Surface Defect Detection in Aircraft Skin & Visual Navigation based on Forced Feature Selection through Segmentation

Tyler B. Hussey

Surface Defect Detection in Aircraft Skin & Visual Navigation based on Forced Feature Selection through Segmentation

THESIS

Tyler B Hussey, Captain, USAF

AFIT-ENG-MS-21-M-049

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-21-M-049

Surface Defect Detection in Aircraft Skin & Visual Navigation based on Forced Feature Selection through Segmentation

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Tyler B Hussey, B.S.E.E., M.S.E.M

Captain, USAF

March 26, 2021

AFIT-ENG-MS-21-M-049

Surface Defect Detection in Aircraft Skin

&

Visual Navigation based on Forced Feature Selection through Segmentation

THESIS

Tyler B Hussey, B.S.E.E., M.S.E.M
Captain, USAF

AFIT-ENG-MS-21-M-049

Abstract

This paper explorers two unique possible applications for CNNs: surface defect detection and improved feature selection. Both topics will be explored in this project. The aircraft owned and maintained by the US Department of Defense continue to age and resources required to catalog and maintain air frame integrity continually increase. Visual inspection of aircraft skin for surface defects is an area of maintenance that is particularly intensive for time and manpower. One novel way to combat this problem is through the use of autonomous scanning of the aircraft with cameras and processing the imagery with ANNs, or more specifically, semantic segmentation via Convolutional Neural Networks (CNN). Semantic segmentation has the ability to localize and classify individual pixels in an image, making it an ideal tool for detecting surface defects in aircraft skin. The research in this thesis evaluates the use of classical and CNN-based tools for semantic segmentation. A classical approach is provided through the statistical measure of Histogram of Orientated Gradients as an image descriptor and Support Vector Machines for image classification. These classification results are then compared to the modern CNN approach utilizing a custom U-Net model for pixel-wise image classification. Due to Covid-19, the data for aircraft skin defects was unavailable, therefore all tests were conducted on a similar application: surface defect detection in steel plates. Given that state-of-the-art segmentation models are highly tailored to the task at hand, the results presented will serve as a baseline for future work. The custom U-net model developed was then shifted to another project and aligned with real world data. Given the world's heavily reliance on global positioning systems (GPS) for navigation there is a large push for alternative methods. One of these methods is through computer-aided visual navigation. The main hurdle up to this point has been to establish meaningful and robust features for consistent and accurate feature matching between aerial and satellite imagery. The research in the paper explores the use of semantic segmentation of aerial imagery as a way to force feature selection onto key areas of an image that might be more likely to correspond under seasonal variations. Utilizing feature selection and matching on the masked aerial image and the satellite image produces a set of reliable key points that can be used for camera pose estimation and visual navigation.

Table of Contents

Page

Page

List of Figures

Figure Page

List of Tables

Surface Defect Detection in Aircraft Skin

&

Visual Navigation based on Forced Feature Selection through Segmentation

I. Introduction

1.1 Overview

This thesis will cover two topics to include steel surface defect detection using semantic segmentation (Problem A) and visual navigation through forced feature matching semantic segmented aerial imagery with satellite imagery (Problem B). The original project was to develop a semantic segmentation model to detect and classify surface defects in the skin of an aircraft. However, due to the spread of COVID-19, collection of any real-world imagery was delayed indefinitely. In the end the decision was made to switch topics to a similar project with readily available data. All work completed, up until the topic switch, was tested and compared against a similar research problem, surface defect detection in steel plates. The second problem presented in this paper involves using semantic segmented images as a way to force feature selection onto unique, easily distinguishable features of an image. Both projects involved semantic segmentation models, so the work on the segmentation model from Problem A was altered slightly and carried over onto the second project. The methodologies and work completed on Problem A will be provided in Chapter 3 along with the preliminary results in Chapter 4. The work and results presented for Problem A do not represent a completed work, but merely serve as documentation and a reference for future work in this area.

1.2 Problem A: Steel surface defect detection

The existence of stealth technology was first announced at a Pentagon news conference on Aug. 22, 1980. Stealth technology was designed in order to reduce an aircraft's signature over a wide spectrum of signals to include: electromagnetic, infrared, visual, acoustic, and radar. The stealth technology today can be broken into two parts. The first part is the shape of the aircraft, stealth aircraft are designed with sharp edges in order to reflect any signals back and away from the source preventing any return signal to the radar. The second key to stealth is through the utilization of "Radar Absorbent Material" (RAM) or "Low Observable" (LO) Material. RAM works by absorbing incoming signals and dispersing the energy as heat, rather than reflecting the signal back to the source. It is the combination of these two key elements that establishes the stealth technology seen on aircraft in the military today.

Over the last few decades stealth has become the predominate strong arm of the Air Force. The Air Force relies heavily on its stealth technology to keep its Airmen safe while simultaneously providing a tactical advantage over its adversaries. However, in order to maintain this advantage the Air Force has had to wage an even greater battle with a far more relentless adversary, time. As stealth coatings are becoming more predominate on today's aircraft the demands for daily upkeep and maintenance have skyrocketed. With today's methods, each aircraft requires 3-4 skilled inspectors to put eyes on every inch of the aircraft, looking for surface defects. This inspection can sometimes take up to 4 hours per aircraft to properly inspect and catalog the defects found on the aircraft. This issue has created a drain on human resources and has opened an opportunity for faster and more accurate inspection techniques.

1.2.1 Research Objectives

The goal of this research is to test the viability of utilizing computer vision to reduce the number of man-hours required to complete a single aircraft inspection. The initial solution is through the utilization of an automated aerial vehicle and an on-board camera. Images will be collected and analyzed, in real-time, to provide an accurate identification and localization of surface defects. The goal of this research will be to develop a set of procedures that will provide proper instructions for image production and segmentation of surface defects. Segmentation involves the selection or creation of an Artificial Neural Network (ANN) to detect and classify surface defects faster than a group of skilled inspectors. This research will focus on comparing results from statistical computer vision methods, such as Histogram of Orientated Gradients (HOG), against more state-of the-art image segmentation techniques utilizing ANNs and various algorithms to detect different type of defects.

1.2.2 Methodology

While the objective of this project was to develop a LO material surface defect detection model, real world data was not available at the time of submitting the project. Therefore, a similar related project, steel surface defect, was substituted in as a replacement database due to the similarities in defect types. The first method tested was based on a traditional statistical approach using Histogram of Orientated Gradients (HOG) and a Support Vector Machine(SVM). The second method focuses on a more modern approach using an ANN model for semantic segmentation.

1.2.3 Assumptions

The selection of the steel surface defect database was based on the characteristics of defects and the availability of data. Given both aircraft skin and steel plates are primarily homogeneous surfaces and contain similar defects the results obtained on steel plates should provide a good baseline for skin aircraft defect detection.

1.3 Problem B: Forced Feature Selection for Visual Navigation

In 1983 Korean Air Lines flight 007 was tragically shot down, by a Soviet fighter, due to a navigational error. Ever since the flight 007 incident the use of global navigation satellite systems has been a staple for navigation in the civilian and military world. Currently GPS is the primary source for all navigation solutions in the military as it is cheap and easy to implement and readily available worldwide. However, the heavy reliance and investment in GPS creates several vulnerabilities in our military. GPS is widely available to the public and, as such, is easily jammed and manipulated by intentional and accidental means, thus creating a need for other alternative navigation methods. In 2015 the Air Force published the Air Force Future Operating Concept for 2035, one of their primary desires was to develop "New concepts and capabilities to counter the increasing technology and proliferation of anti-access and area denial threats, to include multi-domain approaches and systems that can be rapidly modified when adversaries adapt their defenses. [5]" The area denial capabilities is referring to the DoD's heavy reliance on GPS as the sole source of navigation. While GPS has reigned supreme for many years, it is very susceptible to area-denial attacks such as jamming and spoofing. One of the alternative in development that has shown promise as a GPS alternative is through vision-aided navigation.

1.3.1 Research objectives

The primary objective of this research is to test the effectiveness of using semantic segmentation as a way to create and force robust features onto desired areas of an image for the purpose of visual navigation. This involves testing several segmentation algorithms to achieve state-of-the-art segmentation results, and evaluating the effectiveness of feature matching on segmented imagery.

1.3.2 Research focus

The focus of this research will be on the development of a near state-of-the-art semantic segmentation model for aerial imagery that can extract desired buildings and roads from an image. The research will then focus on various feature selection and feature matching algorithms in order to compare the segmented aerial key features with a known database of features from satellite imagery.

1.3.3 Methodology

The methodologies for this project are broken down into 3 parts: semantic segmentation of roads and buildings, feature selection, and feature matching algorithms. The semantic segmentation was conducted using a custom U-net trained on the Inria Aerial dataset for buildings and the Massachusetts dataset for Roads. Using the trained model, locally collected flight test data was fed in as a series of images as the input. The output of the model was a masked version of the collected flight data. These masked images were then reduced down to feature key points using the Scale Invariant Feature Transform (SIFT) algorithm. These key points were then compared against a known database of key points from the satellite imagery to locate matches.

1.3.4 Assumptions

A large-scale PnP algorithm has already been conducted in previous research in order determine a rough location of the aircraft. Working on the assumption that a rough location could be determined, satellite imagery was hand selected from areas along the flight path of the test aircraft. This was done in order to test best case scenario, or proof of concept, when it came to feature matching aerial imagery with satellite imagery.

Due to data currently available and time constraints, the data and imagery used for model training and visual navigation tests were selected for the presence of seasonal invariant features, which in this case are confined to buildings and roads. Buildings and roads were selected based on the assumption, excluding heavy snow, that these features would be seasonally invariant.

1.3.5 Limitations

The developed algorithm would most likely not work as well in a rural environment due to the sparsity of buildings and roads. However, this initial research is not intended to be the end all result, but a proof of concept. Once successful results are demonstrated, we will expand to other categories in the semantic segmentation.

1.4 Hardware and Software

Both projects contained in this thesis were written in Python. The project's design and initial small scale testing was accomplished on an ASUS ROG laptop equipped with an RTX 2070. Once a project achieved acceptable performance, it was transferred to a server located at the Air Force Institute of Technology. The server consisted of four NVIDIA Titan RTXs with 24 Gb RAM and was used for the full scale training and testing of the models.

1.5 Thesis Contributions

- Surface Defect Detection
  - Developed groundwork for statistical and machine-learning approach for surface defect detection applications

  - Researched and procured required hardware for future deployment to drone vehicle for remote scanning of aircraft skin

- Forced Feature Selection
  - Developed baseline building and road semantic segmentation model for aerial imagery
  - Established procedures for post-processing model output for improved feature selection performance
  - Laid groundwork for future work in feature matching between aerial and satellite imagery

1.6 Document Overview

The research objectives and goals will be discussed in more detail in the following chapters. Chapter 2 will cover the literature review for both projects as a majority of the literature can be applied to both applications. Chapter 3 is broken into two sections, each section dedicated to providing a detailed description of the experiments and processes used for both projects. Chapter 4 will also follow the same structure, the first half will cover the preliminary results from the surface defect detection algorithms. The second half of chapter 4 covers the results obtained from the forced feature selection for visual navigation. Finally Chapter 5 will provide the conclusions gathered as well as suggestions for further iterations and continued research.

II. Background and Literature Review

This chapter will provide background and relevant literature for the two problems introduced in Chapter 1. The literature review for surface defect detection will focus on various semantic segmentation algorithms to include model architecture, various loss functions, evaluation metrics, and pre-trained models. Chapter 2 will also cover the basis for various statistical methods that were used as a baseline to evaluate model performance. The second half of the chapter will cover pertinent background for forced feature selection via segmentation models. The segmentation models and various architectures are already well-covered in the first half of the chapter, so the second half will only cover the nuances specific to the second problem and various feature-detection algorithms.

2.1 LO Surface Defect Inspection

The literature surrounding defects in stealth coatings to include defect types, images, and vision-based inspection techniques are almost non-existent or open to the public. Therefore, the scope of the literature review in this area will focus on other various aircraft skin defects and common techniques used for surface defect detection on various homogeneous surfaces.

The following figures show examples of common defects found in aircraft skin and the methods for marking the defects for tracking growth. These marking guides represent the desired goal for the segmentation model and its ability to identify defect boundaries in a precise manner to facilitate accurate tracking over time. Figure 1 shows the proper method for marking missing or damaged material.

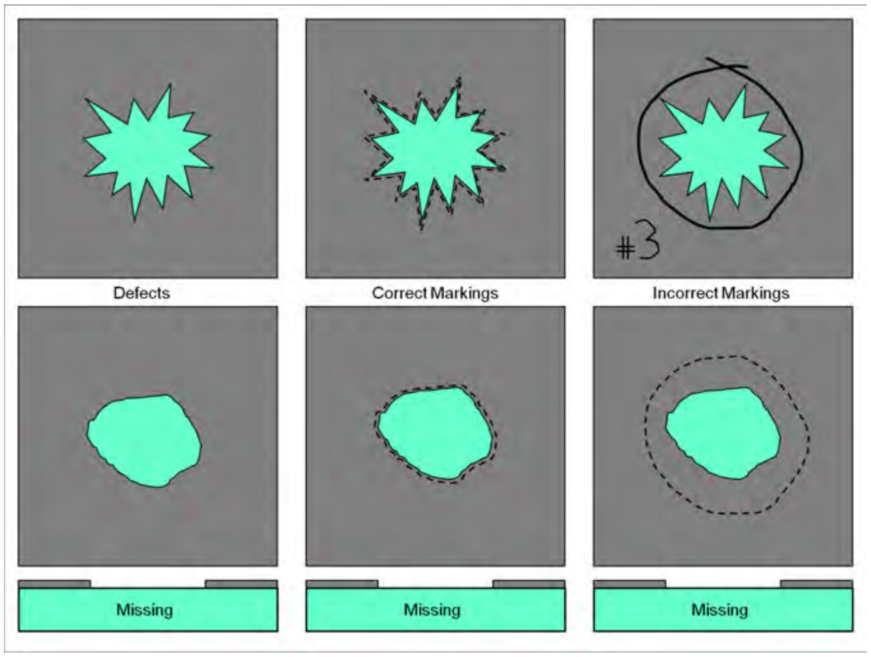


Figure 1: Area Defect marking guide. The grey represents the aircraft skin and blue indicates where the skin may have broken off showing the material underneath. The first Column shows 2 initial defects with the third image being viewed from the side to show that the aircraft skin is missing. The second column demonstrates the correct way to mark a defect and the desired goal of the segmentation model. Column 3 demonstrates the incorrect marking style with a loosely defined boundary.

Figure 2 shows another type of "area" defect, but where the material is neither missing or damaged. This type of defect is referred to as a disbonded area or a "bubble". This particular defect will be the hardest defect to detect as a bubble can be very hard to see with a monocular camera. This issue can be mitigated with harsh lighting conditions and proper placement of the camera. However, the placement of the camera will be dependant on the location of the drone and harsh lighting will require additional equipment.

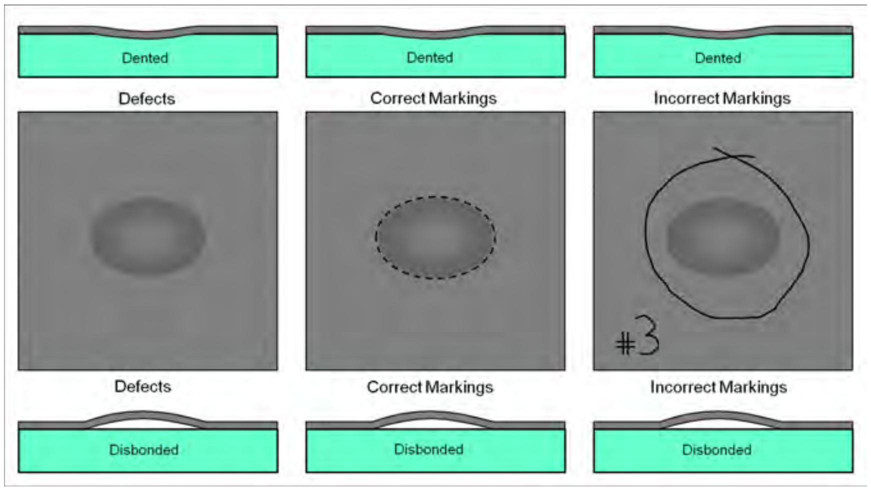


Figure 2: Dis-bonded Defect Marking Guide. This image follows the same formatting for columns as the previous image. However, this image represents a dis-bonded, or bubble, defect

Another defect type found in aircraft material are the defects that are along the edge of a panel. Due to the extreme conditions and speeds of an aircraft, the edge of a panel takes the brunt of the force, often resulting in chips and cracks along the panel edge as seen in Figure 3.

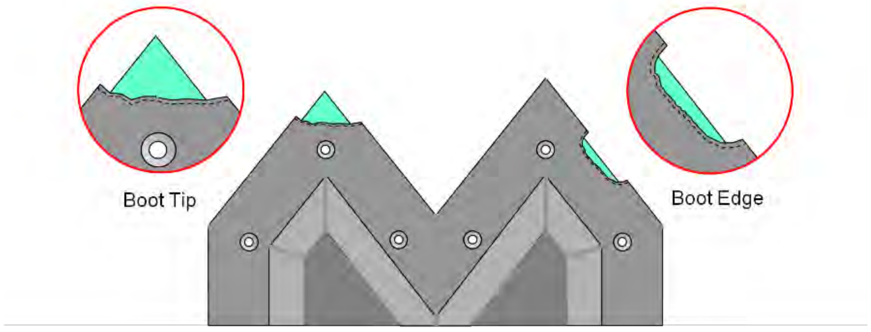


Figure 3: Edge Defect Marking Guide. This image shows common defects that form along the edge or tips or a panel cover.

Another subcategory of surface defects, a byproduct of construction, revolves around the fasteners used to hold the panels together. There are 2 types of fasteners, permanent and removable. Removable fasteners are intended to be removed throughout the course of routine maintenance to allow access to certain panels and doors. Whereas permanent fasteners are not expected to be removed, outside of depot or major unscheduled repair activity, and intended to remain intact for the life of the aircraft. Figures 4 and 5 show the marking guide and several common defects common to fasteners. The main difference between permanent and removable fasteners is the circle around the bolt itself. Permanent fasteners, if covered correctly, should not be visible from the surface. Removable fasteners will have an unavoidable circle around the fastener head. A common defect with removable fasteners is a missing plug that covers the bolt drive. Removable fasteners also create an additional edge in the panel, which results in additional cracks and chips around this edge. It is worth noting that edge defects, including those around fasteners, make up the majority of the defects seen in aircraft skin.

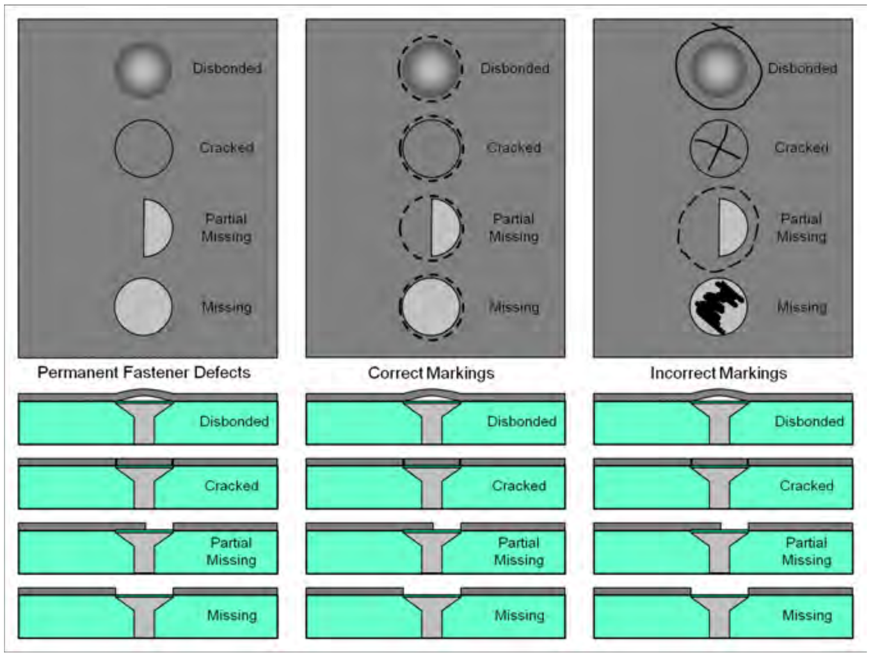


Figure 4: Permanent Fastener Marking Guide. The grey represents the aircraft skin and blue indicates where the skin may have broken off showing the material underneath. The first Column shows initial defects common with permanent fasteners. Permanent fasteners are designed to never be removed, as such, are covered over with aircraft skin. The bottom half of the image shows the fasteners when viewed from the side. The second column demonstrates the correct way to mark a defect and the desired goal of the segmentation model. Column 3 demonstrates the incorrect marking style with a loosely defined boundary.

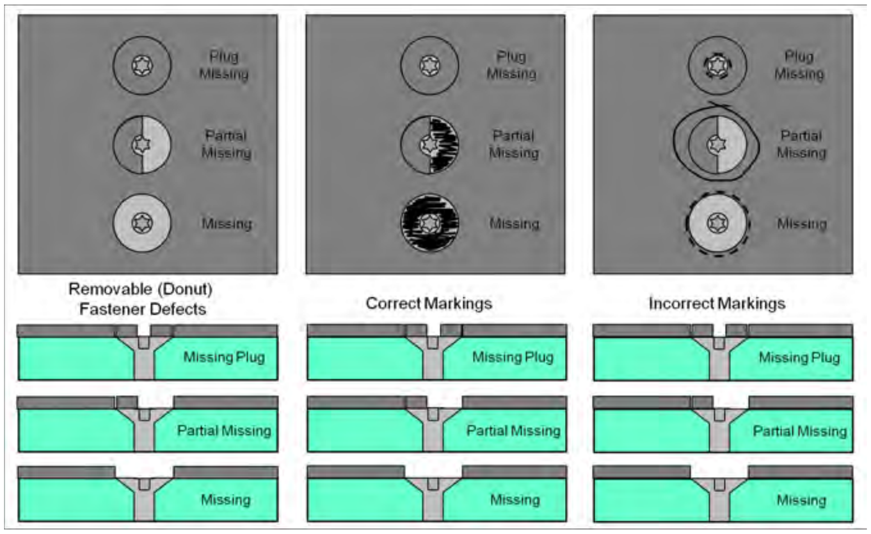


Figure 5: Removable Fastener Marking Guide. The grey represents the aircraft skin and blue indicates where the skin may have broken off showing the material underneath. The first Column shows initial defects common with removable fasteners. Removable fasteners are designed to be taken out, this feature results in an edge along the border of the plug. The bottom half of the image shows the fasteners when viewed from the side. The second column demonstrates the correct way to mark a defect and the desired goal of the segmentation model. Column 3 demonstrates the incorrect marking style with a loosely defined boundary.

2.2 Image Classification

When it comes to image classification there are four different classifications/methods used, each varying in levels of fidelity. As shown in Figure 6 the first method is simply named image classification and focuses on the general question, "What is in the picture?". Image classification aims to apply a single label to the image in its entirety, essentially capturing the general sense of an image. The next step further is referred to as object detection, focusing on the objects in the image and where they are located. Object detection is capable of detecting multiple objects and applying multiple labels to a single image. These object labels are often associated with bounding boxes that aim to locate where an object resides in the image. The third method, and the focus of this research, is semantic segmentation; semantic segmentation is the process of categorizing each pixel into a particular label. Then there is instance segmentation, which works similarly to the previous method except that it tries to apply unique labels to every instance of similar categories. Where semantic segmentation would label all people into a single category, instance segmentation would attempt to identify every unique individual. As each of the four methods expands on their fidelity, so does the difficulty associated with applying each method. The research presented in this paper will primarily focus on semantic segmentation. While instance segmentation can provide a higher fidelity in defect tracking of individual defects, the data set used for training was labeled for semantic segmentation. Semantic segmentation should also be more than sufficient to track defect growth, as two overlapping defects would be considered as a single larger defect.

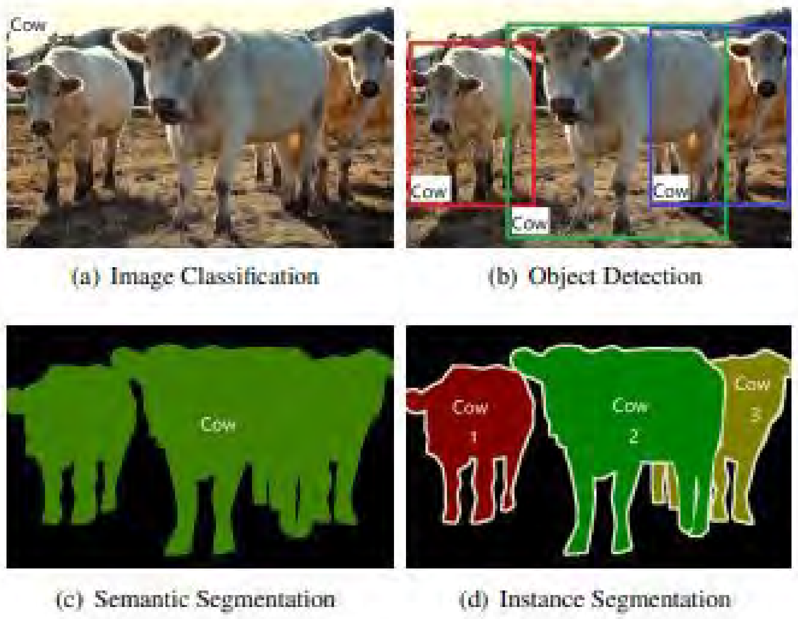


Figure 6: Image classification methods [6]

2.2.1 Histogram of Orientated Gradients

HOG is a feature descriptor that is utilized primarily for object detection in imagery and was first introduced in 2005 by Dalal and Triggs[7]. HOG works by compiling a set of gradient orientations into a histogram that can describe the image. Gradients of an image are useful because the magnitude of a gradient is the greatest around edges and corners, making HOG well suited for surface defect detection. These gradients are calculated by sliding a small window, called a cell, across the image and calculating a local histogram of gradients. The gradients are calculated utilizing the following kernel or discrete derivative mask in both the horizontal and vertical direction.

- Horizontal $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$
- Vertical $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

Given the gradient vector

$$\Delta f = \begin{bmatrix} \Delta fx \\ \Delta fy \end{bmatrix} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix} \tag{1}$$

the gradient's magnitude and orientation are calculated using the following equations.

- Magnitude $= \sqrt{\Delta fx^2 + \Delta fy^2}$
- Orientation:$\theta = arctan \begin{bmatrix} \Delta fy \\ \Delta fx \end{bmatrix}$

The gradients can then sorted by orientation and placed into a set number of bins. The greater the magnitude of a gradient, the greater effect it will have on its respective bin [1]. The HOG descriptor is the concatenation of all these histograms. As shown in Figure 7, the cells make up the size of the window that will be shifted over the image creating multiple HOG features, which are then converted into cell histograms. These cells are then combined into a cell block to normalize the magnitude of adjacent cells and reduce the influence of local lighting and noise conditions.

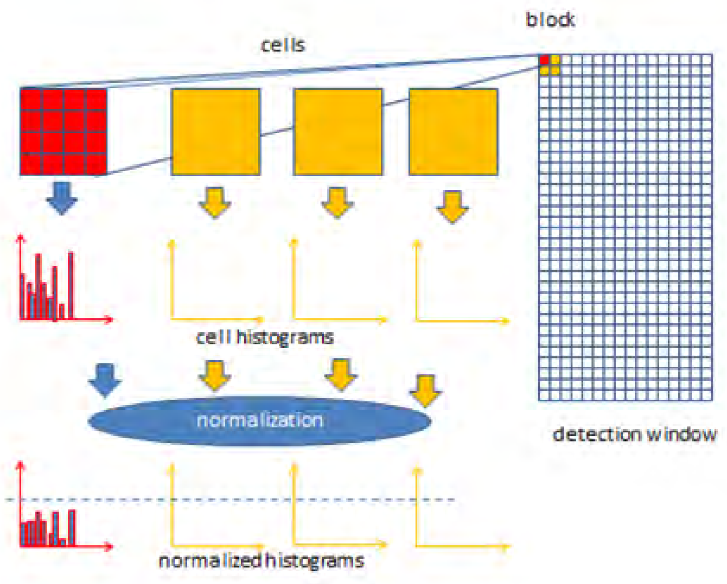


Figure 7: Histogram of Oriented Gradients. This image shows how the information captured in a cell window is transformed into a histogram and then compared against other histograms in a block size. The histograms in a block are then normalized across the board to suppress any external local illumination. [8]

2.2.1.1 Classification

One of the most straightforward ways to classify data into two classes, termed positive and negative, is through the use of a Support Vector Machine (SVM). An SVM determines the optimal hyper plane that separates the positive from the negative data, allowing any new data that falls above or below the hyper plane to easily be classified into one of the two categories, an example of a hyper plane is shown in

Figure 8.

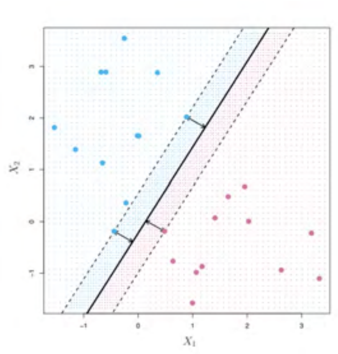


Figure 8: MMC Hyperplane [1].

SVM's are typically binary classifiers, however, there are several methods than can expand the capabilities from binary to multi-class classifications, the most popular methods being "one-vs-one" and "one-vs-all." One-vs-all involves training a single classification against the null combined with the other classifications. One-vs-one involves training a separate classifier for every pair of labels. The superior method comes down to the data class distribution. One-vs-all is more prone to error in imbalanced data sets as it can introduces additional imbalances by combining multiple classifications to create the negative. One-vs-one on the other hand does not introduce any additional imbalances, but requires $\frac{N(N-1)}{2}$ classifiers per N classes, meaning it can be computationally more expensive [9].

2.2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are the primary tools used in machine learning. ANNs, as the name suggests, work and are named after the way the human brain operates. ANNs operate off a series of inputs that are connected through a network of hidden layers and weights to connect a certain input to a desired output. This works in the same way that humans take in sights and sounds to identify an object. The architecture of the model determines the layout and design of these hidden layers and can be carefully crafted to suit a specific task, much like eyes are used for identifying visual objects, and ears for sounds. For vision applications ANNs are mostly comprised of Convolution Neural Networks (CNN).

2.2.2.1 Convolutional Neural Network

A CNN consists of an input layer (the image) and an output layer(label), with several hidden layers that typically consist of convolutions, pooling, fully connected, and normalization layers. CNNs are a small subset of ANNs that primarily deal with imagery and computer vision tasks. A CNN is designed to take in an image as an input, determine the categories and classifications found in the image, and finally output these classifications [?, 10]. The CNN's namesake and its ability to segment an image and determine its parts comes through the use of a convolution layer. A convolutional layer differs from the basic dense layer as the neurons in convolutinal layers are not connected to every single pixel in the input image, but rather only the neurons in their receptive fields, as shown in Figure 9. This feature allows CNNs to concentrate on small low-level features and assembling them into high-level features in subsequent hidden layers.

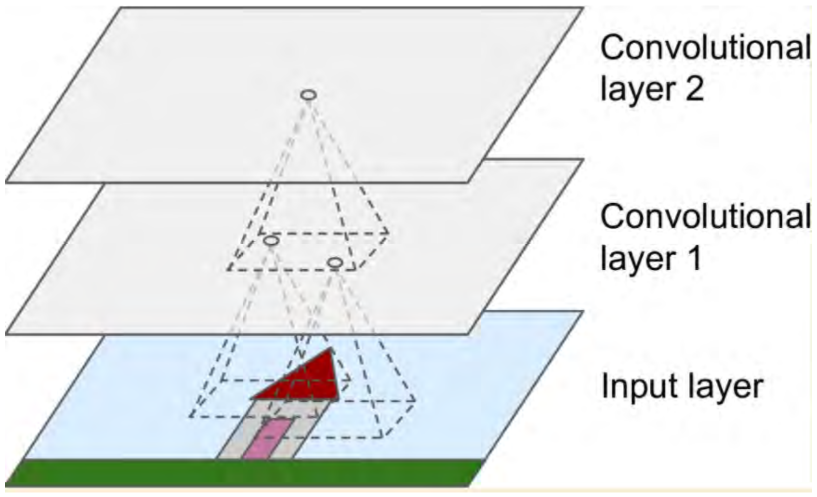


Figure 9: CNN layers with rectangular local receptive fields [?]

The mathematical definition for a convolution is defined as

$$s(t) = \int x(a)w(t-a)da \tag{2}$$

where $x(a)$ is the input, $w(a)$ is the weighting function or kernel, and $s(t)$ is the feature map or output. The size of the receptive field used in the convolution operation is one of the key parameters for defining a convolution. The second defining feature is the number of filters created to encode various aspects of the input data, this feature is also referred to as the depth [**?**].

2.2.3 Strides

Stride is a parameter of the convolution operation that determines how far the receptive field will shift between strides. Figure 10 shows how the kernel operates with a stride value of 1 and 2 and also demonstrates the effect on the output dimension. This reduction in dimension is referred to as down sampling and is what helps the CNN perform semantic segmentation [**?**].

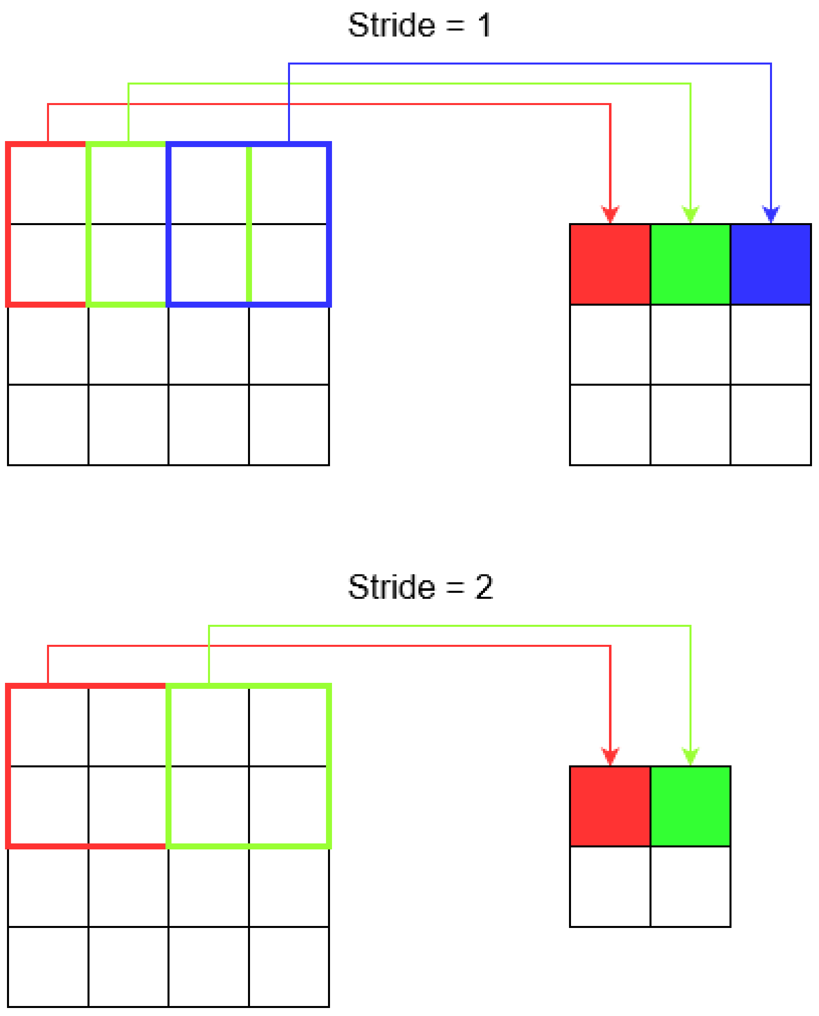


Figure 10: Stride representation [?]

2.2.4 Padding

The padding operation makes it possible for a layer to have the same dimensions as previous layers. There are many type of padding, but the most common is referred to as zero-padding. Zero-padding involves adding zeros around the inputs in order to apply the proper number of strides to take place. SAME padding is the parameter used that ensures the dimensions of the output will be the same as the input; whereas Valid padding focuses on sticking with the kernel size and stride value given [?, 10].

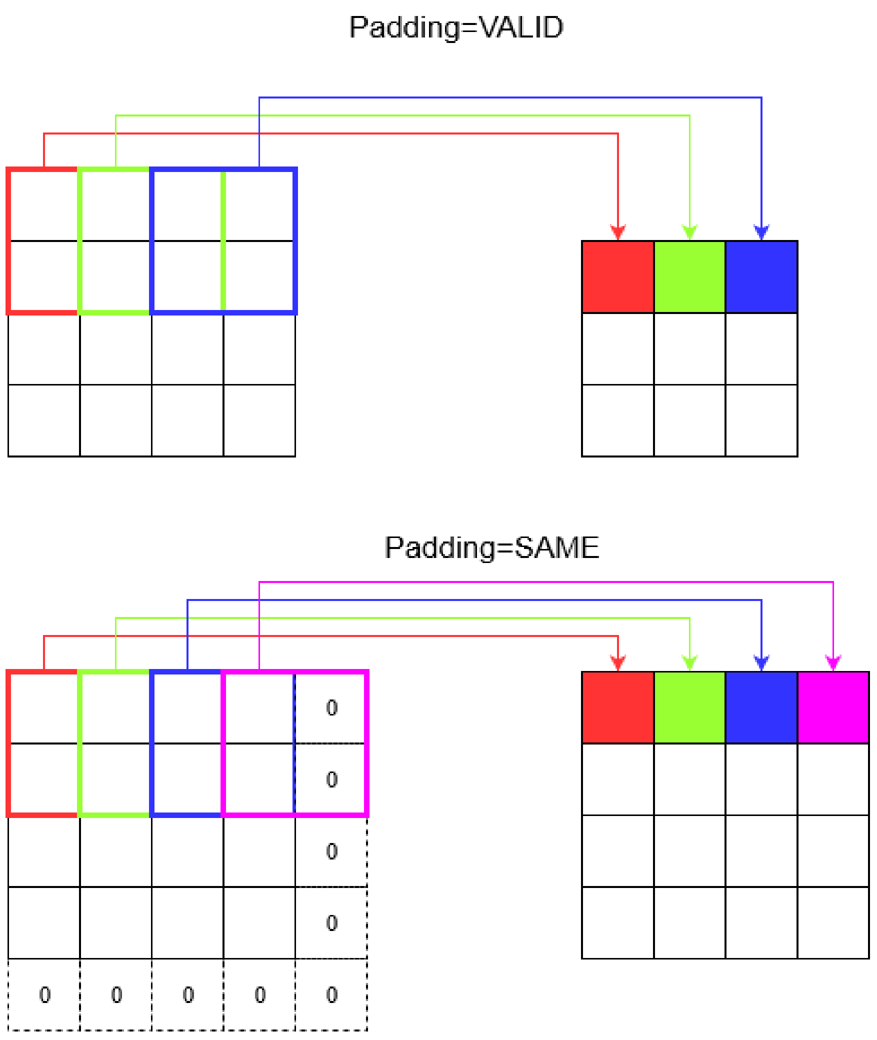


Figure 11: Padding representation [?]

2.2.4.1 Loss Function

The model's capability to learn is determined by its ability to minimize errors between the model output and the truth label. This error is calculated through what is known as a loss function [11]. Simply put, for a model that performs poorly, the loss function will output a higher number. If the model performs well, the output of the loss function will be a smaller number. In order to achieve the best possible performance from a model, it is important that the loss function be designed to optimize the network to the task at hand.

2.2.5 Binary Cross Entropy

When it comes to image segmentation tasks, Binary Cross Entropy is considered one of the primary baseline loss functions for cases where only two labels exist. [12, 13, 11, 14]. The loss function for Binary Cross Entropy is given as

$$Loss_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i))), \qquad (3)$$

where y is the truth label (0 for negative or 1 for positive), and p(y) is the predicted probability of the point being a 1 or positive for all N points. Binary Cross Entropy is defined as the difference between two probability distributions for a given random variable or set of events [13]

2.2.5.1 Data Augmentation

Data augmentation is the art of artificially increasing the size of a training set by generating new realizations of a training image. Data augmentation is most often accomplished by applying simple transforms to an image, such as: Horizontal and Vertical Flips, rotations, scale changes, crops, and translations. Each augmentation technique can train a model to be scale, position, and rotation invariant, allowing a model to detect an object in any setting. Additional methods, such as the addition of noise or adjusting the contrast, can help visual models learn to account for various lightning or environmental effects often found in outdoor imagery [15]. Exact Data Augmentation methods utilized will be discussed in Chapter 3.

2.2.5.2 Morphological Transforms

Unlike data augmentation, Morphological transforms are intended to change the information in the image. While this method is generally not used on images for training, it can be used to great effect on the output of the model. Dilation, in Figure 12, assigns each pixel with the maximum value of all the pixels in the neighborhood. This operation is effective for making objects more visible and filling in small holes in objects. Erosion, pictured in Figure 13, sets the pixel value to the minimum value of all the pixels in the neighborhood which is effective at removing small anomalies leaving behind only significant objects. In surface defect detection, dilation is primarily utilized to enhance the results and bring out smaller features [16]. For this project dilation will primarily be used to fill in holes left in the mask where the model failed to detect a continuity in objects.

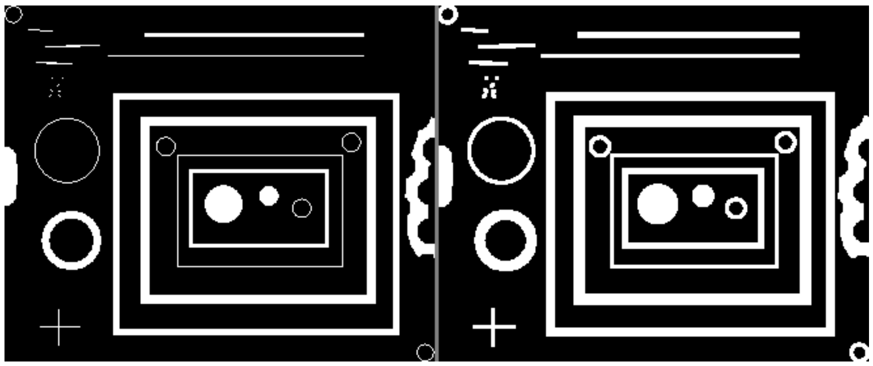


Figure 12: Dilation [17]

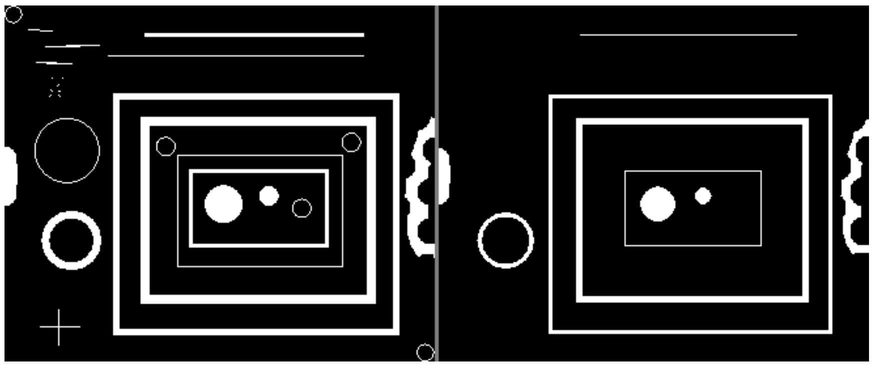


Figure 13: Erosion [17]

2.2.6 U-Net

One of the most popular models for semantic segmentation is a model called a U-net. The U-net was first introduced by Ronneberger as a biomedical image processing algorithm to provide labels for every cell in medical imagery [12]. U-Nets are aptly named for the shape the model creates as shown in Figure 14.

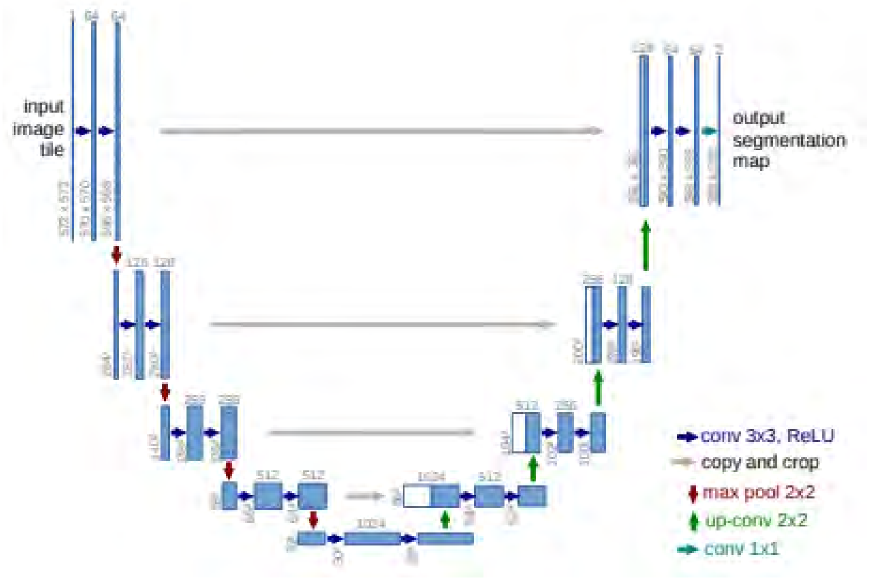


Figure 14: Ronneberger's U-net [12]

The U-net model is made up of two parts, the contracting and the expanding side.

Ronneberger's encoder or contraction (left) side of the U-Net consisted of four encoding blocks, each block made of two back-to-back 3x3 convolutional layers with a ReLu activation function, followed by a single 2x2 max pooling layer. The corresponding decoder or expanding (right) side of the U-Net utilizes transposed convolutional layers. The final output comes from a 1x1 convolutional layer in order to map the feature's vector to the corresponding class. The output of a U-net is an image in the form of a classification mask. The U-net model is also considered to be a Fully-Convolutional Network (FCN) meaning that it can take in images of any size, making it a popular choice for many segmentation applications. The genius behind the U-net architecture is that a U-net can be created with any convolutional model followed by corresponding transposed convolutional layers. This discovery has led many researchers to test the capabilities of placing pre-trained models in the contracting path of the U-Net and then developing the expanding right half of the U-net architecture.

2.2.7 Surface Defect Detection Datasets

As mentioned previously, real world data were unable to be collected for surface defect detection on aircraft skin due to extenuating circumstances. Should data have been collected, a large part of the work and arguably, the most important, would have been in *data set labeling.* Dataset labeling involves attaching a set label to an image, or even to every pixel of an image. These labels are used to show ANNs, during training, what the output of the model should be and are used to compute a loss value for the model. A model's performance is highly correlated to the quality of the dataset being used [18, 19]. For this reason it is important to use the best tools and techniques available when attempting to develop a new dataset. The following section will cover current state-of-the-art image segmentation labeling techniques.

2.2.7.1 Dataset Creation

With the advent of the internet and the collaboration of mega-corporations, Big Data has been readily available for years [20]. Big Data refers to the field of study that revolves around way to process and gather knowledge from datasets that are to large for traditional data processing software applications. The issue with Big Data is that it only solves half the problem, ANN's not only need data to train on, but require labeled data to calculate the loss in the model. Labeling data can be very time, money, and manpower intensive to create [18, 10]. There are two objectives for image segmentation algorithms: localization and classification. Localization focuses on where the defect lies in the image and is accomplished through a process called segmentation. The goal of image segmentation is to make it easier to analyze the contents of an image by finding edges, lines, gradients. The main hurdle with image segmentation comes when there is a lack of data available for training. Labeling data for an image segmentation task can be very time consuming as the labels are applied at the pixel level. In order to label images for segmentation there are two main approaches, supervised and unsupervised labeling.

2.2.7.2 Supervised labeling

Supervised labeling, also referred to as human-aided segmentation is the process of using individuals to label every piece of data for a given task. In the earlier days of working with Big Data, in order to reduce the work load of data labeling, several tools were created to help expedite and increase the performance of supervised labeling, Pauplin, Caleb-Solly, and Smith [21], a group from the Bristol Institute of Technology created an Interactive Parameter Adaptation Tool (IPAT) for image segmentation. Pauplin and his team recognized that the creation of customized image segmentation algorithms was both time consuming and tended to be specialized to a specific task. IPAT was designed as a tool to extract a user's tacit knowledge in order to determine the requirements for the desired segmentation, without any need for specialized knowledge of the underlying machine vision system. IPAT works by presenting the user with several images of a defect that have been pre-segmented, the user then selects the image that they determine is the most accurate. Through several iterations IPAT begins to learn and tweak its parameters to develop an accurate segmentation model built on a trained operators knowledge. This allows models to be tweaked by professionals in the field, without any need to understand the inner-workings of the model. The main drawback to this method is that IPAT requires a working baseline model to iterate through, and results can vary based on the initial model. While IPAT is an excellent tool, given the amount of data needed to train a good model [22, 23, 10], IPAT can be very time intensive for a given project. Within recent years several companies and business have emerged offering image labeling and annotation services. Hasty.ai, Labelbox, Superannotate, and Diffgram are just a few notable business that offer free and paid services.

2.2.7.3 Unsupervised labeling

The second approach is accomplished by allowing the computer to determine the boundaries of the segmentation through a method called computer aided image segmentation. One of the earlier attempts at this was conducted by Elbehiery, Hefnawy, and Elewa, using a statistical approach [16]. Elbehiery worked on creating such a program to handle quality control operations in ceramic tile manufacturing. The quality control operations for tile manufacturing is broken down into three segments, color analysis, dimension verification and surface defect detection. The first two segments are already handled by other computer vision techniques, but surface defect detection was being handled by a human inspector. Elbehiery's goal, therefore, was quality control enhancement through visual image processing [16]. Elbehiery and his team break the edge-finding algorithm into two categories. The first is finding pixels in the image where an edge is likely to occur through discontinuities in gradients. The second is then linking these edge points together to create a description of the edge with lines and curves. This method is best suited for projects without any labeled data as it utilizes statistical methods to determine the presence of defects. While Elbehiery used a statistical approach to map out defect edges, more recently Abhihek Thakur and Rajeec Ranjan used deep learning to label imagery. The process utilized by Thakur and Ranjan is divided into two parts. The first is a machine learning algorithim to detect super pixels. These super pixels are categorized and grouped on the basis of color. The second model was then trained on these color categories, classifying each image into semantic labels [24]. Due to data requirements and the fact that labeled data is not always readily available, in recent years there has been a large push for unsupervised models that can work with unlabeled data using unsupervised labeling [25, 26, 27] .

2.3 Pre-trained Model

2.3.1 VGG16

VGG16 first introduced in 2015 was designed as a method for testing the effects of convolutional network depth and its accuracy on large-scale image recognition [28]. The VGG16 model pushed the depth of traditional models to 16-19 weighted layers winning first and second place in the 2014 ImageNet Challenge. The original VGG16 model is shown below in Figure 15. VGG16 was a breakthrough in image classification tasks in that it showed that model depth can add significant improvements to model performance.

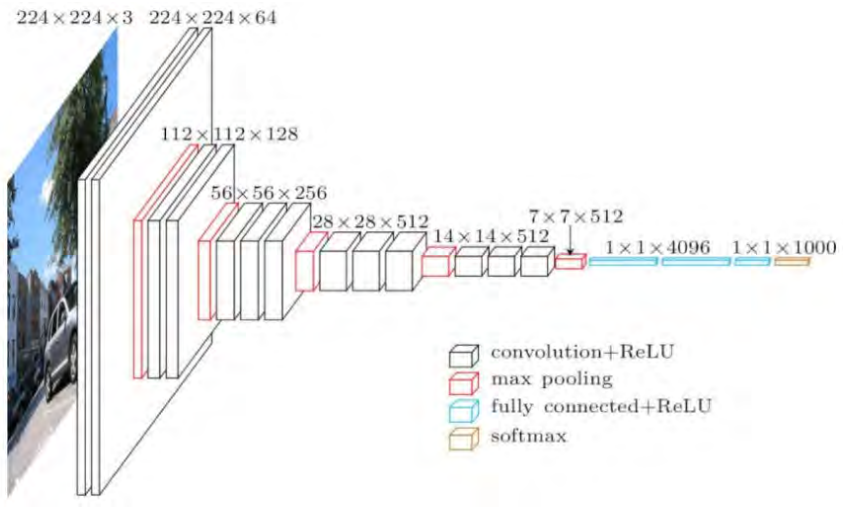


Figure 15: VGG-16 Model [29]

2.3.2 ResNet50

One of the issues with creating larger networks is a problem referred to as exploding/vanishing gradients [30]. However in 2015 a group from Microsoft found a way to incorporate residual learning layers to reference the inputs, instead of unreferenced functions [31]. The original model presented was a 34-layer ResNet model shown in Figure 16; however, in that paper the team also explores models of size 34, 50, 101, 152, and an incredible 1202-layer network. For the project in this paper, surface defect detection, a 50-layer version of ResNet was utilized. ResNet50 and the reason for this model will be discussed further in Chapter 3.

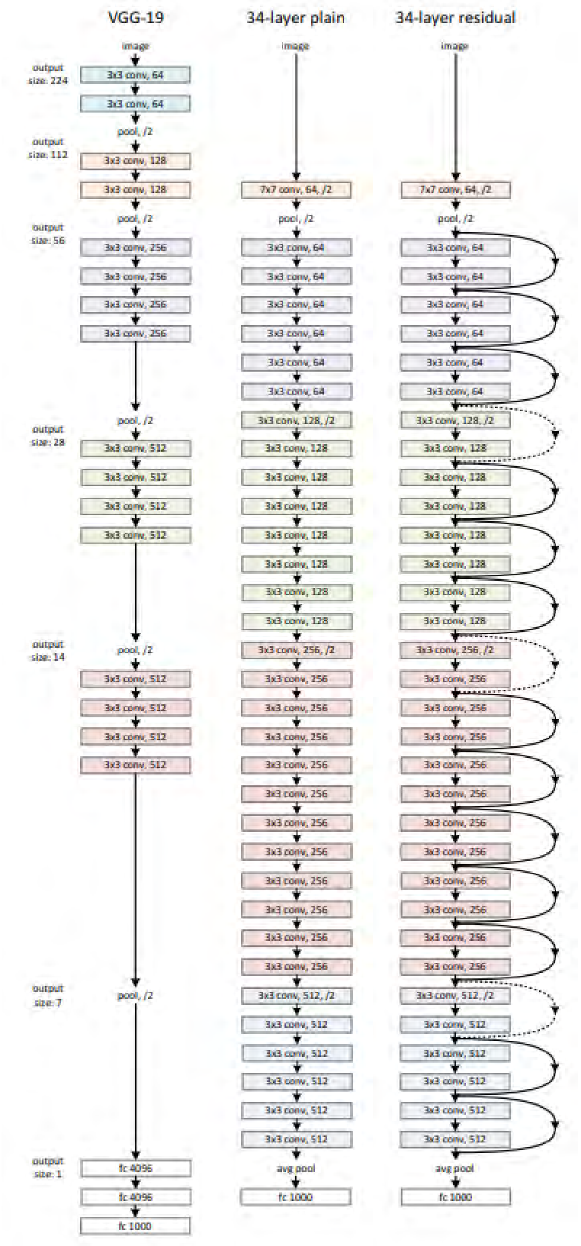


Figure 16: VGG-16 vs. 34-Layer Model vs 34-Layer Resnet [31]

2.3.3 Transfer Learning

Transfer learning is the idea of utilizing knowledge acquired from one task and applying it to a similar application. For example, when attempting to develop a model for object detection/classification, a model can require thousands of images per object just to achieve acceptable results for a single object. Each of these images will need to be labeled in order for the model to learn the association between the image and classification. The labeling of data can take a significant amount of time for even simple applications; this is where transfer learning comes into play. Transfer learning allows the utilization of the knowledge and weights of a model trained on a pre-labeled data set. Utilizing the weights from a pre-trained model can often result in faster training time, more accurate results, and the need for less training data. So when developing a model for object detection/classification, instead of collecting and labeling thousands of images, one can utilize transfer learning with one of the more common models such as, VGG16 or ResNet50, trained on image classification datasets such as ImageNet. While transfer learning can greatly help improve the overall results of a model, the further the application strays from its intended purpose the worse the results.

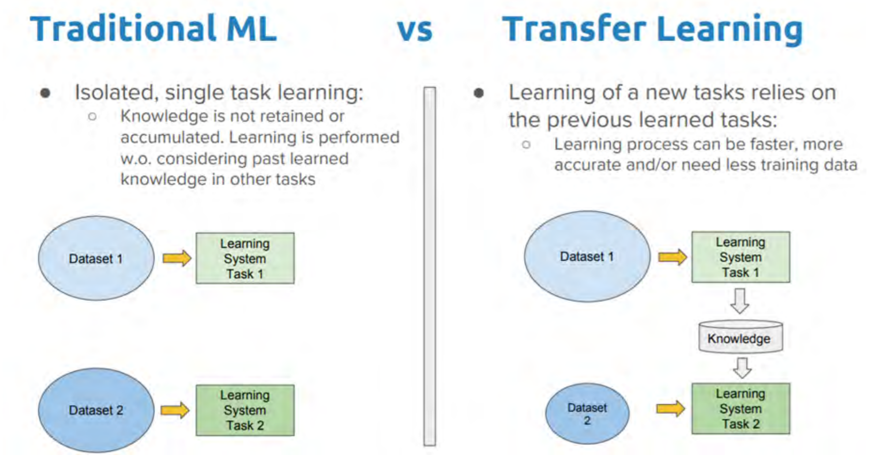

Traditional ML
vs
Transfer Learning
Isolated, single task learning:
Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
Learning of a new tasks relies on the previous learned tasks:
Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System Task 1
Dataset 2
Learning System Task 2
Dataset 1
Learning System Task 1
Knowledge
Dataset 2
Learning System Task 2

Figure 17: Traditional Learning Vs Transfer Learning

2.4 Evaluation Metrics

2.4.1 Accuracy

Accuracy is one of the easiest metrics to understand and one people are most familiar with as it simply measures the number of pixels labeled correctly over the total number of pixels,

$$accuracy = \frac{\#\, of\; correctly\; classified\, pixels}{total\; \#\; of\; pixels}$$

[32].

While accuracy is easy to understand, it is a poor metric for semantic segmentation tasks [10, 33] . For many segmentation projects the number of null pixels, the non-desired pixels labels, vastly outnumber desired labels this can also be referred to as a class imbalance [34, 35]. Given one example of a large class imbalance such as the case where an image consists of 5% positive pixel values and 95% negative pixels. The model trained on a data set suffering from large class imbalances might simple predict the entire image as negative and still achieve 95% accuracy even though the model performs rather poorly in the given task. However, most dataset leaderboards and papers still report accuracy as it can still provide valuable information, but it is not the sole metric for evaluation.

2.4.2 Intersection over Union

Intersect over union (IoU) is one of the primary metrics used for segmentation tasks as it is proven to be a better metric for classification as it focuses more on correctly identifying the positive pixels rather than the negative [36, 10]. The IoU metric can be calculated using equation 4 where A represents the truth label and B represents the predicted label,

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}[37]. \tag{4}$$

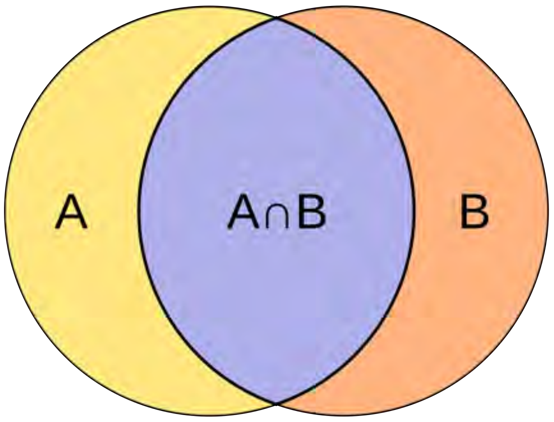


Figure 18: Visual Representation of Jaccard Index

IoU is the preferred metric for object classification tasks and is used in a majority of the image semantic segmentation leaderboards [38, 39, 3, 40, 41]. IoU penalizes more for incorrect classifications and rewards only correct positive classifications, removing true negatives from the equation

$$J(A,B) = \frac{TruePositive}{TruePositive + FalsePositive + FalseNegative}[10]. \tag{5}$$

2.4.3 Dice Coefficient

The other top evaluation metric for segmentation tasks is the Dice Coefficient (DC)[37, 42]. The equation for calculating the DC is shown in equation 6 and visual representation is shown in figure 19,

$$DC(A,B) = \frac{2 * |A \cap B|}{|A| + |B|}. \tag{6}$$

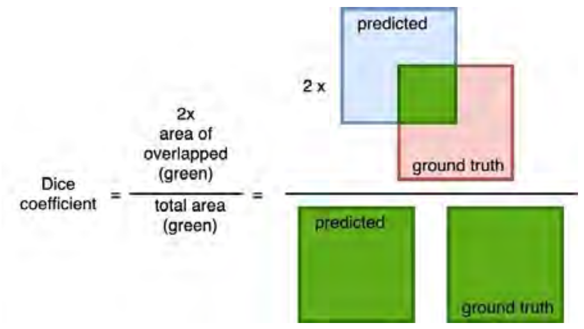


Figure 19: Visual Representation of Dice Coefficient. The blue box represents the model output or in this project the areas of the image the model thinks contains buildings. The red box is the ground truth label. The green area in the numerator represents the area of overlap between what was predicted and the truth label. [12]

The DC is very similar to IoU in that they are both positively correlated. Meaning that if IoU says model A is better than model B then the DC will always agree [43]. Now there are some distinctions between the two and reasons to use one over the other [44, 13], however that is beyond the scope of this project. Both evaluation metrics will be included in the results as certain dataset leader boards have chosen to use one over the other.

2.4.4 Precision vs Recall

Another important evaluation metric is precision and recall [45]. While precision and recall are usually not used to evaluate a models performance they are important to consider when needing to tune a model towards a certain objective. Precision, also referred to as positive predictive value, is a measure of the how the model is at rooting out false positives. Precision can be calculated by dividing the number of True positives by the total number of positives predicted whether they are true or false positives. Or visually represented as,

$$Precision = \frac{TP}{TP + FP}. \tag{7}$$

Recall on the other hand is known as the sensitivity measure. Recall measures the the models ability to not have false negatives as evident by the following equation,

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

. Recall is mathematically defined as dividing the True positives by the True positives added to the False negatives [45].

Both metrics are important and are designed for different tasks. For instance in the medical field a higher Recall value is often preferred [10, 45] . A higher recall aims to ensure the minimal number of false negatives get through, by relaxing the requirements for a positive outcome. Precision on the other hand is more concerned with minimizing the false positives [10, 45, 15]. Precision models aims to reduce the number of false positives by restricting or tightening down on the parameters used to make a positive outcome. The difference between precision and recall is a subtly but often important distinction to make in various machine learning tasks. Often times the best approach is to use a combination of precision and recall to determine model performance, this combination is often referred to as an F1 score or as discussed previously the dice coefficient [10, 46]. The DC can be calculated using precision and recall values and plugging into the following equation,

$$DC = 2 * \frac{Recall * Precision}{Recall + Precision}. \tag{9}$$

Utilizing the equations for precision 7 and recall 8 from above and plugging into equation 9 provides the following results,

$$2 * \frac{Recall * Precision}{Recall + Precision} = 2 * \frac{tp}{2tp + fp + fn} \tag{10}$$

.

Given Union is equal to TP + FN + FP and Intersection is equal to TP, plugging this into the equation 10 and solving gives,

$$2 * \frac{tp}{2tp + fp + fn} = 2 * \frac{Intersection}{Intersection + Union} = Dice\,Coefficient. \tag{11}$$

Equation 11 shows how the unique application and recall and precision is used to produce the F1 score or DC. While the DC is the preferred method for evaluating a models effectiveness based on precision and recall, it is worth keeping these values as a model can be tuned towards specific goals.

2.5 Optimal Feature detection and matching

2.5.1 Feature selection

In order for visual navigation through feature selection to be possible a feature selection tool must be able to identify unique objects and then find the same objects in subsequent imagery. One such method is through the use of the Scale-Invariant Feature Transform (SIFT), SIFT is a feature detection algorithm presented by David Lowe in 1999 [47]. SIFT is unique in that it is invariant to image rotation, affine transformations, intensity, and viewpoint changes in matching features [48] a capability that proves to be invaluable when comparing aerial imagery to north aligned satellite imagery. SIFT works in 4 basic steps. The first step is to estimate scale space extrema through the use of Difference of Gaussian (DoG). The second step is key point localization where the key points are sorted by their low contrast points.

The third step is to assign an orientation based on local image gradients. Finally the last step is to generate the local image descriptor for each key point based on image gradient magnitude and orientation [47].

While there are other types of feature selection such as FAST[49], SURF[50], and ORB[51]; SIFT was primarily used as it is the most commonly used feature selection tool for Visual Navigation [52, 53, 54]

2.5.2 Feature Matching

Once the keypoints, with their features and descriptors, have been collected from and image, the next step is to determine any matches between the two or more images. For the visual navigation project, two feature matching algoithims were tested, the Brute-Force matcher and the Fast Library for Approximate Nearest Neighbors (FLANN) matcher.

2.5.3 Brute-Force Matcher

As the name implies the Brute Force method is a brute-force approach in that it takes the first keypoint found in the first image and compares it to every point in the second image, comparing all possible combinations, and returns the point with the lowest euclidean distance. This method can be computationally expensive but will return the best result [55]. When additional points need to be returned, as such with k-nearest neighbor applications, OpenCv's Brute-Force matcher has a built in function knnMatch. This function takes in the keypoints from the first and second image but also takes in a k value. The k value determines the number of closest matches the Brute-Force matcher will return. The ability to return and rank multiple feature matches will be very important when looking into ways to eliminate incorrect matches.

2.5.4 FLANN Matcher

If computation time is a consideration, FLANN is a collection of algorithms optimized for fast nearest neighbor search. FLANN takes in two parameters, Index and Search. Index establishes the algorithm to be used, with many algorithms designed for SIFT, SURF, and ORB. OpenCv's documentation on FLANN covers the various inputs for each of these algorithms. The second input, search, specifies the number of times the tress in the index should be repeatedly examined. Higher values of the search index can result in higher precision but will start to take more computation time [56].

2.5.5 Ratio Test

While feature matching algorithms can return thousands of matches for complex images, not all of the matches will be correct. In 2004 David Lowe introduced a new process for determining distinctive image features through a method called the Ratio test. Suppose that L1 is the set of keypoints in the first image and L2 is the set of key points for image 2. In order to use the Ratio test, a knn feature matching algorithm must be used in order to store multiple matches to keypoints in L1. In order to eliminate the incorrect matches Lowe works on the assumption that a feature in image 1 can only have a single match in image 2. The Ratio test takes the first keypoint in L1 and atleast two keypoints from L2 and compares the Euclidean distance between the features. Typically the feature with the lowest Euclidean distance is the better match. However, the Ratio test says that if the Euclidean distance between the good match and the bad match is not sufficiently different, then both matches are thrown out. This method ensures that the good matches left behind are easily distinguished from noise, leaving behind only distinctive features. The equation for the Ratio test is given as,

$$d(f_1, f_2)/d(f_1, f_s) < \rho \tag{12}$$

,

which can be defined as the distance between a feature, f_1, in image 1 and the closest match in image 2, f_2. Divided by the distance between the f_1 and the second closest match in image two, f_s. Finally keeping only the matches with a distance ratio that is less that the given ratio threshold ρ [57].

III. Methodology

This chapter will cover the methodology used to build a semantic segmentation model for surface defect detection as well as building and road extraction for visual navigation. Section 3.1 will focus on the techniques utilized to develop a segmentation model for surface defect detection, to include dataset selection, model architecture, evaluation metrics, and loss functions, Section 3.3 will focus on the techniques used to force keypoints onto desired features through semantic segmentation. Section 3.2 will cover the Datasets used, data augmentation, model architecture, and post processing of model output.

3.1 Surface Defect Detection

The following sections will focus on two techniques used for surface defect detection and classification. The first method is based on a traditional statistical approach using Histogram of Orientated Gradients (HOG) and a Support Vector Machine (SVM). The second method focuses on a more modern approach using an CNN trained for semantic segmentation. Chapter IV will then compare the results of the ANN using the traditional approach as the baseline.

3.1.1 Implementation

The following methods for Surface Defect Detection in steel plates was developed using Python and various Machine learning and statistical libraries. The details for each method will be discussed below, but both models used the same general libraries to include: scikit-learn for various data analysis tools and support vector machines(SVMS), scikit-image for image processing to include Histogram of Orientated Gradients(HOG), and tensorflow 2.0 for the development and training of the machine learning model.

3.1.2 Dataset

As previously mentioned, data for aircraft skin surface defects was unavailable due to COVID-19. As such, all surface defect detection techniques were tested on a similar surface defect detection project, steel surface defect detection. The dataset utilized for steel surface defect detection was first presented in Nov 2019 by a company called Severstal, a steel manufacturing company. Severstal recently created the country's largest industrial data lake with petabytes of data. Severstal is now looking to machine learning for ways to improve upon their automation and increase efficiency. The dataset provided consisted of 12,568 (1600 x 256) training observations and 5506 test images. Each image has the possibility of having either zero defects, a single defect, or multiple defects per image broken down into one of four classes. The details for each class and the type of defect involved was not included in the description. While the competition ended in 2019, the truth labels for the test data have not yet been released. Therefore a random selection consisting of 20% of the original 12,568 training images were set aside as the test dataset.

3.1.3 Method 1: HOG/SVM

3.1.3.1 HOG/SVM Data

Each image, sized 1600 x 256, contains 409600 features or predictor measurements in the form of pixel intensity values. Utilizing OpenCV, images were converted to gray scale and then to an array of values between 0 and 255 to describe each image. This resulted in 10,054 training examples consisting of 409600 features per image ranging in values between 0 and 255. By utilizing HOG as a feature descriptor the number of features were reduced and converted into more meaningful statistical measures.

3.1.3.2 Histogram or Orientated Gradients

Table 4.1.1 shows the optimal HOG parameter sets utilized for each defect. Tuning the HOG descriptor for each defect proved to be non-trivial as the cell size is highly dependant on the specific application. If a cell size is too large, small scale detail will be hidden. The opposite also holds true, if a cell block is too small then the ability to suppress local illumination will be lost [58]. As a baseline starting point the HOG for each defect was set at 18 bins with a cell size of 16x16 pixels and each cell block being 2 cells tall and 2 cells wide.

The training data set consists of 4721 clean images and 717 class 1, 197 class 2, 4120 class 3, and 427 class 4 defects, the size disparity can be seen in Figure. 20.

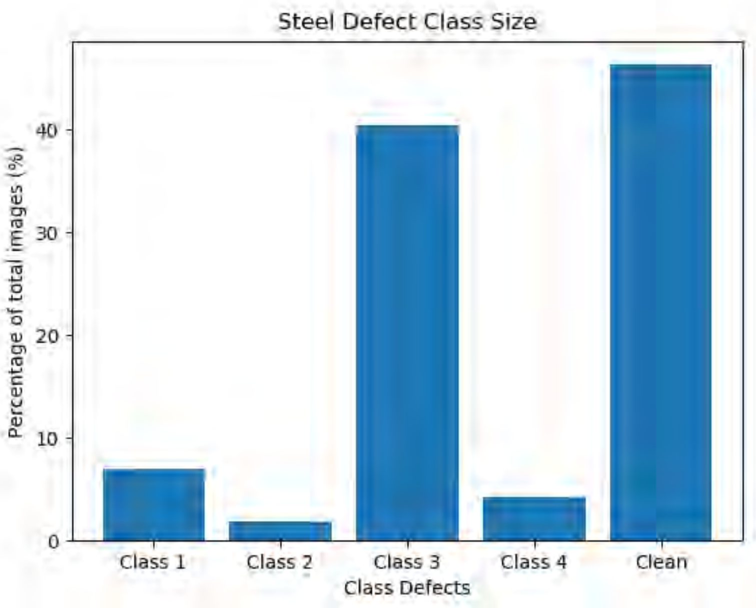


Figure 20: The Steel Surface Defect dataset broken out by defect class to show the disparity in class sizes.

3.1.3.3 Support Vector Machine

Due to hardware memory constraints the Class 3 defects had to be reduced from 4120 to 3,000 images. Utilizing a linear-SVM or SVC for the classifier, a one-vs-one approach was utilized as the data was already susceptible to a large imbalance in class size. Each model created included one of the 4 classes vs the clean dataset. The four models provide a probability on whether or not their class defect is found in the image. The model label with the highest probability was the label assigned to the defect. A probability threshold of 50% was also put in place, meaning that if neither of the 4 models produced a probability of defect greater than 50% then a clean label was was applied to the image. Through training these models, 5 different seeds were used to randomly split the data into different training and validation sets. 80% of the data was used for training and the remaining 20% was used for validation in every split. Within each classifier pair the positive and negative images were balanced 50% containing the defect and the remaining 50% being a clean image. This balancing was done to avoid overpowering the smaller defect classes with a disproportional number of clean images.

3.1.3.4 Confusion Matrix

The results for this approach are provided in the form of a confusion matrix. A confusion matrix is a table that describes the performance of a classification model on a set of test data where true values are known. Figure 21 shows a confusion matrix for a binary classification task. The confusion matrix typically has the actual values down the side with the model predicted values across the top. The confusion matrix table layout simplifies the ability to easily pull out a models true positive, false negative, false positive, and true negative rates. As shown in the figure, the true positives (TP) can be found by finding the "actual positive" row and the "predicted positive" column and the intersection will provide the number of positive samples that were predicted to be positive.

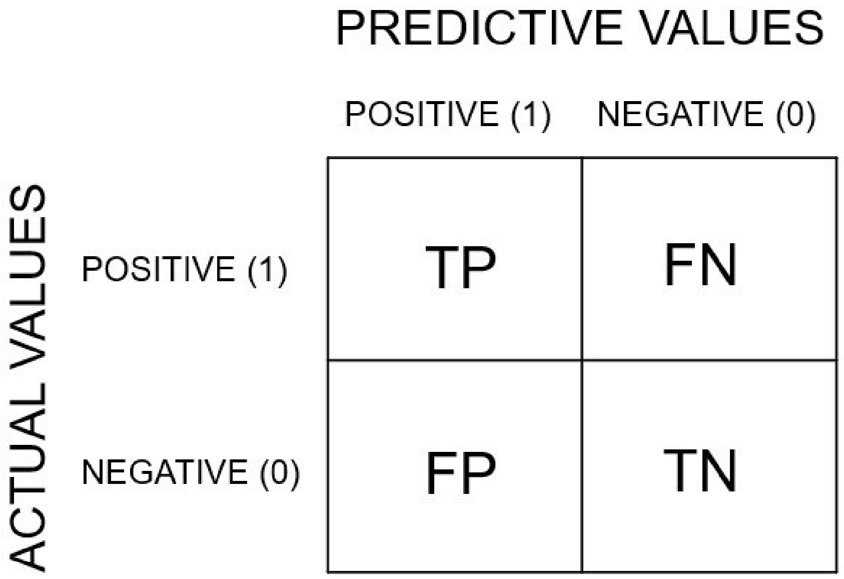


Figure 21: Confusion Matrix for Binary Classification [59]

Using the value provide it is easy to calculate additional statistical measures such as accuracy, $(TN + TP)/totalsamples$, or the misclassification rate, $(FP + FN)/totalsamples$. This statistical approach will help serve as the baseline for comparison against modern techniques.

3.1.4 Method 2: Artificial Neural Network

The second method developed was the construction of an ANN, or more precisely, a fully convolutional U-Net. The U-net was specifically designed for fast and precise segmentation and is well suited for surface defect detection applications.

An important part of building an ANN is to understand the data and the class structure provided. As already pointed out in the previous section the data provided is primarily made up of clean images and class 3 defects.

3.1.4.1 Model Architecture

The model developed for the project consisted of a pre-trained ResNet18 model combined with a tailored transposed convolutional network to create a U-net. Figure 22 below shows the basic structure utilized in a U-net and how it process the original image on the left into a mask prediction on the right. While there are multiple classes in each image the classes themselves are not mutually exclusive, the presence of one defect does not effect the probability of another defect being present, meaning that the model can detect the presence of multiple defects at the same time. Given that the classes are not mutually exclusive the loss function used was binary cross entropy. The loss function will force each binary classifier to be trained independently and then combined at the end for a total loss value [60]. Each residual block of the model contains two consecutive convolutional layers with a ReLu activation followed by a max pooling layer. Due to the size of ResNet, the visual display of the model will be included in the appendix as Steel Defect U-net Model.

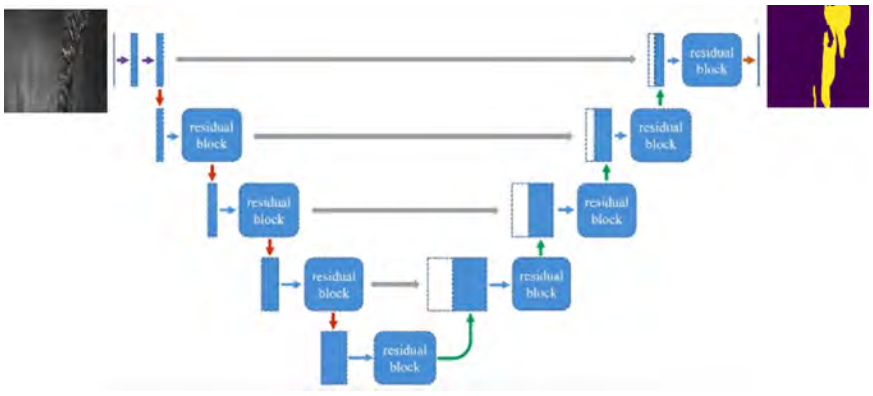


Figure 22: Architecture for the U-net model where each residual block is made up of convolutional layers, followed by a max pooling operation. This series of functions helps transforms the input image on the left into a segmented mask on the right. In this case the mask consists of 0's and 1's where a 1 represents the presence of a defect from that given pixel, with multiple models being trained for each classification task. [61]

3.2 Forced Feature Segmentation Model

This section will cover the methods used for forced feature selection via a segmentation model. The following sections will cover the datasets used for training along with the corresponding pre-processing and augmentation techniques used on the data. This section will also discuss the various model architectures along with evaluation metrics and loss functions used for semantic segmentation of buildings and roads. The final section will cover the feature selection and matching algorithms used in order to compare aerial imagery to satellite imagery.

3.2.1 Dataset

3.2.1.1 Inria Aerial Image Labeling Dataset

The INRIA Aerial Image Labeling dataset was created in order to address a core topic in remote sensing, automatic pixelwise labeling of aerial imagery. The INRIA dataset contains 180 (5000x5000 pixel) color images covering a surface of 1500m x 1500m at a 30 cm resolution, and the training dataset is also paired with with an associated truth mask to delineate between building and not building [2]. An example of the training image and label can be seen below in figure 23. These images are further delineated into groups of 36 tiles, or images, each group covering a specific area. The training dataset contains images from Austin, Chicago, Kitsap County, Western Tyrol, and Vienna. The test dataset contains images from Bellingham, Bloomington, Innsbruckk, San Fancisco, and Eastern Tyrol. These locations were selected for their dense and sparse building infrastructure. Since the test set is only available through online submissions to the INRIA website, the validation data will be used for testing.

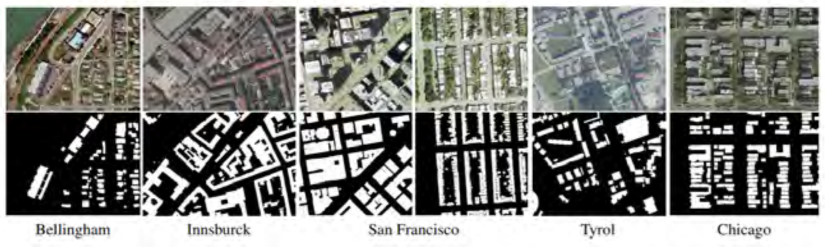


Figure 23: INRIA dataset image and corresponding truth mask [2]

3.2.1.2 SpaceNet 7: Multi-Temporal Urban Development

When developing a model for visual navigation it is important that the model is able to navigate year-round, through seasonal changes. The SpaceNet 7 dataset was released as a competition in 2020 as part of the 2020 NeurlPS conference. The idea behind the competition was to utilize time series data in order to track building construction over time and to assess urbanization. This challenge has broad implications for disaster preparedness, infrastructure development, and epidemic prevention [3]. The dataset consists of satellite imagery mosaics consisting of 24 images (1 image per month over 2 years) covering almost 100 unique areas. Each image covers an area roughly 4000m x 4000m and have an average of 4,700 buildings per image with a pixel resolution of 4m. A sample of the training image and associated binary building mask can be seen below in Figure 24.

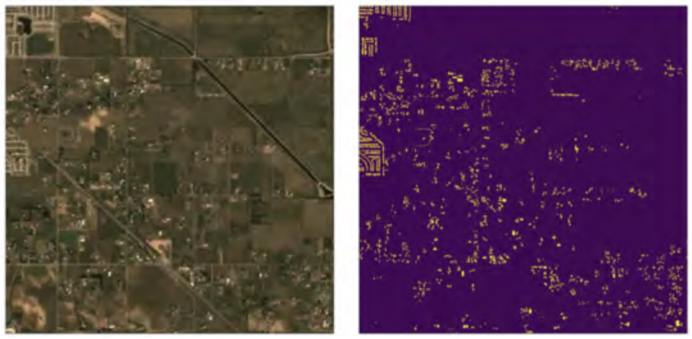


Figure 24: SpaceNet 7 training image and binary mask [3]

The SpaceNet 7 database ended up being excluded from the training set due to its pixel resolution. While it is important to capture seasonal information, the 4m resolution ended up reducing the models performance when combined with the 30 cm resolution of the INRIA dataset. At this point in the project, it was more important to have a good segmentation of the buildings rather than capturing the seasonal information. It is likely future iterations will contain this information once a higher resolution data set is created or made avaialable.

3.2.1.3 Massachusetts Road Dataset

The Massachusetts road dataset is a segmentation dataset designed for road semantic segmentation. The dataset is pre-sectioned into 1109 training images, 14 validation images, and 49 test images. Each image covers 1500m x 1500m with a 1m pixel resolution. Figure 25 shows two images, (a) & (b), along with their corresponding truth labels, (c) & (d) [62].

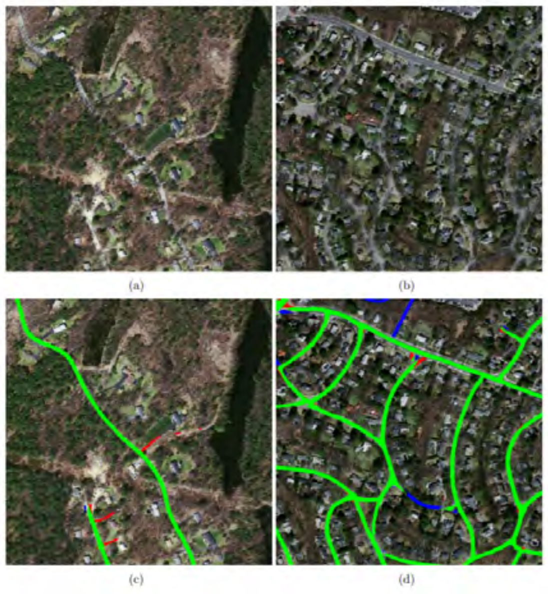


Figure 25: Massachusetts Road Dataset: (a) Rural environment input image. (b) Urban environment input image. (c) Image from (a) overlayed with truth mask depicting roads. (d) Image from (b) overlayed with truth mask depicting roads. [62]

3.2.2 Data Augmentation

While a U-net is designed to take in images of any size as long as image size is evenly divisible by 2N̂, where N is the number of layers in the UNet, in height and width. The main factor on input size though was hardware limitations, which required every image to be cropped from its full resolution into manageable chunks. Later sections will cover the model architecture of the U-net but the final model consisted of 5 levels, meaning that images had to be divisible by 32. Therefore to account for model size and hardware limitations images were cropped into 224x224 image crops. If the original image could not be evenly cropped into 224x224 crops, a border padding was added with a constant 0 value. This padding was considered negligible as only 10% of the training crops produced would contain any padding, those along the edge of the image based on INRIA's 5000x5000 images. Of those 10% of padded crops, only 13% of the crop would contain padded pixels, based on 224x224 image with a 31x224 padded pixel edge. This process was able to transform a single Inria image (5000 x 5000) in 500 (224 x 224) crops.

One of the benefits mentioned about a U-net is that this type of model tends to perform better the more data it is given. To increase the size of the dataset further a few basic data augmentation techniques were used. For this type of data augmentation it was important to maintain the overall structure and geometry of the roads and buildings, therefore only non-transformative augmentation techniques were used. Given a 4 sided image there are 8 unique ways to display an image, this is referred to as a Dihedral group D4 or the symmetry group of a square [4]. Figure 51 shows a visual representation of the D4 transformations which consist of the original image, rotation of the image by 90, 180, and 270 degrees, along with a horizontal, vertical, left and right diagonal reflections creating the 8 unique displays.

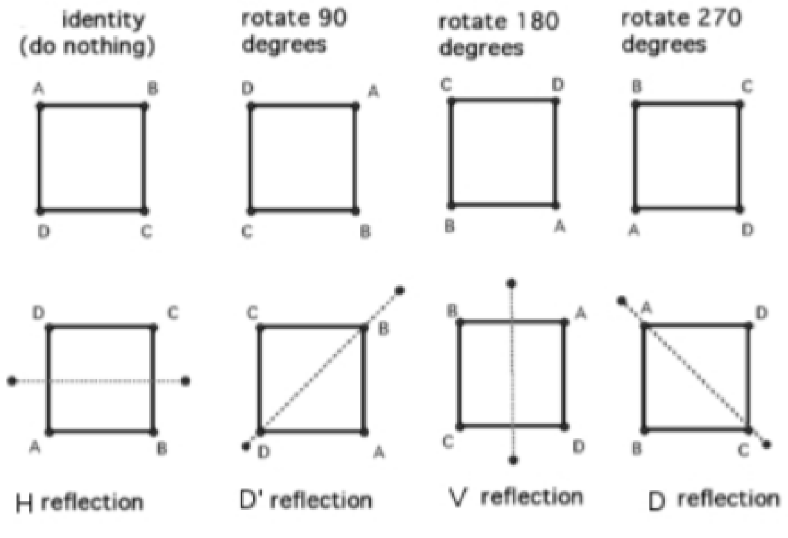


Figure 26: Dihedral group D4 visual representation [4]

While additional techniques such as the addition of Gaussian noise or contrast changes can account for various weather conditions and lighting changes, these augmentation techniques were not included. Future iterations will most likely incorporate contrast changes in order to make the model more robust to lighting changes.

3.2.3 Model Architecture

3.2.3.1 U-Net

The structure for the U-net architecture utilized for this project is roughly based on the first U-net introduced by Ronneberger. The difference being the the addition of a 6th layer allowing the images to be contracted down to a 7x7 layer before being expanded to the full size image. Each residual block in the contracting path of the network consists of a single 3x3 convolution block with "SAME" padding and batch normalization followed by a rectified linear unit and finally a 2x2 max pooling layer. Dropout was also utilized on layers 1-5 and their associated expanding counterpart. As the contracting side of the model gets deeper, the number of filters on the conventional layers also increases to match the complexity of the image. The number of filters was found to have a direct correlation to the performance of the model, the higher number of filters resulted in better performance. The first layer in the contraction side of the model has 120 filters with each subsequent layer doubling the number of filters, with the exception of layers 4 and 5 both containing the same number of filters (960). Due to hardware limitations the number of filters for the first layer was limited to 120, leading to the final layer having 1,920 filters. The expanding half of the U-net was designed to mirror the number of filters on the contraction side.

3.2.3.2 Evaluation Metrics

The evaluation metrics used consists of accuracy, Intersect over Union (IoU), and Dice coefficient (DC) with the latter two being the standard for image segmentation tasks. While IoU and DC are highly correlated, meaning they will always agree on the best model, both were used in order to compare model performance with other model leaderboards and papers. The model with the lowest validation loss and the highest IoU or DC score will be the final model utilized as it will be more likely to outperform the other models when presented with new data.

3.2.3.3 Loss function

The loss function used for this experiment was binary cross entropy as it is the industry standard for binary semantic segmentation tasks [12, 13, 11, 14]. It is worth noting that some datasets and papers utilize intersection over Union as a loss function. However, IoU cannot inherently be used as a loss function as it is not differentiable. IoU expects an output of 1's and 0's, while the models outputs an array of probabilities. The Jaccard loss function was created in order to utilize IoU as a loss function, but valuable results were not obtained utilizing the standard Jaccard loss in Keras or a custom version. Therefore, only binary cross-entropy was utilized in the given results in Chapter 4.

3.2.4 Morphological Operations

Upon review of the initial output of the model, the model struggled to detect the geometry and continuity of a building, often times leaving out large chunks of buildings in the center when dealing with large structures. However, the model performed exceedingly well when detecting the edges of buildings. The resulting mask eliminates a lot of desired key features on buildings and created a harsh boarder on the building that attracted many features artificially. To account for these obstacles, it was determined that the best method would be through the use of morphological operations. Dilating the masks by a set number of pixels caused any holes found in the continuity of buildings to be filled in and extended the border around buildings. This spacing allowed for more features to be drawn to additional keypoints on the buildings resulting in a greater number of features, and consequently, feature matches.

3.2.5 Feature Selection/Matching

The final step of the project involves running several feature detection algorithms on collected aerial imagery, to determine whether or not feature selection can be improved by first removing unnecessary sections of the image through image segmentation. The flight test data was collected by the Air Force Research Lab located in Dayton OH. The data contained 3000, 1024 x 1024, images collected over a 10 min flight test. The collected imagery was processed through the building and road segmentation models. The output mask was converted to a binary mask using 50% probability as the cutoff. The binary mask was finally padded using a 5x5 kernel to correct for any continuity errors in building surfaces. This final padded mask then combined with the original image to create the desired output as shown in Figure 27.

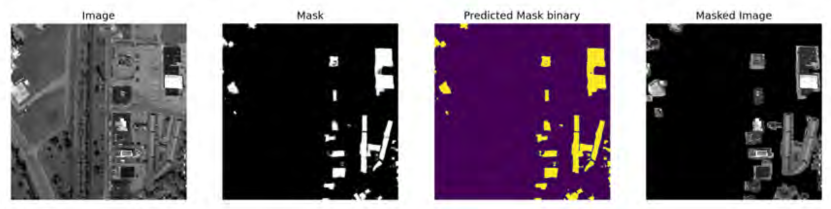


Figure 27: Imagery masks

Finally feature selection was accomplished through the use of SIFT as a feature detector. Chapter IV will discuss the results of using SIFT and compare the number of keypoints that were removed from non-desired parts of the imagery and how many keypoints were gained in desired areas.

3.2.5.1 Feature matching

Given the focus of this research is proof of concept the decision was made to use Brute Force (BF) matcher as the feature matching algorithm. The BF matcher is computationally more expensive than FLANN, but will try every possible combination, meaning it will find the best results. But it is worth noting that a FLANN algorithm will most likely be used in future iterations in order to operate in real-time. The Satellite imagery was taken from the Digital Globe website and was provided in the form of a GeoTIFF allowing for latitude and longitude coordinates to be pulled out for every pixel of the image. Keypoints from the satellite imagery were matched with aerial imagery using the BF matcher. This method of matching returned thousands of matches between the two images, not all of them being correct. Using equation 12 for the Ratio Test, discussed in Chapter 2, helped to minimize any incorrect matches. The ratio test was tuned, to 0.56, in order to eliminate all incorrect matches.

IV. Results

Preamble

The beginning of this chapter discusses the overall results for the surface defect detection model and its ability to accurately detect and classify surface defects. The surface defect model's performance was measured using the Dice Coefficient (DC), as it was the preferred metric from Severstal [42]. The sections in this Chapter will cover the results obtained from the statistical approach as well as the training and test results from the segmentation model. Examples of the segmentation outputs will also be included for both methods. The second half of this chapter will review the results from the force feature segmentation model and its ability to force keypoints onto robust features of an image. These latter sections will cover several examples from the data augmentation methods used as well as the training and test results from the building/road segmentation model. The sections will also examine the number of features forced onto desired areas of an image through segmentation and some of the drawbacks with feature matching.

4.1 Steel Surface Defect Detection

4.1.1 Method 1: HOG

Figure 28 shows the histogram descriptors for a clean area and an area with a defect, respectively. Due to the surface of the metal being homogeneous the clean histogram shows relatively even gradient magnitudes across the spectrum. In the defect histogram a clear increase is observed in the magnitude between 110 and 120 degrees indicating a significant change in the gradient and the presence of a defect.

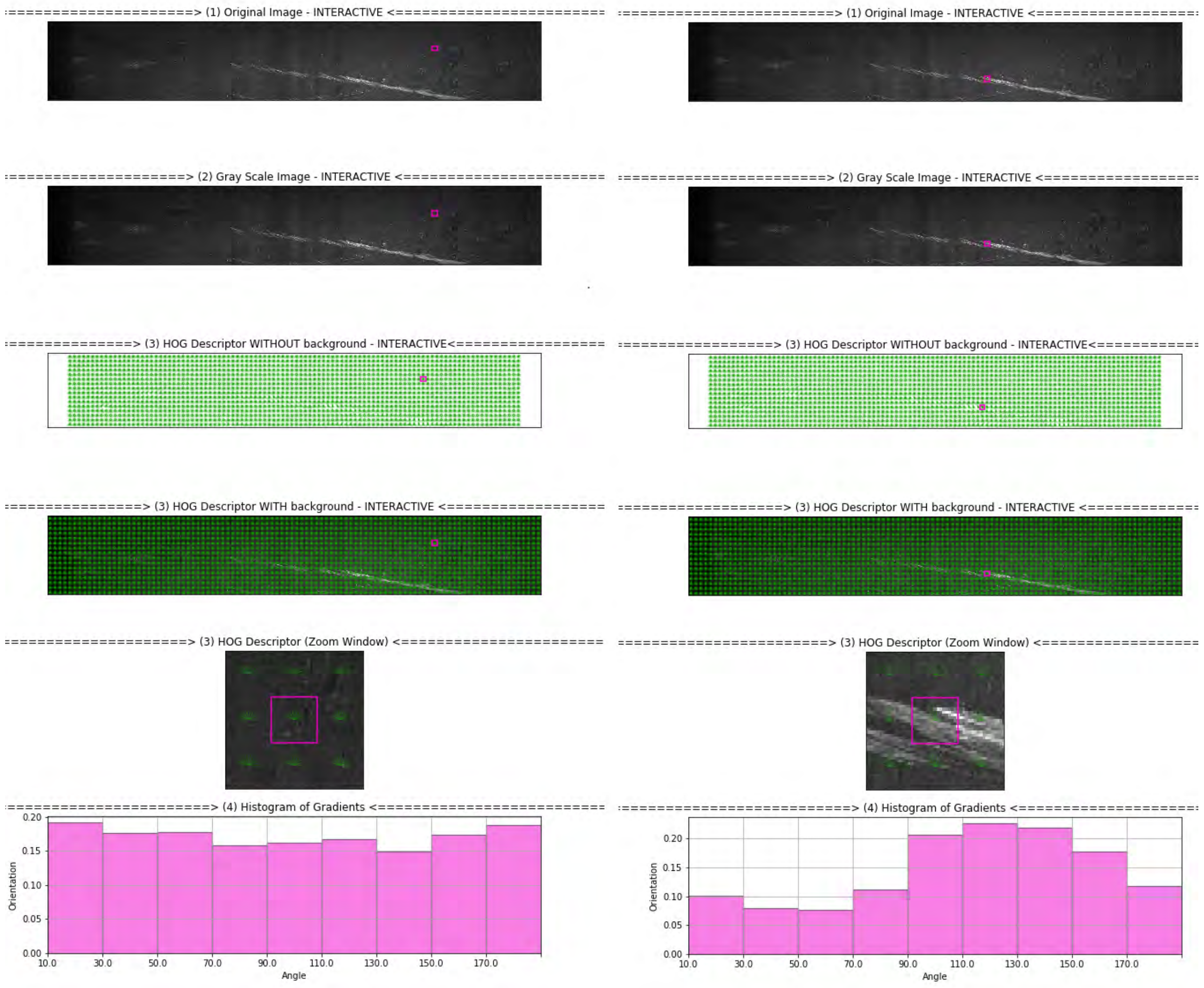


Figure 28: The top two images show the location being evaluated by the HOG kernel. The left image shows an area of the plate that is clean, and the right image shows a defective area. The middle figures show the grid of locations evaluated by the HOG. The second lowest figure shows the pixels being evaluated by the HOG kernel and the lowest image shows the histogram. The relatively flat histogram indicates a clean area of the plate, whereas the right image shows a clear gradient around 120 degrees.

The HOG descriptors shown above were used to train an SVM to automatically classify pixels between defect and no defect. The results for the performance of each

SV (one for each class of defect) can be found in Table 1. The accuracy for each model across the 5 different seeds are shown in Figure **??**

Table 1: Overall, the linear SVM classifiers were able to distinguish defects fairly accurately: between 80% and 93% DC.

Defect 1	**HOG Parameters**			
Vs Null	***Precision***	***Recall***	***F1 Score***	***support***
Negative (Null)	0.823	0.76	0.79	143
Positive (Defect)	0.78	0.84	0.81	144
Accuracy			**0.80**	287

Defect 2	**HOG Parameters**			
Vs Null	***Precision***	***Recall***	***F1 Score***	***support***
Negative (Null)	0.94	0.87	0.90	38
Positive (Defect)	0.89	0.95	0.92	41
Accuracy			**0.91**	79

Defect 3	**HOG Parameters**			
Vs Null	***Precision***	***Recall***	***F1 Score***	***support***
Negative (Null)	0.81	0.84	0.83	606
Positive (Defect)	0.83	0.80	0.82	594
Accuracy			**0.82**	1200

Defect 4 Vs Null	**HOG Parameters**			
	Precision	***Recall***	***F1 Score***	***support***
Negative (Null)	0.93	0.93	0.93	136
Positive (Defect)	0.93	0.93	0.93	128
Accuracy			**0.93**	256

Each table above provides the average results, across 5 random seeds, for each defect type. The fist table presents the results from feeding the SVM with 143 negative, clean, and 144 positive, defective, samples given by the support numbers. The precision and recall was calculated using equations 8 & 7 for both the negative and positive datasets. When dealing with defects in a object or material it is usually preferred to error on the side of caution and over-classify things as defects, so as not to let any defects through. Given this assumption, the models should have a higher recall value for the positive dataset and a higher precision value for the negative dataset. Recall means that out of all the defects in the defective class, "how many of the defects were predicted correctly?" Whereas the precision means out of all the images predicted clean by the model, "how many many times was the image actually clean?" This combination of precision and recall and the results presented in the tables demonstrates and provides a higher confidence that these models are properly tuned for defect detection. Finally the F1 score, or DC, is the combination of precision and recall, using equation 10, to provide a single quantification of model performance.

The results above are based on the validation data. The final step, after tuning, was to run the original test images, set aside in the beginning, through each of the 4 models. Each image was given a confidence score from the 4 models and the model with the highest score had that respective label assigned to the image. The success of the overall model was assessed using accuracy, measuring the number of correct classifications over the total number of images being classified. To further tune the model a probability threshold for each model was set, as discussed in section 3.1.3. By setting a higher threshold, the models become more strict in their criteria for assigning their respective labels, resulting in a greater number of clean images and increasing the probability of a false negative. However, to maximize recall, a lower threshold should be utilized. A lower threshold will loosen the restrictions on applying a class label and would catch more defects but would also increase the number of false positives. A threshold value of 50% was applied to gather initial results for model correction. Threshold tuning should be conducted when applying this model to the final application.

The confusion matrix in Table 2 shows the results from running all the images through the four models with a probability threshold of 50%. All values are normalized against the total number of samples in each class. This table shows that there is a flaw in model 4, as a majority of the class 4 defects where labeled as class 3.

Table 2: Confusion Matrix

		Predicted Value					
		1	***2***	***3***	***4***	***Clean***	**Total**
Actual Value	**1**	**0.47**	0.11	0.08	0.00	0.34	1.00
	2	0.06	**0.72**	0.12	0.00	0.10	1.00
	3	0.35	0.04	**0.57**	0.06	0.27	1.00
	4	0.00	0.01	0.52	**0.27**	0.20	1.00
	Clean	0.08	0.09	0.05	0.03	**0.75**	1.00
	Total	1.36	4.06	0.73	0.88	1.07	

One method to correct the ambiguity between class 3 and class 4, was the addition of a 5th model. This model was trained the same way as the previous 4 models, except this model used class 4 as the positive images and class 3 as the negative images. This model was created to learn the smaller nuances between the two classes in hopes that the model could delineate between the two. Table 3 shows that with the additional model the accuracy for class four greatly increased reducing the number of class 3 misclassifications from 52% to 10%, and increasing the number of correctly classified images from 27% to 69%. Using equations 8 & 7 provides a total precision and recall score of 73.67% and 68.9% respectively.

Table 3: Confusion Matrix w/ additional model

		Predicted Value					
		1	***2***	***3***	***4***	***Clean***	**Total**
Actual Value	**1**	**0.47**	0.11	0.08	0.00	0.34	1.00
	2	0.06	**0.72**	0.12	0.00	0.10	1.00
	3	0.06	0.04	**0.58**	0.06	0.27	1.00
	4	0.00	0.01	0.10	**0.69**	0.20	1.00
	Clean	0.08	0.09	0.05	0.03	**0.75**	1.00
	Total	1.07	3.16	0.67	1.26	1.07	

As mentioned previously, these results can further be tuned for precision or recall based on the desired needs. With no additional information given from Severstal on preference, I opted to maximize recall by attempting to bring the false negative rate to 10%. This decision was based on the project, surface defect detection in USAF aircraft, where maximizing recall would be preferable as a false negative would have greater consequences than false positives. One way to reduce the false negative rate in the model was to loosen the restrictions and increase the recall capabilities of the model. This was done by lowering the probability threshold for each model. To bring the false negative rate down to 10% several tests were run on threshold parameters, finally landing at 9% probability threshold. Table 4 shows the results from setting the threshold to 9% so that the new false negative rate falls from 32% to 10%. The new hyper-parameters also bring the recall score up to 89.3% while still maintaining a precision score of 73.67%.

Table 4: Confusion Matrix w/ 10% Probability Threshold

		Predicted Value					
		1	***2***	***3***	***4***	***Clean***	**Total**
Actual Value	**1**	**0.68**	0.12	0.09	0.01	0.11	1.00
	2	0.10	**0.74**	0.14	0.00	0.02	1.00
	3	0.10	0.06	**0.68**	0.07	0.09	1.00
	4	0.00	0.01	0.11	**0.83**	0.04	1.00
	Clean	0.10	0.10	0.07	0.03	**0.7**	1.00
	Total	1.93	4.7	0.79	1.48	0.82	

4.1.2 Method 2: ANN

Figure 29 shows the initial results of the image segmentation ResNet18 Model for all four class defects. Where class 1, 2, 3, and 4 are represented by yellow, green, blue, and magenta outlines on the left side. The black areas on the left side indicates the absence of any plate material. Having these image shifted to the right or the left and including the black areas helps the model to ignore background information should a test plate not cover an entire image. Including images with some background information will most likely play a larger roll in air craft inspection, especially around the edges of the aircraft, as the background for aircraft skin inspection could be rather busy in certain circumstances.

The model had a low false negative rate and proved capable at finding defects, however, it was misclassifying many of the defects resulting in a dice coeffienct score of 70%. After digging further into the results, it was apparent that the segmentation model was able to locate and segment the defects, but it was classifying every defect as a class 3. Figure 30 is a set of images overlayed with only a class 3 mask, meaning only class 3 defects (blue) from the left image should be highlighted on the right. However, all of the defects are still being highlighted regardless of class. This shows that there was mostly likely an imbalance in the data and the model was over-tuned for class 3 defects.

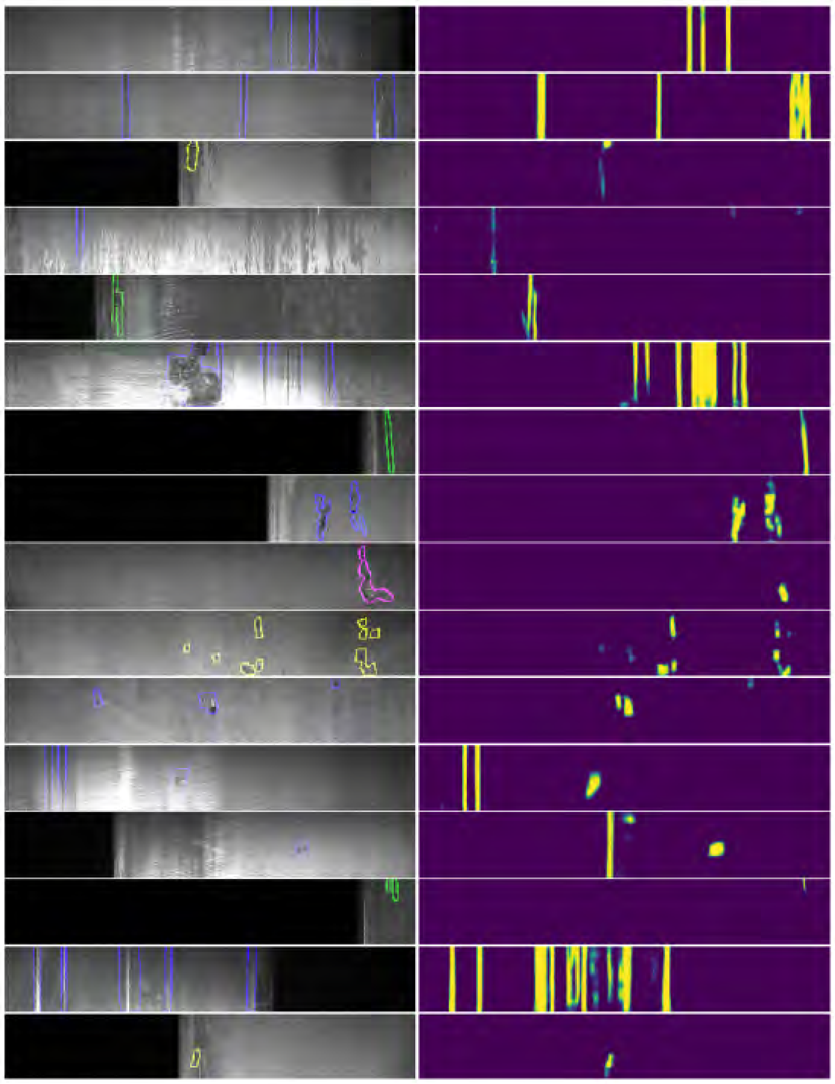


Figure 29: Multi-class segmentation mask representing all 4 classes on the left and the detections on the right

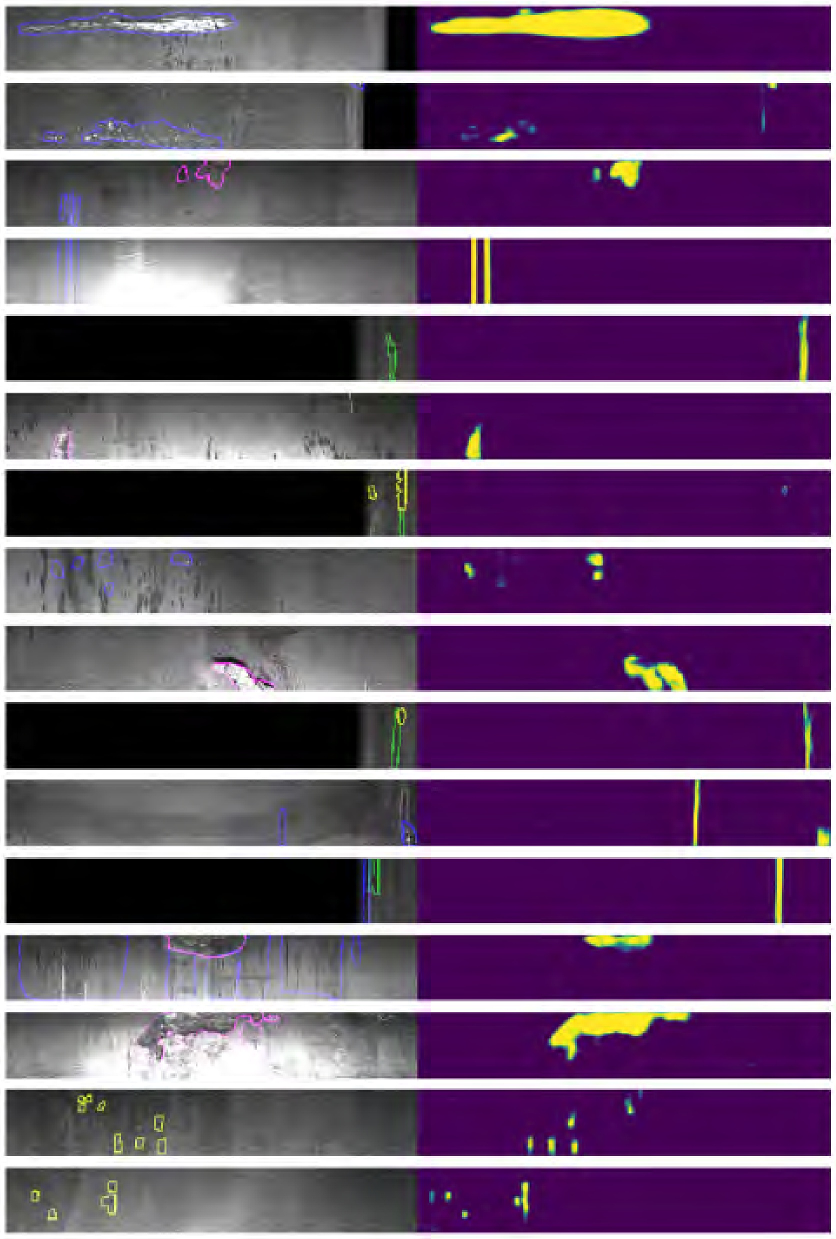


Figure 30: Left side shows various samples of steel plates containing all types of defects represented by the different color outlines. The right side shows the output when the samples are run through only the class 3 defect detection model. Class 3 defects are represented by the Blue lines on the left side.

The outcome highlighted by Figure 30 is most likely the result of 70% of the images being from class 3. This first model resulted in over-classification of class 3 defects, which gave a false accuracy reading of around 70% on the validation set when in reality it was defining every image as a class 3 defect. Unfortunately, it was at this time that the decision was made to switch the segmentation project onto building and road segmentation. Given more time an easy implementation would be to reduce the number of class 3 images from the dataset, under-sampling, and to increase the size of the smaller classes through augmentation, over-sampling. It is also worth noting that this issue is only relevant to the steel defect dataset, and not the intended final project. The benefit of having the data for surface defect detection in aircraft skin locally sourced, is that this provides control over the dataset class sizes and ensures an equal balance in data.

4.1.2.1 HOG and ANN Comparison

Both the HOG and ANN methods performed exceedingly well in defect detection. The statistical method using HOG showed that it was robust and highly tuneable to meet the needs of any application. The HOG was able to achieve, a self-set goal of, 90% recall while still maintaining a 73.6% precision score, giving an overall dice score of 80.978%. While the ANN only scored an initial dice score of 70%, due to defect classification, the figures presented showed that the segmentation model performed exceedingly well in defect detection. Given these results, both models should be able to perform well on surface defect detection on aircraft skin. In the future, given more control of the data, the ANN will most likely outperform the HOG method.

4.2 Forced Feature Selection through Segmentation

This section will cover all of the results obtained from the Forced Feature Selection (FSS) segmentation model, outlined in Section 3.2.3.1, on the INRIA aerial building dataset and the Massachussets Road dataset. This section will also cover results obtained from the feature selection and feature matching algorithms. One way to test the effects of segmentation on feature selection and matching was to use the FFS segmentation model on sequential imagery collected from flight test data. Using images collected from the same camera, at the same time of day helped to set a baseline for comparison when feature matching aerial imagery with satellite imagery.

4.2.1 Training

The following sections discuss the training performance for each dataset using the FFS model evaluation metrics of accuracy, Dice coefficient, and IoU, as well as the associated loss curves. The metric presented will follow suit with the leader board scores for the given dataset.

4.2.1.1 Building Segmentation: INRIA and SpaceNet

The FFS model was trained for 82 epochs and was able to achieve a 95.42% accuracry, 72.34% mIoU, and 87.4% DC. Table 5 Shows the 5 latest submissions on the Inria leaderboard as well as the top score of 81.0697 mIoU achieved by Eugene Khvedchenya compared to the FFS Model developed for this project. Table 6 shows the results obtained from several different papers that used the DC score instead of the traditional mIoU.

Table 5: Inria Leaderboard scores

Team	***Date***	***mIoU***	***Acc.***
Eugene Khvedchenya ods.ai	03 September 2020	**81.0697**	**97.25**
Alireza Abedin The University of Adelaide	21 October 2020	68.55	95.23
Heng-Qing Huang GuangXi University	22 Septmeber 2020	79.22	96.94
Mateusz Gomulski	23 September 2020	74.42	96.23
Alireza Abedin The University of Adelaide	27 October 2020	76.18	96.52
FFS Model: U-Net	No Submission	72.34	95.42

Table 6: Inria Dice Coefficient scores

Team	***Date***	***Dice Coef.***	***Acc.***
Yaroslavl State University [63]	April 2019	77	96.31
VGG16:U-Net Mendili Lamiae [64]	28 January 2020	**88.24**	**96.23**
FFS Model: U-Net	No Submission	87.4	95.42

From these results the FFS model was unable to achieve the latest state-of-the-art results, but it is very competitive for being considered a good segmentation model on the Inria database. As previously stated, the goal of the research was not to achieve a top score, but to achieve a good segmentation model for forced feature selection.

Examples of the segmentation output can be seen in Figure 31. From left to right, the figure shows the input image, the truth label, the output mask from the model, and binary mask conversion of the output. The output of the model is converted to a binary mask by changing pixel values to 1 if the probability of a building being present was greater than 50%, which is also referred to as the *probability threshold*. This value was chosen as an arbitrary starting point, but didn't require any further changes as alterations to the masks were conducted through mask padding.

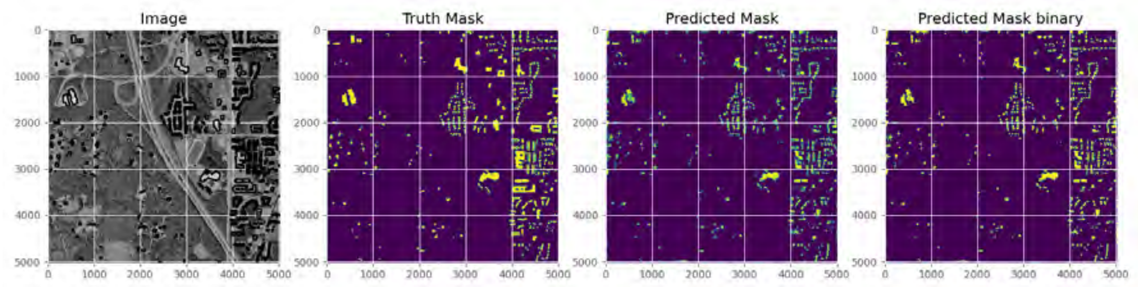


Figure 31: The images in this figure,from left to right, depict: the original input image, provided truth label, model output, model output converted to binary mask

Figure 32 shows a similar image but instead shows the input image on the left followed by the model output mask, the binary converted mask, and the last image is the binary mask overlayed with the original input image.

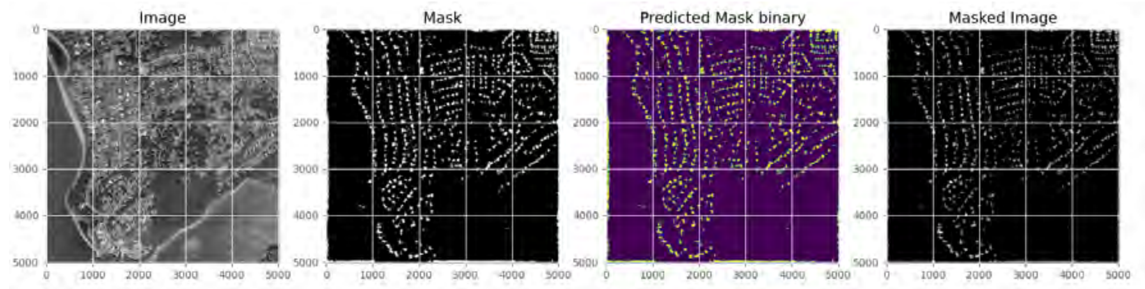


Figure 32: The images in this figure,from left to right, depict: the original input image, model output, model output converted to binary mask based on 50% probability, the binary mask overlayed with original input image

Figure 33 shows a zoomed in view of same image from Figure 32. Figure 34 shows an example of the final mask overlayed with the original image. This final output was padded by a 5x5 kernel to expand the mask boundaries just past the actual building boundaries. Early testing showed that the edges and corners of buildings are a primary source for unique and robust key features, but by having the mask border on the edge of the building some of these features were corrupted with the negative space from the mask. The padding not only helped push the edge boundary away from the building edge, but it also helped fill in continuity gaps of the building structure.

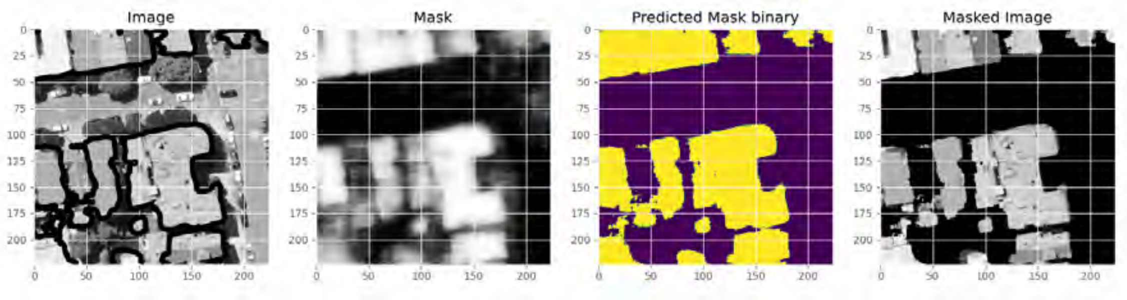


Figure 33: The images in this figure,from left to right, depict: the original input image, model output, model output converted to binary mask based on 50% probability, the binary mask overlayed with original input image

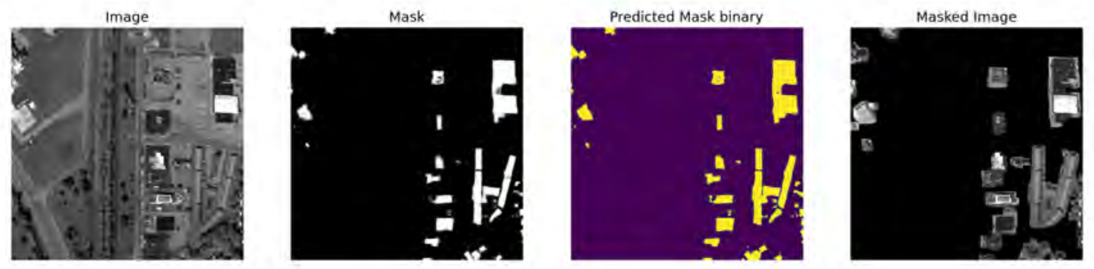


Figure 34: The images in this figure,from left to right, depict: the original input image, model output, model output converted to binary mask based on 50% probability, the binary mask overlayed with original input image and padded with a 5x5 dilation kernel

The steps described above lay down the ground work for forced feature selection.

Typical feature selection algorithms would pick up on the trees and cars in the image, however, these features can cause issues when conducting feature matching between images taken at different times of the day or year. Advanced techniques such as Random Sample Consensus (RANSAC) can be used to try and narrow down these mismatches but can often fail when the number of inliers is less than 50%, which can happen often in areas without unique features, or noisy areas such as dense areas of trees.

4.2.1.2 Road Segmentation: Massachusetts Database

The FFS model was trained on the Massachusetts Road Dataset for 42 epochs and achieved 91.87% accuracry, 55.39% mIoU, and 72.43% DC. Table 7 shows other top leader board scores and the architecture used. Given the time required to train a single model the VGG16 and ResNet50 models were unable to be trained. Only a single paper was found to use both the VGG16 and ResNet50 on the Massachusetts road dataset, but the results provided were not valid given an error they experienced during training. The U-Net model that scores a mIoU of 59.85 used a very similar architecture as the FSS model except for the input size of the image. The FFS model cropped the Road dataset model into 224x224 size images before training and the U-Net model on the leaderboard utilized 64x64 images. This results suggests that the smaller images are able to provide better training for the model. This is further evident by reviewing the top leaderboard score and their use of Atrous Spatial Pyramid Pooling (ASSP) [65]. ASSP is a semantic segmentation method for sampling certain layers at multiple rates as a way to classify regions of the image at varying scales [66]. Future iterations of the FSS model could incorporate this concept and include several methods for capturing and handling data at multiple scales. At this point the current model, while not quite reaching state-of-the-art results, when combined with the dilation morphological operations the road segmentation results are more than sufficient and are competitive for enhanced guided feature extraction.

Table 7: Massachusetts Leader board scores

Team	***mIoU***	***Acc.***
U-Net	59.85	95.66
LinkNet	67.92	96.78
DeepLabv3+	66.72	96.67
D-LinkNet	67.03	96.75
ResNet34 + ASPP + LinkNet [65]	68.78	96.72
FFS Model	55.39	91.87

Examples of the segmentation output can be seen in figures 36, 36, 37. The format and process follows the same as presented in the INRIA dataset. From left to right, the image shows the input image, the truth label, the output mask from the model, and binary mask conversion of the output. The output of the model is converted to a binary mask with a 50% probability threshold. From these figures the model appears to struggle in areas that are heavily crowded with trees or areas of the road covered or shadowed by larger buildings. This is noticeable in the right image where black lines are present without the yellow highlights, meaning that the model failed to identify a road.

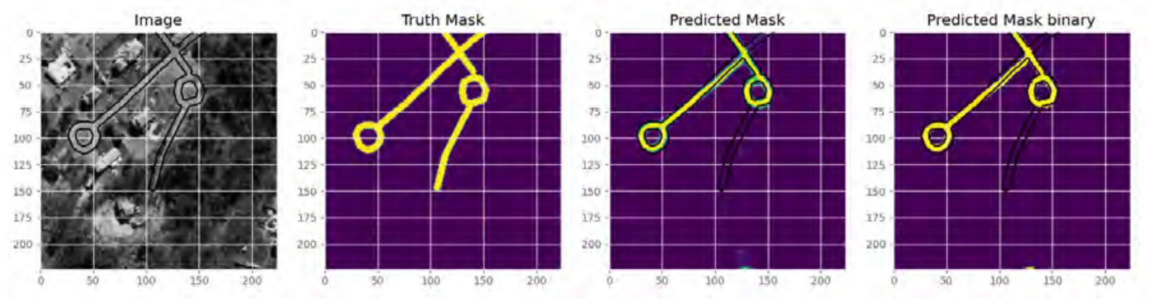


Figure 35: The images in this figure,from left to right, depict: the original input image, truth mask (also outlined in subsequent frames), model output, model output converted to binary mask based on 50% probability

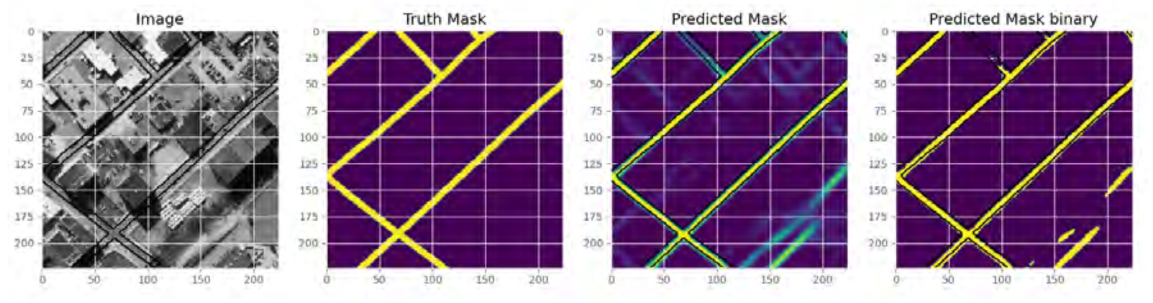


Figure 36: The images in this figure,from left to right, depict: the original input image, truth mask (also outlined in subsequent frames), model output, model output converted to binary mask based on 50% probability

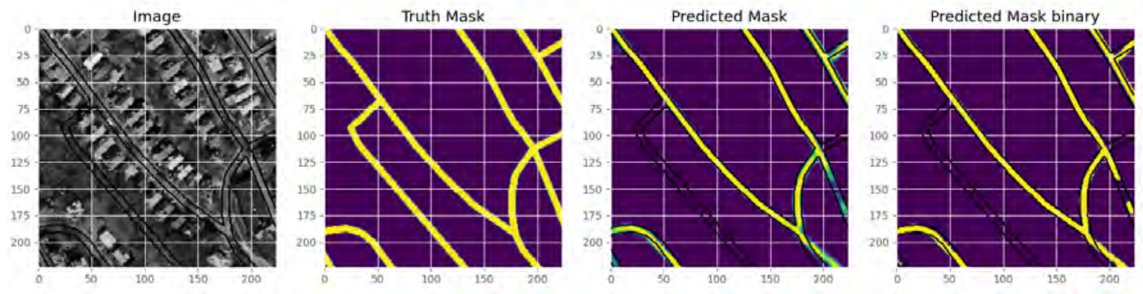


Figure 37: The images in this figure,from left to right, depict: the original input image, truth mask, model output (also outlined in subsequent frames), model output converted to binary mask based on 50% probability

4.2.2 Post-Processing

A single model could have been trained on the combined datasets containing buildings and roads, however, it has been shown that a combined model often performs worse than an ensemble of individually dedicated models [16, 67, 68]. The way around this issue was to take the output of each model and combine the mask outputs into a single mask. The combined masks can then be overlayed with the original image to segment out both roads and buildings. Figure 38 shows the same image but padded by a 5x5 kernel to fill in any gaps and expand the outline of the mask.

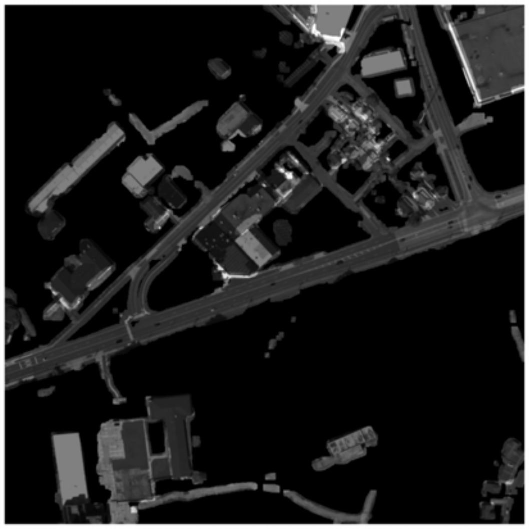


Figure 38: Road and building Segmentaiton mask overlayed with the original image and padded by a 5x5 kernel shows a complete removal of all non-essentail features

4.3 Optimal Feature detection and matching

4.3.1 Feature selection

Figure 39 shows a comparison of the results of the SIFT feature selection algorithm on a single image collected from the aerial flight test data before and after masking the image. In the left image there are numerous features attached to trees along the middle and in the fields, but the right image has limited all features to the buildings.

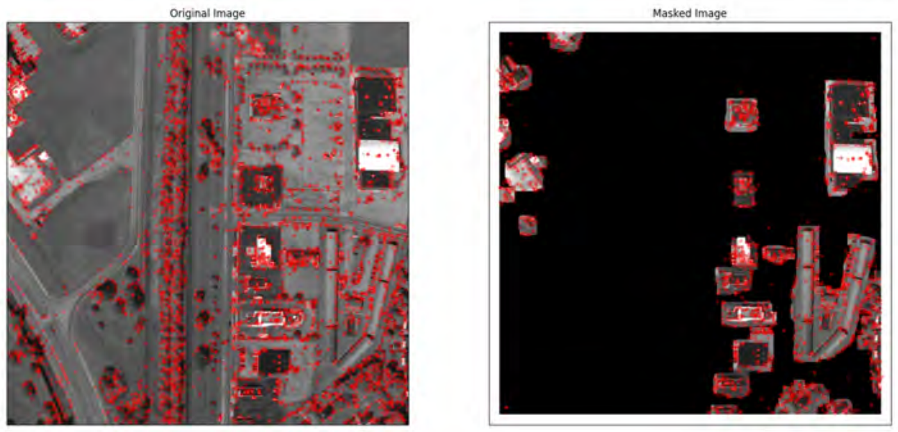


Figure 39: SIFT feature selection on original image, left, and the masked image, right.

4.3.2 Feature Matching

In order to remove certain variables, introduced by images from differing sources, feature matching was first conducted on subsequent imagery from flight test data. Figure 40 shows SIFT features with the BF matcher on the subsequent imagery. It was important to show the base image prior to masking to see how feature selection and feature matching was affected by the segmentation. With the two images being of the same source, the matching algorithms had no problem finding perfect matches. However, this image shows that many features were attached to the trees and the shadows cast by objects in the image. When comparing the aerial imagery to satellite imagery, the trees may be different given the time of the year and shadows will vary by the hour meaning these matches will most likely not appear in the final test. This comparison will be covered further in the chapter.

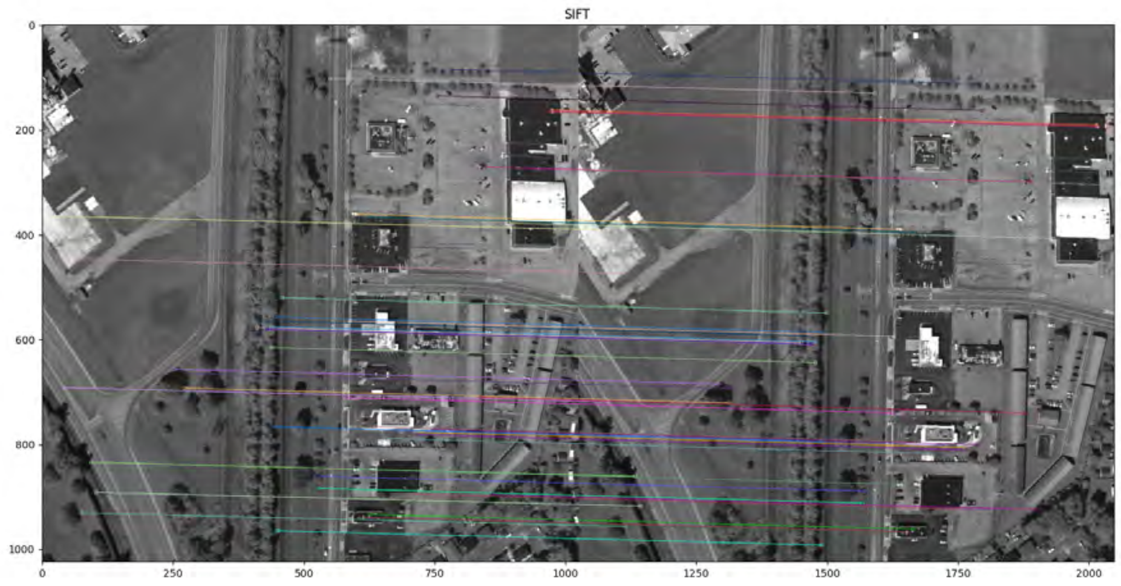


Figure 40: SIFT feature selection and matches from subsequent aerial imagery

Figure 41 shows the same imagery as Figure 40 but the right image is masked using the FFS segmentation model. While the model was able to force all the features onto the desired areas of the image, the descriptors were being corrupted by the hard border of the building perimeter and the mask, introducing a few mismatched descriptors.

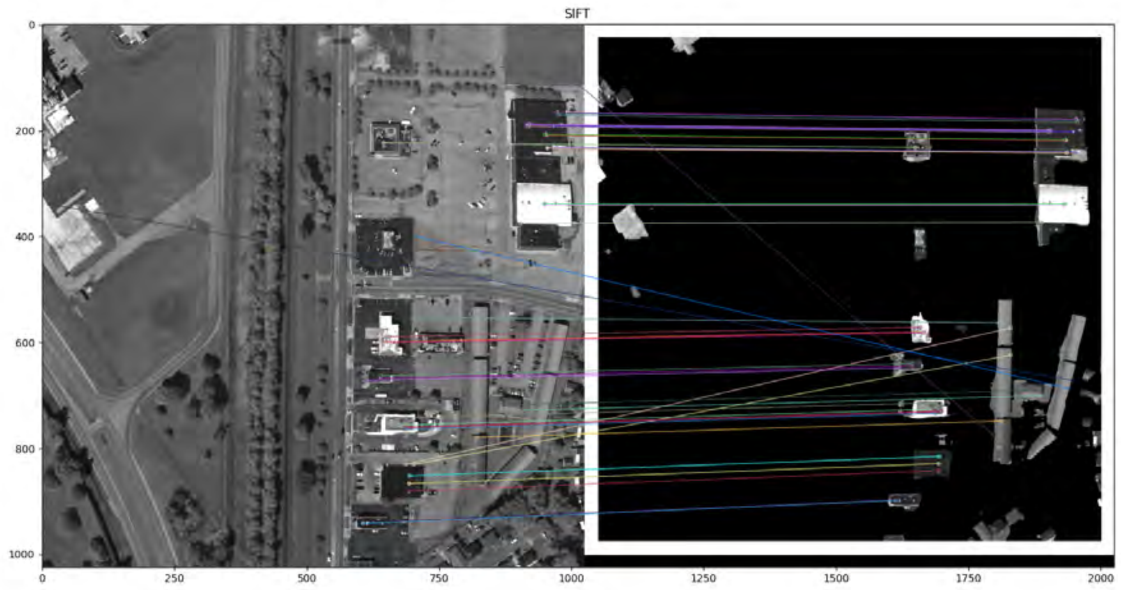


Figure 41: SIFT feature selection and matches from subsequent aerial imagery with the later image being the masked output from the FFS model

The solution to this problem was found through the use of a morphological operation, called dilate, to expand the mask by a set kernel size. To preserve the resolution of the original image and enlarge the mask it was important to apply the dilation operation prior to overlaying the mask with original image. Figure 42 shows the same image and mask as Figure 41 except the mask had been expanded by a 10x10 kernel.

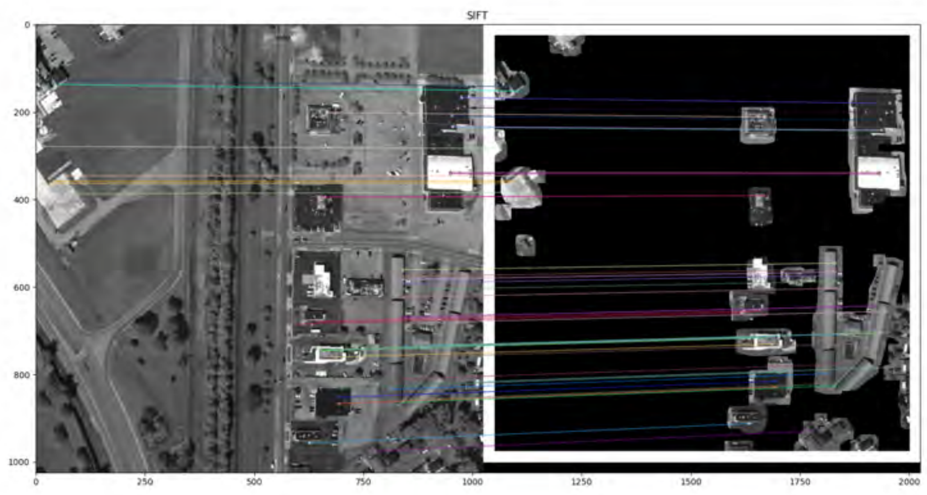


Figure 42: SIFT feature selection and matches from subsequent aerial imagery with the later image being the masked padded output from the FFS model

These results shows that by using the FFS segmentation model it is possible to move key features onto desired areas of an image and with the use of the dilation morphological operation one can expect quality feature matching results. The final test will employ this process and compare the results when feature matching between aerial flight test data and local satellite imagery.

4.4 Test

The final test for this project involved feature matching aerial imagery with local satellite imagery. The previous section proved that through segmentation it is possible to force feature selection onto specific areas of an image. Comparing these specific areas to a known satellite image, one can correlate geographical coordinates from the satellite imagery to the aerial imagery. These coordinates can then be used to solve for the camera pose estimation using a PnP algorithm [69].

4.4.1 Feature Matching Aerial vs Satellite imagery baseline

The first step was to develop a baseline comparison by conducting feature matching on Aerial and Satellite imagery without any segmentation to force feature selection. Figure 43 shows the aerial image on the left and the local satellite image on the right. The BF matcher identified 3727 matches from the two samples. The image shows the top 50 matches from the SIFT features. While there are a few correct matches, a majority of the matches failed to find any correlation. Given the size and resolution difference between the images there is likely more noise in the descriptors than when comparing the two aerial images in the previous section. One way to account for this difference is through a quality control method, such as the Ratio test.

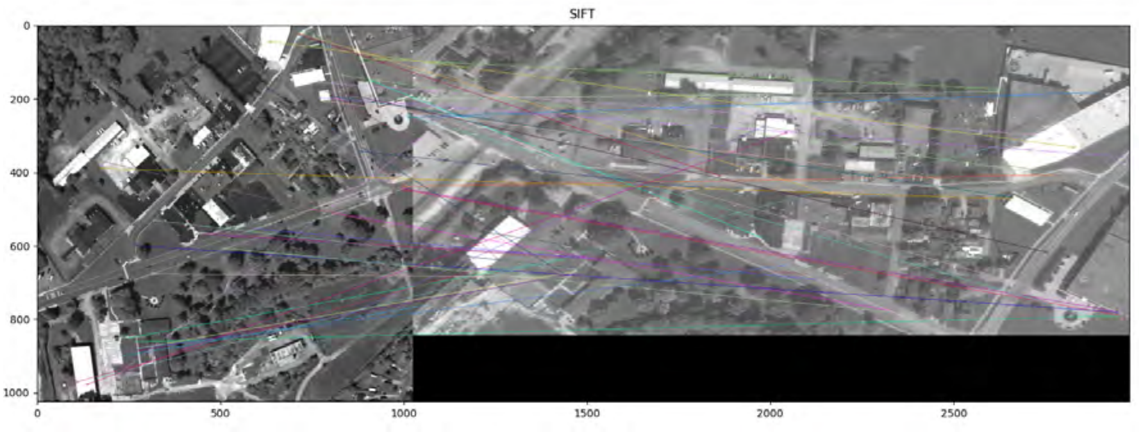


Figure 43: This figure depicts the SIFT matches between the aerial image, left, and the satellite image, right, collected from Digital Globe.

4.4.2 Feature Matching Aerial vs Satellite imagery using Ratio Test

Using the Ratio test, discussed in section 2.5.5, reduced the number of matches by eliminating noisy features. Figure 44 shows the same comparison but with the addition of the ratio test tuned to .56. The ratio test reduced the number of matches from 3727 down to 7 with 5 of them being correct. Lower values for the Ratio test resulted in correct matches being eliminated rather than the 2 incorrect matches and higher values introduced more incorrect than correct matches. The ratio test value of .56 was selected to provide the highest ratio of correct to incorrect matches through visual inspection.

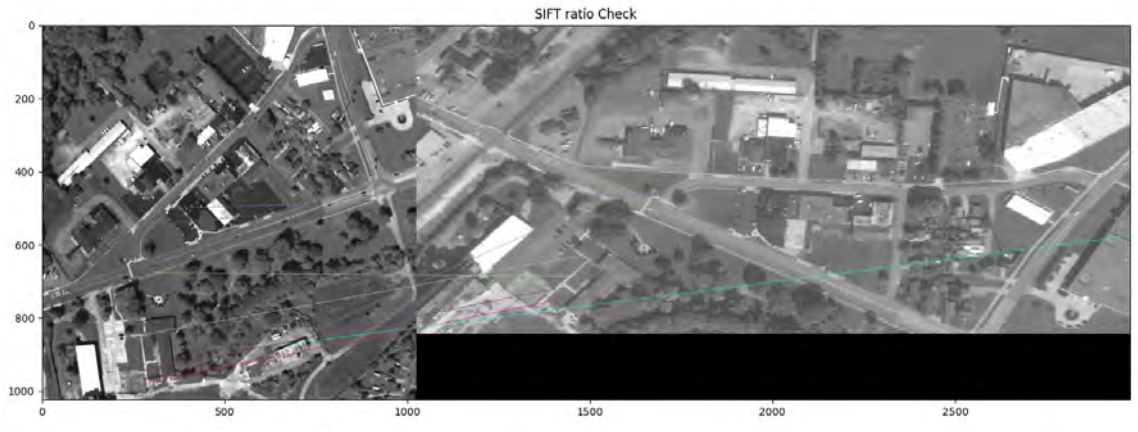


Figure 44: SIFT matches between the aerial and satellite imagery after being processed through the Ratio test

While the quality of the matches improved, the problem with this solution is that none of the matches found were attached to buildings. This is further evident by Figure 4.4.4 which shows zero matches between the masked aerial and satellite imagery using SIFT features and the Ratio Test. The addition of the road segmentation model could help find additional matches, however, there should be matches between the buildings. The addition of the road model would only obscure the issue which appears to be in the disparity between SIFT features for the aerial and satellite imagery.

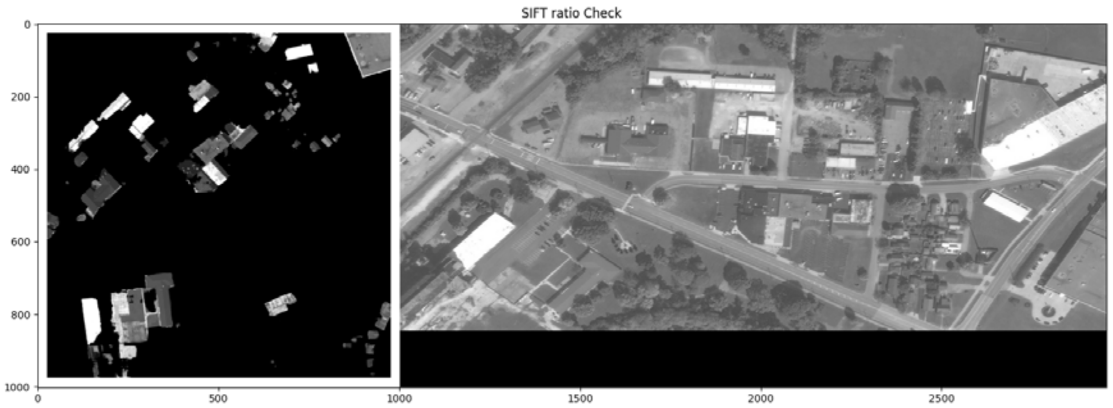


Figure 45: Image shows that zero matches were found between the aerial and satellite imagery when comparing buildings alone.

4.4.3 Feature Matching North-aligned Aerial vs Satellite imagery

SIFT is defined as being scale, translation, rotation, and illumination invariant. However one way to improve upon the disparity between SIFT descriptors is to begin by removing several of these transformations. The first test involved using the aircraft's roll, pitch, and yaw to north rectify the aerial imagery to align with the satellite imagery. Initial tests showed improved results when removing the rotation variable. Using the attitude collected from the aircraft Figure 46 shows an aerial image that has been north aligned to match the satellite imagery. North aligning the image was able to produce 7 out of 7 correct matches.

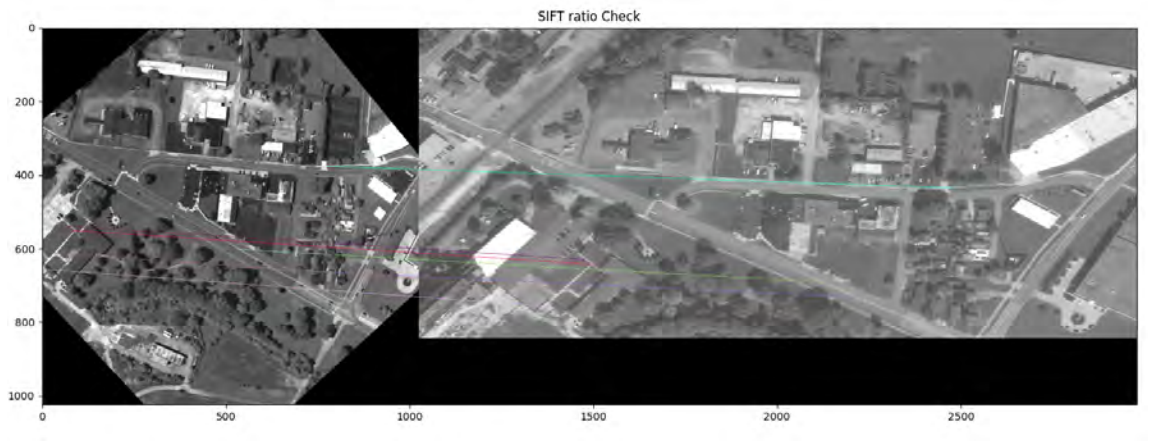


Figure 46: SIFT feature matching with the Ratio Test. Left image is a north aligned aerial photo and the right is a satellite image of the same local area

4.4.4 Feature Matching FFS Segmentation Aerial vs Satellite imagery

The last step is to apply the entire process once again to the aerial and satellite imagery. Figure 47 shows an aerial imagery that has been segmented with the FFS model for buildings and roads, padded with a 10x10 kernel, and north aligned to match the satellite imagery. Processing the imagery with the process described above provides 2 matches between the aerial imagery and the satellite imagery, up from the previous 0 matches in figure . However, a minimum of 3 points is needed for a PnP solution, although additional points will be needed for an accurate pose estimation.

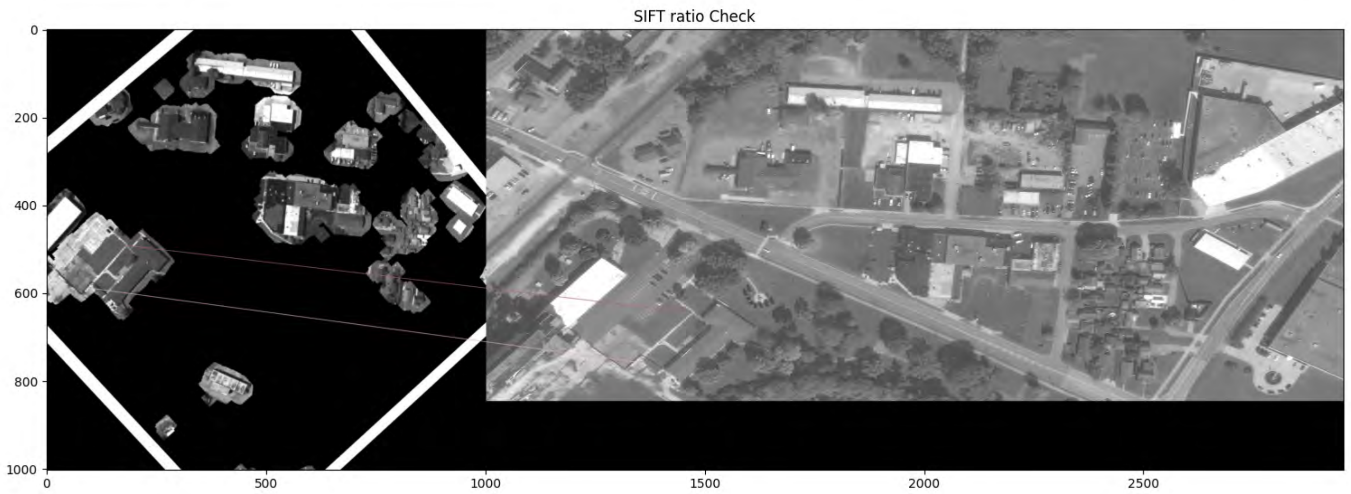


Figure 47: Padded North Aligned Aerial imagery vs Satellite

4.4.5 Feature Matching FFS Segmentation Aerial vs Subsequent Aerial imagery

While north aligned aerial imagery helped improve the number of matches, replacing the satellite image in the previous figure with the next aerial image in the time series shows that SIFT and BF matcher should have no issues in finding matches that are not aligned. Figure 48 shows the results of SIFT and BF matcher on subsequent images from the same source. Given this result, while SIFT descriptors are scale, translation, rotation, and illumination invariant, it is apparent that SIFT descriptors are unable to overcome the difference between the aerial and satellite imagery.

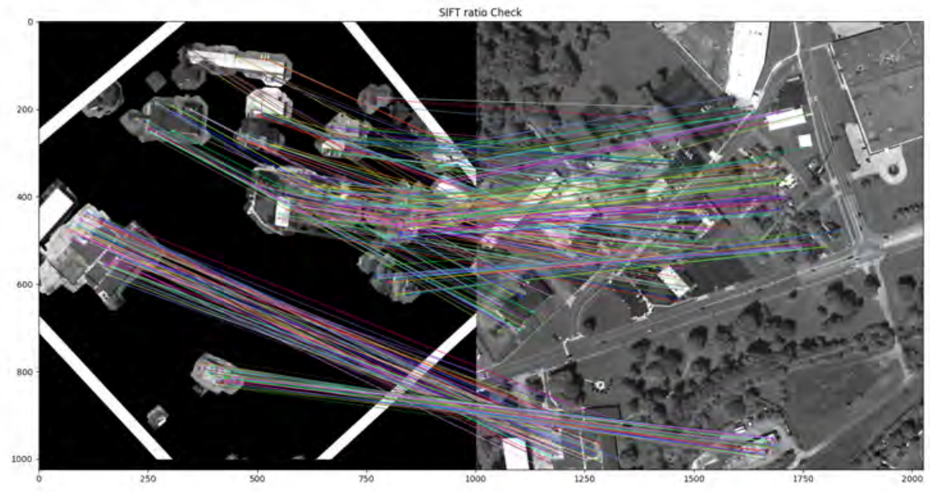


Figure 48: Padded North Aligned Aerial imagery vs subsequent Aerial Imagery

4.4.6 Review

The disparity between the aerial and satellite imagery being an issue for feature matching was an area of research that was never considered in the planning for this project. As such, future research will need to be accomplished in this area before a consistent and accurate camera pose estimation can be realized. At present given the wide variability in the number and quality of feature matches, the camera pose estimation error varies between 10-300m. Once the validity of the features matching can be improved, the range of error in the camera pose estimation should narrow down closer to the 10m range, making this method of alternative visual navigation approaching the capabilities of GPS.

V. Conclusions

The first project's focus was to train various U-net models for the purpose of surface defect detection in aircraft skin. However, due to extenuating circumstances the data for the aircraft material was not available. Therefore all tests were conducted on the steel surface defect detection dataset provided by Severstal. Two methods were utilized in this projects, a traditional statistical approach, using a HOG and SVM, and a more modern machine learning approach. Due to time constraints and switching projects neither method was able to achieve state-of-the-art results, however the performance of each model can serve as a baseline for future work in this area.

The second project set out to test the viability of using semantic segmentation as a way to force features on to keypoints of an image in order improve upon a PnP solution. Ultimately the force feature selection provided to be a viable method and successfully forced features onto desired areas of an image. However, further research will need to be conducted on feature matching algorithms that can overcome the nuances of images from vastly different sources. The feature matching algorithm needs to be able to overcome differences in sources that can be seen as resolution , scale, rotation, and illumination disparities found in the descriptors of the key points. The lack of valid matches between the two sources inhibited the PnP algorithim to achieve an adequate camera pose that could rival GPS.

5.1 Future Work

5.1.1 Steel Surface Defect Detection

The findings that I have presented for this project suggests that in the absence of adequate data to use modern deep learning techniques, histogram of orientated gradients and support vector machines can be utilized for surface defect detection applications. It was also shown that the model was robust and could be tuned and tailored for different applications based on user need.

5.1.1.1 Training and Tuning

One of the many limitations to this project was the time needed to train and tune models. A true one-vs-one SVM architecture requires 10 models to be trained for 5 classifications. The initial 4 models trained against the null showed adequate results with relatively low misclassifications, however the extra models, class 3 vs class 4, showed that a drastic increase in accuracy could be obtained with additional models. The extra models for this project were abandoned due to time constraints and that the whole approach was unlikely to beat a modern machine learning approach. However, given the HOG and SVMs ability to work on relatively small sets of data it could be worth pursuing. Further work can be conducted in various way to tune bin sizes in order to improve upon the ability of the model to pull out nuances of each defect given size, shape, and color. Little time was spent on tuning HOG hyper-parameters as this aspect is very dependant on the attributes of the defect. This step was set aside until more information could be obtained from the surface defects in aircraft skin imagery. Given that a significant amount of time and resources were spent training the various models, little time was left to expand the research to explore other SVM architectures. The data obtained in this project was achieved utilizing a linear SVM, non-linear SVMS were not explored in this research.

5.1.1.2 Data Imbalance

The last limitation that could not be addressed, given the scope of this project, was an imbalance in the data. Without working directly with Severstal, acquiring additional data was out of the question. Due to data imbalances some classes only had 300 images for train/test while others had over 3000. Given the opportunity, a more balanced representation of data could drastically improve upon accuracy across the board and even remove the need for additional models beyond the basic class vs null models. The class imbalance also ended up causing significant degradation in the U-Net model performance. A first approach at a segmentation model resulted in a model with excellent segmentation abilities but struggled with adequate classification. Due to the class imbalance in data, the model struggled to overcome this imbalance and often over-classified with the class 3 defect, even after introducing a weighted system. This was most likely due to the relatively small dataset, 12,000 images. One technique that was utilized by some of the top leaderboard scores, on the Kaggle steel defect competition, utilized Pseudo-Labeling to balance out the classes and increase the data size.

5.1.1.3 Pseudo-Labeling

Pseudo-Labeling is a semi-supervised labeling technique first introduced by Lee in 2013 [70]. This semi-supervised labeling method trains a model on a batch of labeled data and then uses the trained model to predict labels on a new batch of unlabeled data. The predicted labels from the unlabeled data are then used to calculate the loss on the unlabeled data. The labeled loss values are then combined with the unlabeled loss and then back-propagated. The winners of the Kaggle competition used the submissions from other competitors as their unlabeled data to improve their model performance.

5.1.1.4 Stacked Models

The final technique that can be utilized to improve the score is through the use of stacked models. This project utilized a single U-net model to produce the given results, however many of the top leaderboard scores utilize a U-Net model followed by up to three Feature Pyramid Networks (FPN). An FPN is very computational and memory expensive and therefore is often avoided. However an FPN excels at detecting objects at different scales. Facebook AI Research (FAIR) at the Cornell University developed a method of using an FPN as a basic Faster R-CNN system to achieve state-of-the-art results on the COCO detection dataset "utilizing a single model without bells and whistles" [71].

5.1.1.5 Final Thoughts For Defect Detection

The model and techniques presented in this paper will serve as a starting point for future work in surface defect detection on aircraft skin. While the model architecture will need to be designed and tailored for specific defect types, U-Net models have been proven to provide decent prediction results for various projects across the object detection spectrum. Therefore the U-net model and results presented in this paper can serve as a good baseline for future work in surface defect detection applications.

5.1.2 Forced Feature Segmentation Model

While the segmentation results for buildings and roads did not achieve state-of-the-art results, the overall performance of the segmentation models proved to be comparable to other recent scores and exceeded the needs of the project. By padding the output masks and overlaying the mask with the original image the results showed that a significant number of features could be forced onto desired key points of an image. However, when it came to feature matching, the BF Matcher failed to find connections between the aerial image descriptors and the satellite image descriptors. Several feature selection methods were tested to see if a connection could be found in the descriptors of the two images but none of the tested methods produced an adequate number of matches. A majority of the work done was completed with SIFT as a feature selection method as it produced the best results when paired with the Ratio Test. Future work will most likely be in exploring other various non-standard algorithms or hyper-parameters for feature selection and matching.

5.1.2.1 Effective Matching of Multi-source Optical Satellite Imagery

One approach for future research is through a method developed by Yuxuan Liu, Fan Mo, and Pengjie Tao called Effective Matching of Multi-source Optical Satellite Imagery (EMMOSI). The goal of their research was to develop a method to solve the multi-source feature matching problem. Feature matching images of different sources can be burdened by issues such geometric distortions and intensity differences. The solution to this problem developed in [72] involved three steps, the first two steps following an standard feature matching process. The first step utilized uniform robust scale invariant feature transform (UR-SIFT) detector as a way to extract feature points. Second, the initial matching was conducted based on the Euclidean distance to obtain a few correct matches and the initial projection transformation between the image pair. The finally step that the group introduced involved two matching strategies to propagate matches and produce more reliable matching results. The first strategy was to use the geometric relationship between the image pair, geometric correspondence matching found more matches than the initial UR-SIFT feature points. The second method was through probability relaxation matching which propagated new matches around the initial UR-SIFT feature points. The results of their work showed that the EMMOSI model, when ran against the Chinese ZY3 and GaoFen satellite images, outperformed the standard UR-SIFT model [72]. While the two datasets used in this dataset both consisted of satellite imagery, the process introduced by the EMOSSI model could translate to the aerial imagery presented in this paper.

5.1.2.2 Generative Adversarial Network

In the absence of current feature matching algorithms working, another alternative would be to create a new feature matching technique through machine learning. A Generative Adversarial Network (GAN) is a machine learning model, that runs on game theory, in which two neural networks compete with each to achieve the best possible predictions. A GAN is devised of two models, the generator and the discriminator. The generators goal is to artificially create data that could be mistaken for real world collected data. The discriminators job is then to identify which outputs it receives are real or which have been artificially created. The feedback loop created by the two adversarial networks is essentially a way for GANs to create their own training data.

The ability of a GAN to produce additional training data will be extremely vital to developing a feature matching algorithm that can be tailored to the exact datasets that will be used in a real world test, as well as provide a framework for easy conversion to other datasets in the future.

Appendix A. Appendix A. U-Net model visual representations

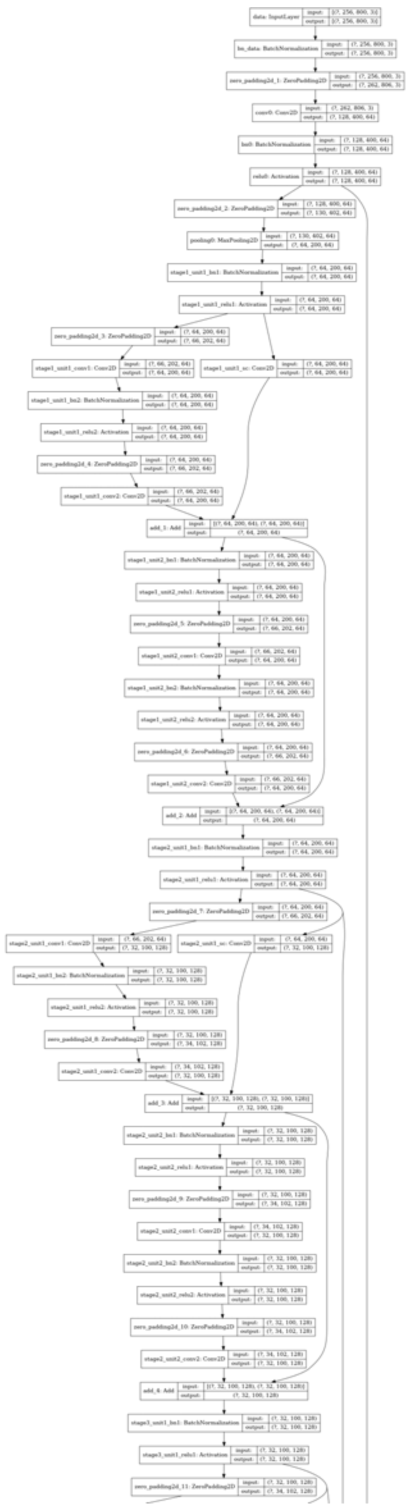

data: InputLayer
input: [(?, 256, 800, 3)]
output: [(?, 256, 800, 3)]
bn_data: BatchNormalization
input: (?, 256, 800, 3)
output: (?, 256, 800, 3)
zero_padding2d_1: ZeroPadding2D
input: (?, 256, 800, 3)
output: (?, 262, 806, 3)
conv0: Conv2D
input: (?, 262, 806, 3)
output: (?, 128, 400, 64)
bn0: BatchNormalization
input: (?, 128, 400, 64)
output: (?, 128, 400, 64)
relu0: Activation
input: (?, 128, 400, 64)
output: (?, 128, 400, 64)
zero_padding2d_2: ZeroPadding2D
input: (?, 128, 400, 64)
output: (?, 130, 402, 64)
pooling0: MaxPooling2D
input: (?, 130, 402, 64)
output: (?, 64, 200, 64)
stage1_unit1_bn1: BatchNormalization
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
stage1_unit1_relu1: Activation
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
zero_padding2d_3: ZeroPadding2D
input: (?, 64, 200, 64)
output: (?, 66, 202, 64)
stage1_unit1_conv1: Conv2D
input: (?, 66, 202, 64)
output: (?, 64, 200, 64)
stage1_unit1_sc: Conv2D
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
stage1_unit1_bn2: BatchNormalization
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
stage1_unit1_relu2: Activation
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
zero_padding2d_4: ZeroPadding2D
input: (?, 64, 200, 64)
output: (?, 66, 202, 64)
stage1_unit1_conv2: Conv2D
input: (?, 66, 202, 64)
output: (?, 64, 200, 64)
add_1: Add
input: [(?, 64, 200, 64), (?, 64, 200, 64)]
output: (?, 64, 200, 64)
stage1_unit2_bn1: BatchNormalization
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
stage1_unit2_relu1: Activation
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
zero_padding2d_5: ZeroPadding2D
input: (?, 64, 200, 64)
output: (?, 66, 202, 64)
stage1_unit2_conv1: Conv2D
input: (?, 66, 202, 64)
output: (?, 64, 200, 64)
stage1_unit2_bn2: BatchNormalization
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
stage1_unit2_relu2: Activation
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
zero_padding2d_6: ZeroPadding2D
input: (?, 64, 200, 64)
output: (?, 66, 202, 64)
stage1_unit2_conv2: Conv2D
input: (?, 66, 202, 64)
output: (?, 64, 200, 64)
add_2: Add
input: [(?, 64, 200, 64), (?, 64, 200, 64)]
output: (?, 64, 200, 64)
stage2_unit1_bn1: BatchNormalization
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
stage2_unit1_relu1: Activation
input: (?, 64, 200, 64)
output: (?, 64, 200, 64)
zero_padding2d_7: ZeroPadding2D
input: (?, 64, 200, 64)
output: (?, 66, 202, 64)
stage2_unit1_conv1: Conv2D
input: (?, 66, 202, 64)
output: (?, 32, 100, 128)
stage2_unit1_sc: Conv2D
input: (?, 64, 200, 64)
output: (?, 32, 100, 128)
stage2_unit1_bn2: BatchNormalization
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
stage2_unit1_relu2: Activation
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
zero_padding2d_8: ZeroPadding2D
input: (?, 32, 100, 128)
output: (?, 34, 102, 128)
stage2_unit1_conv2: Conv2D
input: (?, 34, 102, 128)
output: (?, 32, 100, 128)
add_3: Add
input: [(?, 32, 100, 128), (?, 32, 100, 128)]
output: (?, 32, 100, 128)
stage2_unit2_bn1: BatchNormalization
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
stage2_unit2_relu1: Activation
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
zero_padding2d_9: ZeroPadding2D
input: (?, 32, 100, 128)
output: (?, 34, 102, 128)
stage2_unit2_conv1: Conv2D
input: (?, 34, 102, 128)
output: (?, 32, 100, 128)
stage2_unit2_bn2: BatchNormalization
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
stage2_unit2_relu2: Activation
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
zero_padding2d_10: ZeroPadding2D
input: (?, 32, 100, 128)
output: (?, 34, 102, 128)
stage2_unit2_conv2: Conv2D
input: (?, 34, 102, 128)
output: (?, 32, 100, 128)
add_4: Add
input: [(?, 32, 100, 128), (?, 32, 100, 128)]
output: (?, 32, 100, 128)
stage3_unit1_bn1: BatchNormalization
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
stage3_unit1_relu1: Activation
input: (?, 32, 100, 128)
output: (?, 32, 100, 128)
zero_padding2d_11: ZeroPadding2D
input: (?, 32, 100, 128)
output: (?, 34, 102, 128)

Figure 49: Defect U-net based on ResNet18 - Part 1

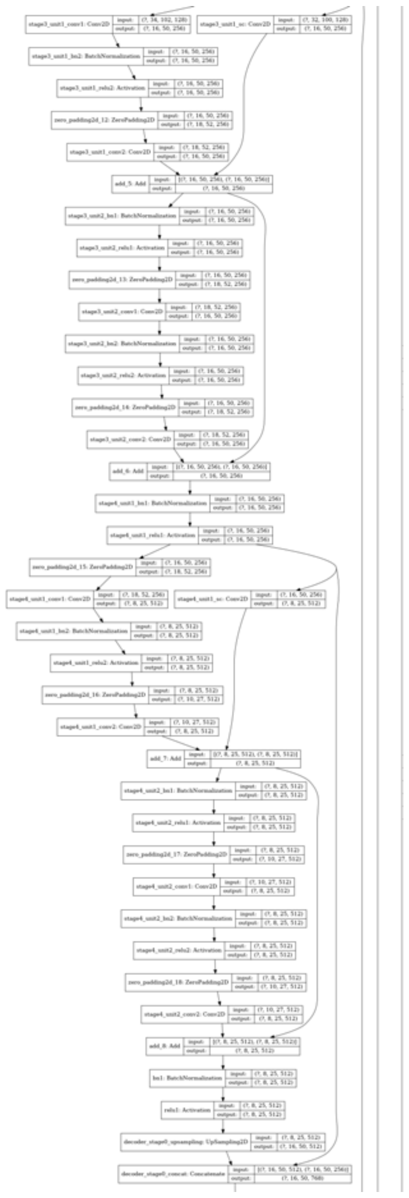


Figure 50: Defect U-net based on ResNet18 - Part 2

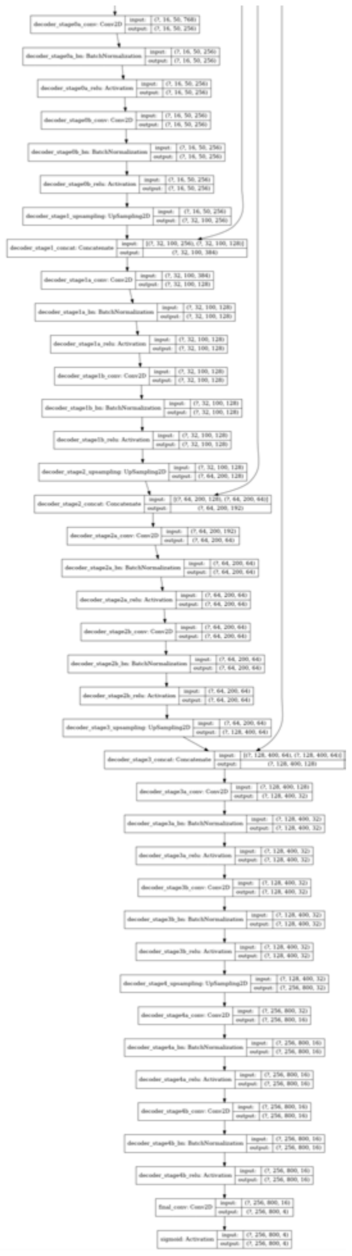

decoder_stage0a_conv: Conv2D | input: (?, 16, 50, 768) | output: (?, 16, 50, 256)
decoder_stage0a_bn: BatchNormalization | input: (?, 16, 50, 256) | output: (?, 16, 50, 256)
decoder_stage0a_relu: Activation | input: (?, 16, 50, 256) | output: (?, 16, 50, 256)
decoder_stage0b_conv: Conv2D | input: (?, 16, 50, 256) | output: (?, 16, 50, 256)
decoder_stage0b_bn: BatchNormalization | input: (?, 16, 50, 256) | output: (?, 16, 50, 256)
decoder_stage0b_relu: Activation | input: (?, 16, 50, 256) | output: (?, 16, 50, 256)
decoder_stage1_upsampling: UpSampling2D | input: (?, 16, 50, 256) | output: (?, 32, 100, 256)
decoder_stage1_concat: Concatenate | input: [(?, 32, 100, 256), (?, 32, 100, 128)] | output: (?, 32, 100, 384)
decoder_stage1a_conv: Conv2D | input: (?, 32, 100, 384) | output: (?, 32, 100, 128)
decoder_stage1a_bn: BatchNormalization | input: (?, 32, 100, 128) | output: (?, 32, 100, 128)
decoder_stage1a_relu: Activation | input: (?, 32, 100, 128) | output: (?, 32, 100, 128)
decoder_stage1b_conv: Conv2D | input: (?, 32, 100, 128) | output: (?, 32, 100, 128)
decoder_stage1b_bn: BatchNormalization | input: (?, 32, 100, 128) | output: (?, 32, 100, 128)
decoder_stage1b_relu: Activation | input: (?, 32, 100, 128) | output: (?, 32, 100, 128)
decoder_stage2_upsampling: UpSampling2D | input: (?, 32, 100, 128) | output: (?, 64, 200, 128)
decoder_stage2_concat: Concatenate | input: [(?, 64, 200, 128), (?, 64, 200, 64)] | output: (?, 64, 200, 192)
decoder_stage2a_conv: Conv2D | input: (?, 64, 200, 192) | output: (?, 64, 200, 64)
decoder_stage2a_bn: BatchNormalization | input: (?, 64, 200, 64) | output: (?, 64, 200, 64)
decoder_stage2a_relu: Activation | input: (?, 64, 200, 64) | output: (?, 64, 200, 64)
decoder_stage2b_conv: Conv2D | input: (?, 64, 200, 64) | output: (?, 64, 200, 64)
decoder_stage2b_bn: BatchNormalization | input: (?, 64, 200, 64) | output: (?, 64, 200, 64)
decoder_stage2b_relu: Activation | input: (?, 64, 200, 64) | output: (?, 64, 200, 64)
decoder_stage3_upsampling: UpSampling2D | input: (?, 64, 200, 64) | output: (?, 128, 400, 64)
decoder_stage3_concat: Concatenate | input: [(?, 128, 400, 64), (?, 128, 400, 64)] | output: (?, 128, 400, 128)
decoder_stage3a_conv: Conv2D | input: (?, 128, 400, 128) | output: (?, 128, 400, 32)
decoder_stage3a_bn: BatchNormalization | input: (?, 128, 400, 32) | output: (?, 128, 400, 32)
decoder_stage3a_relu: Activation | input: (?, 128, 400, 32) | output: (?, 128, 400, 32)
decoder_stage3b_conv: Conv2D | input: (?, 128, 400, 32) | output: (?, 128, 400, 32)
decoder_stage3b_bn: BatchNormalization | input: (?, 128, 400, 32) | output: (?, 128, 400, 32)
decoder_stage3b_relu: Activation | input: (?, 128, 400, 32) | output: (?, 128, 400, 32)
decoder_stage4_upsampling: UpSampling2D | input: (?, 128, 400, 32) | output: (?, 256, 800, 32)
decoder_stage4a_conv: Conv2D | input: (?, 256, 800, 32) | output: (?, 256, 800, 16)
decoder_stage4a_bn: BatchNormalization | input: (?, 256, 800, 16) | output: (?, 256, 800, 16)
decoder_stage4a_relu: Activation | input: (?, 256, 800, 16) | output: (?, 256, 800, 16)
decoder_stage4b_conv: Conv2D | input: (?, 256, 800, 16) | output: (?, 256, 800, 16)
decoder_stage4b_bn: BatchNormalization | input: (?, 256, 800, 16) | output: (?, 256, 800, 16)
decoder_stage4b_relu: Activation | input: (?, 256, 800, 16) | output: (?, 256, 800, 16)
final_conv: Conv2D | input: (?, 256, 800, 16) | output: (?, 256, 800, 4)
sigmoid: Activation | input: (?, 256, 800, 4) | output: (?, 256, 800, 4)

Figure 51: Defect U-net based on ResNet18 - Part 3

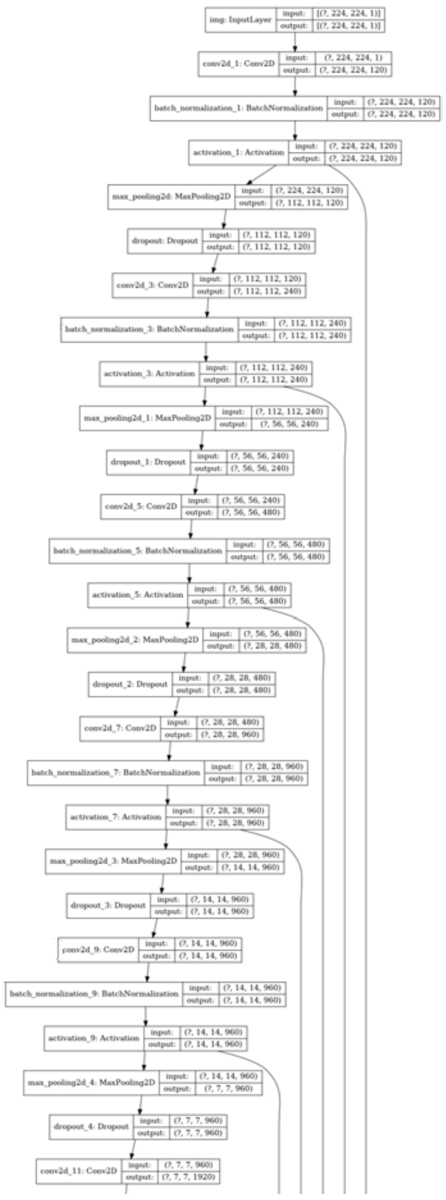
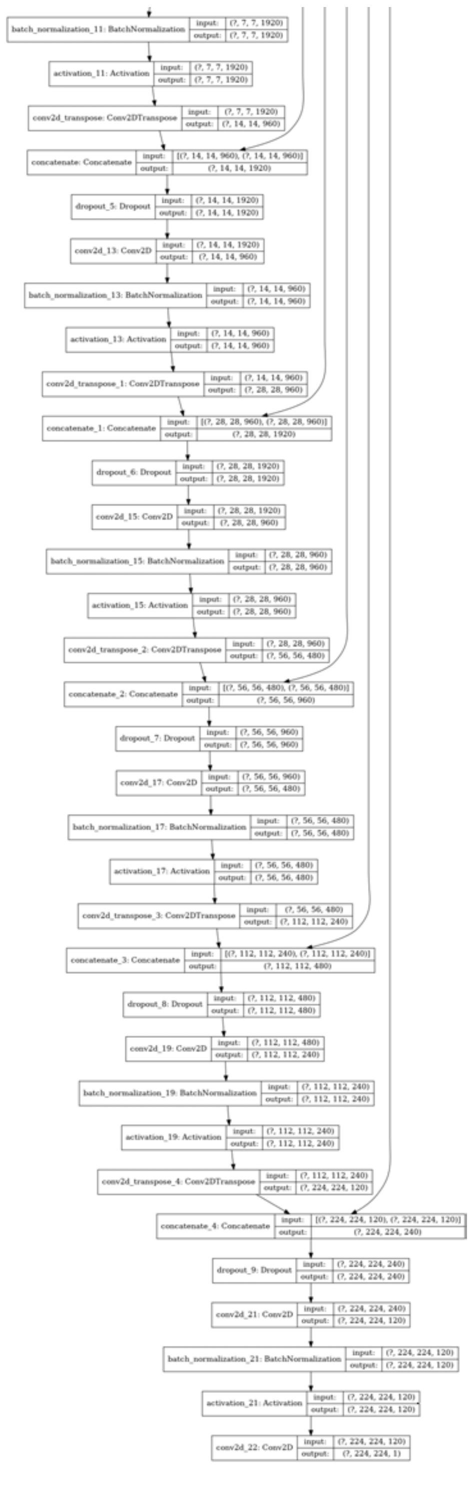

img: InputLayer | input: [(?, 224, 224, 1)] | output: [(?, 224, 224, 1)]
conv2d_1: Conv2D | input: (?, 224, 224, 1) | output: (?, 224, 224, 120)
batch_normalization_1: BatchNormalization | input: (?, 224, 224, 120) | output: (?, 224, 224, 120)
activation_1: Activation | input: (?, 224, 224, 120) | output: (?, 224, 224, 120)
max_pooling2d: MaxPooling2D | input: (?, 224, 224, 120) | output: (?, 112, 112, 120)
dropout: Dropout | input: (?, 112, 112, 120) | output: (?, 112, 112, 120)
conv2d_3: Conv2D | input: (?, 112, 112, 120) | output: (?, 112, 112, 240)
batch_normalization_3: BatchNormalization | input: (?, 112, 112, 240) | output: (?, 112, 112, 240)
activation_3: Activation | input: (?, 112, 112, 240) | output: (?, 112, 112, 240)
max_pooling2d_1: MaxPooling2D | input: (?, 112, 112, 240) | output: (?, 56, 56, 240)
dropout_1: Dropout | input: (?, 56, 56, 240) | output: (?, 56, 56, 240)
conv2d_5: Conv2D | input: (?, 56, 56, 240) | output: (?, 56, 56, 480)
batch_normalization_5: BatchNormalization | input: (?, 56, 56, 480) | output: (?, 56, 56, 480)
activation_5: Activation | input: (?, 56, 56, 480) | output: (?, 56, 56, 480)
max_pooling2d_2: MaxPooling2D | input: (?, 56, 56, 480) | output: (?, 28, 28, 480)
dropout_2: Dropout | input: (?, 28, 28, 480) | output: (?, 28, 28, 480)
conv2d_7: Conv2D | input: (?, 28, 28, 480) | output: (?, 28, 28, 960)
batch_normalization_7: BatchNormalization | input: (?, 28, 28, 960) | output: (?, 28, 28, 960)
activation_7: Activation | input: (?, 28, 28, 960) | output: (?, 28, 28, 960)
max_pooling2d_3: MaxPooling2D | input: (?, 28, 28, 960) | output: (?, 14, 14, 960)
dropout_3: Dropout | input: (?, 14, 14, 960) | output: (?, 14, 14, 960)
conv2d_9: Conv2D | input: (?, 14, 14, 960) | output: (?, 14, 14, 960)
batch_normalization_9: BatchNormalization | input: (?, 14, 14, 960) | output: (?, 14, 14, 960)
activation_9: Activation | input: (?, 14, 14, 960) | output: (?, 14, 14, 960)
max_pooling2d_4: MaxPooling2D | input: (?, 14, 14, 960) | output: (?, 7, 7, 960)
dropout_4: Dropout | input: (?, 7, 7, 960) | output: (?, 7, 7, 960)
conv2d_11: Conv2D | input: (?, 7, 7, 960) | output: (?, 7, 7, 1920)

Figure 52: Custom U-net model built for semantic segmentation of Buildings and Roads - Part 1



batch_normalization_11: BatchNormalization | input: (?, 7, 7, 1920) | output: (?, 7, 7, 1920)
activation_11: Activation | input: (?, 7, 7, 1920) | output: (?, 7, 7, 1920)
conv2d_transpose: Conv2DTranspose | input: (?, 7, 7, 1920) | output: (?, 14, 14, 960)
concatenate: Concatenate | input: [(?, 14, 14, 960), (?, 14, 14, 960)] | output: (?, 14, 14, 1920)
dropout_5: Dropout | input: (?, 14, 14, 1920) | output: (?, 14, 14, 1920)
conv2d_13: Conv2D | input: (?, 14, 14, 1920) | output: (?, 14, 14, 960)
batch_normalization_13: BatchNormalization | input: (?, 14, 14, 960) | output: (?, 14, 14, 960)
activation_13: Activation | input: (?, 14, 14, 960) | output: (?, 14, 14, 960)
conv2d_transpose_1: Conv2DTranspose | input: (?, 14, 14, 960) | output: (?, 28, 28, 960)
concatenate_1: Concatenate | input: [(?, 28, 28, 960), (?, 28, 28, 960)] | output: (?, 28, 28, 1920)
dropout_6: Dropout | input: (?, 28, 28, 1920) | output: (?, 28, 28, 1920)
conv2d_15: Conv2D | input: (?, 28, 28, 1920) | output: (?, 28, 28, 960)
batch_normalization_15: BatchNormalization | input: (?, 28, 28, 960) | output: (?, 28, 28, 960)
activation_15: Activation | input: (?, 28, 28, 960) | output: (?, 28, 28, 960)
conv2d_transpose_2: Conv2DTranspose | input: (?, 28, 28, 960) | output: (?, 56, 56, 480)
concatenate_2: Concatenate | input: [(?, 56, 56, 480), (?, 56, 56, 480)] | output: (?, 56, 56, 960)
dropout_7: Dropout | input: (?, 56, 56, 960) | output: (?, 56, 56, 960)
conv2d_17: Conv2D | input: (?, 56, 56, 960) | output: (?, 56, 56, 480)
batch_normalization_17: BatchNormalization | input: (?, 56, 56, 480) | output: (?, 56, 56, 480)
activation_17: Activation | input: (?, 56, 56, 480) | output: (?, 56, 56, 480)
conv2d_transpose_3: Conv2DTranspose | input: (?, 56, 56, 480) | output: (?, 112, 112, 240)
concatenate_3: Concatenate | input: [(?, 112, 112, 240), (?, 112, 112, 240)] | output: (?, 112, 112, 480)
dropout_8: Dropout | input: (?, 112, 112, 480) | output: (?, 112, 112, 480)
conv2d_19: Conv2D | input: (?, 112, 112, 480) | output: (?, 112, 112, 240)
batch_normalization_19: BatchNormalization | input: (?, 112, 112, 240) | output: (?, 112, 112, 240)
activation_19: Activation | input: (?, 112, 112, 240) | output: (?, 112, 112, 240)
conv2d_transpose_4: Conv2DTranspose | input: (?, 112, 112, 240) | output: (?, 224, 224, 120)
concatenate_4: Concatenate | input: [(?, 224, 224, 120), (?, 224, 224, 120)] | output: (?, 224, 224, 240)
dropout_9: Dropout | input: (?, 224, 224, 240) | output: (?, 224, 224, 240)
conv2d_21: Conv2D | input: (?, 224, 224, 240) | output: (?, 224, 224, 120)
batch_normalization_21: BatchNormalization | input: (?, 224, 224, 120) | output: (?, 224, 224, 120)
activation_21: Activation | input: (?, 224, 224, 120) | output: (?, 224, 224, 120)
conv2d_22: Conv2D | input: (?, 224, 224, 120) | output: (?, 224, 224, 1)

Figure 53: Custom U-net model built for semantic segmentation of Buildings and Roads - Part 2

Bibliography

1. Blaise Ngendangenzwa. Defect detection and classification on painted specular surfaces. Technical report, 2017.

2. Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark. Technical report, 2017.

3. SpaceNet. spacenet.ai – Accelerating Geospatial Machine Learning.

4. Cody Clifton. Commutativity in non-Abelian Groups. Technical report, 2010.

5. Robert Putnam. Air Force Future Operating Concept, 2015.

6. Xiongwei Wu, Doyen Sahoo, and Steven C H Hoi. Recent Advances in Deep Learning for Object Detection. Technical report.

7. Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference*, 2005.

8. Intel Integrated Performance Primitives. *Histogram of Oriented Gradients (HOG) Descriptor*.

9. Trevor et. all. Hastie. *Springer Series in Statistics The Elements of Statistical Learning*, volume 27. 2009.

10. Andrew Ng, Kian Katanforoosh, and Younes Mourri. Deep Learning Specialization.

11. Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss Functions for Neural Networks for Image Processing. nov 2015.

12. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, pages 234–241. Springer Verlag, may 2015.

13. Shruti Jadon. A survey of loss functions for semantic segmentation. pages 1–7, jun 2020.

14. Bowei Shan and Yong Fang. A cross entropy based deep neural network model for road extraction from satellite images. *Entropy*, 22(5):535, may 2020.

15. Francois Chollet. *Deep Learning with Python.* Manning, 1st editio edition, 2017.

16. H. Elbehiery, A. Hefnawy, and M. Elewa. Surface defects detection for ceramic tiles using image processing and morphological techniques. *Proceedings - WEC'05: 3rd World Enformatika Conference*, 5:158–162, 2005.

17. MathWorks. Types of Morphological Operations - MATLAB & Simulink.

18. Valerie Sessions and Marco Valtorta. THE EFFECTS OF DATA QUALITY ON MACHINE LEARNING ALGORITHMS. Technical report.

19. Wei Dai, Kenji Yoshigoe, and William Parsley. Improving Data Quality through Deep Learning and Statistical Models. *Advances in Intelligent Systems and Computing*, 558:515–522, oct 2018.

20. Mark Rijmenam. A Short History Of Big Data, 2013.

21. Olivier Pauplin, Praminda Caleb-Solly, and Jim Smith. INTERACTIVE parameter adaptation tool for image segmentation. *MCCSIS'08 - IADIS Multi Conference on Computer Science and Information Systems; Proceedings of Com-*

puter Graphics and Visualization 2008 and Gaming 2008: Design for Engaging Experience Soc. Interaction, pages 134–141, 2008.

22. J. Castillo-Navarro, N. Audebert, A. Boulch, B. Le Saux, and S. Lefevre. What data are needed for semantic segmentation in earth observation? In *2019 Joint Urban Remote Sensing Event, JURSE 2019*. Institute of Electrical and Electronics Engineers Inc., may 2019.

23. Rosa L. Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H. Ngo. Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, 12(1), 2012.

24. Abhishek Thakur and Rajeev Ranjan. *Image Segmentation and Semantic Labeling using Machine Learning*. PhD thesis, 2019.

25. Jaehoon Choi, Minki Jeong, Taekyung Kim, and Changick Kim. Pseudo-Labeling Curriculum for Unsupervised Domain Adaptation. *arXiv*, aug 2019.

26. Biplab Banerjee, Francesca Bovolo, Avik Bhattacharya, Lorenzo Bruzzone, Subhasis Chaudhuri, and B. Krishna Mohan. A new self-training-based unsupervised satellite image classification technique using cluster ensemble strategy. *IEEE Geoscience and Remote Sensing Letters*, 12(4):741–745, 2015.

27. Yue Zhang, Shun Miao, Tommaso Mansi, and Rui Liao. Unsupervised X-ray image segmentation with task driven generative adversarial networks. *Medical Image Analysis*, 62:101664, may 2020.

28. Karen Simonyan and Andrew Zisserman. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. Technical report, 2015.

29. Muneeb Hassan. VGG16 - Convolutional Network for Classification and Detection.

30. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

31. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778. IEEE Computer Society, dec 2016.

32. Seyma Tas. How To Evaluate Image Segmentation Models? — by Seyma Tas — Towards Data Science.

33. Frank Harrell. Damage Caused by Classification Accuracy and Other Discontinuous Improper Accuracy Scoring Rules — Statistical Thinking.

34. Gongping Yang, Xinjian Guo, Yilong Yin, Cailing Dong, and Guangtong Zhou. On the Class Imbalance Problem A Systematic Review of Finger Vein Recognition Techniques View project On the Class Imbalance Problem *. 2008.

35. Joffrey L. Leevy, Taghi M. Khoshgoftaar, Richard A. Bauder, and Naeem Seliya. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5(1):42, dec 2018.

36. Adrian Rosebrock. Intersection over Union (IoU) for object detection - PyImageSearch.

37. Nigel Parsad. Deep Learning in Medical Imaging V — by Nigel M. Parsad — Data Driven Investor — Medium, 2018.

38. Karlsruhe Institute of Technology. The KITTI Vision Benchmark Suite.

39. Leaderboard – Inria Aerial Image Labeling Dataset.

40. Kaggle. 2019 FIRE171 ASN11 Image Segmentation Challenge V2, 2019.

41. 2018 Data Science Bowl — Kaggle.

42. Severstal: Steel Defect Detection, 2019.

43. Ekin Tiu. Metrics to Evaluate your Semantic Segmentation Model — by Ekin Tiu — Towards Data Science, aug 2019.

44. Gabriela Csurka, Diane Larlus, and Florent Perronnin. CSURKA, LARLUS, PERRONNIN: EVALUATION OF SEMANTIC SEGMENTATION What is a good evaluation measure for semantic segmentation? Technical report.

45. Classification: Precision and Recall — Machine Learning Crash Course.

46. David M. W. Powers. What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes. mar 2015.

47. David G Lowe. Object Recognition from Local Scale-Invariant Features. Technical report, 1999.

48. Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. Technical report.

49. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. Technical report.

50. Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Lecture Notes in Computer Science (including subseries Lecture Notes in*

Artificial Intelligence and Lecture Notes in Bioinformatics), volume 3951 LNCS, pages 404–417. Springer, Berlin, Heidelberg, 2006.

51. Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 2011.

52. Qiang Fang, Dai Bing Zhang, and Tian Jiang Hu. Vision-aided localization and navigation based on trifocal tensor geometry. *International Journal of Micro Air Vehicles*, 9(4):306–317, dec 2017.

53. J. Beich and M. Veth. Tightly-coupled image-aided inertial relative navigation using Statistical Predictive Rendering (SPR) techniques and a priori world models. In *Record - IEEE PLANS, Position Location and Navigation Symposium*, pages 552–560, 2010.

54. Diomidis Katzourakis, Nikos I. Vitzilaios, and Nikos C. Tsourveloudis. Vision aided navigation for unmanned helicopters. pages 1245–1250. Institute of Electrical and Electronics Engineers (IEEE), jul 2009.

55. Amila Jakubović and Jasmin Velagić. Image feature matching and object detection using brute-force matchers. In *Proceedings Elmar - International Symposium Electronics in Marine*, volume 2018-Septe, pages 83–86. Croatian Society Electronics in Marine - ELMAR, nov 2018.

56. Vineetha Vijayan and Pushpalatha Kp. FLANN Based Matching with SIFT Descriptors for Drowsy Features Extraction. In *Proceedings of the IEEE International Conference Image Information Processing*, volume 2019-Novem, pages 600–605. Institute of Electrical and Electronics Engineers Inc., nov 2019.

57. David G Lowe. Accepted for publication in the. Technical report, 2004.

58. Extract histogram of oriented gradients (HOG) features - MATLAB extractHOGFeatures.

59. Johshua Sortino. Decoding the Confusion Matrix. Understand the Confusion Matrix and… — by Prateek Sharma — Towards Data Science.

60. Jason Brownlee. Multi-Label Classification with Deep Learning, aug 2020.

61. Alexander Teplyuk. PyTorch Starter (U-Net ResNet) , nov 2020.

62. Volodymyr Mnih. Machine Learning for Aerial Image Labeling. Technical report, 2013.

63. Leonid Ivanovsky, Vladimir Khryashchev, Vladimir Pavlov, P G Demidov, and Anna Ostrovskaya. Building Detection on Aerial Images Using U-NET Neural Networks. Technical report.

64. El Mendili Lamiae, Chougrad Mehdi, Sebari Imane, and Puissant Anne. Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks. *ISPRS Journal of Photogrammetry*, (2), 2017.

65. Aziguli Wulamu, Zuxian Shi, Dezheng Zhang, and Zheyu He. Multiscale Road Extraction in Remote Sensing Images. *Computational Intelligence and Neuroscience*, 2019, 2019.

66. Ir Michael Heffels and Joaquin Vanschoren. Aerial Imagery Pixel-level Segmentation. Technical report.

67. S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer Berlin Heidelberg, 1990.

68. Funda Gunes. Why do stacked ensemble models win data science competitions? - The SAS Data Science Blog, may 2017.

69. Donald Venable and John Raquet. Large scale image aided navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 52:2849–2860, 12 2016.

70. D. Lee. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. 2013.

71. Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. Technical report.

72. Yuxuan Liu, Fan Mo, and Pengjie Tao. Matching Multi-Source Optical Satellite Imagery Exploiting a Multi-Stage Approach. *Remote Sensing*, 9(12):1249, dec 2017.

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

1. REPORT DATE *(DD–MM–YYYY)* 26–03–2020	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2019 — Mar 2021

4. TITLE AND SUBTITLE Surface Defect Detection in Aircraft Skin & Visual Navigation based on Forced Feature Selection through Segmentation	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Hussey, Tyler, Capt, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-21-M-049
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Visual inspection of aircraft skin for surface defects is an area of maintenance that is particularly intensive for time and manpower. One novel way to combat this problem is through the use of computer vision and the advent of Artificial Neural Networks(ANN), or more specifically, semantic segmentation via Convolutional Neural Networks (CNN). The research in the paper explores the use of semantic segmentation of aerial imagery as a way to force feature selection onto key areas of an image that might be more likely to correspond under seasonal variations. Utilizing feature selection and matching on the masked aerial image and the satellite image produces a set of reliable key points that can be used for camera pose estimation and visual navigation.

15. SUBJECT TERMS

artificial neural network (ANN), convolutional neural network (CNN), deep learning, defect detection, visual navigation, multi-source feature matching

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Robert Leishman, AFIT/ENG
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	105	(937) 255-3636 x4755; robert.leishman@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18