4
       1.2.2 CubeSat Detection Using the Faster R-CNN .......... 5
   1.3 CubeSat Attitude Estimation Using the Detection Results From the
       Faster R-CNN ..................................................... 7
   1.4 Dissertation Organization .................................... 8

2. DEEP LEARNING FOR OBJECT DETECTION .................... 9
   2.1 Background .................................................. 9
       2.1.1 Machine Learning ....................................... 9
       2.1.2 Features .................................................. 11
   2.2 Neural Networks .............................................. 13
   2.3 Deep Neural Networks ....................................... 14
       2.3.1 Back-Propagation ....................................... 16
       2.3.2 Activation Functions ................................... 17
   2.4 Deep Learning ............................................... 18
       2.4.1 Graphics Processing Unit ............................... 18
       2.4.2 Types of GPUs .......................................... 19
2.4.3 Parallel Computing ................................................. 19
2.4.4 GPU Programming ................................................ 19
2.4.5 CUDA ................................................................. 20

2.5 Computer Vision .................................................... 21
2.5.1 Object Detection .................................................. 21

2.6 CNNs for Object Detection ........................................ 23
2.6.1 Convolutional Operation ......................................... 23
2.6.2 Local Receptive Fields ........................................... 27
2.6.3 Pooling Layers .................................................... 29
2.6.4 Fully Connected Layer ........................................... 29

2.7 Images as High-Dimensional Points ............................... 30
2.7.1 Classifier ........................................................... 31

2.8 Region-Based Convolutional for Object Detection .............. 31
2.8.1 Intersection-Over-Union ........................................ 32
2.8.2 Non-Maximum-Suppression ..................................... 33
2.8.3 Region-of-Interest Pooling ..................................... 33
2.8.4 Role of Object Proposals in Object Detection ............... 34

2.9 R-CNN Family .......................................................... 35
2.9.1 R-CNN Architecture ............................................... 36
2.9.1.1 Drawbacks of the R-CNN ................................. 36
2.9.2 Fast R-CNN Architecture ....................................... 37
2.9.3 Faster R-CNN Architecture ..................................... 37

2.10 Training a CubeSat Detection Model on the Faster R-CNN ... 38
2.10.1 Feature Extractors ............................................... 41
2.10.1.1 LeNet ......................................................... 41
2.10.1.2 AlexNet ...................................................... 42
2.10.1.3 ZFNet ......................................................... 42
2.10.1.4 VGGNet ...................................................... 42
2.10.1.5 GoogleNet .................................................... 42
2.10.1.6 ResNet ......................................................... 42
2.10.2 Loss Functions .................................................... 44

2.11 Evaluation Metrics .................................................. 44
2.11.1 Precision-Recall Curves ........................................ 45
2.11.2 Average Precision ............................................... 46

2.12 Alternatives for the Faster R-CNN ................................. 47

3. GPU BASED CUBESAT DETECTION ................................. 48

3.1 Background ........................................................... 48
3.2 Data ................................................................. 49
3.3 Data Augmentation ................................................... 54
3.3.1 Image Annotation .................................................. 54
3.4 Experimental Setup .................................................. 55
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.1</td>
<td>Important Considerations</td>
<td>55</td>
</tr>
<tr>
<td>3.5</td>
<td>Generalization of the Developed CubeSats Detection Models</td>
<td>63</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Improved Detection Results</td>
<td>65</td>
</tr>
<tr>
<td>3.6</td>
<td>Modifications to the RPN Layer</td>
<td>69</td>
</tr>
<tr>
<td>3.7</td>
<td>CAD Models for CubeSats Detection</td>
<td>70</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Analyzing Results of CAD-based CubeSats Models</td>
<td>71</td>
</tr>
<tr>
<td>3.8</td>
<td>Detection Accuracy After Increasing the Number of CubeSat Classes: 1U, 2U, 3U, and 6U CubeSats</td>
<td>73</td>
</tr>
<tr>
<td>3.9</td>
<td>Discussion</td>
<td>78</td>
</tr>
<tr>
<td>4.1</td>
<td>Background</td>
<td>80</td>
</tr>
<tr>
<td>4.2</td>
<td>Two-Step Classification</td>
<td>81</td>
</tr>
<tr>
<td>4.3</td>
<td>Combination of Evidence</td>
<td>83</td>
</tr>
<tr>
<td>4.4</td>
<td>Mass, Belief, and Plausibility</td>
<td>84</td>
</tr>
<tr>
<td>4.5</td>
<td>Mass Assignments for CubeSat Classification</td>
<td>85</td>
</tr>
<tr>
<td>4.6</td>
<td>Dempster’s and Yager’s Rules</td>
<td>89</td>
</tr>
<tr>
<td>4.7</td>
<td>Illustrative Examples</td>
<td>90</td>
</tr>
<tr>
<td>4.8</td>
<td>CubeSat Detection Results</td>
<td>91</td>
</tr>
<tr>
<td>4.8.1</td>
<td>Results of Dempster’s Rules</td>
<td>93</td>
</tr>
<tr>
<td>4.8.2</td>
<td>Results of Yager’s Rules</td>
<td>94</td>
</tr>
<tr>
<td>4.9</td>
<td>Discussion</td>
<td>95</td>
</tr>
<tr>
<td>5.1</td>
<td>Background</td>
<td>97</td>
</tr>
<tr>
<td>5.2</td>
<td>Algorithm Development</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>Illustrative Examples</td>
<td>100</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Object Space Versus World Space</td>
<td>100</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Camera Frame</td>
<td>101</td>
</tr>
<tr>
<td>5.3.2.1</td>
<td>Focal Length</td>
<td>101</td>
</tr>
<tr>
<td>5.3.2.2</td>
<td>Angle of View</td>
<td>102</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulated Scenario 1: 1U CubeSats Detection</td>
<td>102</td>
</tr>
<tr>
<td>5.5</td>
<td>Simulated Scenario 2: 3U CubeSats Detection</td>
<td>104</td>
</tr>
<tr>
<td>5.6</td>
<td>Discussion</td>
<td>107</td>
</tr>
<tr>
<td>6.1</td>
<td>CONCLUSIONS</td>
<td>108</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td></td>
<td>110</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

2.1 Numerical representation for different colors ........................................... 12

2.2 Four validation metrics: positives versus negatives ................................. 45

3.1 CubeSats detection results on different image scales (tested on the Web-
searched augmented image dataset) .......................................................... 56

3.2 Performance with training images batch size (tested on the Web-searched
augmented image dataset) .......................................................................... 57

3.3 CubeSats detection results by varying number of object proposals (tested on
the Web-searched augmented image dataset) ................................................. 57

3.4 Performance of each feature extractor on GTX-1080 GPU (tested on the
Web-searched augmented image dataset) ...................................................... 59

3.5 Detection results on different training methods (trained and tested on the
Web-searched augmented datasets) .............................................................. 63

3.6 Detection results on the improved Web-Searched augmented trained dataset 65

3.7 Detection results after modifying the RPN layer (trained on Web-searched
augmented image dataset) ......................................................................... 67

3.8 CubeSats detection results on CAD-with-texture, Web-searched, and CAD-
no-texture test datasets, Table shows different CubeSats models capabilities
to detect CubeSats when adding and removing features (Extractor: VGG-16) 72
3.9 mAP after increasing the number of CubeSat classes: 1U, 2U, 3U, and 6U in Web-searched dataset

B.1 Conflicting mass functions

B.2 Extremely conflicting mass functions

B.3 Less conflicting mass functions

B.4 Identical mass functions

B.5 Weights in case 1

B.6 Weights in case 2

B.7 Weights in case 3

B.8 Bayes’ rule/Dempster’s rule (all weights, simple sets) in case 1

B.9 Bayes’ rule/Dempster’s rule (all weights, simple sets) in case 2

B.10 Bayes’ rule/Dempster’s rule (all weights, simple sets) in case 3

B.11 Dempster’s rule (one weight, simple sets) in case 1

B.12 Dempster’s rule (one weight, simple sets) in case 2

B.13 Dempster’s rule (one weight, simple sets) in case 3

B.14 Dempster’s rule (all weights, complex sets) in case 1

B.15 Dempster’s rule (all weights, complex sets) in case 2

B.16 Dempster’s rule (all weights, complex sets) in case 3

B.17 Dempster’s rule (one weight, complex sets) in case 1

B.18 Dempster’s rule (one weight, complex sets) in case 2

B.19 Dempster’s rule (one weight, complex sets) in case 3

B.20 Yager’s rule (all weights, simple sets) in case 1
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Computer designed 1U CubeSat structure</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Challenges involved when preparing an image database (See Appendix A.1)</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Computer readable image representation</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Brain cell versus neural network</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Image classification</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>Deep neural networks</td>
<td>15</td>
</tr>
<tr>
<td>2.6</td>
<td>Back-propagation [7]</td>
<td>16</td>
</tr>
<tr>
<td>2.7</td>
<td>Activation functions: the sigmoid, tanh and ReLU</td>
<td>18</td>
</tr>
<tr>
<td>2.8</td>
<td>Deep learning libraries [10]</td>
<td>20</td>
</tr>
<tr>
<td>2.9</td>
<td>Object detection (See Appendix A.1)</td>
<td>22</td>
</tr>
<tr>
<td>2.10</td>
<td>Convolutional operation</td>
<td>24</td>
</tr>
<tr>
<td>2.11</td>
<td>Edge detection using convolutional operation</td>
<td>25</td>
</tr>
<tr>
<td>2.12</td>
<td>Enlarge part of the image for edge detection</td>
<td>25</td>
</tr>
<tr>
<td>2.13</td>
<td>Convolutional neural network</td>
<td>26</td>
</tr>
<tr>
<td>2.14</td>
<td>Feature representations in conv.-3 layer of the AlexNet</td>
<td>28</td>
</tr>
<tr>
<td>2.15</td>
<td>Pooling layer</td>
<td>29</td>
</tr>
<tr>
<td>2.16</td>
<td>Fully connected layers with a CubeSat classifier</td>
<td>30</td>
</tr>
</tbody>
</table>
2.17 Representation of images in high-dimensional feature space 31
2.18 Intersection-over-union 32
2.19 Non-maximum-suppression (See Appendix A.1) 34
2.20 RoI pooling layer 35
2.21 R-CNN [9] 37
2.22 Fast R-CNN [8] 38
2.23 Anchor generation [38] 39
2.24 Faster R-CNN for object detection [38] 39
2.25 VGG architecture [44] 43
2.26 Positives and negatives (See Appendix A.1) 46
3.1 Web-searched images (See Appendix A.1) 50
3.2 Data augmented techniques: rotation, gray, flip, and jitter (See Appendix A.1) 51
3.3 3D-CAD model images 52
3.4 Data annotation tool: LabelImg. All annotated files for positive images are in pascal challenge format 53
3.5 Number of object proposals versus mAP on the Web-searched augmented test image dataset 58
3.6 Memory usage of GTX-1080 GPU by each feature extractor (trained and tested on the Web-searched augmented image datasets) 60
3.7 mAP of two feature extractor (trained and tested on the Web-searched augmented image datasets) 61
3.8 Precision and recall curves for different test datasets 62
LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

AP Average Precision
CAD Computer-aided Design
CNN Convolutional Neural Network
CPU Central Processing Unit
FC Fully Connected
FN False Negative
FP False Positive
GPS Global Positioning System
GPU Graphics Processing Unit
IoU Intersection-over-Union
LOS Line-of-Sight
mAP mean Average Precision
NMS Non-Maximum-Suppression
NN Neural Network
R-CNN Region-based Convolutional Neural Network
ReLU Rectified Linear Unit
RoI Region-of-Interest
RPN Region Proposal Network
SVD Singular Value Decomposition
TN True Negative
TP True Positive
CHAPTER 1
INTRODUCTION

1.1 Background

Space situational awareness usually refers to the process of detection, tracking, and characterization of non-resolved space objects using radar and/or optical sensors. Another interesting class of space situational awareness problems arise from close proximity operations such as rendezvous and docking, space debris capture, and asteroid landing, in which it is crucial for a spacecraft to understand its environments and the objects surrounding it. A basic task towards situational awareness in close proximity is object detection in information-rich images or videos. Therefore, in this dissertation, CubeSats are chosen as the object-of-interest primarily for their standardized configurations, popularity, and concerns about them becoming space debris.

Since 1960, there have been numerous launches of different types of satellites (Sats) aiming at studying different disciplines, mainly focusing on engineering applications and atmospheric chemistry\(^1\). The Aerospace Corporation is one of the major industries that constructed and launched CubeSats for the purpose of technological demonstration\(^2\). The structure of a CubeSat is significant to resolving the problem of space object identification. The CubeSat reference design, proposed by Professor Jordi Puig-Suari from California

\(^1\)See [https://en.wikipedia.org/wiki/CubeSat](https://en.wikipedia.org/wiki/CubeSat)

\(^2\)[http://www.aerospace.org/about-us/]
Polytechnic State University and Professor Bob Twiggs from Stanford University in 1999, aimed to build a spacecraft with similar capabilities to the first spacecraft, Sputnik\textsuperscript{[1]} A CubeSat consists of one or more 10 cm by 10 cm by 11.35 cm units with mass no more than 1.33 kg per unit\textsuperscript{[2]}. In this research, CubeSats with various dimensions are referred as “1U CubeSat,” “3U CubeSat,” “6U CubeSat,” and so on. 1U CubeSats are building blocks of larger CubeSats. Larger CubeSat platforms have been proposed from time to time. Among them, common are the 6U and 12U CubeSats, which are used for academic and technological validation applications\textsuperscript{[1]}. Figure 1.1 shows a computer-aided design (CAD) model of the 1U CubeSat designed by Pumpkin Inc\textsuperscript{[3]}

The feasibility and the success of CubeSats detection missions using deep learning techniques were mainly challenged by the availability of image data on CubeSats. Currently, there is no CubeSats image database available to the vision community. This research was focused on generating new CubeSats databases and development of CubeSats detection models using the faster region-based convolutional neural network (Faster R-CNN) \textsuperscript{[38]}. The main objective of this work is to understand the attainable performances of the Faster R-CNN as a space object detection tool.

Vision-based space object detection is a basic guidance, navigation, and control task for proximity operations such as capture and servicing. Here, the object detection refers to the process of localizing all objects of interest in images or videos (localization) and determining the classes to which the objects belong (classification). Faster R-CNN models described in this research for CubeSat detection were trained and tested on both Web-
searched images and synthetic images rendered using CAD models. A series of sensitivity analyses were performed to measure the accuracy of the developed CubeSats detection models. It showed good mean average precision (mAP) and has the potential to be a general CubeSat detection tool.

Figure 1.1

Computer designed 1U CubeSat structure

Although higher accurate object detection is becoming important, there are many loopholes in current object detection methods. Therefore, robust object detection systems are needed to precisely understand the environment. With the development of the software industry, there are many advances that have been done in the fields of computer vision and machine learning. One major contribution is Faster R-CNN [38] which shows promising results. This context motivated the researcher to participate in this research and to develop robust Faster R-CNN-based object detection models that will perform a key role in the computer vision community.
1.2 Object Detection Using Web-Searched Images and CAD-based Images

The development of image datasets used to train artificial neural networks (NNs) progresses with the computer vision demand. Computer vision is a research field that is used to perform many object detection experiments with image datasets [6, 24, 28, 31, 48, 51]. The use of visual data from the internet is a good source to develop a vision-based system. There is good amount of literature in the computer vision community [29, 32, 47, 56] that has been devoted to designing object detection systems using images’ texture and shape cues. With the recent improvements, the convolutional neural networks (CNNs) have been successfully used on red-green-blue (RGB) images for a variety of tasks [3, 5, 8, 9, 11, 19, 35, 38] in computer vision (e.g., classification and object detection). CNNs have the power of learning features accurately. After learning from a large database like ImageNet [38], CNNs have the ability to generalize the learned features on new image datasets as well. However, there are drawbacks of these systems with the limited data availability. Because of this, researchers were focused on developing detection models using synthetic image data [34, 49, 54].

1.2.1 Synthetic Images for Training

In this subsection, discussions about the combination of two topics in the computer vision community (Faster R-CNN and the three-dimensional (3D) CAD models) to solve a vision task are presented. This research introduces an innovative path (using Autodesk Maya\(^4\)) to render images from 3D-CAD models for Faster R-CNN training purposes. Many

\(^4\)See [https://www.autodesk.com/products/maya/overview](https://www.autodesk.com/products/maya/overview)
researchers tried to use synthetic images to train CNNs due to the lack of training images [15, 33, 45, 53]. There have been a few works published on shape descriptors considering the 3D-CAD data representation [25, 43, 52]. There are many ways to represent the shape information in a vision system [1, 2]. This research had been focused on a few more experiments to increase the accuracy of the object detection process using 3D-CAD-based images. Thereby, it evaluates how texture and shape cues affect to developing accurate Faster R-CNN-based CubeSat models.

This research was focused on both the automatic synthesis procedure and the large collection of 3D-CAD models that scale up the CubeSat detection system with higher generalization ability. When preparing a massive image dataset using 3D-CAD models, the dataset needs to consist of a large number of images with higher variation in features (e.g., color, scale, texture, etc.) to increase the accuracy of training with higher learning capacity [17]. When rendering an image, many different parameters such as different lighting conditions and camera configurations can be used.

1.2.2 CubeSat Detection Using the Faster R-CNN

Given an image, the objective of CubeSat detection is to find all the CubeSats in the image, label them (classification), and draw bounding boxes enclosing them (localization). The detection result contains useful information for navigation, pose estimation etc. For example, a coarse range single-point attitude estimation can be obtained from the centroids of the bounding boxes surrounding the CubeSats in the image. This range information would be unavailable if the CubeSats were just a cloud of featured points. The more
accurate relative pose may be inferred by combining high-level object detection with low-level feature points. Faster R-CNN is a state-of-the-art object detection method capable of near real-time object detection in real-world environments [38]. It uses a region proposal network (RPN) for generating region proposals and uses Fast R-CNN [8] for classifying the proposed regions into object classes and backgrounds [38]. Compared with R-CNN [9] and Fast R-CNN [8], Faster R-CNN significantly reduces the running time by generating proposals using an RPN. In Faster R-CNN, both RPN and Fast R-CNN share a common set of layers. A deep CNN consists of many layers [17], which generate feature maps, downsample features (the pooling layer), and increase the nonlinearity (the rectified linear unit (ReLU) layer) among other functions[5]. Two of the CNN models that have been used in Faster R-CNN are the Zeiler and Fergus model [56] and the Simonyan and Zisserman model [44]. This work aims to understand the CubeSat detection performance of Faster R-CNN models trained on a small set of real CubeSat images collected from the Web and synthetic CubeSat images using 3D-CAD models. The problem is challenging because many images have complex backgrounds and because the appearances and orientations of 3D CubeSats have a wide range of variations. Detection of a 3D object in an image is more difficult than detection or recognition of 2D objects such as hand-written letters, numbers, and traffic signs. Although the training images are not CubeSat images taken in space, good performance of this Faster R-CNN trained on these images will indicate a high likelihood of good performance in space operations.

The CubeSat detection problem is to detect all CubeSats in the image but not mistake other objects or backgrounds for CubeSats. For the sake of simplicity, only four CubeSat classes are assumed in Web-searched CubeSat detection: 1U, 2U, 3U, and 6U. However, the CubeSat detection task can be extended to other types of CubeSats such as 12U and 27U. The input of the problem is a colored or gray-scaled image in which one or more CubeSats are present. No prior information about the CubeSat locations in the image or unique features (e.g., surface patterns and logos) is available. The problem is not to recognize a specific CubeSat but to detect all kinds of CubeSats. The output consists of axis-aligned rectangular bounding boxes and the associated class labels and scores. The scores typically range from 0 to 1. A detection result is considered positive only when the score exceeds a threshold (e.g., 0.7). Detection differs from segmentation in that the latter outputs the boundaries or contours of the objects instead of rectangular bounding boxes.

1.3 CubeSat Attitude Estimation Using the Detection Results From the Faster R-CNN

Increasing the number of CubeSats launches leads to an increase in the number of small Sats in the lower-earth orbit (LEO). This causes an increase in space traffic and the possibility of the collision of Sats and asteroids. This requires more service missions to remove Sats’ debris. To better service these missions, it is important to understand the environment. Therefore, fast and accurate space objects detection and attitude estimation methods need to be developed. Sat’s attitude can be determined by using several different ways. In this research, the singular value decomposition (SVD) method [30] has been
examined to estimates a spacecraft attitude by minimizing Wahba’s loss function [50].

In this dissertation, a coarse range of attitude estimation is obtained for a Sat using the centroids of the bounding boxes of detected CubeSats in the environment.

1.4 Dissertation Organization

The remainder of this dissertation focuses on providing theories and applications necessary for understanding the final results. Chapter 2 provides a comprehensive review of artificial intelligence, deep learning theories, and the Faster R-CNN. Chapter 3 presents collected image datasets which are used to develop CubeSats detection models. Discussions are included in Chapter 3 for experimental setups to achieve goals of developing accurate CubeSats detection models and final detection results outputs by the improved Faster R-CNN. Chapter 4 provides discussions for experiments carried out to detect a CubeSat in close proximity using 3D-CAD models and the Faster R-CNN. Chapter 5 focuses on an attitude estimation problem of a Sat using the detection results from the Faster R-CNN. Finally, Chapter 6 provides a conclusion to the dissertation.
CHAPTER 2
DEEP LEARNING FOR OBJECT DETECTION

2.1 Background

This chapter provides the theoretical background necessary for understanding CNNs and region-based CNNs and includes discussions relevant to machine learning, neural networks, and computer vision which are categorized as artificial intelligence. In addition, the development of a vision system, as a combination of the Faster R-CNN and artificial intelligence, is discussed.

2.1.1 Machine Learning

Computer vision community uses many learning algorithms for vision-based applications. Many of these vision-based applications use images. The basics of machine learning algorithms, which help to develop a computer vision system, are discussed in this subsection.\(^1\) The basic purpose of a machine learning algorithm is to develop models which have artificial intelligence. Machines do not have thinking capabilities to identify objects in the environment accurately. One possible way to teach machines to think is by adding artificial intelligence (conceptual knowledge), which would help machines (e.g., robots) make decision based on the environment. Teaching a machine to have thinking capability (cognitive

\(^1\)See [http://www.deeplearningbook.org/contents/ml.html](http://www.deeplearningbook.org/contents/ml.html)
thinking/conceptual knowledge) is called machine learning\(^2\). With the development of the computational power with massive image, text, and sounds datasets, machine learning has become more practical over the years.

There are two main parts of machine learning: supervised and unsupervised learning\(^3\). Supervised learning uses data that has been annotated (predefined images: training images are marked with the locations and classes of objects-of-interest) according to a prior knowledge. A learning algorithm (e.g., Faster R-CNN) will learn from these predefined images and make predictions of the correct class (e.g., 1U CubeSats, 3U CubeSats) for unseen test images \[^{17}\].

![Figure 2.1](https://commons.wikimedia.org/wiki/File:Figure_2.1.png)

Challenges involved when preparing an image database (See Appendix A.1)[10]

---


\(^{3}\)See [https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d](https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d)
There are many challenges involved in building a computer vision system. Some are related to image datasets. The following are the main challenges associated with developing a vision system using images:

- viewpoint variation: object instance with different camera viewpoints,
- scale variation: variation in the size of the object,
- intra-class variation: different models of the same object,
- occlusion: the object-of-interest can be occluded,
- illumination: the effects of lighting condition,
- background clutter: similarity of the background texture to the object-of-interest.

Figure 2.1 shows some of these challenges. All Web-based CubeSats/non CubeSats images included in this dissertation are obtained through the Google search engine.[5] (See Appendix A.1 copy and paste Web-links in the Web-browser). The second type of machine learning method is unsupervised learning[17]. An example of unsupervised learning is data clustering[6].

2.1.2 Features

Images are the most important part of any vision system. Images are used to teach deep learning algorithms about the environment. A computer reads an image using numerical numbers as shown in Figure 2.2 [7]. In computer vision, converting the images into a
computer readable data format is called “feature extraction”\textsuperscript{8}. The gray-scaled images are represented by a matrix with integers ranging from 0 (black) to 255 (white) \textsuperscript{17}. Color images are represented by values of RGB\textsuperscript{9}. Table 2.1 shows numerical representation for different colors with the assigned number to RGB channels\textsuperscript{10}.

Table 2.1

Numerical representation for different colors

<table>
<thead>
<tr>
<th>Color</th>
<th>Red number</th>
<th>Green number</th>
<th>Blue number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>255</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Purple</td>
<td>255</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>White</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>Black</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 2.2

Computer readable image representation\textsuperscript{8, 11}

\textsuperscript{8}See https://en.wikipedia.org/wiki/Feature_extraction
\textsuperscript{9}https://www.mathworks.com/help/matlab/creating_plots/image-types.html
\textsuperscript{10}https://www.google.com/search?q=1U+cubesat&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjOrP7Zu4PcAhXBJJAKHQD0AisQ_AUICigB&biw=1697&bih=834
2.2 Neural Networks

Neural networks (NNs) are traditional machine learning algorithms which provide acceptable solution to many problems in computer vision. A special case of a neural network named the Faster R-CNN is the primary focus of this chapter. With the introduction of the AlexNet for object classification, CNNs have played a major role in many computer vision related projects. This chapter is mainly focused on the path to explore the ability of the Faster R-CNN in estimating both Web-searched images and 3D-CAD-based images for CubeSats detection challenge. Before diving deep into the Faster R-CNN, it is worth to understand the concept of a regular neural network.

![Brain cell versus neural network](https://isaacchanghau.github.io/post/activation_functions/)

NNs are inspired by the brain. Figure 2.3 shows a comparison between an NN and a brain cell. Figure 2.3 shows the mathematical model of a neuron with an input data

---


12 [https://isaacchanghau.github.io/post/activation_functions/](https://isaacchanghau.github.io/post/activation_functions/)
Each neuron has a weight \( w_i \) and a bias term, \( b \). As shown in Figure 2.3, inputs and weights are linearly combined and summed. The result of this summation is passed through a predefined activation function (e.g., sigmoid) that produces the output \( y(k) \) of the neuron as shown in equation 2.1:

\[
y(k) = f \left( \sum_i w_i x_i + b \right).
\] (2.1)

In the training step, neurons are trained to learn the most acceptable weights to produce the required output for each input. Figure 2.4 shows the visual representation of classifying a CubeSat image using trained weights and bias. In Figure 2.4, every row in the \( W \) matrix represents a single class (1U CubeSat, 2U CubeSat, 3U CubeSat). By formatting the CubeSat image into a vector \( X \) as shown in the Figure 2.4 and matrix calculation between the weight matrix \( W \) and the image vector \( X \), the output, \( f(X; W, b) \) is calculated to find the most accurate class (e.g., green: 1U CubeSat, yellow: 2U CubeSat, and pink: 3U CubeSat).

2.3 Deep Neural Networks

The basic structure of a deep NN is the combination of artificial neurons\(^1\). These neurons are grouped into three layers: input layer, hidden layers, and the output layer.\(^1\) Figure 2.5 shows a flow diagram of a NN.\(^1\) The first layer is the input layer. The input layer passes the image data to the first hidden layer without modifying it.\(^1\) Hidden layers carry the heavy computational tasks.\(^1\) The last layer is the output layer, which takes

\(^1\) See [http://cs231n.github.io/linear-classify/](http://cs231n.github.io/linear-classify/)
Figure 2.4

Image classification

Figure 2.5

Deep neural networks
inputs from the final hidden layer. Some basic calculations are involved in a deep NN of this output conversion process: back-propagation, activation function, and parameter tuning [17].

2.3.1 Back-Propagation

NNs are trained by selecting weights of all neurons where the main goal of this training is to find the best weights for the targeted inputs [15]. In NNs, the technique back-propagation is used to find the optimal values for these weights [17]. Another method for finding effective weights is the gradient descent optimization [38].

Figure 2.6

Back-propagation [7]

15See https://brilliant.org/wiki/backpropagation/
The basic steps in a back-propagation are as follows\footnote{See \url{https://skymind.ai/wiki/backpropagation}}. In the first phase of the algorithm, the weights are randomly initialized. Then in the next step, with the predefined loss function (e.g., mean squared error) the error with the desired output is calculated. The gradient of the loss function is then computed\footnote{\url{https://en.wikipedia.org/wiki/Backpropagation}}. Figure 2.6 shows the basic flow-diagram of the back-propagation \cite{lecun1998}. After calculating the error values using the predefined loss function, the error is back propagated through the network and the weights are updated \cite{lecun1998} (In Figure 2.6, $X_1...X_n$: input data; $W_{ji}, W_{kj}$: randomly initialized weights). Learning rate ratio is used in back-propagation algorithm to control the weights update process\footnote{\url{http://cs231n.github.io/neural-networks-3/}}.

\subsection*{2.3.2 Activation Functions}

The activation function ($f$) gives a nonlinearity to a neural network\footnote{\url{https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092}}. There are many options for an activation function\footnote{\url{https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning}}. Common nonlinear functions are: the sigmoid function, tanh, and rectified linear-unit (ReLU)\footnote{\url{https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning}}. Figure 2.7 shows three main activation functions\footnote{\url{https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning}} commonly used in NNs \cite{lecun1998}. ReLUs have become more common compared to the sigmoid and tanh activation functions due to its flexibility of combining with the back-propagation algorithm\footnote{\url{https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning}}. Equation 2.2 shows the mathematical concept for the ReLU activation:

$$f(x) = \max(0, x).$$

\footnote{\url{https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning}}
2.4 Deep Learning

All NNs used nowadays are deep NNs. The two main issues prevented the training of networks with deep hidden layers are the computational power and unavailability of big data (images, text or sounds). However, within the last few years, deep NNs have come into play with the availability of powerful computers and larger datasets. One milestone was discovering efficient usage of graphics processing units (GPUs) for computer vision-based applications.

2.4.1 Graphics Processing Unit

A GPU is a computer chip that performs rapid mathematical calculations and widely used in processing graphics. Before GPUs come into play the central processing unit (CPU) performs these calculations. Nvidia, AMD, and Intel are some of the major proces-
sors in the GPU market nowadays. With the development of the information technology GPUs have been modified progressively towards implementing any complex algorithm.

2.4.2 Types of GPUs

Different graphics cards are made for different tasks. Nvidia GTX 10-series graphics cards are popular in offering higher speed compared to AMD Radeon RX series. GeForce GTX 1080 Ti, Titan Xp, and Titan V are the best Nvidia GPUs available nowadays. Radeon RX, Vega 56, and Vega 64 are best Radeon graphics cards that use for heavy computational tasks.

2.4.3 Parallel Computing

Parallel computing helps in computing many calculations simultaneously. A GPU is able to solve complex algorithms faster than a CPU because of its parallel processor. CPU can perform simple calculation faster than a GPU as it has a higher clock speed. Therefore, when implementing algorithms on a GPU, it is necessary to consider the suitability of using a GPU because a CPU can be a better solution.

2.4.4 GPU Programming

When it comes to a GPU’s programming, it cannot be done with any language. There is a need to have special libraries that support the device. For example, CUDA only supports Nvidia drives. Figure shows deep learning libraries proposed over time which can

\[\text{See https://en.wikipedia.org/wiki/Graphics_processing_unit}\]
\[\text{https://www.pcgamer.com/the-best-graphics-cards/}\]
\[\text{https://en.wikipedia.org/wiki/Parallel_computing}\]
be used to run on a GPU [10]. The following software’s can be used to run a program on a GPU [10]:

1. CUDA: GPU programming API by Nvidia,
2. OpenCL: multi-vendor version of CUDA,
3. PyCUDA: Python bindings to CUDA driver,
4. PyOpenCL: PyCUDA for OpenCL,
5. Theano: Python-based library,
6. TensorFlow,
7. Trust,

![Deep learning libraries](image)

**2.4.5 CUDA**

CUDA was introduced by Nvidia as a programming interface to GPUs [20]. CUDA refers to a CPU as the host and a GPU as the device, which makes the GPU look like another programmable device [20].
2.5 Computer Vision

Computer vision is a sub-topic in artificial intelligence that extracts useful and meaningful information from images or videos. With the discovery of CNNs, many detection algorithms were developed using deep learning techniques. Among them, the combination of images with CNNs plays a major role.

2.5.1 Object Detection

There are two parts to computer vision, namely classification and object detection. Object detection has a long history in computer vision and is considered a difficult task. When developing a vision-based system, there is a need to focus on the following areas:

- data type,
- accuracy/efficiency, and
- the real-time detection.

When building an accurate computer vision system, there are many barriers to deal with, such as different illumination conditions, viewpoint of the object, and scale of the image, etc. In order to develop an accurate vision system, it is very important to accurately localize the object-of-interest (e.g., CubeSats) in an image. Nowadays there are many new developments to localize an object in an image. The basic steps to develop an accurate object detection system is as follows:

First, it is necessary to define the location and size of the object-of-interest (e.g., CubeSats) using any accurate image annotation.

---

25 See https://en.wikipedia.org/wiki/Computer_vision
26 https://en.wikipedia.org/wiki/Object_detection
Figure 2.9

Object detection (See Appendix A.1)
technique (e.g., bounding box). These annotated images are then classified by using a learning algorithm (e.g., Faster R-CNN) in artificial intelligence techniques \[17\]. Second, the trained models need to test on unseen image data. Figure 2.9 shows an output result of the Faster R-CNN for 1U and 3U CubeSats detection with a high precision. The following sections of this chapter describe the main components of a CNN and how to develop an efficient computer vision system using region-based CNNs.

2.6 CNNs for Object Detection

One of the common uses of CNNs is to classify and detect objects in images. However, the process becomes challenging when it is required to detect several classes of object that vary in different illumination levels and/or occluded objects\[28\]. With the advancement of region-based object proposals by Girshick et.al., \[9\] there were many research works developed in the computer vision community. The most significant work was proposed by Shaoqing et.al., \[38\] which is able to generate object proposals by itself. This contribution is named as the RPN \[38\]. Before going into depth in this region-based CNNs, basic components of a CNN architecture are described in the following subsections.

2.6.1 Convolutional Operation

Convolutional is a widely used technique in image processing that reduces the number of less significant parameters in a CNN\[29\]. Three main components in a CNN that differ from NNs are the pooling layer, local receptive field, and weight sharing \[17\]. Figure 2.10

---

28 See https://gluon.mxnet.io/chapter08_computer-vision/object-detection.html
Convolutional operation shows the convolutional operation with the selected receptive field (e.g., $5 \times 5$ filter) to represent one pixel in the first hidden layer. In the convolutional process, this receptive field runs all over the image similar to a sliding window as shown in Figure 2.10.

In image processing, images can be filtered using convolutional operation to detect different kind of features (e.g., edges, lines, parts of an object, etc.). Figures 2.11 and 2.12 show how a convolutional filter detects edges in an image, functioning similar to a receptive field. Figure 2.12 is an enlarged version of the image which represents different image pixels by different colored squares. The numbers in the matrix in Figures 2.11 and 2.12 represent a image filter that is used to detect edges in the image. The convolutional operation between an image and a filter matrix is defined as:

$$h[x, y] = f[x, y] \cdot g[x, y] = \sum_m \sum_n f[n, m] \cdot g[x - n, y - m].$$  (2.3)
Figure 2.11

Edge detection using convolutional operation

Figure 2.12

Enlarge part of the image for edge detection
In equation 2.3, \( f \) is the image and \( g \) is the filter.\(^{33}\) The size of the receptive field (i.e., activation map) is a user-defined input which can be controlled by the size of the filter matrix \( g \). To avoid image shrinking, each convolutional step needs to be properly padded.\(^{30}\) A common method is “zero padding” as required\(^{30}\) to avoid image shrinkage.

![Convolutional neural network](image)

**Figure 2.13**

Convolutional neural network\(^\text{10}\)

Figure 2.13 shows the basic component of a convolutional neural network. When building a CNN, several convolutional filters can be used depending on the design requirement to detect different features (e.g., edges, lines, parts of an object, etc.) or to get different outputs.\(^{31}\) A standard method of feature extraction is by dividing images into blocks of pixels.\(^{34}\) After dividing images into blocks of pixels, each block may be described by a color, texture, or shape, etc.\(^{35}\) It is worth it to analyze how these features are represented in each layer of a CNN. It is an open question to research experts how to control the

\(^{33}\)See [http://neuralnetworksanddeeplearning.com/index.html](http://neuralnetworksanddeeplearning.com/index.html)


\(^{35}\) [https://en.m.wikipedia.org/wiki/Feature_extraction](https://en.m.wikipedia.org/wiki/Feature_extraction)
features inside a complex CNN to get the desired output. Many researchers are working towards understanding the process behind this “black box.” To increase the understanding inside this black box, in this chapter the researcher evaluated CNNs by visualizing how features are represented in layers of a CNN, providing knowledge to understand the different behaviours of a CNN. Figure 2.14 shows a screen-shot using the Deep Visualization Toolbox\textsuperscript{36} processing a 1U CubeSat. Figure 2.14 shows how the learned features are represented in “conv.3” layer in the AlexNet \textsuperscript{22} architecture for a test CubeSat image. In the first layer of a CNN, it tries to learn basic patterns like lines and edges\textsuperscript{37}. In hidden layers a CNN tries to combine these basic patterns to more high level features such as parts of a CubeSat as shown in Figure 2.14. High-level reasoning (detecting more meaningful patterns for CubeSats) happens in the last layers of a CNN.

### 2.6.2 Local Receptive Fields

If the input image size (length and width) is $a \times a$, then there are $a \times a$ neurons for the first layer of a CNN\textsuperscript{38}. Computational cost increases with the size of the $a$. To avoid such heavy computations, the CNN can make connections in small regions (e.g., $n \times n$, where $n < a$) of the input image\textsuperscript{38}. These small regions in the input image are named as “local receptive fields\textsuperscript{38}.” The concept, “stride” \textsuperscript{17}, will determine the movements of the receptive field on an image at a time.

\textsuperscript{36}See \url{http://yosinski.com/deepvis}

\textsuperscript{37}https://skymind.ai/wiki/neural-network

\textsuperscript{38}http://cs231n.github.io/convolutional-networks/
Figure 2.14

Feature representations in $conv_3$ layer of the AlexNet

- Input image
- High activation
- Artificially synthesize images to the activation
- Images from the training set to activate this neuron the most
- Deconv training images
2.6.3 Pooling Layers

Building a high accurate object detection system requires a high computational cost. Computational cost can be reduced by reducing the size (height and width) of the less significant data volume. The concept “pooling” has been used in CNNs to reduce the less significant data volume. Pooling layers are generally used after the convolutional layers. There are a few options for pooling, namely the max-pooling and average pooling. The best use is the max-pooling. The theory behind the max-pooling is, it outputs the maximum activation in a selected region as illustrated in Figure 2.15.

![Pooling Layer](https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/pooling_layer.html)

2.6.4 Fully Connected Layer

The final layer in a CNN is the fully connected (FC) layer. Each neuron in the FC layer is connected to the entire volume of the last convolutional layer. Figure 2.16 shows Figure 2.16 shows

![Fully Connected Layer](https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/fc_layer.html)
a flow diagram for classification of a CubeSat through FC layers. As shown in Figure 2.16, after the last convolutional layer, the output is connected to the FC layer. By connecting the FC layer with the softmax or support vector machine (SVM)\textsuperscript{41}, it is possible to create a classifier to a deep learning network.

![Flow diagram for classification of a CubeSat through FC layers.](image)

**Figure 2.16**

Fully connected layers with a CubeSat classifier\textsuperscript{7}

### 2.7 Images as High-Dimensional Points

In deep learning, images can be represented in a high-dimensional feature space\textsuperscript{41}. Figure 2.17 shows how images are represented in higher dimensional feature space and how different classes have been classified\textsuperscript{41}. As shown in Figure 2.17, each line (red, green, and blue) visualizes cut-off limits for each class and shows how the term bias ($b$) in equation 2.1 helps to avoid crossing all lines across the (0,0) coordinates of the graph\textsuperscript{41}.

\textsuperscript{41}See [http://cs231n.github.io/linear-classify/](http://cs231n.github.io/linear-classify/)
2.7.1 Classifier

After the FC layer, the final classification happens in a CNN. SVM and softmax are common choices for image classification. A classifier that has been used in a CNN is the softmax function. The softmax function gives a probability (e.g., 1U CubeSat: 0.2 and 2U CubeSat: 0.8) for a certain input class \( y_i \) as shown in equation (2.4):

\[
Q_i = -\log\left( \frac{e^{f_{y_i}}}{\sum_m e^{f_m}} \right)
\]  (2.4)

where the \( f_m \): \( m \)-th element of class scores \( f \), and the loss function is \( Q_i \).

2.8 Region-Based Convolutional for Object Detection

In this section, discussions are provided for different object detection methods which have been utilized in deep learning. In particular, the section includes discussions that
combine CNNs with cost-free object proposals for the object detection challenge \[8,9,38\]. It also includes discussions of how to generate accurate image segments to object detection challenge. In addition, discussion on a few mathematical concepts like intersection-of-union (IoU), non-maximum-suppression (NMS), and region-of-interest (RoI) pooling, which help to region-based CNNs \[9\] are included.

### 2.8.1 Intersection-Over-Union

In R-CNN, the IoU has been used to evaluate the accuracy of predicted object proposals with respect to the ground-truth\[42\]. Equation 2.5 shows the mathematical representation for the IoU \((Q_C)\) where \(Q_A\) is the ground-truth and \(Q_B\) is the predicted proposal \[42\].

\[
Q_C = \frac{(Q_A \cap Q_B)}{(Q_A \cup Q_B)}.
\]  

(2.5)

![Intersection-over-union](https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/)

Figure 2.18

\[42\] See [https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/](https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/)
Figure 2.18 shows a few IoUs of the predicted bounding-box. The predicted bounding-boxes (red) that largely overlap (e.g., Excellent) with the ground-truth bounding-boxes (green) are good proposals for region-based CNNs.

2.8.2 Non-Maximum-Suppression

Another important concept that is commonly used in the R-CNN is the NMS. The concept of the NMS is as follows. Figure 2.19 shows a detection results of a CubeSat. However, the detection results are with a total of three bounding boxes. A good CubeSat classifier should not results many CubeSats detection when there is only one CubeSat. However, this is an acceptable situation as it would be not favorable if the detector either reported:

- a false positive/negatives or
- failed to detect a CubeSat.

In situations like many CubeSats detection, NMS helps to neglect the poor detection (mis-aligned bounding boxes) over the same CubeSat and keep the highest detection probability as shown in Figure 2.19.

2.8.3 Region-of-Interest Pooling

RoI pooling is another concept that has been used in the region-based object detection challenge. It was first proposed by Ross Girshick. It also helps to maintain a high detection accuracy. The RoI layer takes two inputs.

---

2.8.4 Role of Object Proposals in Object Detection

Object detection requires localizing objects within an image. Many object detection algorithms in computer vision community had the requirement of cost-free object proposal methods to get accurate candidate proposals with the likelihood of containing an object [8, 9, 38]. In order to identify an object in the feature map, a detection system needs to have the ability of detect the presence of an object in the feature map [45]. In computer vision,
the likelihood of containing objects (e.g., CubeSats) in a feature map is named as object proposals. In Section 2.9, discussions are provided for fast and efficient cost-free object proposal methods with different CNN architectures developed over time.

2.9 R-CNN Family

Over time, there were many modifications proposed to region-based CNNs. To develop an accurate object detection system, sliding window methods were used in the past. The advancement of the object proposal methods lead to a new chapter in the computer vision community with results in successful CNNs architectures. Sermanet et.al.,

See [https://towardsdatascience.com/review-faster-r-cnn-object-detection-f5685cb30202](https://towardsdatascience.com/review-faster-r-cnn-object-detection-f5685cb30202)
illustrated that the end-to-end trained CNNs architectures can be used to solve object detection challenge efficiently.

One proposed by Girshick et.al., [9] was named R-CNN. With the development of the R-CNN, research began in the object detection field, focusing on reducing computational cost [9, 12]. With the discovery of the category independent object proposal generation step in the Faster R-CNN [38] the computer vision community achieved a milestone. This contribution is called RPN. The latest finding of R-CNN family, the Faster R-CNN, achieves near real-time performance using deep CNNs while generating cost-free object proposals. It does so by using the last shared convolutional layer [38].

### 2.9.1 R-CNN Architecture

All the revolution of object detection happens with the R-CNN architecture. The R-CNN architecture has several steps as shown in Figure 2.21. First, R-CNN will generate RoIs to a given image [9]. The R-CNN uses the method named selective-search [47] to generate the object proposals. As shown in Figure 2.21, the image is down-scaled to match the designed input size of the CNN architecture, and then fed to the CNN [9]. After extracting the features from the input image, the SVM has been used in the R-CNN for the final classification [9]. R-CNN uses ImageNet pre-trained CNN weights to fine tune the CNN on the “pascal_voc” dataset [9].

#### 2.9.1.1 Drawbacks of the R-CNN

Even though the R-CNN is an important method, it has a few drawbacks. The main drawback of the R-CNN is low-speed object detection rates [9]. This is because of the
R-CNN forward computational demand [9]. A solution to drawbacks of the R-CNN is the Fast R-CNN.

2.9.2 Fast R-CNN Architecture

In the Fast R-CNN, an image and RoIs are inputted into a CNN. This paper [8] introduced the “RoI pooling” concept as described in Subsection 2.8.3. The network has two outputs per RoI [38]:

1. softmax probabilities and
2. bounding-box coordinates.

Bounding-box coordinates is a method that outputs four co-ordinates for each object proposal to draw a rectangular box to the detected object [38]. Figure 2.22 illustrates the flow diagram of the Fast R-CNN architecture.

2.9.3 Faster R-CNN Architecture

The main modification in the Faster R-CNN is the author replaced the selective-search method with a neural network named RPN [38]. A flow diagram of the Faster R-CNN is illustrated in Figure 2.24. An RPN takes an image (of any size) as the input. Outputs are a
set of rectangular object proposals; each with an objectness score (likelihood of containing an object on a selected feature map) \([38]\). There are a few convolutional layers that can be used to generate the object proposals. For high accuracy, researchers use the last convolutional layer \([8, 9, 38]\). ZF, VGG-16 network architectures are used as feature extractors in the Faster R-CNN \([38]\). The Faster R-CNN uses three scales and aspect ratios to generate a different number of anchors \([38]\), as shown in Figure 2.23.

### 2.10 Training a CubeSat Detection Model on the Faster R-CNN

In this section, discussions are provided for the training procedure of the Faster R-CNN for the CubeSats detection challenge. The steps to follow to develop an efficient CubeSats detection model are as follows.

1. Collect and annotate images and divide them into a training dataset and a test dataset.

   This is the most labor-intensive step. It is standard practice to augment the datasets by image transformations such as flips, crops, jitters, translations, and rotations. To
Figure 2.23

Anchor generation [38]

Figure 2.24

Faster R-CNN for object detection [38]
the network, these transformations are non-trivial. The purpose of augmentation is to multiply training and test samples and increase diversity. For each image, an annotation file is created that contains the correct class labels and bounding-boxes of all CubeSats in the image. The images and the associated annotation files provide the ground truth for training and performance evaluation.

2. Train the Faster R-CNN model on the training dataset. Training is the process of optimizing the parameters of the model so that it yields the desired bounding-boxes and class labels that is, learning from data what CubeSats look like. Transfer learning is a standard technique to accelerate training and ensure performance. The idea is that the low-level feature extractors of a pre-trained model that has been trained on large datasets are transferable; that is, they are good low-level feature extractors for new datasets, too. By use of a pre-trained model, only the parameters of the problem-specific last layers of the network need to be tuned. The other parameters are frozen during training. A CNN can be trained using the back-propagation and stochastic gradient descent algorithm [17]. The Faster R-CNN, which consists of a RPN and a Fast R-CNN, is trained by either alternating training or approximate joint training [38]. The former alternates between RPN training and Fast R-CNN training. The latter trains the RPN and the Fast R-CNN as one merged network.

3. Test the trained model on the test dataset. The trained network processes the test images one by one. For each image, the model predicted bounding-boxes and class labels are compared with the ground-truth bounding-boxes and class labels in the
annotation files. Intersection-over-union is used to determine how well the predicted and ground-truth bounding-boxes match. A detection result may be true positive, true negative, false positive (false alarm), and false negative (missed detection). Precision is the ratio of the number of true positives to the sum of the true positives and false positives [17]. For each object class, the “average precision” (AP) is calculated. Their average gives the “mean average precision” (mAP). AP and mAP are used as the performance metrics.

2.10.1 Feature Extractors

In all R-CNN architectures, it is necessary to apply a feature extractor to the input image to obtain different kind of features[47]. For each feature extractor, a different number of parameters need to be selected in order to use it in the feature extraction process. Some of them are [21] selecting the filter size, nonlinear units, pooling type, and the layer to generate object proposals, etc. Over time, there were many proposals for feature extractors with many accuracy levels. This subsection provides discussions for a few famous feature extractors that have been used in the object detection challenge.

2.10.1.1 LeNet

The LeNet is a seven-level convolutional network proposed by LeCun et.al., [23] to recognize the hand-written gray-scaled numbers. This technique was limited to detect low-resolution images such as 32×32 pixels.

2.10.1.2 AlexNet

AlexNet \[22\] developed by Krizhevsky et.al., is a milestone in the computer vision community. The AlexNet is much deeper than the LeNet \[22\]. AlexNet uses the softmax layer as an output classifier \[22\].

2.10.1.3 ZFNet

The ILSVRC 2013 winner is known as the ZFNet \[56\]. The Zeiler and Fergus model (ZF) has five shareable convolutional layers.

2.10.1.4 VGGNet

The Simonyan and Zisserman model (VGG-16) \[44\] has 13 shareable convolutional layers. Figure 2.25 shows the different architectures of the VGG model proposed by the Simonyan and Zisserman.

2.10.1.5 GoogleNet

The winner of the ILSVRC 2014 competition is the GoogleNet, proposed by Google \[46\]. This architecture consisted of 22 layers.

2.10.1.6 ResNet

The residual neural network (ResNet) is proposed by Kaiming He et. al., and introduces the concept of “skip connections” \[13\]. The ResNet has a high accuracy which beats the human recognition performance on the ILSVRC 2015 dataset \[48\] \[13\].

\[48\] See https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8
Figure 2.25

VGG architecture [44]
2.10.2 Loss Functions

Least absolute deviations (L1) and least square errors (L2) are two loss functions used to minimized the training error while learning from a dataset. The loss function chosen in the Faster R-CNN [38] is shown in equation 2.6:

$$ Q(a_i, b_i) = \frac{1}{N_{cls}} \sum_i Q_{cls}(a_i, a_i^*) + \lambda \frac{1}{N_{reg}} \sum_i a_i^* Q_{reg}(b_i, b_i^*) $$ (2.6)

where $i$: the index of anchor, $b_i^*$: the ground-truth with reg. loss and cls. loss, $a_i$: the prediction probability, $b_i$: the predicted bounding-box, $\lambda$ is a tunable parameter. The regression head $Q_{reg}$ [38] calculated using the smooth L1 function as shown in equations 2.7 and 2.8

$$ Q_{reg} = smooth_{L1}(b_i - b_i^*) $$ (2.7)

$$ smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise.} \end{cases} $$ (2.8)

2.11 Evaluation Metrics

The common way to measure the performance of many vision systems in image processing is by calculating the metric mAP [38]. There are four validation metrics which are used to measure the performances of a vision system, namely true positive, false positive,
true negative, and false negative. Equation 2.9 defines the precision and equation 2.10 defines the recall:

\[
\text{precision} = \frac{tp}{tp + fp} \tag{2.9}
\]

\[
\text{recall} = \frac{tp}{tp + fn} \tag{2.10}
\]

where \(tp\) : true positives, \(fp\) : false positives and \(fn\) : false negatives. Table 2.2 shows how to quantify these four validation metrics and Figure 2.26 shows the visual representation of these four validation metrics.

Table 2.2

<table>
<thead>
<tr>
<th>Label</th>
<th>Actual positive</th>
<th>Actual negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted positive</td>
<td>true positive (TP)</td>
<td>false positive (FP)</td>
</tr>
<tr>
<td>Predicted negative</td>
<td>false negative (FN)</td>
<td>true negative (TN)</td>
</tr>
</tbody>
</table>

2.11.1 Precision-Recall Curves

By plotting the precision-recall curve for test images it is possible to get an understanding about the accuracy of a vision system. The precision-recall curve is a plot of the precision \(p\) against the recall \(t\).

2.11.2 Average Precision

An accurate method to evaluate the performance of a vision system is by analyzing the precision and recall curves. However, the AP also used to evaluate the performances (as a single number) of a vision system as shown in equation

\[
\int_{0}^{1} p(t) \, dt. \tag{2.11}
\]

AP is equal to sum of the area under the curve as shown in equation

\[
\sum_{q=0}^{M} P(q) \Delta r(q) \tag{2.12}
\]

\[\text{See } \url{https://sanchom.wordpress.com/tag/average-precision/}\]
where $P(q)$ is the precision at a cutoff of $q$ images, and $\Delta r(q)$ is the change in recall between cutoff $q - 1$ and cutoff $q$, $M$ is the total number of images in the collection.\textsuperscript{51}

### 2.12 Alternatives for the Faster R-CNN

You-only-look-once (YOLO)\textsuperscript{52} is a good alternative for the Faster R-CNN. With the higher speed object detection, the YOLO has become a state-of-the-art for near real-time object detection. The YOLO has a low accuracy level of the mAP compared to the Faster R-CNN with higher localization errors. The recurrent neural network (RNN) is another good substitute to the Faster R-CNN. Over the years, researchers have developed more sophisticated types of RNNs. Some of them are bi-directional RNNs, deep (bi-directional) RNNs, and LSTM networks\textsuperscript{53}. In addition, the Mask R-CNN\textsuperscript{54} is another good substitute for the Faster R-CNN\textsuperscript{14}. The focal loss for dense object detection\textsuperscript{26}, single shot detectors\textsuperscript{27}, YOLO9000\textsuperscript{36}, and YOLOv3\textsuperscript{37} methods are also good alternative methods for the Faster R-CNN when considering the near real-time object detection. Many deep learning platforms have been proposed over time to implement CNNs models. The CNTK, TensorFlow, Theano, and PyTorch are some of them\textsuperscript{55}.

\textsuperscript{51}See https://pjreddie.com/darknet/yolo/
\textsuperscript{52}https://en.wikipedia.org/wiki/Recurrent_neural_network
\textsuperscript{53}https://github.com/matterport/Mask_RCNN
\textsuperscript{54}http://dlbench.comp.hkbu.edu.hk/s/pdf/dlbench_v7.pdf
\textsuperscript{55}
CHAPTER 3
GPU BASED CUBESAT DETECTION

3.1 Background

Real-time object detection is crucial for many space-related applications. Recently, Faster R-CNNs have been used as a powerful tool for recognizing image content and are widely considered in the computer vision community [38]. One disadvantage of Faster R-CNNs is that it is computationally demanding, which requires a GPU that requires higher power consumption. In this research, the Faster R-CNN, a state-of-the-art algorithm is applied for CubeSats detection. Latest development the Faster R-CNN achieves near real-time performances using deep networks [38]. Preliminary work of this research has been focused on studying the Faster R-CNN. Many experiments were run to improve detection results, incorporating different datasets from Web-based texture images to CAD images (without and with texture). All Web-based CubeSats images included in this dissertation are obtained through the Google search engine[1] (See Appendix A.1 copy and paste Web-links in the Web-browser).

The primary use of the Faster R-CNN is the detection of objects in images. This is challenging when it comes to CubeSats detection as these objects vary in size with multiples of 10 cm x 10 cm x 11.35 cm units with more similarities. Given an image, the

[1]See https://www.bing.com/images/search?q=Nasa+CubeSats+images&id=5B4B1098CF718AF0F42E61064C313D444938E4E56FORM=IQPRBA
proposed detection system will draw rectangular boxes around CubeSats or CubeSat-like objects and provide probabilities for those objects being CubeSats. Results presented in this section are mainly focused on two CubeSats classes: 1U CubeSat and 3U CubeSat. The deep learning library “Caffe” has been used in this research to develop CubeSats detection models.

3.2 Data

For the preliminary works in this research three new datasets are collected. They are Web-searched CubeSats images, 3D-CAD models-based images without texture, and 3D-CAD models-based images with texture. Following subsections include discussions about datasets, annotation process, and experimental setup in detail.

1. Web-searched dataset

This dataset is collected as a part of the research to detect CubeSats in Web-searched images (See Figure 3.1). The dataset is divided into two formats: (a) original images and (b) augmented images with corresponding annotation files.

Main problems when working with images are the partial observability, scale, and recognition of the correct shape of the object with different viewpoints. However, these problems could be solved by introducing more information such as increasing the training data with various data augmentation techniques. For the first stage of the CubeSats detection research, a range of experiments are conducted by preparing

---

2See http://caffe.berkeleyvision.org/
3https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3349939/
various CubeSats datasets. The results are analyzed by modifying the Faster R-CNN architecture with many fine tuning techniques.

**Augmented Web-Searched Dataset**

This dataset is prepared by incorporating data augmented techniques, as shown in Figure 3.2. The dataset is divided into two:

- original images (970 images) and
- augmented images (9,067 images)

with corresponding annotation files. 10,037 images (with data augmentation) are used for training and four types of image data sets are used for testing.

2. CAD images dataset: without texture

In this experiment to prepare 2D-CAD images, Autodesk FreeCAD has been used\(^4\). Figure 3.3-(a) shows a 3D-CAD model image (without texture) that has been used in this experiment. Same as the Web-searched dataset, there are two formats: (a)

\(^4\)See https://www.freecadweb.org/
Data augmented techniques: rotation, gray, flip, and jitter (See Appendix A.1)

original images (1,570 images) and (b) augmented images (3,021 images) with corresponding annotation files. 4,591 images (with data augmentation) are used for the training set. This dataset contained images of shapes similar to 1U CubeSats and 3U CubeSats without any texture on their surfaces.

3. CAD images dataset: with texture

This dataset is collected using 3D-CAD models to detect CubeSats in Web-searched images. This dataset is prepared by including texture on the surface of the CAD model (See Figure 3.3-(b)). Same as the Web-searched dataset, there are two formats: (a) original images (773 images) and (b) augmented images (8,503 images)
with corresponding annotation files. 9,276 images are used for the training set. The CAD model is obtained from Pumpkin Inc., to prepare this dataset.

4. Test dataset

Test datasets include 255 Web-searched gray images, 317 CAD with texture images, 313 CAD images without texture, and 255 Web-searched color images. In addition, for a fair comparison 1,014 Web-searched test images are prepared using all data augmented techniques. These images are collected to evaluate the ability of the trained CubeSats models on detecting correct CubeSats classes.

See http://www.pumpkinspace.com/about-us.html
Data annotation tool: LabelImg. All annotated files for positive images are in pascal challenge format.

Figure 3.4
3.3 Data Augmentation

Data augmentation techniques rotation, jitter, gray, and flip are used to increase the Web-searched training dataset to 10,037 images with corresponding annotation files. There are many layers in the Faster R-CNN with millions of parameters [38]. Because of this huge number of parameter, a Faster R-CNN model can easily over-fits to a small dataset [38]. A way to overcome over-fitting problem is by applying data augmentation techniques to increase the size of the dataset. One main advantage of using CAD models is that it is possible to augment images in many ways by changing the texture, orientation, scale, etc., of a 3D-CAD model. When preparing CAD datasets, random viewpoints are chosen from 3D-CAD models to prepare a reasonable CubeSat dataset. Chapter describes more experiments on CubeSats detection using 3D-CAD models.

3.3.1 Image Annotation

Training and testing CubeSat datasets are annotated as follows. Inputs to the annotation process are a CubeSat image, predefined classes (e.g., 1U CubeSat, 3U CubeSat), and a user defined bounding box around the object-of-interest in the image. When it comes to image annotation process, two assumptions are made about the images:

1. the object class (e.g., 1U CubeSat, 3U CubeSat) and
2. the location of the object in each image.

Figure shows the annotation process using the “LabelImg” tool. Each CubeSat image is annotated by drawing rectangular boxes for each predefined objects (e.g., Cube-

---

6See https://ieeexplore.ieee.org/document/8388338
7https://www.quora.com/What-is-image-annotation
8https://futurism.com/newly-created-tiny-satellites-are-key-to-space-exploration
9https://github.com/tzutalin/labelImg
Sats) and labeled every object-of-interest with predefined keywords (1U CubeSats, 3U CubeSats) found in its image. All annotated files for positive images are in the pascal challenge format [52].

3.4 Experimental Setup

Faster R-CNN has the capability of learning powerful image data patterns. All these image data patterns are hidden under huge number of parameters [38]. It is worth it to study what these parameters represent to understand the behaviour of the Faster R-CNN. In this chapter, the Faster R-CNN is evaluated on two tasks:

1. Web-searched images-based CubeSats detection and
2. CAD images-based CubeSats detection.

The mAP, precision and recall curves are reported to evaluate the accuracy of trained CubeSats models. Subsection 3.4.1 includes discussions for important considerations that are followed towards developing an accurate CubeSats detection system [16].

3.4.1 Important Considerations

1. How to handle the scale of an image

One of the biggest challenges when training the Faster R-CNN is the scale of images [17]. Sometimes, the CubeSats detection process fails due to the difficulty of detecting CubeSats. One limitation of Web-searched images for CubeSats detection is that there are a limited number of images available for CubeSats. To overcome this situation, one possibility is preparing the CubeSat dataset including CubeSats images with different scales [16]. In order to evaluate how the scale of the image
effects on Web-searched-based CubeSats dataset, the Faster R-CNN is trained at the image scale of 600×1,000 (default scale) and results are tested at different scales [16]. Results show that image scaling largely affects the detection process (See Table 3.1). When changing the scaling of the image, resolution of the image drops by reducing the CubeSat detection probability.

Table 3.1

CubeSats detection results on different image scales (tested on the Web-searched augmented image dataset)

<table>
<thead>
<tr>
<th>Trained data</th>
<th>Trained scale</th>
<th>Extractor</th>
<th>Tested scale</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-searched</td>
<td>600×1,000</td>
<td>VGG-16</td>
<td>300×300</td>
<td>85.0</td>
</tr>
<tr>
<td>Web-searched</td>
<td>600×1,000</td>
<td>VGG-16</td>
<td>300×1,000</td>
<td>93.7</td>
</tr>
<tr>
<td>Web-searched</td>
<td>600×1,000</td>
<td>VGG-16</td>
<td>600×1,000</td>
<td>95.8</td>
</tr>
<tr>
<td>Web-searched</td>
<td>600×1,000</td>
<td>VGG-16</td>
<td>600×600</td>
<td>93.0</td>
</tr>
<tr>
<td>Web-searched</td>
<td>600×1,000</td>
<td>VGG-16</td>
<td>1,000×300</td>
<td>85.0</td>
</tr>
</tbody>
</table>

2. Orientation of the CubeSat in an image

When it comes to CubeSat detection it cannot be expected to see a CubeSat from the same angle all the time. Sometimes, different viewpoints (front, back views) will lead to different images [16]. To evaluate the detection accuracy with different viewpoints, experiments are conducted by incorporating data augmentation techniques (e.g., rotation) to training images and tested CubeSat images at different rotations. Table 3.2 shows that the Faster R-CNN detection accuracy on Web-searched Cube-Sats dataset is increasing when incorporating the data augmentation techniques to training images.
Table 3.2
Performance with training images batch size (tested on the Web-searched augmented image dataset)

<table>
<thead>
<tr>
<th>Method</th>
<th>Extractor</th>
<th>Trained data</th>
<th>1U (%)</th>
<th>3U (%)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end</td>
<td>VGG-16</td>
<td>Web-searched: original (970)</td>
<td>84.8</td>
<td>82.5</td>
<td>83.7</td>
</tr>
<tr>
<td>End-to-end</td>
<td>VGG-16</td>
<td>Web-searched: augmented (10,037)</td>
<td>95.8</td>
<td>95.8</td>
<td>95.8</td>
</tr>
</tbody>
</table>

3. How object proposal matters

In the Faster R-CNN, detection accuracy depends on the number of object proposals [16]. The Faster R-CNN architecture is flexible on choosing the number of object proposals to be sent to the classifier at test-time [38]. Experiments are conducted on the number of the object proposal at test time to find out how the accuracy changes in the Web-searched CubeSats detection system (See Table 3.3). In this experiment, the test-time number of object proposals vary between 10 and 1,000 [16]. Figure 3.5 shows variations of the mAP with a different number of object proposals.

Table 3.3
CubeSats detection results by varying number of object proposals (tested on the Web-searched augmented image dataset)

<table>
<thead>
<tr>
<th>Model</th>
<th>Trained data</th>
<th>Trained proposals</th>
<th>Tested proposals</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>Web-searched</td>
<td>2,000</td>
<td>10</td>
<td>89.8</td>
</tr>
<tr>
<td>VGG-16</td>
<td>Web-searched</td>
<td>2,000</td>
<td>100</td>
<td>95.7</td>
</tr>
<tr>
<td>VGG-16</td>
<td>Web-searched</td>
<td>2,000</td>
<td>200</td>
<td>95.7</td>
</tr>
<tr>
<td>VGG-16</td>
<td>Web-searched</td>
<td>2,000</td>
<td>300</td>
<td>95.8</td>
</tr>
<tr>
<td>VGG-16</td>
<td>Web-searched</td>
<td>2,000</td>
<td>500</td>
<td>95.6</td>
</tr>
<tr>
<td>VGG-16</td>
<td>Web-searched</td>
<td>2,000</td>
<td>1,000</td>
<td>95.6</td>
</tr>
</tbody>
</table>
4. Iterative training

The Faster R-CNN is an iterative method [38]. Experiments are conducted to monitor how the iterative ways improve the accuracy of the CubeSats detection [16]. Multiple networks are trained with two different feature extractors: the VGG-M and VGG-16. VGG-M is a smaller CNN architecture with seven layers [38]. For the Web-searched image dataset, it showed that increasing the number of iterations does not largely help to improve the accuracy of the CubeSats detection model (See Table 3.4). For the VGG-16, it is worth applying early stopping at 70K to prevent unnecessary computation [10]. In addition, the researcher measured the memory consumption of GTX-1080 GPU for the VGG-M and VGG-16 feature extractors. Also,
the size of CubeSats models generated by each extractor is recorded. The graph shown in Figure 3.6 shows that the VGG-16 has higher memory consumption compared to VGG-M feature extractor.

Table 3.4

Performance of each feature extractor on GTX-1080 GPU (tested on the Web-searched augmented image dataset)

<table>
<thead>
<tr>
<th>Type</th>
<th>Trained data</th>
<th>VGG-16</th>
<th>VGG-M</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU memory consumption (MiB)</td>
<td>Web-searched</td>
<td>6,834</td>
<td>1,945</td>
</tr>
<tr>
<td>Size of the model (MB)</td>
<td>Web-searched</td>
<td>546.9</td>
<td>349.8</td>
</tr>
<tr>
<td>Testing time for an image (seconds)</td>
<td>Web-searched</td>
<td>&lt; 0.2</td>
<td>&lt; 0.2</td>
</tr>
<tr>
<td>Iteration 70K (%) mAP (one stage)</td>
<td>Web-searched</td>
<td>95.8</td>
<td>—</td>
</tr>
<tr>
<td>Iteration 80K (%) mAP (two stage)</td>
<td>Web-searched</td>
<td>—</td>
<td>90.6</td>
</tr>
<tr>
<td>Iteration 100K (%) mAP (one stage)</td>
<td>Web-searched</td>
<td>95.2</td>
<td>—</td>
</tr>
</tbody>
</table>

5. Performances of the CubeSats detection process

In this chapter, the mAP [38] is reported for all of the trained CubeSats models. Figure 3.7 shows different mAP values of each class for different feature extractors trained. The detection accuracy of CubeSats models is evaluated by plotting the precision and recall curve. Figure 3.8 shows how the precision and recall curve change with the type of tested image datasets. Figures 3.8-(a), 3.8-(b) and 3.8-(c) show higher accuracy while maintaining a high precision with a high recall compared to Figure 3.8-(d). Figure 3.8-(d) shows that the Web-searched CubeSats detection model shows a very low accuracy when tested on CAD-no-texture CubeSats image dataset compared to other precision and recall curves. Rendered images from 3D-CAD models are lack of realistic nature which significantly reduces the performance
when testing on Web-searched CubeSat model [34]. It is possible to overcome such situations to some extent by adding real image texture on to CAD models [34]. This process is time-consuming and needs supervision to select the appropriate texture for each CubeSat category [34].

Figure 3.6

Memory usage of GTX-1080 GPU by each feature extractor (trained and tested on the Web-searched augmented image datasets)

6. Training methods

To train the Faster R-CNN both the “approximate joint training” (end-to-end) and the “alternating training” [38] methods can be used. Table 3.5 shows results of both training methods. From results on Table 3.5, it shows that deep architectures like VGG-16 trained using the one-stage method learned better than the small architecture like VGG-M which has used the two-stage training method [38]. What makes
Figure 3.7

mAP of two feature extractor (trained and tested on the Web-searched augmented image datasets)

alternating training special is it first trains the RPN and uses the proposals to train the Fast R-CNN [38]. For the alternating training, the learning rate is fixed at 0.001, momentum to 0.9, and trains for 80K iterations, then lowers the learning rate to 0.0001 and trains for another 40K iterations [38]. For the approximate joint training (end-to-end), the learning rate is fixed at 0.0001, momentum to 0.9, and has trained for both 70K and 100K iterations. The IoU threshold for the NMS is fixed at 0.7 to get around 2,000 proposal regions per image [38]. The experiments are conducted on a Dell desktop computer with 32GB RAM, an i7-6700 Intel CPU, and an Nvidia GeForce GTX-1080 GPU. It took below 0.2 seconds to process a test image by all CubeSats models on the GTX-1080 GPU (See Table 3.4). This training process applied widely used Faster R-CNN pre-trained weights for 1,000 object categories on ImageNet [38].
(a) VGG-16 trained and tested on Web-searched augmented image datasets

(b) VGG-16 trained on Web-searched and tested on CAD with texture image datasets

(c) VGG-16 trained on Web-searched images and tested on Web-searched gray image datasets

(d) VGG-16 trained on Web-searched and tested on CAD-no-texture images

Figure 3.8

Precision and recall curves for different test datasets
Table 3.5

Detection results on different training methods (trained and tested on the Web-searched augmented datasets)

<table>
<thead>
<tr>
<th>Method</th>
<th>Extractor</th>
<th>1U (%)</th>
<th>3U (%)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximate (One-stage)</td>
<td>VGG-16</td>
<td>95.8</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>Alternative (Two-stage)</td>
<td>VGG-M</td>
<td>90.7</td>
<td>90.5</td>
<td>90.6</td>
</tr>
</tbody>
</table>

7. Effect of the training batch size

In order to measure the impact of the size of image datasets, two CubeSats models are trained with Web-searched image datasets varying from 970 original images to 10,037 images with data augmentation techniques [16]. VGG-16 feature extractor is evaluated on the training batch size (See Table 3.2). The performances are increased considerably (from mAP of 83.7 percent to 95.8 percent for VGG-16) when increasing the size of training images dataset [16].

3.5 Generalization of the Developed CubeSats Detection Models

When developing a vision-based model, it is important to have a way to measure the accuracy of the developed vision system to handle unseen test data. Supervised learning models which are acquired from a dataset can be categorized into three types:

1. under-fitted,
2. well-trained, and
3. over-fitted models.

Figure 3.9 shows a graphical representation of three type of models which are acquired from a dataset that can be found in supervised learning. Overly simple models named

---

as under-fitted models. Overly complex training methods lead to over-fitted models by learning noisy data, which then leads to bad generalization. With the experiments carried out, the researcher found that some trained CubeSats models are not performing with higher accuracy. The CubeSats detection failed many times when the test image data is made of complex unseen images. One reason for such a high rate of false positives and false negatives is poor data preparation techniques (repeated identical images of original images and lack of data augmented techniques) being used to prepare CubeSats datasets. Because of these poor data preparation techniques, CubeSats detection models are failed to generalize for a wide range of unseen CubeSats images.

With the observed errors in the Web-searched CubeSats dataset, training’s are conducted to improve the CubeSats detection accuracy by incorporating data augmentation techniques and by using filtering methods to remove identical images from the training and testing phases. By this way, it is possible to get a clear understanding about the detection

---

12See http://www.pixelbeat.org/fslint/
tion accuracy of trained CubeSats models and how data preparation techniques affect the training and testing process.

Table 3.6

Detection results on the improved Web-Searched augmented trained dataset

<table>
<thead>
<tr>
<th>Tested data</th>
<th>No. images</th>
<th>Extractor</th>
<th>1U (%)</th>
<th>3U (%)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-searched (Augmented)</td>
<td>1,014</td>
<td>VGG-16</td>
<td>96.2</td>
<td>89.9</td>
<td>93.0</td>
</tr>
<tr>
<td>Web-searched (Original)</td>
<td>255</td>
<td>VGG-16</td>
<td>94.3</td>
<td>93.0</td>
<td>93.7</td>
</tr>
<tr>
<td>Web-searched-gray</td>
<td>255</td>
<td>VGG-16</td>
<td>87.3</td>
<td>86.3</td>
<td>86.8</td>
</tr>
<tr>
<td>CAD-with-texture</td>
<td>317</td>
<td>VGG-16</td>
<td>72.5</td>
<td>86.3</td>
<td>79.4</td>
</tr>
</tbody>
</table>

3.5.1 Improved Detection Results

Test results on the modified Web-searched dataset showed considerably higher improvement. Table 3.6 shows the mAP on the Web-searched CubeSats model tested on four types of datasets. Increasing the number of images in a dataset is increasing the distribution of different CubeSats models in the dataset\(^\text{[iii]}\). This leads the network to learn different CubeSats models accurately. One observation of this augmented dataset is even gray images helped to improve the detection accuracy of the CubeSats, simultaneously causing the false detection to increase when the test images are difficult gray images, as shown in Figure 3.10-(c). The cylinder shown in Figure 3.10-(c)\(^\text{[13]}\) is detected as a 3U CubeSat. To overcome this situation, the training is conducted by removing the gray augmented images from the training set. Results are shown in Figure 3.10-(d) without the gray images in the training dataset. Detection results varied when removing the gray images from the Web-

(a) Misdetection trained and tested on Web-searched augmented image datasets

(b) View point-false detection trained and tested on Web-searched augmented image datasets

(c) With gray images trained and tested on Web-searched augmented image datasets

(d) Without gray images: trained and tested on Web-searched augmented image datasets

Figure 3.10

Detection results
searched CubeSats dataset. This is because gray images make the training process harder by leading the Faster R-CNN to learn wrong information as the target output.

In this section, research is performed to improve the detection accuracy of the CubeSats detection model. The developed CubeSat model has a considerably higher accuracy with the ability to distinguish different CubeSats in its environment in near real-time. One observation of these detection results is that the Faster R-CNN suffers from identifying CubeSats based on the dimensions of the CubeSat. It also suffers from detecting small-scaled CubeSats as well. Figure 3.10(a) shows how the CubeSats detection model failed to detect a small-scale CubeSat[14] Figure 3.10(b) shows that the CubeSats model failed to detect the correct class with viewpoints of the CubeSat[15] One CubeSat in Figure 3.10(b) is a false detection. One reason for this is in object detection, a viewpoint of the object that plays a major role. In this research, solutions are proposed to solve these main problems and improved the detection and localization accuracy of the Faster R-CNN.

Table 3.7
Detection results after modifying the RPN layer (trained on Web-searched augmented image dataset)

<table>
<thead>
<tr>
<th>Tested data</th>
<th>No. images</th>
<th>Extractor</th>
<th>1U (%)</th>
<th>3U (%)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-searched (Augmented)</td>
<td>1,014</td>
<td>VGG-16</td>
<td>95.8</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>Web-searched (Original)</td>
<td>255</td>
<td>VGG-16</td>
<td>94.6</td>
<td>97.5</td>
<td>96.0</td>
</tr>
<tr>
<td>Web-searched-gray</td>
<td>255</td>
<td>VGG-16</td>
<td>85.1</td>
<td>90.3</td>
<td>87.7</td>
</tr>
<tr>
<td>CAD-with-texture</td>
<td>317</td>
<td>VGG-16</td>
<td>80.2</td>
<td>84.4</td>
<td>82.3</td>
</tr>
</tbody>
</table>

[14] See https://sst-soa.arc.nasa.gov/03-power
Figure 3.11

Modifications to RPN layer to incorporate wide range of scales and aspect ratios

(a) Misdetection: trained and tested on the Web-searched augmented image datasets
(b) Small scale: trained and tested on the Web-searched augmented image datasets

Figure 3.12

Improved detection results
3.6 Modifications to the RPN Layer

If the RPN layer is not able to propose the likelihood of containing a CubeSat in the feature map (in the object proposal generation stage), the trained CubeSats model will not be able to detect that CubeSat in the final classification [38]. Images with small-scaled CubeSats are facing these problems due to the inability to propose the presence of a CubeSat in the feature map by the RPN layer. To improve the CubeSats detection accuracy, it is necessary to make modifications to the object proposal generation stage. The modification made to the RPN layer is as follows[^16].

The Faster R-CNN uses a sliding window to run spatially on the feature maps to generate the object proposal[^38]. For each sliding window, the researcher increases the number of anchors generated (with a wide range of scales and aspect ratio) so it can detect from small-scaled CubeSats to a high-scaled CubeSats. The goal is to increase the range of scale of anchors (See Figure 3.11) so it can detect a wide range of scales of CubeSats[^16]. This leads to a reduction in the mis-detection of CubeSats when the RPN layer proposes the likelihood of containing a CubeSat to the final detection process[^17].

The detection results are shown in Figure 3.12 after modifying the RPN layer to incorporate a wide range of scales of anchors. When compared to the fault detection results presented in Figure 3.10-(a), (b) the network with the modified RPN layer is able to detect CubeSats with higher accuracy. Table 3.7 show the mAP for CubeSats detection with the modified RPN layer. Even though there are no much significance difference of mAP values

compared to the Table 3.6, the new model with the modified RPN layer shows a high localization accuracy on detecting small-scale CubeSats. Results are presented in Figure 3.13 for the modified RPN layer (See Appendix A.1).

3.7 CAD Models for CubeSats Detection

This section tends to connect topics of the Faster R-CNN, texture, and CAD models by trying to solve a vision task. Many vision projects, which are available in the computer vision community, used real images to train their detection models [29, 47, 56]. One objective of this research is to find the Faster R-CNN generalization capabilities on CAD image-based CubeSats detection. The architecture used in this research consisted of the same major components [38]:

- the RPN Layer and
the Fast R-CNN detector.

In the past many research work has been done to monitor the performance of CNNs. Specifically, how texture, shape, color, and other factors affect the detection process (See Figure 3.14) [34]. If the CNN is not invariant to the texture feature, the recognition capabilities will be different [34]. As a consequence, the detector trained on CAD images without the texture feature (using only shape cues) will perform worse on Web-searched images [34]. In the following subsection, discussions are provided to the experiments conducted to identify CubeSats using simple 3D-CAD models.

3.7.1 Analyzing Results of CAD-based CubeSats Models

In this series of research, other than the Web-searched CubeSats model, two types of CAD-based CubeSats models (without and with texture) are trained to detect the 1U and 3U CubeSats. The detection results are evaluated on how the training image type (CAD images and Web-searched images) will affect to develop accurate CubeSats detection models. CAD images with different scales, random angles have been used to prepared a reasonable synthetic 2D image dataset. When preparing texture-based CAD images the researcher used 3D-CAD models offered by Pumpkin Inc., to add realistic nature to the images [18]. After extracting features from the convolutional step, a CubeSat detector has been trained for each CubeSat category (1U and 3U CubeSats), and the CubeSat detector is tested for both Web-searched and CAD-based images. For a fair comparison with the Web-searched-based CubeSats detection models, the VGG-16 feature extractor has been

[18] See http://www.cubesatkit.com/content/design.html
used for the CAD-based CubeSat detection. To extract more accurate features the “cov5,3”
layer of VGG-16 architecture (same as the Web-searched CubeSats detection model) has
been used \[38\]. To find out which features (shape, texture, color, etc.) are affective in
developing an accurate CubeSats model, the researcher designed a series of experiments
by using CAD images without and with texture features (See Table 3.8).

Table 3.8

CubeSats detection results on CAD-with-texture, Web-searched, and CAD-no-texture test
datasets, Table shows different CubeSats models capabilities to detect CubeSats when
adding and removing features (Extractor:VGG-16)

<table>
<thead>
<tr>
<th>Trained data</th>
<th>Tested data</th>
<th>Images</th>
<th>1U (%)</th>
<th>3U (%)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAD-with-texture</td>
<td>Web-searched (Original)</td>
<td>255</td>
<td>61.6</td>
<td>20.6</td>
<td>41.1</td>
</tr>
<tr>
<td>CAD-with-texture</td>
<td>CAD-no-texture</td>
<td>313</td>
<td>82.9</td>
<td>65.5</td>
<td>74.2</td>
</tr>
<tr>
<td>CAD-with-texture</td>
<td>CAD-with-texture</td>
<td>317</td>
<td>96.4</td>
<td>90.9</td>
<td>93.6</td>
</tr>
<tr>
<td>Web-searched</td>
<td>Web-searched (Augmented)</td>
<td>1,014</td>
<td>95.8</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>Web-searched</td>
<td>Web-searched (Original)</td>
<td>255</td>
<td>94.6</td>
<td>97.5</td>
<td>96.0</td>
</tr>
<tr>
<td>Web-searched</td>
<td>Web-searched-gray</td>
<td>255</td>
<td>85.1</td>
<td>90.3</td>
<td>87.7</td>
</tr>
<tr>
<td>Web-searched</td>
<td>CAD-with-texture</td>
<td>317</td>
<td>80.2</td>
<td>84.4</td>
<td>82.3</td>
</tr>
<tr>
<td>CAD-no-texture</td>
<td>Web-searched (Original)</td>
<td>255</td>
<td>9.1</td>
<td>4.3</td>
<td>6.7</td>
</tr>
<tr>
<td>CAD-no-texture</td>
<td>CAD-no-texture</td>
<td>313</td>
<td>98.0</td>
<td>66.9</td>
<td>82.5</td>
</tr>
<tr>
<td>CAD-no-texture</td>
<td>CAD-with-texture</td>
<td>317</td>
<td>35.8</td>
<td>54.2</td>
<td>45.0</td>
</tr>
</tbody>
</table>

From certain viewpoints (front and back views are identical for 1U and 3U CubeSats),
CAD-based CubeSats models failed to distinguish the correct shape. The mAP drops dras-
tically on Web-searched test images when the trained CubeSats images are the CAD-no-
texture (See Table 3.8). In some cases (front view) the Faster R-CNN confused with 1U
and 3U CubeSats \[32\]. In addition, the CAD-with-texture-based CubeSats model detection
is lower when tested on Web-searched images. Lack of realistic nature of the trained CAD
images to tested Web-searched images is at the origin to drop the accuracy of CAD-based
CubeSats detection models. To improve the accuracy of CAD-based CubeSats models on Web-searched images, it is possible to add different CubeSat CAD models (intra-class variant) to the training dataset [34]. By adding similar textures as in real CubeSat images, it is possible to increase the accuracy when testing on Web-searched images. These techniques help CAD-based CubeSats detection models to learn different CubeSats models/texture, improving the accuracy on the learning process [34].

One conclusion from this work is by increasing the image data variations with different CubeSats models on the CAD-based dataset, the accuracy of the CAD-based CubeSats detection process increases. Chapter 4 describes the experiments carried using 3D-CAD models to detect CubeSats in close proximity by incorporating image data variations with different CubeSats models.

3.8 Detection Accuracy After Increasing the Number of CubeSat Classes: 1U, 2U, 3U, and 6U CubeSats

The most important use of the Faster R-CNN is detection of objects in an image. When it comes to CubeSats configuration, there are different CubeSats configurations such as “2U” and “6U” CubeSats. In this section, the researcher set a goal of developing a CubeSat detection model by increasing the number of the CubeSat classes: “1U, 2U, 3U, and 6U” CubeSats. The task is challenging due to the visual difference between the four types of CubeSats which is only the dimensions of the CubeSats. In order to start the process, the researcher collected a dataset for four different classes of CubeSats. The dataset contains images from the Web using the Google search engine. 29,210 images (with data augmentation) are used for the training process. Data augmentation techniques rotation, jitter, gray
Shape versus texture: If trained CubeSats models preliminary depends on shape information and other features (scale, model variation, and texture, etc.) will improve detection results.
and flip are used to increase the training dataset to 29,210 images with corresponding annotation files. 4201 images (with data augmentation) are used for the testing process. The CubeSat detection model is trained using the “approximate joint training” method. The IoU, threshold for NMS, set at 0.7 for this experiment to get around 2000 proposal regions per image [38]. For the approximate joint training, the learning rate is fixed at 0.0001, momentum to 0.9, and has trained for 100K iterations. The size of the sliding window used in this experiment is $3 \times 3$, as it is a good scale to detect the likelihood of the presence of a CubeSat in the proposal generation stage [38]. To extract more accurate features the “cov5_3” layer of VGG-16 architecture has been used. It took below 0.2 seconds to process a test image on the GTX-1080 GPU.

**Table 3.9**

<table>
<thead>
<tr>
<th>Tested data</th>
<th>No. images</th>
<th>1U (%)</th>
<th>2U (%)</th>
<th>3U (%)</th>
<th>6U (%)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-searched</td>
<td>4201</td>
<td>96.3</td>
<td>73.4</td>
<td>82.5</td>
<td>78.6</td>
<td>82.7</td>
</tr>
</tbody>
</table>

The following experiments are conducted to evaluate the performance of the trained CubeSat model: when the test image is that of a “1U, 2U, 3U, and/or, 6U” CubeSats, when the test image is an asteroid or a planet, and if there are no CubeSats at all to find out when the trained CubeSat detection model fails. The detection results are shown in Figure 3.16 after modifying the RPN layer to incorporate a wide range of scales and aspect ratios of anchors in the training process. Compared to the detection results presented in Figure 3.13, the modified Faster R-CNN is able to detect four different classes of CubeSats.
with a considerable accuracy. mAP for detecting four CubeSats classes are shown on Table 3.9. Figure 3.15 shows the precision and recall curve after increasing the number of CubeSat classes: 1U, 2U, 3U, and 6U CubeSats.

Figure 3.15

Precision and recall curve after increasing the number of CubeSat classes: 1U, 2U, 3U, and 6U

In this section, the researcher investigated a wide range of experiments to develop an accurate CubeSats detection system. Experiments ran till developing a CubeSat detection model with four different classes of CubeSats, utilizing Web-searched CubeSat images. One of the biggest challenges with these experiments is the detection of the correct shapes of CubeSats. Sometimes the CubeSats detection process fails due to the difficulty of detecting the correct shape of CubeSats.
Figure 3.16
Detection results after increasing the number of CubeSat classes: 1U, 2U, 3U, and 6U (See Appendix A.1)

Figure 3.17
Trained for 1U, 2U, 3U, and 6U CubeSats detection, no red-box means no detection (See Appendix A.1)
3.9 Discussion

This chapter proposes and implements two frameworks that decomposed the images based on their texture and shape cues using Web-searched and CAD-based CubeSats images. For Web-searched images the researcher trained and evaluated a few CubeSats detection models to demonstrate the qualitative and quantitative results of the CubeSats detection process. A set of typical detection results are shown in Figures 3.13 and 3.16. The Web-searched CubeSats detection model can detect CubeSats that appear in similar or complex backgrounds to those in the training images with higher accuracy. Table 3.8 shows the mAP for different types of test datasets evaluated to check trained CubeSats detection models accuracy.

To understand why CubeSats detection models failed on some of the images, the researcher analyzed the low probability and false positive images. There are a few error patterns. For example, the front view of CubeSats (1U and 3U CubeSats are identical in shape with zero altitude angle) are hard to distinguish\[19\]. The second reason is that images of unseen models (CAD-based CubeSats models tested on Web-searched images) are hard to classify correctly \[42, 17\]. When it comes to CAD-based CubeSats models, Web-searched images also used in the testing process to check the CubeSat detection model ability to detect near-real CubeSats. It can be clearly seen that the lack of a realistic nature of the CAD images is at the origin of higher false recognition.

In order to evaluate the performance of the CubeSats detection process of Web-searched CubeSats models, a few experiments were conducted. CubeSats detection results are evalu-
ated on how illumination affects the detection process and the prediction probability when there are no CubeSats at all. If the Web-searched CubeSats detection model has to detect low-resolution images (See Table 3.1) there is a high chance for a false detection. Figure 3.17 shows that the Web-searched CubeSats detection models assigned lower probabilities to non-CubeSat objects\textsuperscript{20,21,22}. Due to the intra-class variation of CubeSats models available in the Web-searched image dataset, Web-searched-based CubeSats detection model learned the 1U, 2U, 3U, and 6U CubeSats shapes with higher accuracy than that of CAD-based CubeSats model. There are false positives for Web-Searched-based CubeSats detection system. The Web-searched CubeSats detection model detected cylinders (e.g., Hubble space telescope) as 3U CubeSats (See Figure 3.17, Appendix A.1). Sometimes, the Web-searched-based CubeSats detection models detect wrong shape (e.g., a 2U CubeSat as a 3U CubeSat) as shown in Figure 3.17. These are some major challenges with the developed CubeSat models which focus to solve in the future works. However, the developed Web-searched-based CubeSats detection models have a high capability on rejecting irregular shapes and circles (asteroids and planets) as shown in Figure 3.17. When it comes to CubeSats configuration, there are other configurations such as the 12U and 27U CubeSats. To expand the range of CubeSats detection, future works will include the 12U and 27U CubeSats as well in the training process.

\textsuperscript{20}See https://www.ibtimes.com/nasa-asteroid-tracker-2-massive-asteroids-zip-incredibly-close-earth-today-2798552

\textsuperscript{21}http://spaceref.com/onorbit/nasa-awards-$350000-to-winning-astronaut-glove-designers.html

\textsuperscript{22}https://www.forbes.com/sites/startswithabang/2019/05/09/we-have-now-reached-the-limits-of-the-hubble-space-telescope/1f4331863208
CHAPTER 4
DETECTING A CUBESAT IN CLOSE PROXIMITY

4.1 Background

Detecting space objects in close proximity is a problem arising from capturing, servicing, and other proximity operations. Vision-based object detection refers to the process of localizing all objects of interest in images or videos (localization in the camera space) and determining the categories to which the objects belong (classification). The localization result can aid relative navigation. The classification result is crucial for situation assessment and high-level control, planning, and decision making.

Faster R-CNN is a state-of-the-art single-view object detection method [38]. It automatically extracts low- and high-level features and is capable of near real-time object detection in real-world environments. It uses a RPN for generating region proposals and uses Fast R-CNN [8] for classifying the proposed regions into object classes and background [38]. Compared with the closely related R-CNN [9] and Fast R-CNN [8], Faster R-CNN significantly reduces the running time by generating proposals using a RPN.

All detection systems yield false results. Faster R-CNN is no exception. The basic idea of this work is to reduce false results of Faster R-CNN by combining the classification results from multiple images using the Dempster-Shafer theory of evidence, [4, 41, 55] a
general framework for fusing information from multiple sources under certainty. Neither Faster R-CNN nor the evidence theory requires knowledge of the camera parameters.

The main contribution of this work is the two-step classification method and its application to rotating CubeSat detection. The CubeSat classes are limited to 1U, 2U, 3U, and 6U CubeSats. They differ mainly in dimension ratios and may be indistinguishable from unfavorable viewpoints, rendering single-view classification difficult or impossible. An obvious limitation of the method should be pointed out: Without scale information, it cannot tell a CubeSat from a larger or smaller satellite with the same appearance.

4.2 Two-Step Classification

The objective of the CubeSat detection problem is to detect a rotating or tumbling CubeSat from multiple images taken by a camera in close proximity. The intrinsic or external parameters of the camera are unknown or not used. The images are denoted by $I_k$, $k = 1, ..., n$, and the image set by $I = \{I_1, I_2, \cdots, I_n\}$. The order of the images does not change the classification result. It is assumed that four CubeSat classes exist: 1U, 2U, 3U, and 6U, whose dimension ratios are approximately 1:1:1, 1:1:2, 1:1:3, and 1:2:3, respectively. Note that the alternative 6U CubeSat with dimension ratio 1:1:6 is not included.

The space $\Omega$ of the classification problem is therefore:
\[ \Omega \equiv \{ \text{"1U CubeSat"}, \text{"2U CubeSat"}, \text{"3U CubeSat"}, \text{"6U CubeSat"}, \text{"Background"} \} \]

where the “Background” class accounts for non-CubeSat objects or background in the image. A class label \( c \) is an element of the set: \( c \in \Omega \). A partition of the set is:

\[
\{ \text{"1U CubeSat"} \} \cup \{ \text{"2U CubeSat"} \} \cup \{ \text{"3U CubeSat"} \} \cup \{ \text{"6U CubeSat"} \} \cup \{ \text{"Background"} \} = \Omega.
\]

In the first step, the Faster R-CNN for CubeSat detection processes all the images one by one. Given an image \( I_k \), the Faster R-CNN adds bounding boxes around all possible CubeSat objects in the image. In the ideal case, the number of bounding boxes equals the number of real CubeSats and each bounding box contains a real CubeSat. If the Faster R-CNN believes that no CubeSat is present in the image, no bounding boxes are drawn. The locations of the bounding boxes as well as the features contain important relative navigation information but does not contribute to classification. For each bounding box, the weights or scores of the five possible labels are calculated and the label with the largest weight is assigned to the bounding box. The weights are normalized: \( \sum_c w_k(c) = 1 \). For sake of simplicity, the researcher assumes that there is one and only one positive detection (one bounding box) in each and every image.

In the second step, the classification results (labels and weights) of the individual images are combined using the Dempster-Shafer theory of evidence. The weights are in-
terpreted as either standard probabilities in Bayesian inference or mass functions of the
theory of evidence. The former requires that weights on all classes are available, not just
the largest weight for each image. The latter does not need this requirement. The out-
put of the second step is the aggregated probabilities of or beliefs in the five classes. The
class with the highest probability or belief will be selected. The rules of combination are
reviewed in the next section.

4.3 Combination of Evidence

4.3.1 Bayes’ Rule

When the weights on all five classes are available from the first-step classification,
Bayes’ rule can be used. The weights are viewed as single-event probabilities over Ω:

\[ P_k(c) \equiv P_k(c|I_k) = w_k(c) \]  

(4.3)

where \( P_k(c) \) is the probability that the object belongs to the class designated by the
label \( c \). That is,

\[ P_k(c = “1U CubeSat”) = w_k(“1U CubeSat”) \]  

(4.4a)

\[ P_k(c = “2U CubeSat”) = w_k(“2U CubeSat”) \]  

(4.4b)

\[ P_k(c = “3U CubeSat”) = w_k(“3U CubeSat”) \]  

(4.4c)

\[ P_k(c = “6U CubeSat”) = w_k(“6U CubeSat”) \]  

(4.4d)

\[ P_k(c = “Background”) = w_k(“Background”). \]  

(4.4e)
Let the probability conditioned on the image set $I$ be $P(c) \equiv P(c|I)$. Under the independence assumption:

$$P(c) \propto \prod_{k=1}^{n} P_k(c).$$ (4.5)

### 4.4 Mass, Belief, and Plausibility

Three interdependent basic functions of the Dempster-Shafer theory are the mass function, the belief function, and the plausibility function. The mass function is defined for all subsets of $\Omega$ and satisfy:

$$m(\emptyset) = 0$$ (4.6)

$$\sum_{A \subseteq \Omega} m(A) = 1$$ (4.7)

where $\emptyset$ denotes the empty set and $A$ every subset of $\Omega$. The belief function $\text{Bel}(\cdot)$ and the mass function $m(\cdot)$ are related by:

$$\text{Bel}(A) = \sum_{B \subseteq A} m(B)$$ (4.8)

$$m(A) = \sum_{B \subseteq A} (-1)^{|A-B|} \text{Bel}(B)$$ (4.9)

where $|A - B|$ is the cardinality difference between subsets $A$ and $B$. The plausibility function $\text{Pl}(\cdot)$ is related to the mass function $m(\cdot)$ and $\text{Pl}(\cdot)$ by:

$$\text{Pl}(A) = \sum_{A \cap B \neq \emptyset} m(B)$$ (4.10)
\[ Pl(A) = 1 - Bel(A^C) \] (4.11)

with \( A^C \) the complementary set of \( A \). Clearly, \( Bel(A) \leq Pl(A) \). The interval \([Bel(A), Pl(A)]\) contains the unknown true probability of \( A \). The difference \( Pl(A) - Bel(A) \) represents ignorance (resulting from incomplete information). Total ignorance corresponds to \([0, 1]\).

If \([Bel(A), Pl(A)] = [0, 0]\), then \( A \) is impossible. If \([Bel(A), Pl(A)] = [1, 1]\), then \( A \) is certain.

Note that if the subsets \( A \) of \( \Omega \) with \( m(A) > 0 \) form a partition of \( \Omega \), then \( Bel(A) = Pl(A) = m(A) \) and the belief function in the Dempster-Shafer theory can be treated as the probability in Bayesian inference.

### 4.5 Mass Assignments for CubeSat Classification

Given the weights \( w_k(c) \) from the image \( I_k \), the mass function is formed:

\[ m_k(A(c)) = w_k(c) \] (4.12)

with \( A(c) \subset \Omega \).
The masses on all other subsets of $\Omega$ are zero. When the weights on all classes are available from the first-step classification, the natural choice of $A(c)$ is:

$$A(\text{“1U CubeSat”}) = \{\text{“1U CubeSat”}\} \quad (4.13a)$$
$$A(\text{“2U CubeSat”}) = \{\text{“2U CubeSat”}\} \quad (4.13b)$$
$$A(\text{“3U CubeSat”}) = \{\text{“3U CubeSat”}\} \quad (4.13c)$$
$$A(\text{“6U CubeSat”}) = \{\text{“6U CubeSat”}\} \quad (4.13d)$$
$$A(\text{“Background”}) = \{\text{“Background”}\}. \quad (4.13e)$$

The corresponding mass function is:

$$m_k(\{\text{“1U CubeSat”}\}) = w_k(\text{“1U CubeSat”}) \quad (4.14a)$$
$$m_k(\{\text{“2U CubeSat”}\}) = w_k(\text{“2U CubeSat”}) \quad (4.14b)$$
$$m_k(\{\text{“3U CubeSat”}\}) = w_k(\text{“3U CubeSat”}) \quad (4.14c)$$
$$m_k(\{\text{“6U CubeSat”}\}) = w_k(\text{“6U CubeSat”}) \quad (4.14d)$$
$$m_k(\{\text{“Background”}\}) = w_k(\text{“Background”}). \quad (4.14e)$$
An alternative choice of $A(c)$ is:

\[
A(\text{"1U CubeSat"}) = \{\text{"1U CubeSat"}\} \cup \{\text{"2U CubeSat"}\} \cup \{\text{"3U CubeSat"}\} \quad (4.15a)
\]

\[
A(\text{"2U CubeSat"}) = \{\text{"2U CubeSat"}\} \cup \{\text{"3U CubeSat"}\} \cup \{\text{"6U CubeSat"}\} \quad (4.15b)
\]

\[
A(\text{"3U CubeSat"}) = \{\text{"3U CubeSat"}\} \cup \{\text{"6U CubeSat"}\} \quad (4.15c)
\]

\[
A(\text{"6U CubeSat"}) = \{\text{"6U CubeSat"}\} \quad (4.15d)
\]

\[
A(\text{"Background"}) = \{\text{"Background"}\}. \quad (4.15e)
\]

The motive is to take label ambiguity into consideration. That is to say, a CubeSat labeled “1U CubeSat” may be a 1U, 2U, or 3U CubeSat, a CubeSat labeled “2U CubeSat” may be a 2U, 3U, or 6U CubeSat, and a CubeSat labeled “3U CubeSat” may be a 3U or 6U CubeSat. The mass function is chosen as:

\[
m_k(\{\text{"1U CubeSat"}\} \cup \{\text{"2U CubeSat"}\} \cup \{\text{"3U CubeSat"}\}) = w_k(\text{"1U CubeSat"})
\quad (4.16a)
\]

\[
m_k(\{\text{"2U CubeSat"}\} \cup \{\text{"3U CubeSat"}\} \cup \{\text{"6U CubeSat"}\}) = w_k(\text{"2U CubeSat"})
\quad (4.16b)
\]

\[
m_k(\{\text{"3U CubeSat"}\} \cup \{\text{"6U CubeSat"}\}) = w_k(\text{"3U CubeSat"})
\quad (4.16c)
\]

87
Both choices of $A(c)$ are appropriate for certain camera viewpoints, but not for all viewpoints. Other alternatives are possible but not tested in this dissertation. For example, $w_k$ (“3U CubeSat”) may be assigned to two masses:

$$m_k\left(\{\text{“3U CubeSat”}\} \cup \{\text{“6U CubeSat”}\}\right) = pw_k(\text{“3U CubeSat”}) \quad (4.18a)$$
$$m_k(\{\text{“3U CubeSat”}\}) = qw_k(\text{“3U CubeSat”}) \quad (4.18b)$$

with $p, q > 0$ and $p + q = 1$.

When only one label and one weight (the largest weight) are available from the first-step classification, the mass function is defined as:

$$m_k(A(c_{\text{max}})) = w_k(c_{\text{max}}) \quad (4.19a)$$
$$m_k(\Omega) = 1 - w_k(c_{\text{max}}) \quad (4.19b)$$

where $c_{\text{max}}$ denotes the class corresponding to the largest weight and $w_k(c_{\text{max}})$ the associated weight. The second equation ensures that the masses add up to one.
4.6 Dempster’s and Yager’s Rules

Many rules exist for combining evidence from multiple sources \cite{39}. Rules of combining two mass functions are presented. Rules of combining more mass functions can be obtained by repeated application of the rules for two mass functions. Dempster’s rule is defined by:

\[
m_1 \oplus m_2(\emptyset) = 0 \tag{4.20a}
\]

\[
m_1 \oplus m_2(A) = \sum_{A_i \cap B_j = A} \frac{m_1(A_i)m_2(B_j)}{\sum_{A_i \cap B_j \neq \emptyset} m_1(A_i)m_2(B_j)}, A = \emptyset \tag{4.20b}
\]

\[
m_1 \oplus m_2(A) = \frac{\sum_{A_i \cap B_j = A} m_1(A_i)m_2(B_j)}{1 - \sum_{A_i \cap B_j = \emptyset} m_1(A_i)m_2(B_j)}. \tag{4.21}
\]

Note that when \( m_1(\cdot) \) and \( m_2(\cdot) \) are defined over a partition of \( \Omega \), Dempster’s rule of combination yield the same result as Bayes’ rule and the interval \([Bel(A), Pl(A)]\) collapses to a point. An alternative to Dempster’s rule is Yager’s rule, which handles conflicting information differently. Yager’s rule is defined through the ground probability assignment function \( q(A) \), given by:

\[
q(A) = \sum_{A_i \cap B_j = A} m_1(A_i)m_2(B_j). \tag{4.22}
\]
Note that \( q(\emptyset) = \sum_{A_i \cap B_j = \emptyset} m_1(A_i) m_2(B_j) \) is positive in general and increases as more evidence is aggregated. Yager’s rule is given by:

\[
m^Y(\emptyset) = 0 \tag{4.23a}
\]

\[
m^Y(\Omega) = q(\emptyset) + q(\emptyset) \tag{4.23b}
\]

\[
m^Y(A) = q(A), A \neq \emptyset, A \neq \Omega. \tag{4.23c}
\]

Dempster’s rule can be calculated from \( q(A) \) and \( q(\emptyset) \):

\[
m(A) = \frac{q(A)}{1 - q(\emptyset)}. \tag{4.24}
\]

4.7 Illustrative Examples

Four examples are presented to compare Dempster’s and Yager’s rules. The results are given in Appendix, Tables B.1-4. Two mass functions are defined over the subsets given by Equation 4.14. Since the subsets form a partition of \( \Omega \), Dempster’s rule yields the same result as Bayes’ rule. The results of Dempster’s rule are points instead of intervals.

The first example mimics a classical example by Zadeh [4]. The results of this extreme example are shown in Appendix, Table B.1. Dempster’s rule yields the undesired result \( m(“3U\text{ CubeSat”}) = 1 \). The result of Yager’s rule assigns a low belief (0.0001) to the “3U CubeSat” class but zero belief to the other classes. If a hard decision must be made based on the highest belief, both rules are in favor of the “3U CubeSat” class. In this sense, they are consistent. The second example represents the most extreme case, where all the
masses are 0 and 1 and the two pieces of evidence completely conflict with each other. Dempster’s rule (as well as Bayes’ rule) is not defined in this case. Yager’s rule yields total ignorance for all classes, as shown in Appendix, Table B.2. The rules do not support or oppose any decision. The third example is likely to occur in real systems. As shown in Appendix, Table B.3, the mass functions are still conflicting, but all masses are nonzero except for the “Background” class. The two rules are again consistent: They no longer support the “3U CubeSat” class and equally support the “1U CubeSat” and “3U CubeSat” classes. The last example represents the case in which evidence from the two sources is in perfect agreement. As expected, Dempster’s rule and Yager’s rule are in good agreement if the highest belief is the parameter for decision making. Appendix, Table B.4 also shows that Yager’s rule tends to yield low beliefs even in seeming benign cases.

4.8 CubeSat Detection Results

A Faster R-CNN with four CubeSat classes is built on Caffe [18]. A total of 90,956 synthetic training images in 1,000 random camera viewpoints are generated using Autodesk Maya. In Maya, the default method of rotation is Euler \(^1\). Euler rotations are calculated using three Euler angles, which represent rotations about the X, Y, and Z axis, with an order of rotation \(^1\). The user can specify the order of rotation for an object (CubeSat) by setting its rotation order (e.g., XYZ). For example, if the user sets a CubeSat’s rotation order to XZY, the CubeSat will first rotate on the X axis, then the Z axis, and finally the Y axis. The synthetic test images of rotating CubeSats are simulated based on rigid-body kinematics.

and dynamics under zero external torque. The camera is assumed to be sufficiently close to the CubeSat, but the relative translational motion is not simulated. Bayes’, Dempster’s, and Yager’s rules as well as the Faster R-CNN are tested in three cases. The weights in the three cases are shown in Appendix, Tables B.5-7. The “Background” weights are not given but can be easily calculated from the row sums. The image sequences in the three cases are shown in Appendix, Figures B.1-3. The Faster R-CNN may yield false results when the CubeSat is in unfavorable views (Appendix, Figures B.1 and B.3) and relatively small weights when the CubeSat is in dark environments (Appendix, Figure B.2). In Case 1, the true CubeSat is a 3U CubeSat but is mistaken as a 2U CubeSat in the third image.

Case 2 has a 3U CubeSat in dark environments. The largest weights are under 0.65. Although the “3U CubeSat” class receives the largest weights, there exist competing hypotheses that the CubeSat is 6U or 2U. The CubeSat in Case 3 is a 6U CubeSat, but in the first four images, it appearance is closer to a 3U CubeSat than to a 6U CubeSat, which results in four false classifications. There are four Dempster’s rules with four different mass functions, depending on whether all weights are available and how the labels are interpreted:

1. Dempster’s rule (all weights, simple sets),
2. Dempster’s rule (one weight, simple sets),
3. Dempster’s rule (all weights, complex sets),
4. Dempster’s rule (one weight, complex sets).
When all weights are available, the mass function is defined by Equation 4.14 or 4.16 and 4.17. When only one weight (the largest weight) is available, the mass function is given by Equation 4.19. The designations “simple sets” and “complex sets” mean that the subsets of $\Omega$ with nonzero mass are defined by Equation 4.13 and 4.15 respectively. The first Dempster’s rule always yields the same result as Bayes’ rule. Therefore, the results of Bayes’ rule are not listed separately.

The four Yager’s rules are defined in the same way:

1. Yager’s rule (all weights, simple sets),
2. Yager’s rule (one weight, simple sets),
3. Yager’s rule (all weights, complex sets),
4. Yager’s rule (one weight, complex sets).

### 4.8.1 Results of Dempster’s Rules

The results of the first Dempster’s rule in the three test cases are given by Appendix, Tables B.8-10. The results of the second Dempster’s rule in the three test cases are given by Appendix, Tables B.11-13. The results of the third Dempster’s rule in the three test cases are given by Appendix, Tables B.14-16. The results of the fourth Dempster’s rule in the three test cases are given by Appendix, Tables B.17-19. The results can be summarized as follows:

1. When all the weights are available, the Dempster’s rule with the labels interpreted as simple sets, which yields the identical results as Bayes’ rule, provides strong evidence for the correct classes in all three cases.
2. When only the largest weight is available, the Dempster’s rule with the labels interpreted as simple sets works equally well. The lack of complete information does not affect the classification results.

3. When the labels are interpreted as complex sets, Dempster’s rule is much less assertive and provides weaker evidence for all classes. Despite that, it firmly and correctly rejects the “1U CubeSat”, “2U CubeSat”, and “Background” classes. It is biased toward the “6U CubeSat” class, however, and this bias leads to incorrect classification in the first two cases. That is not surprising because the “6U CubeSat” label provides evidence against the “3U CubeSat” class, but the “3U CubeSat” label equally supports the “3U CubeSat” and “6U CubeSat” classes. The complex set interpretation of the labels is not suited for the rotating or tumbling CubeSats.

4.8.2 Results of Yager’s Rules

The results of the first Yager’s rule in the three test cases are given by Appendix, Tables B.20-22. The results of the second Yager’s rule in the three test cases are given by Appendix, Tables B.23-25. The results of the third Yager’s rule in the three test cases are given by Appendix, Tables B.26-28. The results of the fourth Yager’s rule in the three test cases are given by Appendix, Tables B.29-31. Overall, these Yager’s rules yield low beliefs in all classes. The classification results are similar to those of the Dempster’s rules, if classification is based on the highest belief. The summary follows:

1. When all the weights are used, the Yager’s rule with the labels interpreted as simple sets yields low beliefs and large ignorance for all three cases. That is due to the presence of conflicting information. Compared with Dempster’s rules, it provides weak evidence in support of the correct classification in the first two cases. In the third
case, all five classes receive the same result of total ignorance. Closer examination of the beliefs or the lower bounds of the intervals shows that the belief in the correct class, “6U CubeSat”, is orders of magnitude stronger than in the other classes. The beliefs in the five classes are $1 \times 10^{-38}$, $1 \times 10^{-33}$, $6 \times 10^{-16}$, $5 \times 10^{-8}$, $2 \times 10^{-40}$, respectively. If the relative magnitude of the belief is used, this rule is correct in all three cases.

2. When only the largest weight is used, the Yager’s rule with the labels interpreted as simple sets yield stronger evidence in the first two cases than when all the weights are used. In the third case, the beliefs in the five classes are also $1 \times 10^{-38}$, $1 \times 10^{-33}$, $6 \times 10^{-16}$, $5 \times 10^{-8}$, $2 \times 10^{-40}$, respectively. When the relative magnitude of the belief is used, this rule is correct in all three cases.

3. When the labels are interpreted as complex sets, like Dempster’s rule, Yager’s rule rejects the “1U CubeSat”, “2U CubeSat”, and “Background” classes but is biased toward the “6U CubeSat” class.

4.9 Discussion

Integrating the Dempster-Shafer theory of evidence with Faster R-CNN provides a simple but effective way of detecting a rotating CubeSat in close proximity from multiple images. The two-step classification method yields correct and reliable classification even when the CubeSat is in unfavorable views or in dark environments. Dempster’s rule of combination is well suited for this classification problem, which is unlikely to have completely conflicting evidence from consecutive images. Since the rule can handle incomplete information with ease, it only needs the label and the associated weight from a first-step
classification result. The future work includes a model with more space object classes and extensive more realistic tests.
CHAPTER 5
ATTITUDE ESTIMATION BASED ON CUBESAT DETECTION

5.1 Background

This chapter presents a single-point coarse attitude estimation method based on the CubeSat detection results by a spacecraft in the close proximity of two or more CubeSats. The method can be used as a contingent attitude estimation solution for the spacecraft.

The attitude is defined not as the orientation of the spacecraft relative to one of the CubeSats in close proximity, but the orientation of the spacecraft with respect to a global reference frame such as the Earth-Centered Inertial frame or the Earth-Centered Earth-Fixed frame. Determining the former would require the use of CubeSat surface feature points in the image, which are unavailable from the output of the CubeSat detection system. Instead, the attitude determination system leverages the coordinates of the bounding boxes surrounding the CubeSats. The centroid of a bounding box can be readily calculated from the coordinates of the four corners of the bounding box. These bounding box centroids approximate the centroids of the CubeSats in the image, which in turn approximates the center of mass of the CubeSats in the image.

5.2 Algorithm Development

The attitude estimation algorithm is developed under the following assumptions:
• The spacecraft body frame is the same as the camera frame.

• The CubeSat image is provided by a pin-hole camera with known focal length on-board the spacecraft.

• Two or more CubeSats are detected in the image.

• The position vectors of the detected CubeSats and the spacecraft are provided by GPS.

• The position information of the detected CubeSats is shared with the spacecraft.

The bounding box coordinates and the camera focal length determine the Line-of-Sight (LOS) vectors from the spacecraft to the CubeSats in the body frame. The GPS data are used to determine the LOS vectors in the reference frame. Then, the three-axis attitude is obtained by solving Wahba’s problem [50].

Suppose there are \( n \geq 2 \) bounding boxes with centroids \((\tilde{x}_i, \tilde{y}_i), i = 1, \ldots, n\). The \( n \) LOS vectors in the body frame are given by:

\[
\hat{b}_i = \frac{1}{\sqrt{\tilde{x}_i^2 + \tilde{y}_i^2 + f^2}} \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ f \end{bmatrix}
\]  

(5.1)

where \( f \) denotes the focal length of the camera. The LOS vectors in the reference frame are given by:

\[
\tilde{r}_i = \frac{\tilde{R}_i - \tilde{R}_c}{\|\tilde{R}_i - \tilde{R}_c\|}
\]  

(5.2)

where \( \tilde{R}_i \) are the CubeSat positions in the reference frame and \( \tilde{R}_c \) is the spacecraft position in the reference frame.
The attitude matrix estimate \( \hat{A} \) is the solution to Wahba’s problem, which minimizes the following cost function:

\[
L(A) = \frac{1}{2} \sum_{i=1}^{2} \frac{1}{\sigma_i^2} \| \tilde{b}_i - A \tilde{r}_i \|^2
\]

subject to the constraint

\[
AA^T = A^T A = I_{3 \times 3}, \quad \text{det}(A) = 1
\]

where the superscript \( T \) denotes matrix transpose, \( \text{det} \) denotes matrix determinant, \( I_{3 \times 3} \) is the three-dimensional identity matrix, and \( \sigma_i^2 \) is the effective noise level. Suppose the noise variances of \( \tilde{b}_i \) and \( \tilde{r}_i \) are \( \sigma_{ri}^2 \) and \( \sigma_{bi}^2 \), respectively. Therefore, \( \sigma_i^2 \approx \sigma_{ri}^2 + \sigma_{bi}^2 \).

Many solutions to Wahba’s problem exist. In this research Markley’s singular value decomposition method [30] has been used to solve the attitude estimation problem. First, an attitude profile matrix \( B \) is constructed:

\[
B = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \tilde{b}_i \tilde{r}_i^T.
\]

Then, the singular value decomposition of \( B \) gives:

\[
B = USV^T
\]

where \( U \) and \( V \) are orthogonal matrices and \( S \) is a diagonal matrix

\[
S = \begin{bmatrix}
    s_1 & 0 & 0 \\
    0 & s_2 & 0 \\
    0 & 0 & s_3
\end{bmatrix}
\]

The attitude estimate is given by:
\[ \hat{A} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} V^T. \] (5.8)

The last diagonal element \( d = \det(U) \det(V) \) can take on two possible values: \( \pm 1 \). The loss function of \( \hat{A} \) is:

\[ L(\hat{A}) = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} - s_1 - s_2 - ds_3. \] (5.9)

The 3 × 3 attitude error covariance matrix is given by:

\[ P = \left[ \sum_{i=1}^{n} \frac{1}{\sigma_i^2} (I_{3 \times 3} - b_i b_i^T) \right]^{-1} \] (5.10)

where \( b_i \) are the noise-free LOS vectors in the body frame.

### 5.3 Illustrative Examples

Two examples are used to show the attitude determination process. Simulated images are generated using Autodesk Maya, a 3D software application developed by Autodesk.

#### 5.3.1 Object Space Versus World Space

Maya has two coordinate systems: the local coordinate system and global coordinate system. The local coordinate system is called object space and the global coordinate system is called world space. In Maya, the world coordinate system is always fixed. Figure 5.1 shows three CubeSats representation in the world frame and XYZ coordinates of a one CubeSat in the world frame. It is necessary for each CubeSat to have its own axis inde-
pendent of the world-axis. This is called the object space/local-axis. When a CubeSat rotates or moves, its object space/local-axis rotates or moves with it.

5.3.2 Camera Frame

When a user creates a camera in Maya, its view is perspective. To render a scene the user needs to create a rendering camera as shown in Figure 5.2.

5.3.2.1 Focal Length

In Maya, the focal length is represented in millimeters (mm). The object’s (CubeSat’s) size in the rendering frame is proportional to the focal length of the camera. Therefore, the user needs to be careful when selecting a focal length as it cause the CubeSat to appears larger or smaller in the rendering frame.

5.3.2.2 Angle of View

When trying to render a CubeSat image using the Maya, the user can decide the size of a CubeSat by adjusting the lens of the camera to a longer or shorter focal length. This is what causes CubeSats to get larger or smaller in the rendering frame.

5.4 Simulated Scenario 1: 1U CubeSats Detection

In the simulated scenario 1, three 1U CubeSats are present in the close proximity of a camera onboard a spacecraft. The detection result is shown in Figure 5.3.

The centroids of the bounding boxes are [units: centimeters (cm)]:

\[
\begin{align*}
\tilde{x}_1 &= 12.7604, & \tilde{y}_1 &= 4.6158 \\
\tilde{x}_2 &= 18.2434, & \tilde{y}_2 &= 8.3937 \\
\tilde{x}_3 &= 24.1238, & \tilde{y}_3 &= 3.9172.
\end{align*}
\]

See [https://knowledge.autodesk.com/support/maya](https://knowledge.autodesk.com/support/maya)
The focal length of the camera is $f = 3.5$ cm. Thus, the three LOS vectors in the body frame are:
\[
\tilde{b}_1 = \begin{bmatrix} 0.9106 \\ 0.3294 \\ 0.2498 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} 0.8950 \\ 0.4118 \\ 0.1717 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} 0.9771 \\ 0.1587 \end{bmatrix}.
\] (5.14)

The position vectors of the CubeSats and the spacecraft are (cm):

\[
\tilde{R}_1 = \begin{bmatrix} -6.78 \\ 4.24 \\ 6.77 \end{bmatrix}, \quad \tilde{R}_2 = \begin{bmatrix} 1.79 \\ -1.18 \\ 2.16 \end{bmatrix}, \quad \tilde{R}_3 = \begin{bmatrix} 9.26 \\ 3.85 \\ 7.23 \end{bmatrix}, \quad \tilde{R}_c = \begin{bmatrix} 6.46 \\ -53.69 \\ 10.32 \end{bmatrix}.
\] (5.15)

From the position vectors, the LOS vectors in the reference frame are:

\[
\tilde{r}_1 = \begin{bmatrix} -0.2224 \\ 0.9731 \\ 0.0596 \end{bmatrix}, \quad \tilde{r}_2 = \begin{bmatrix} -0.0875 \\ 0.9843 \\ -0.1530 \end{bmatrix}, \quad \tilde{r}_3 = \begin{bmatrix} 0.0485 \\ 0.9974 \\ -0.0536 \end{bmatrix}.
\] (5.16)

The attitude estimate is given by:

\[
\hat{A} = \begin{bmatrix} 0.2601 & 0.9656 & 0.0081 \\ -0.6471 & 0.1806 & -0.7407 \\ -0.7167 & 0.1874 & 0.6718 \end{bmatrix}.
\] (5.17)

### 5.5 Simulated Scenario 2: 3U CubeSats Detection

In the simulated scenario 2, three 3U CubeSats are present in the close proximity of a camera onboard a spacecraft. The detection result is shown in Figure 5.4.
The centroids of the bounding boxes are (cm):

\[
\begin{align*}
\tilde{x}_1 &= 16.4729, \quad \tilde{y}_1 = 9.8152 \tag{5.18} \\
\tilde{x}_2 &= 13.2392, \quad \tilde{y}_2 = 4.6372 \tag{5.19} \\
\tilde{x}_3 &= 23.8533, \quad \tilde{y}_3 = 5.0917. \tag{5.20}
\end{align*}
\]

The focal length of the camera is \( f = 3.5 \text{ cm} \). Thus, the three LOS vectors in the body frame are:

\[
\tilde{b}_1 = \begin{bmatrix} 0.8451 \\ 0.5035 \\ 0.1796 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} 0.9157 \\ 0.3207 \\ 0.2421 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} 0.9681 \\ 0.2066 \\ 0.1420 \end{bmatrix}. \tag{5.21}
\]

The position vectors of the CubeSats and the spacecraft are (cm):

\[
\begin{align*}
\tilde{R}_1 &= \begin{bmatrix} 6.88 \\ 2.62 \\ 6.15 \end{bmatrix}, \quad \tilde{R}_2 = \begin{bmatrix} -4.04 \\ 2.44 \\ 2.00 \end{bmatrix}, \quad \tilde{R}_3 = \begin{bmatrix} -2.17 \\ -10.59 \\ 5.74 \end{bmatrix}, \quad \tilde{R}_c = \begin{bmatrix} -32.95 \\ 31.4 \\ 19.29 \end{bmatrix}. \tag{5.22}
\end{align*}
\]

From the position vectors, the LOS vectors in the reference frame are:

\[
\begin{align*}
\tilde{r}_1 &= \begin{bmatrix} 0.7830 \\ -0.5658 \\ -0.2583 \end{bmatrix}, \quad \tilde{r}_2 = \begin{bmatrix} 0.6508 \\ -0.6519 \\ -0.3892 \end{bmatrix}, \quad \tilde{r}_3 = \begin{bmatrix} 0.5721 \\ -0.7805 \\ -0.2519 \end{bmatrix}. \tag{5.23}
\end{align*}
\]

The attitude estimate is given by:
3U CubeSats detection

\[
\hat{A} = \begin{bmatrix}
0.3293 & -0.8463 & -0.4186 \\
0.8568 & 0.4541 & -0.2441 \\
0.3967 & -0.2783 & 0.8747
\end{bmatrix}
\]  \quad (5.24)
5.6 Discussion

From the attitude-error covariance matrix in Eq. (5.10), the attitude estimation method has two primary error sources: GPS and bounding box coordinates. The effect of the GPS positioning error on \( \sigma_{ri} \) is well understood. Roughly speaking, \( \sigma_{ri} \approx \sigma_{GPS}/r_i \), where \( r_i \) is the distance from the camera to the \( i \)-th CubeSat. The noise level \( \sigma_{bi} \) is a complicated function of the attitude of the CubeSats and the tightness of the bounding box and need to be determined by experimentation. In addition, \( \sigma_{bi} \) is approximately proportional to the pixel size of the camera and inversely proportional to the field of view of the camera. The field of view of the camera limits the number of CubeSats that simultaneously appear in the image. When the number drops below two, the attitude cannot be uniquely determined.
CHAPTER 6
CONCLUSIONS

The goal of this research framework is to provide an introduction and crucial knowledge to develop accurate CubeSats detection models using the Faster R-CNN. It also covers an example problem such as attitude estimation of a CubeSat using the detection results from the Faster R-CNN. This dissertation offers a comprehensive literature survey for the Faster R-CNN based on CubeSats detection, and tries to combine both CubeSats detection and attitude estimation to solve a vision task.

This dissertation provides discussions for difficulties and important considerations that need to be considered to develop accurate CubeSats detection models. The aim of this task is to provide a sound background to the Faster R-CNN inherent properties in order to obtain accurate features from an image. Therefore, research is performed to explore and develop CubeSats detection models using Web-searched and CAD images. In this dissertation, a two-step method is presented for detecting a rotating CubeSat in close proximity using 3D CAD images. For the first step, a wide range of experiments is conducted to develop accurate CubeSats detection models. For the preliminary work of this research, a CubeSats detection model using the Faster R-CNN with Web-searched images is developed. Then, experiments are analyzed CAD-based CubeSats detection models with and without texture features. The biggest challenges with these experiments are to detect small-scale CubeSats and to detect the correct shape of CubeSats. There are situations where the CubeSats detection process failed due to the difficulty of detecting the CubeSat. Therefore, in this
dissertation, the researcher proposed modifications to the Faster R-CNN to improve the accuracy of CubeSats detection models.

In the first step, Faster R-CNN for CubeSat detection processes all the images to locate the CubeSat in each image. In the second step, the classification results of the individual images are combined using the Dempster-Shafer theory of evidence. The researcher of this work proposed an effective way to integrate the Dempster-Shafer theory of evidence with the Faster R-CNN to detect a rotating CubeSat in close proximity from multiple images. The output of the second step is the aggregated probabilities of or beliefs in the predefined classes. The proposed method is tested on simulated scenarios (using Autodesk Maya) where the rotating 3U and 6U CubeSats are with different viewpoints and illumination levels which is well suited for this classification problem (See Appendix B).

Another objective of this research is to solve an attitude estimation problem using the detection results from the Faster R-CNN. When it comes to attitude determination, spacecraft attitude must be stabilized and controlled for a different number of reasons. In this dissertation, the researcher of this work proposed a coarse single-point attitude estimation method utilizing the centroids of the bounding boxes surrounding the CubeSats in the image. In this research, the SVD method has been examined to estimates a spacecraft attitude by minimizing Wahba’s loss function. The proposed estimation concept is tested on simulated scenarios (for 1U and 3U CubeSats) using Autodesk Maya.

Future research will focus on generalizing the CubeSats detection model to an extent that can detect more CubeSats classes (12U, 27U) in dynamic environments while optimizing the Faster R-CNN network. As a future problem to be solved, plan to evaluate other object detection methods (e.g., image segmentation methods) to solve the CubeSats detection problems in hand. Moreover, in order to increase the estimation accuracy, future works will focus on more quantitative error analysis for the attitude estimation problem.
BIBLIOGRAPHY


APPENDIX A
CUBESATS/NON-CUBESATS IMAGES
A.1 Website Links to CubeSats/Non-CubeSats Images

All CubeSats/Non-CubeSats images included in this dissertation are obtained through following websites (copy and paste Web-links in the Web-browser).

- [https://www.google.com/search?q=1U+cubesat&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj0rP7Zu4PcAhXBjJAKHQD0AisQ_AUICigB&biw=1697&bih=834](https://www.google.com/search?q=1U+cubesat&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj0rP7Zu4PcAhXBjJAKHQD0AisQ_AUICigB&biw=1697&bih=834)
- [https://www.sciencenewsforstudents.org/article/diamonds-and-more-suggest-unusual-origins-asteroids](https://www.sciencenewsforstudents.org/article/diamonds-and-more-suggest-unusual-origins-asteroids)
- [http://spaceref.com/nasa-hack-space/swarming-cubesats-for-science.html](http://spaceref.com/nasa-hack-space/swarming-cubesats-for-science.html)
- [https://amsat-uk.org/edsn-cubesat-swarm-nasa/](https://amsat-uk.org/edsn-cubesat-swarm-nasa/)
- [http://kawakatsu.isas.jaxa.jp/htmls/projectENG.html](http://kawakatsu.isas.jaxa.jp/htmls/projectENG.html)
- [https://www.google.com/search?q=astronauts+gloves&source=lnms&tbm=isch&sa=X&ved=0ahUKEwie0fuUnOLiAhWnTt8KHe1mCUqQ_AUIESgC&biw=1867&bih=904#imgrec=8TU7ig1Qzr0MUM:](https://www.google.com/search?q=astronauts+gloves&source=lnms&tbm=isch&sa=X&ved=0ahUKEwie0fuUnOLiAhWnTt8KHe1mCUqQ_AUIESgC&biw=1867&bih=904#imgrec=8TU7ig1Qzr0MUM:)
- [https://www.gaussteam.com/gallery-tupod-integration/](https://www.gaussteam.com/gallery-tupod-integration/)
- [http://www.aerospacetechnical.com/about](http://www.aerospacetechnical.com/about)
• http://www.madeinepal.com/2017/01/kits-multinational-birds-cubesats-are.html
• https://picswe.net/pics/-navsat-satellite-b5.html
• https://directory.eoportal.org/web/eoportal/satellite-missions/p/pw-sat
• https://space.skyrocket.de/doc_sdat/mysat-1.htm
• https://space.skyrocket.de/doc_sdat/sense.htm
• https://appel.nasa.gov/2010/04/12/the-next-big-thing-is-small/
• https://slideplayer.com/slide/5295053/
• https://www.amsat.org/wordpress/wp-content/uploads/2016/07/FoxLabsLive.jpg
• http://w6trw.com/index.php/tag/cubesat/
• http://culair.weebly.com/small-satellites.html
• https://www.jpl.nasa.gov/cubesat/
• https://www.jpl.nasa.gov/cubesat/missions/
• https://www.osa-opn.org/home/articles/volume_30/january_2019/features/cubesats_tiny_platforms_for_orbiting_optics/
B.1 Figures for Close Proximity Detection

Figure B.1

3U CubeSat
Figure B.2

3U CubeSat in dark environments
Figure B.3

6U CubeSat in unfavorable views
### B.2 Tables for Close Proximity Detection

**Table B.1**

Conflicting mass functions

<table>
<thead>
<tr>
<th></th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m_1(A))</td>
<td>0.99</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(m_2(A))</td>
<td>0</td>
<td>0</td>
<td>0.01</td>
<td>0.99</td>
<td>0</td>
</tr>
<tr>
<td>(m(A))</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(m^Y(A))</td>
<td>[0, 0.9999]</td>
<td>[0, 0.9999]</td>
<td>[0.0001, 1.000]</td>
<td>[0, 0.9999]</td>
<td>[0, 0.9999]</td>
</tr>
</tbody>
</table>

**Table B.2**

Extremely conflicting mass functions

<table>
<thead>
<tr>
<th></th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m_1(A))</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(m_2(A))</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(m(A))</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(m^Y(A))</td>
<td>[0, 1]</td>
<td>[0, 1]</td>
<td>[0, 1]</td>
<td>[0, 1]</td>
<td>[0, 1]</td>
</tr>
</tbody>
</table>
Table B.3

Less conflicting mass functions

<table>
<thead>
<tr>
<th></th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1(A)$</td>
<td>0.9890</td>
<td>0.0005</td>
<td>0.0100</td>
<td>0.0005</td>
<td>0</td>
</tr>
<tr>
<td>$m_2(A)$</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.0100</td>
<td>0.9890</td>
<td>0</td>
</tr>
<tr>
<td>$m(A)$</td>
<td>0.4540</td>
<td>0.0002</td>
<td>0.0918</td>
<td>0.4540</td>
<td>0</td>
</tr>
<tr>
<td>$m^Y(A)$</td>
<td>[0.0005, 0.9994]</td>
<td>[0, 0.9989]</td>
<td>[0.0001, 0.9990]</td>
<td>[0.0005, 0.9994]</td>
<td>[0, 0.9989]</td>
</tr>
</tbody>
</table>

Table B.4

Identical mass functions

<table>
<thead>
<tr>
<th></th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1(A)$</td>
<td>0.8</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m_2(A)$</td>
<td>0.8</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m(A)$</td>
<td>0.9412</td>
<td>0.0588</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$m^Y(A)$</td>
<td>[0.64, 0.96]</td>
<td>[0.04, 0.36]</td>
<td>[0, 0.32]</td>
<td>[0, 0.32]</td>
<td>[0, 0.32]</td>
</tr>
</tbody>
</table>

Table B.5

Weights in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0002</td>
<td>0.1135</td>
<td>0.8506</td>
<td>0.0356</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0042</td>
<td>0.9155</td>
<td>0.0801</td>
</tr>
<tr>
<td>3</td>
<td>0.0407</td>
<td>0.7812</td>
<td>0.1688</td>
<td>0.0082</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.0146</td>
<td>0.8296</td>
<td>0.1548</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.0007</td>
<td>0.8719</td>
<td>0.1273</td>
</tr>
<tr>
<td>6</td>
<td>0.0001</td>
<td>0.0014</td>
<td>0.9744</td>
<td>0.0241</td>
</tr>
</tbody>
</table>

Table B.6

Weights in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0003</td>
<td>0.0042</td>
<td>0.5844</td>
<td>0.411</td>
</tr>
<tr>
<td>2</td>
<td>0.0002</td>
<td>0.0182</td>
<td>0.621</td>
<td>0.3579</td>
</tr>
<tr>
<td>3</td>
<td>0.0849</td>
<td>0.2851</td>
<td>0.6067</td>
<td>0.0219</td>
</tr>
<tr>
<td>4</td>
<td>0.001</td>
<td>0.0062</td>
<td>0.6053</td>
<td>0.3872</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.0005</td>
<td>0.6337</td>
<td>0.3657</td>
</tr>
<tr>
<td>6</td>
<td>0.003</td>
<td>0.2304</td>
<td>0.453</td>
<td>0.3135</td>
</tr>
</tbody>
</table>
Table B.7
Weights in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.0002</td>
<td>0.9868</td>
<td>0.013</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0001</td>
<td>0.9929</td>
<td>0.007</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.9959</td>
<td>0.004</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.0003</td>
<td>0.8591</td>
<td>0.1406</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.0002</td>
<td>0.0015</td>
<td>0.9982</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0.0001</td>
<td>0.9998</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9999</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0.0001</td>
<td>0.9998</td>
</tr>
</tbody>
</table>

Table B.8
Bayes’ rule/Dempster’s rule (all weights, simple sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0002</td>
<td>0.1135</td>
<td>0.8506</td>
<td>0.0356</td>
<td>0.0001</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0006</td>
<td>0.9958</td>
<td>0.0036</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.0028</td>
<td>0.997</td>
<td>0.0002</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0.9999</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.9
Bayes’ rule/Dempster’s rule (all weights, simple sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0003</td>
<td>0.0042</td>
<td>0.5844</td>
<td>0.411</td>
<td>0.0001</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0001</td>
<td>0.7115</td>
<td>0.2884</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.0001</td>
<td>0.9855</td>
<td>0.0144</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0.9907</td>
<td>0.0093</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0.9946</td>
<td>0.0054</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0.9963</td>
<td>0.0037</td>
<td>0</td>
</tr>
</tbody>
</table>
Table B.10
Bayes’ rule/Dempster’s rule (all weights, simple sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.0002</td>
<td>0.9868</td>
<td>0.013</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.9999</td>
<td>0.0001</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0.7386</td>
<td>0.2614</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0.0001</td>
<td>0.9999</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.11
Dempster’s rule (one weight, simple sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 0.1494]</td>
<td>[0.8506, 1.0000]</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 0.1494]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 0.0126]</td>
<td>[0.9874, 1.0000]</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 0.0126]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0121]</td>
<td>[0.0431, 0.0552]</td>
<td>[0.9448, 0.9569]</td>
<td>[0.0000, 0.0121]</td>
<td>[0.0000, 0.0121]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0021]</td>
<td>[0.0076, 0.0098]</td>
<td>[0.9902, 0.9924]</td>
<td>[0.0000, 0.0021]</td>
<td>[0.0000, 0.0021]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0010, 0.0013]</td>
<td>[0.9987, 0.9990]</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0000, 0.0003]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>

Table B.12
Dempster’s rule (one weight, simple sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 0.4156]</td>
<td>[0.5844, 1.0000]</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 0.4156]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 0.1575]</td>
<td>[0.8425, 1.0000]</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 0.1575]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 0.0620]</td>
<td>[0.9380, 1.0000]</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 0.0620]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 0.0245]</td>
<td>[0.9755, 1.0000]</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 0.0245]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0090]</td>
<td>[0.0000, 0.0090]</td>
<td>[0.9910, 1.0000]</td>
<td>[0.0000, 0.0090]</td>
<td>[0.0000, 0.0090]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 0.0049]</td>
<td>[0.9951, 1.0000]</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 0.0049]</td>
</tr>
</tbody>
</table>
Table B.13
Dempster’s rule (one weight, simple sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 0.0132]</td>
<td>[0.9868, 1.0000]</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 0.0132]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0001]</td>
<td>[0.9999, 1.0000]</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0001]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.8399, 0.8399]</td>
<td>[0.1601, 0.1601]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>7</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0003, 0.0003]</td>
<td>[0.9997, 0.9997]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>8</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>

Table B.14
Dempster’s rule (all weights, complex sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0002]</td>
<td>[0.0000, 0.1137]</td>
<td>[0.0000, 0.9643]</td>
<td>[0.0356, 0.9996]</td>
<td>[0.0001, 0.0001]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0005]</td>
<td>[0.0002, 0.8872]</td>
<td>[0.1128, 0.9998]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0004]</td>
<td>[0.0365, 0.8840]</td>
<td>[0.1160, 0.9635]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0310, 0.7512]</td>
<td>[0.2488, 0.9690]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0272, 0.6582]</td>
<td>[0.3418, 0.9728]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[<strong>0.0266, 0.6428</strong>]</td>
<td>[<strong>0.3572, 0.9734</strong>]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>

Table B.15
Dempster’s rule (all weights, complex sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0000, 0.0044]</td>
<td>[0.0000, 0.5889]</td>
<td>[0.4110, 0.9996]</td>
<td>[0.0001, 0.0001]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0003, 0.3777]</td>
<td>[0.6223, 0.9997]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0342, 0.3900]</td>
<td>[0.6100, 0.9658]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0216, 0.2423]</td>
<td>[0.7577, 0.9784]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0138, 0.1549]</td>
<td>[0.8451, 0.9862]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[<strong>0.0100, 0.1071</strong>]</td>
<td>[<strong>0.8929, 0.9900</strong>]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>
### Table B.16

Dempster's rule (all weights, complex sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0000, 0.9870]</td>
<td>[0.0130, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.9802]</td>
<td>[0.0198, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.9763]</td>
<td>[0.0237, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.8390]</td>
<td>[0.1610, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0015]</td>
<td>[0.9985, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>7</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>8</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>

### Table B.17

Dempster's rule (one weight, complex sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.1494]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0126]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0028]</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0028]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0005]</td>
<td>[0.0000, 0.0022]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0005]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0001]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0001]</td>
</tr>
</tbody>
</table>

### Table B.18

Dempster's rule (one weight, complex sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.4156]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.1575]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0620]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0245]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0090]</td>
<td>[0.0000, 0.0090]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0090]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0049]</td>
</tr>
</tbody>
</table>
Table B.19

Dempster’s rule (one weight, complex sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0132]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0001]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0018]</td>
<td>[0.9982, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>7</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>8</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>

Table B.20

Yager’s rule (all weights, simple sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0002</td>
<td>0.1135</td>
<td>0.8506</td>
<td>0.0356</td>
<td>0.0001</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.2180]</td>
<td>[0.0005, 0.2185]</td>
<td>[0.7787, 0.9967]</td>
<td>[0.0028, 0.2208]</td>
<td>[0.0000, 0.2180]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.8682]</td>
<td>[0.0004, 0.8685]</td>
<td>[0.1314, 0.9996]</td>
<td>[0.0000, 0.8682]</td>
<td>[0.0000, 0.8682]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.8910]</td>
<td>[0.0000, 0.8910]</td>
<td>[0.1090, 1.0000]</td>
<td>[0.0000, 0.8910]</td>
<td>[0.0000, 0.8910]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.9049]</td>
<td>[0.0000, 0.9049]</td>
<td>[0.0951, 1.0000]</td>
<td>[0.0000, 0.9049]</td>
<td>[0.0000, 0.9049]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.9074]</td>
<td>[0.0000, 0.9074]</td>
<td>[0.0926, 1.0000]</td>
<td>[0.0000, 0.9074]</td>
<td>[0.0000, 0.9074]</td>
</tr>
</tbody>
</table>

Table B.21

Yager’s rule (all weights, simple sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0003</td>
<td>0.0042</td>
<td>0.5844</td>
<td>0.411</td>
<td>0.0001</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.4899]</td>
<td>[0.0001, 0.4900]</td>
<td>[0.3629, 0.8528]</td>
<td>[0.1471, 0.6370]</td>
<td>[0.0000, 0.4899]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.7766]</td>
<td>[0.0000, 0.7766]</td>
<td>[0.2202, 0.9968]</td>
<td>[0.0032, 0.7798]</td>
<td>[0.0000, 0.7766]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.8655]</td>
<td>[0.0000, 0.8655]</td>
<td>[0.1333, 0.9988]</td>
<td>[0.0012, 0.8667]</td>
<td>[0.0000, 0.8655]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.9151]</td>
<td>[0.0000, 0.9151]</td>
<td>[0.0845, 0.9995]</td>
<td>[0.0005, 0.9155]</td>
<td>[0.0000, 0.9151]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.9616]</td>
<td>[0.0000, 0.9616]</td>
<td>[0.0383, 0.9999]</td>
<td>[0.0001, 0.9617]</td>
<td>[0.0000, 0.9616]</td>
</tr>
</tbody>
</table>
### Table B.22

Yager’s rule (all weights, simple sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.0002</td>
<td>0.9868</td>
<td>0.013</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0202]</td>
<td>[0.0000, 0.0202]</td>
<td>[0.9797, 0.9999]</td>
<td>[0.0001, 0.0203]</td>
<td>[0.0000, 0.0202]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0242]</td>
<td>[0.0000, 0.0242]</td>
<td>[0.9758, 1.0000]</td>
<td>[0.0000, 0.0242]</td>
<td>[0.0000, 0.0242]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.1618]</td>
<td>[0.0000, 0.1618]</td>
<td>[0.8382, 1.0000]</td>
<td>[0.0000, 0.1618]</td>
<td>[0.0000, 0.1618]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.9987]</td>
<td>[0.0000, 0.9987]</td>
<td>[0.0013, 1.0000]</td>
<td>[0.0000, 0.9987]</td>
<td>[0.0000, 0.9987]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
</tr>
<tr>
<td>7</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
</tr>
<tr>
<td>8</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
</tr>
</tbody>
</table>

### Table B.23

Yager’s rule (one weight, simple sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 0.1494]</td>
<td>[0.8506, 1.0000]</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 0.1494]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 0.0126]</td>
<td>[0.9874, 1.0000]</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 0.0126]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.7741]</td>
<td>[0.0099, 0.7840]</td>
<td>[0.2160, 0.9901]</td>
<td>[0.0000, 0.7741]</td>
<td>[0.0000, 0.7741]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.7800]</td>
<td>[0.0017, 0.7817]</td>
<td>[0.2183, 0.9983]</td>
<td>[0.0000, 0.7800]</td>
<td>[0.0000, 0.7800]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.7810]</td>
<td>[0.0002, 0.7813]</td>
<td>[0.2187, 0.9998]</td>
<td>[0.0000, 0.7810]</td>
<td>[0.0000, 0.7810]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.7812]</td>
<td>[0.0000, 0.7812]</td>
<td>[0.2188, 1.0000]</td>
<td>[0.0000, 0.7812]</td>
<td>[0.0000, 0.7812]</td>
</tr>
</tbody>
</table>

### Table B.24

Yager’s rule (one weight, simple sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 0.4156]</td>
<td>[0.5844, 1.0000]</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 0.4156]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 0.1575]</td>
<td>[0.8425, 1.0000]</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 0.1575]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 0.0620]</td>
<td>[0.9380, 1.0000]</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 0.0620]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 0.0245]</td>
<td>[0.9755, 1.0000]</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 0.0245]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0099]</td>
<td>[0.0000, 0.0099]</td>
<td>[0.9910, 1.0000]</td>
<td>[0.0000, 0.0099]</td>
<td>[0.0000, 0.0099]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 0.0049]</td>
<td>[0.9951, 1.0000]</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 0.0049]</td>
</tr>
</tbody>
</table>
Table B.25

Yager’s rule (one weight, simple sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 0.0132]</td>
<td>[0.9868, 1.0000]</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 0.0132]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0001]</td>
<td>[0.9999, 1.0000]</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0001]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.9982]</td>
<td>[0.0000, 0.9982]</td>
<td>[0.0018, 1.0000]</td>
<td>[0.0000, 0.9982]</td>
<td>[0.0000, 0.9982]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0002]</td>
<td>[0.0000, 0.1137]</td>
<td>[0.0000, 0.9643]</td>
<td>[0.0356, 0.9996]</td>
<td>[0.0001, 0.0001]</td>
</tr>
<tr>
<td>7</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
</tr>
<tr>
<td>8</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
</tr>
</tbody>
</table>

Table B.26

Yager’s rule (all weights, complex sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0002]</td>
<td>[0.0000, 0.1137]</td>
<td>[0.0000, 0.9643]</td>
<td>[0.0356, 0.9996]</td>
<td>[0.0001, 0.0001]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0004]</td>
<td>[0.0000, 0.0009]</td>
<td>[0.0002, 0.8872]</td>
<td>[0.1128, 0.9998]</td>
<td>[0.0000, 0.0004]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0062]</td>
<td>[0.0000, 0.0065]</td>
<td>[0.0362, 0.8847]</td>
<td>[0.1153, 0.9638]</td>
<td>[0.0000, 0.0062]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0127]</td>
<td>[0.0000, 0.0127]</td>
<td>[0.0306, 0.7544]</td>
<td>[0.2456, 0.9694]</td>
<td>[0.0000, 0.0127]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0167]</td>
<td>[0.0000, 0.0167]</td>
<td>[0.0267, 0.6639]</td>
<td>[0.3361, 0.9733]</td>
<td>[0.0000, 0.0167]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0174]</td>
<td>[0.0000, 0.0174]</td>
<td>[0.0261, 0.6490]</td>
<td>[0.3510, 0.9739]</td>
<td>[0.0000, 0.0174]</td>
</tr>
</tbody>
</table>

Table B.27

Yager’s rule (all weights, complex sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0000, 0.0044]</td>
<td>[0.0000, 0.5889]</td>
<td>[0.4110, 0.9996]</td>
<td>[0.0001, 0.0001]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0030]</td>
<td>[0.0000, 0.0030]</td>
<td>[0.0003, 0.3795]</td>
<td>[0.6205, 0.9997]</td>
<td>[0.0000, 0.0030]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0571]</td>
<td>[0.0000, 0.0571]</td>
<td>[0.0322, 0.4248]</td>
<td>[0.5752, 0.9678]</td>
<td>[0.0000, 0.0571]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0704]</td>
<td>[0.0000, 0.0704]</td>
<td>[0.0201, 0.2957]</td>
<td>[0.7043, 0.9799]</td>
<td>[0.0000, 0.0704]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0778]</td>
<td>[0.0000, 0.0778]</td>
<td>[0.0127, 0.2207]</td>
<td>[0.7793, 0.9873]</td>
<td>[0.0000, 0.0778]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0843]</td>
<td>[0.0000, 0.0843]</td>
<td>[0.0091, 0.1823]</td>
<td>[0.8177, 0.9909]</td>
<td>[0.0000, 0.0843]</td>
</tr>
</tbody>
</table>
### Table B.28

Yager’s rule (all weights, complex sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0003]</td>
<td>[0.0000, 0.9870]</td>
<td>[0.0130, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.9802]</td>
<td>[0.0198, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.9763]</td>
<td>[0.0237, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.8390]</td>
<td>[0.1610, 0.9999]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.8390]</td>
<td>[0.9983, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.9763]</td>
<td>[0.9997, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>7</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.9763]</td>
<td>[0.9997, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>8</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0001, 0.9763]</td>
<td>[0.9997, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>

### Table B.29

Yager’s rule (one weight, complex sets) in case 1

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 0.1494]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.1494]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 0.0126]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0126]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>

### Table B.30

Yager’s rule (one weight, complex sets) in case 2

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 0.4156]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.4156]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 0.1575]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.1575]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 0.0620]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0620]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 0.0245]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0245]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0090]</td>
<td>[0.0000, 0.0090]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0090]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 0.0049]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0049]</td>
</tr>
</tbody>
</table>
Table B.31

Yager’s rule (one weight, complex sets) in case 3

<table>
<thead>
<tr>
<th>k</th>
<th>“1U CubeSat”</th>
<th>“2U CubeSat”</th>
<th>“3U CubeSat”</th>
<th>“6U CubeSat”</th>
<th>“Background”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 0.0132]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0132]</td>
</tr>
<tr>
<td>2</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 0.0001]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0001]</td>
</tr>
<tr>
<td>3</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>4</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>5</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0018]</td>
<td>[0.9982, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>6</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>7</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
<tr>
<td>8</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[0.0000, 0.0000]</td>
<td>[1.0000, 1.0000]</td>
<td>[0.0000, 0.0000]</td>
</tr>
</tbody>
</table>