Design and Exploration of a Computer Vision Based Unmanned Aerial Vehicle for Railroad Health Applications

Jay Matthew Frauenthal

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
In partial fulfillment of the requirements for the degree of

Master of Science

In

Mechanical Engineering

Kevin B. Kochersberger, Chair

Devi Parikh

Steve Southward

May 19, 2015

Blacksburg, Virginia

Keywords: UAS, Computer Vision, Railroad Health Monitoring, Image Transformation, Defect Detection

Design and Exploration of a Computer Vision Based Unmanned Aerial Vehicle in Railroad Health Applications

Jay Matthew Frauenthal

(ABSTRACT)

Railroad tracks require consistent and periodic monitoring to ensure safety and reliability. Unmanned Aerial Vehicles (UAVs) have great potential because they are not constrained to the track, allowing trains to continue running while the UAV is inspecting. Also, they can be quickly deployed without human intervention. For these reasons, the first steps towards creating a track-monitoring UAV system have been completed.

This thesis focuses on the design of algorithms to be deployed on a UAV for the purpose of monitoring the health of railroad tracks. Before designing the algorithms, the first steps were to design a rough physical structure of the final product. A small multirotor or fixed-wing UAV will be used with a gimbaled camera mounted on the belly. The camera will take images of the tracks while the onboard computer processes the images. The computer will locate the tracks in the image as well as perform defect detection on those tracks.

Algorithms were implemented once a rough physical structure of the product was completed. These algorithms detect and follow rails through a video feed and detect defects in the rails. The rail following algorithm is based on a custom-designed masking technique that locates rails in images. A defect detection algorithm was also created. This algorithm detect defects by analyzing gradient data on the rail surface.

Acknowledgments

This is an accumulation of two years of my work, but it would not be what it is without the people who helped along the way. They are what made the difference. Dr. K is a quality advisor and person. He encouraged me and directed me in my graduate studies all along the way. My time would be much less pleasant and more difficult if not for him.

The graduate students who work in the USL make the lab what it is. They are a group of smart, talented, encouraging individuals who believe in each other and push each other's limits. They have always been willing to help with anything that comes up. I am proud to call them friends. Thanks Gordon Christie, Haseeb Chaudhry, Scott Radford, Evan Smith, Jp Stewart, Alfred Mayalu, Kenneth Kroeger and Justin Stiltner.

My academic success might not have been possible without the people in other parts of my life encouraging me as well. I am who I am because of my parent's love, teaching and guidance through my whole life. They always put me before themselves and pointed me towards the only thing that really matters: God. I can do anything through Christ who strengthens me. Which leads me to the most important of all. The greatest gift God has given me. My best friend and wife, Lauren. You are the biggest encouragement of all. You are beautiful inside and out. Thank you for sticking with me and trusting God. I look up to you and will keep on looking up to you. I love you Lauren.

All photos by author, 2015

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

According to the Federal Railroad Administration (FRA) Office of Safety Analysis [1] 8,188 out of 24,223 train accidents were caused by track issues between January 2004 and January 2014. The FRA understands that this is a problem and is actively tightening rail inspection requirements. In the past, the FRA regulations required tracks to be inspected for physical defects at specified intervals. These intervals were determined by a maximum number of days and tonnage that could be hauled over a section of track before inspection. In some cases, this could result in multiple track inspections per week. Now, in the past year, the FRA issued new regulations requiring performance-based inspection methods. These generally require more frequent inspections than previous standards [2] .With inspection standards tightening, there is more demand for new and improved track inspection methods and tools.

While inspection standards are tightening, Unmanned Aircraft Systems (UAS) are becoming more readily available. The Federal Aviation Administration (FAA) recently decided upon six test sites around the United States for the purpose of safely integrating UAS into the US airspace. Specifically, these test sites are helping find solutions for "sense and avoid, command and control, ground control station standards and human factors, airworthiness, lost link procedures and the interface with the air traffic control system" [3]. This constitutes a big step in the direction of commercial access to airspace and it illustrates the recent national trend.

Both of these trends point towards a UAS-based track inspection system. UAS offer a great deal of benefits over standard track inspection systems. First, they have the potential for eliminating train intervention time. UAS can be flown at a safe altitude eliminating the need to interrupt train movement. This results in much cheaper inspection because the inspection does not affect the efficiency of the railroad. They are also very versatile. A complete UAS would require little to no human intervention during the inspection process. The UAS could simply take data at a pre-determined time and the results could be sent to the user.

UAS are a promising new direction in railway monitoring that has been explored very little. This work is the beginning of research and testing to enable UAS to be used for railroad inspection and monitoring.

1.2 UAS Niche in the Market

One very essential aspect of the project is finding the best way that UAS could be used for track inspection. To begin, market research was conducted to determine the ways in which companies inspect tracks. Currently, the FRA requires periodic visual inspections and periodic internal defect searches. It is possible that the FRA will change the requirements in the future to allow UAS with cameras to substitute for visual inspections, but this cannot be counted on in the near future. Many railroads also have track inspection vehicles that are equipped with a variety of high-resolution sensors. Some of these vehicles can run 30 mph or more and analyze large portions of track at a time. Some disadvantages of these vehicles include that they are very

expensive and the track cannot be used while they are inspecting. Many railroads do not own a track inspection vehicle because they are so expensive.

Through this research a gap was found in the market that UAS will fit into. Disaster response is an area that is lacking in the railroad health monitoring market. All of the defect detection methods are either slow or use the tracks for transportation. UAS can travel fast and work independently of track conditions. So, after a tornado, hurricane, earthquake, severe temperature shift, etc. the UAS could be sent to run a sweep over the tracks and verify that there is no damage from the event. This would constitute detecting objects, kinks, severe cracks, and so on in the track.

1.3 Vision for Final Implemented System

This section explains the long-term vision for this project. This is the vision for the final implemented disaster response system. The research that is outlined in this paper is the first step towards implementing this long-term vision.

This work accomplishes the first steps towards designing and implementing a system that can automatically detect and analyze railroad tracks for defects in the track structure and geometry. Eventually, this system will work without human interaction, but will instead notify an operator the location of any defects and the type of defect at the location. The system will display an image of the location of track that contains a defect along with an estimate of the type of defect. This will allow an operator to take appropriate action.

The core of this final implemented system will be a small-scale unmanned plane (on the order of 50in wingspan), a camera and a small, lightweight processor. The plane will be able to fly long distances on the order of tens of miles or greater. Along the way, the camera will be taking sequential images of the track. The camera will be mounted on a gimbal that is set to keep the camera at a nadir view of the tracks. This simplifies the computer vision algorithms because the rails would remain approximately parallel to each other. A camera-based system will be implemented as opposed to a laser-based system due to cost and weight advantages.

GPS will be the primary localization method used for navigation of the plane, but GPS is not a robust localization method for this application. Due to the design of commercial GPS, it has an inherent inaccuracy of about 3 meters. [4] Also, for any number of reasons, the number of connected satellites could be lost and the localization accuracy could become worse. These possibilities of inaccuracy are unacceptable in this application because the tracks must be in the camera's view at all times. Also, there is a high likelihood of obstacles such as trees and poles being located near either side of the track. Therefore, these GPS inaccuracies could cause a defect to be missed or the system to crash. For these reasons, a secondary navigation controller will be implemented. The computer vision algorithms will be the core of the controller. Once the tracks are located in an image, the location of the tracks relative to the vehicle can be found. The onboard computer can then create a new desired trajectory and fine-tune the vehicle's localization relative to the tracks. In this way, the GPS can bring the plane in the vicinity of the tracks and the camera can keep the plane over them.

In order for this analysis to be possible, the on-board computer must be able to locate the tracks, calculate the new trajectory and send flight commands in at least 1Hz speeds, ideally on the order of 30Hz. So, the algorithms must be quick enough to locate the track at these high speeds: however, the defect analysis can be done at slower speeds. The defect analysis can be post-processed for detailed investigation.

1.4 Description of Specific Work, Objectives and Goals

This research did not attempt to fully accomplish the previously outlined system. It simply set out to validate the concept and begin developing and testing the first portions of the system. Research was done on the portions of the project that seemed to have the greatest risk of failure. So, if those portions could be implemented, it is likely that the whole system could be created. The following were the goals for the project:

- Implement and test a working prototype rail detection algorithm
- Implement and test a working prototype rail head anomaly detection algorithm
- Create an image library that would be used for future testing

The vision system was most likely the point of greatest risk in the project. Using computer vision systems to analyze railroad track is a very new field of research. Other work has been completed in this area, but the camera is always mounted to a railcar of some sort. By using railcars, lighting conditions can be controlled as well as the position of the camera relative to the tracks. Compared to these systems, the proposed vision system will be much more complex. Lighting conditions can only be controlled by what time of day the vehicle is flown and the camera position relative to the rail cannot be fixed. So, a large portion of this work was done with the goal of developing a rail finding computer vision system. This rail finding system needed to be very robust because if the rails cannot be found in an image, there is no chance of following or finding defects in the track.

This paper also shows a variety of other analysis and research with regards to this project. Different types of defect detection were implemented and tested. An analysis of how much camera resolution is needed for the implemented defect detection method was performed. For future testing of the system, a broad, diverse image library was developed. This combination of tasks would give a good base for the development of a UAS with the purpose of railroad defect detection.

Chapter 2 Literature Review

The following sections detail the background research that was conducted for this project.

2.1 Track Inspection

2.1.1 Required Inspection Methods

The FRA requires periodic track inspections [5] which are outlined in the Track Safety Standards (TSS) document. The frequency and type of inspections vary based upon track class and usage. The following sections outline the different inspection methods that are required by this document.

2.1.1.1 Visual

The FRA requires regular visual inspections either on foot or from a hi-rail vehicle. [5] A single inspector can inspect up to two tracks at a time and up to two inspectors can inspect from the same vehicle at a time. Inspectors must be qualified personnel and are expected to look for all defects included in the TSS document.

It has been shown that many defects can be detected purely by eyesight. Some of the defects that can be detected visually are as follows: corrugations on rail head, rail anomalies, surface rail cracks, wheel burns, insufficient ballast, tie cracks, missing fasteners, missing tie plates, missing joint bars, obstacles on the track and warped rails. The significance of the defects that can be visually detected ranges from minor to severe.

On the other hand, loose bolts on joint bars can rarely be detected through vision alone: on a high rail vehicle or otherwise. However, an experienced inspector can detect a loose joint from the sound it makes as the vehicle traverses the joint. At other times the inspector must use indirect visual cues to detect issues. Detecting an ineffective fastener or a rail seat abrasion is very difficult to see directly, but as a loaded rail car traverses such a defect, there may be a tendency for the rail to roll slightly. Once the roll is observed, the inspector must view the area more closely to determine the specific defect.

The frequency of visual inspections is dependent on the class of track and track usage. Visual inspection requirements range from once monthly to three times per week. Tracks must also be visually inspected in the case of special occurrences such as fire, flood, severe storm or another occurrence that may have damaged the track. [6]

2.1.1.2 Internal Inspection Techniques

The FRA also requires internal inspection techniques, but the specifics as to which techniques are applied are not outlined. Dependent on the class of track, the FRA requires up to two annual internal inspections or one inspection for every 40 million gross tons (mgt) that passes over the rail [6]. Any of the following internal inspection techniques would meet these requirements.

There are many inspection techniques that qualify as internal techniques. The most common are Ultrasonic and induction methods. Ultrasonic techniques usually consist of a mechanical means of striking the rail, thus inducing sound waves that travel through the rail. Receivers then pick up the sound waves. Anomalies in the rail cause certain anomalies in the sound waves and algorithms determine the specific defect from the sound wave anomalies. Ultrasonic methods are typically implemented with a push-cart that is attached to the rails. The operator walks along the track collecting and recording data. Ultrasonic methods are very reliable at detecting defects within the rail, but do not work well to locate defects on the surface, near the surface or on the edges of the rail. They are also sensitive to flaw orientation, so they provide details about the type of flaw. Ultrasonic methods give a very accurate view of the center of the rail, but do not profile the edges well. [7] [8] [9] [10]

Induction testing is a typical internal inspection technique that is used to complement ultrasonic testing. These tests are normally done by placing an electric coil with a current running through it near the track. Then, when the magnetic field interacts with the rail, an eddy current is produced. This current can be monitored by a receiver and interpreted to locate defects. Induction methods are complementary to ultrasonic methods because they work well at detecting surface and near surface defects. They also are more robust at profiling the entire rail width. [10] [11] [12]

There are also less typical, more specialized internal testing methods. Some examples are magnetic particle inspection, radiography and electromagnetic acoustic transducer methods. These methods are used to analyze bolt holes, weld areas or perform more detailed inspection. [13]

2.1.2 Laser-Based Systems

The most commonly used laser-based system is the Track Geometry Measurement System (TGMS). These systems, which are mounted on a rail inspection car, use non-contact optical laser sensors located close to each rail. Many different geometry parameters can be measured: gage, alignment, vertical surface profile, cross-level, twist, warp, and curvature. All these parameters are important because track geometry generally characterizes the health of the track structure. Flawed geometry could be a result of failed ties, failed rail fasteners, degraded ballast or many other defects. Another advantage of TGMS is that measurements are taken when the rail is under load. There are a number of manual techniques that measure track geometry, but they do not offer insight into how the track is deformed by a load. [14] [5]

2.1.3 Rail Inspection Cars

Rail inspection cars are the industry standard for high mileage rail inspection. These cars are manufactured in three styles: self-propelled, towed and hi-rail. [15] Hi-rail vehicles are vehicles that can operate both on rail tracks and conventional road. In each case, the inspection cars are equipped with computers and sensors to perform automated testing. Typically, these cars perform ultrasound and induction testing of the rail at speeds up to 30 mph. [13] Many cars are also equipped with lasers to model a geometry profile of the track. These lasers commonly make up a TGMS, as explained previously. Cameras are also typically used and mounted below the

rail inspection car to take images of the track. These images are rarely used to automatically detect defects, but are instead used to create a visual record of the rails. This allows an operator to compare the automated defect information to an image at any location they desire. These cars were originally designed by Dr. Elmer A. Sperry who founded Sperry Rail Service. Sperry Rail Service is still the industry leader of rail inspection cars. [16]

There are multiple downfalls to rail inspection cars. First, they occupy paths that could otherwise be used for commercial traffic. They also require highly skilled staff to operate the vehicles. Finally, only relatively large railway organizations can afford these cars. [17] These are some of the motivations for an alternative inspection platform.

2.1.4 Current Uses of Computer Vision in Track Inspection

Using computer (machine) vision for track inspection is a relatively new practice. This type of research was widely unheard of until around the year 2000: Beena Vision began developing and manufacturing products in the year 2000 and the FRA began developing a machine-vision-based joint-bar inspection system in 2002. Cameras were used in track inspection before this time, but they were not used to directly determine flaws. Images were simply taken so that an operator could look back at the track at a specific location and time. In fact, this is still the case today. Very few track inspection systems exist that contain a computer vision element because the concept is so new.

The Sawadisavi paper [18] does a great job of outlining the accomplishments of companies and universities in terms of computer vision track inspection. Some of the leaders in computer vision track inspection research are University of Illinois [18] [19] [20], MERMEC Inc. [21], Georgetown Rail [22], University of Central Florida [23] [24] and Beena Vision [25]. Overall, these organizations have been very successful. They have been able to create systems that can be operated in the range of 30 to 99 mph, they have been able to detect tie type and movement, inspect and classify rail fastenings, measure rail gap, check for ballast irregularities, locate vegetation, inspect surface cracks and more. Some of these systems are still in development and some have been shown to have very high accuracy for certain types of detection (85-95%).

To this date, almost all track inspection cameras are mounted on a hi-rail or a rail car. The camera takes pictures of the track as the vehicle drives along the track. One reason this type of system works well is that lighting conditions can be controlled and the position of the camera relative to the track is approximately known. Both of these conditions greatly reduce the complexity of any computer vision algorithms.

Different techniques have been implemented for conducting track inspection. Texture is often used to distinguish ballast, rails and ties. Template matching is often used to detect specific objects such as tie plates. Edge and line detectors have been used to locate the boundaries of object as well as locating rail anchors. These techniques are complemented by the proximity of the camera to the tracks, the controlled lighting conditions and the controlled camera position.

2.2 Common Computer Vision Concepts and Techniques

The following sections detail computer vision concepts and techniques that were used in this project.

2.2.1 Gradients

Similar to a derivative, the gradient represents a slope. This slope is the tangent at a certain point of a function. More precisely, the gradient points in the direction of the greatest rate of increase of the function and its magnitude is the slope of the graph in that direction. [26] Gradients are used to measure the amount of change on a surface in a desired direction. [27] With regards to image processing, gradient is the directional change in intensity or color in an image. Since the intensity function of a digital image is only known at discrete points, derivatives of this function cannot be defined unless assumptions are made. The most common way to approximate the image gradient is to convolve an image with a kernel, such as the Sobel or Prewitt operator. [28] [29]

2.2.2 Color Spaces

A color space maps a range of physically produced colors to an objective description of color sensations registered in the eye. The human eye uses three different kinds of cone cells to perceive color. Each different type of cone cell is sensitive to a different wavelength spectrum: short (S, 420-440 nm), middle (M, 530-540 nm) and long (L, 560-580 nm). This theory is the basis upon which all color spaces were created; color can be broken up and displayed with a combination of three wavelength spectrums. So, in principle, three parameters can describe any color sensation. These three parameters can be classified into a 3-dimensional space: LMS. All color spaces are based upon this concept and are an attempt to classify a range of light wavelength spectrum. [30]

2.2.2.1 XYZ

The XYZ color space encompasses all color sensations that an average person can experience. When judging relative brightness, humans tend to perceive light within green parts of the spectrum as brighter than red or blue light of equal power. The XYZ model capitalizes on this fact by defining Y as luminance. In the same way, Z is very similar to blue stimulation, or the S cone response. Finally, X is a linear combination of nonnegative cone response curves. The XYZ color space is based upon the research of W. David Wright and John Guild. In the 1920s they independently conducted a series of experiments on human sight that laid the foundation of this color space. [30] [31] [32]

2.2.2.2 sRGB

The sRGB color space is an additive color space that is defined by red, green and blue. This is the most widely used color space, particularly in consumer grade cameras, HD video cameras and computer monitors. It is considered adequate for most consumer grade applications, but is not typically used by professionals because many highly saturated colors are left out. Having

most devices use the same color space is convenient because images do not need to be converted from one color space to another. [33] [34] [35]

2.2.2.3 HSV

HSV (Hue-Saturation-Value) is one of the most common cylindrical coordinate representations of points in an RGB color model. This model was developed in the 1970s for computer graphics applications. HSV is commonly used today in color pickers, image editing software, image analysis and computer vision. The goal of HSV was to create a model that was more intuitive to manipulate. Hue is a rough measure of color without regard to the color's attributes: intensity, brightness, etc. Saturation is colorfulness relative to brightness and value is a measure of brightness. These representations allow users to easily edit colors. For example, if a brighter color is desired, the user can simply increase the saturation. If a different color is desired, the user can simply change the hue value. This makes editing more intuitive and also makes it easier to locate certain traits. [36] [37]

2.2.2.4 Lab

Lab is a color opponent space. Color opponent theory suggests that there are three components: red versus green, blue verses yellow and black versus white. Dimension L is the lightness (black versus white) component while a and b are the color opponent dimensions: a roughly represents redness versus greenness and b roughly represents blueness versus yellowness. The Lab color space represents a much wider range of wavelengths than most other spaces: some even outside of human perception. For this reason, data is either lost or extrapolated when converting from sRGB to Lab. [38] [39]

2.2.2.5 CMYK

This color space is the most common space used in the printing process because it describes what kinds of inks need to be applied to produce a certain color: Cyan, Magenta, Yellow and black. The K in CMYK stands for Key and it represents the amount of black ink to use. Key is used because using black ink is cheaper than creating black with a combination of the other three. This is an additive color model that is opposite of RGB. In RGB, white is the combination of all primary colored lights while in CMYK black is the combination of all primaries. [40] [41]

2.2.3 Transforming Images

Transforming images, also known as image warping, is commonly used in image processing and forming mosaics. Arguably the most common application is creating panoramic images. To do this, images must be warped and stitched together to make a larger, seamless image.

2.2.3.1 Homography

The first step in transforming an image is creating a homography matrix. A homography is a 3x3 matrix that represents the transformation from one point in an image to a corresponding point to in another image. The 9 different values in the homography make up the different possible

transformations that can be made to an image: rotation, translation, aspect, affine and projective. A homography is made up of the components shown in the following equation:

$$H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \qquad \textit{Equation 2.1}$$

where a, b, d and e make up a combination of rotation, aspect and affine components, c and f make up the translation, and g, h and i make up the perspective components. This matrix is typically calculated using the following equation:

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = H * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \textit{Equation 2.2}$$

where x' and y' are point coordinates in one image that correspond to x and y in the other image. The homography is calculated by inserting a number of point pairs into both sides of the equation and solving for H. At least 8 total points are typically used. [42] [43]

2.2.3.2 Homogeneous Coordinates

To properly create a homography matrix, the input points must first be converted to homogeneous coordinates as shown in Equation 2.2. Typical Cartesian coordinates have the flaw of not being able to represent points at infinity. So, operations such as dividing by zero are unable to be completed. On the other hand, homogeneous coordinates do not have this disadvantage. Operations of this nature sometimes are necessary when calculating a homography matrix and homogeneous coordinates are required. Homogeneous coordinates (x_1, x_2, x_3) of a finite point (x, y) in the plane are any three numbers for which the following equations hold true:

$$\frac{x_1}{x_3} = x \qquad \textit{Equation 2.3}$$

$$\frac{x_2}{x_3} = y \qquad \textit{Equation 2.4}$$

[44] [45]

2.2.3.3 Transformation Implementation

Once a homography is calculated, it is not as simple as plugging the image points into Equation 2.2. If this is done, it is highly likely that the transformed pixel will land on a location 'between' two pixels. This location 'between' pixels cannot be assigned a value, therefore the color must be distributed between neighboring pixels. Another technique for solving this problem is called inverse warping. Inverse warping is done by first taking the pixel location to be 'filled in' and plugging it into Equation 2.2. This will almost certainly land on a location in the original image 'between' pixels. Then, the color to be 'filled in' can be calculated by taking the proportion of the colors around the 'between' location. Doing this process will allow an image to be transformed using a homography matrix.

2.3 Other Common Methods Used

2.3.1 Normalized Difference Vegetation Index

Scientists have tested and verified a method to determine the health of plants. This method is called Normalized Difference Vegetation Index. The technique is based upon which wavelengths of light are reflected from plants. The pigment in plant leaves, chlorophyll, strongly absorbs visible light other than green (from 0.4 to 0.5 and 0.6 to 0.7 μm) for use in photosynthesis. The cell structure of the leaves, on the other hand, strongly reflects near-infrared light (from 0.7 to 1.1 μm). [46] The health of the plant and its number of leaves affect the reflected wavelengths as well. In general, if there is much more reflected radiation in near-infrared wavelengths than in visible wavelengths, then the pixel is likely to represent vegetation. This concept is illustrated in Equation 2.5:

$$NDVI = \frac{(NIR - VIS)}{(NIR + VIS)} \qquad \textit{Equation 2.5}$$

where NIR is a pixel value from a near infrared camera and VIS is the value of the same pixel taken from a visible spectrum camera. If this calculation yields a value near 1, then it is likely that the pixel represents live vegetation. On the other hand, if this calculation is near zero it is likely that the pixel is of dead vegetation or not vegetation at all. [47] [48]

Chapter 3 Image Library Design

This chapter addresses the design of the image library used to test and evaluate the rail and defect detection computer vision system. The quality of the system is dependent upon the quality of the image library it was designed with and tested against. The goal was to obtain a library that was diverse, but still contained relevant and unique images.

The algorithms and techniques outlined in this paper were tested using only the first image set and the video as detailed in section 3.1.6. The first image set is the ideal situation, so these images were used to design and test this work's algorithms. Some algorithms were also tested with the video. If the algorithm would work in these ideal situations, this proves the concept of the final system.

3.1 Library Images and Methods Used to Obtain Them

As stated previously, a diverse library was desired. So, images were taken from 3 different cameras, at 5 different altitudes and from 4 different aerial vehicles. They were also taken at 3 different locations and 5 different days/times of day. Table 3.1 contains all of the details of how the images were taken. The reasoning and details behind each set of images can be found in the following sections. The total image count in the library is 106.

Table 3.1 This table shows the details of how images in the image library were obtained.

Set	1	2	3	4	5	Video
Vehicle	Turnigy H.A.L Quadcopter	ARF MXP230	SIG Rascal	Taken by hand	3DR Spektre Prototype	Iris +
Camera	Raspberry PI camera module	Canon A810	Canon A810	Canon A810	GoPro Hero 3+ Black	GoPro Hero 4 Silver
Resolution	2MP	4MP	4MP	4MP	12MP	0.5MP
Mounting Style	Vibration-Isolated	Hard-Mounted	Hard-Mounted	N/A	Stabilized/ Vibration Isolated	Stabilized/ Vibration Isolated
Vehicle Speed	~1 m/s	~1 m/s	~15 m/s	0 m/s	~2 m/s	~4 m/s
Altitude	~10 meters	~15 meters	~30 meters	~15 meters	~15 meters	~10 meters
Time of Day	10 am	5pm	5 pm	12 noon	10 am	10 am
Weather Conditions	Calm and sunny	Windy and sunny	Calm and sunny	Calm and cloudy	Calm and sunny	Calm and sunny
Number of Images	20	15	34	19	18	707

3.1.1 Image Set 1

Figure 3.1 is an example of an image in set 1. This set of images was designed to be as simple as possible for an algorithm to work on. These images were taken at a relatively low altitude with a vibration-isolated camera. The flight was taken on a stable quadcopter on a sunny day with low wind speeds. All of these factors create a bright, crisp image of a track with very little distortion.

This set of images is the most extensively used throughout this work. Any proof-of-concept studies were conducted with this set and any extensive testing began with this set. If an algorithm worked, it worked best on this data set. If it did not work on this data set, then most likely did not work on any of the others either. So, showing that an algorithm worked with this set indicated that the algorithm would work under ideal conditions.

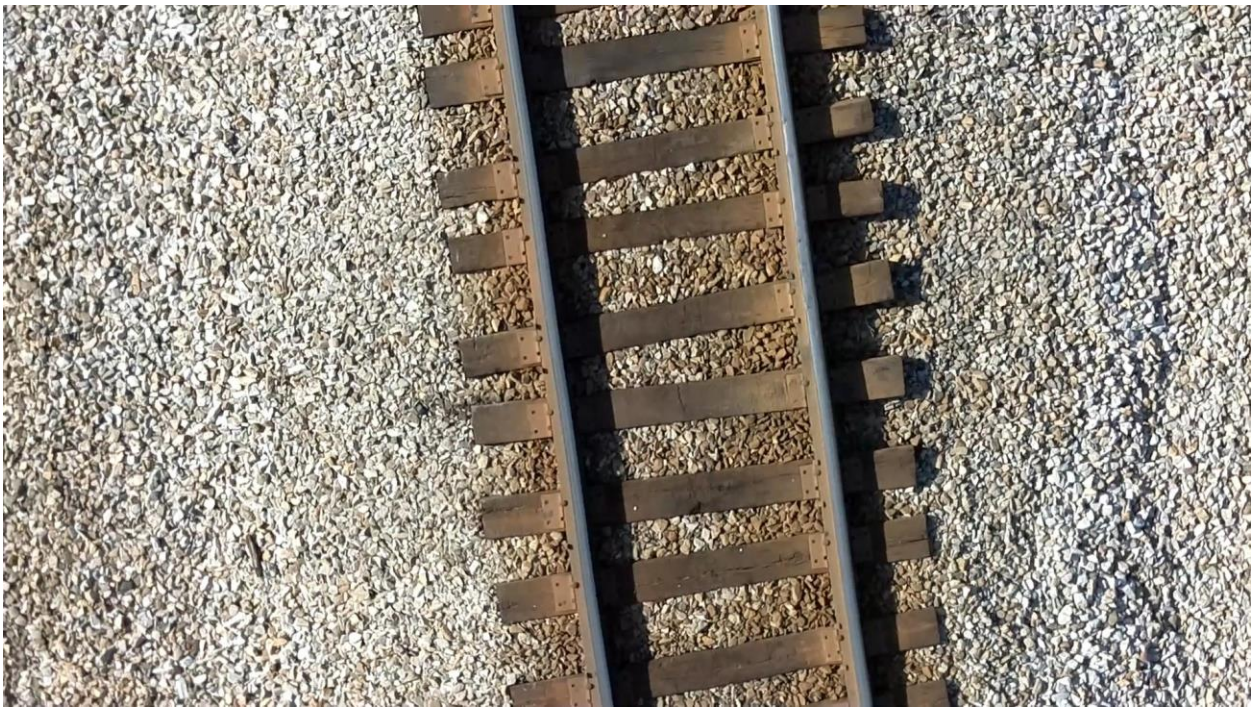

Figure 3.1 This is an example of an image taken with set 1 as outlined in Table 3.1.

3.1.2 Image Set 2

Figure 3.2 is an example of an image in set 2. This set was designed to be similar to the images in set 1, but to be a little more difficult. These images were taken over the same run of track, but on a vehicle that was more unstable and on a windier day. This caused the images to stray from an ideal nadir view and the quadcopter to stray from flying directly over the tracks. So, more distortion is present and the rail position moves around in the image more severely.

Figure 3.2 This is an example of an image taken with set 2 as outlined in in Table 3.1.

3.1.3 Image Set 3

Figure 3.3 is an example of an image in set 3. This set was also designed to be similar to set 1, but different variables were changed than that of set 2. Similar to the previous sets, the same stretch of track were used on a sunny day. This time however, the vehicle flew at a much higher altitude and at a faster speed.

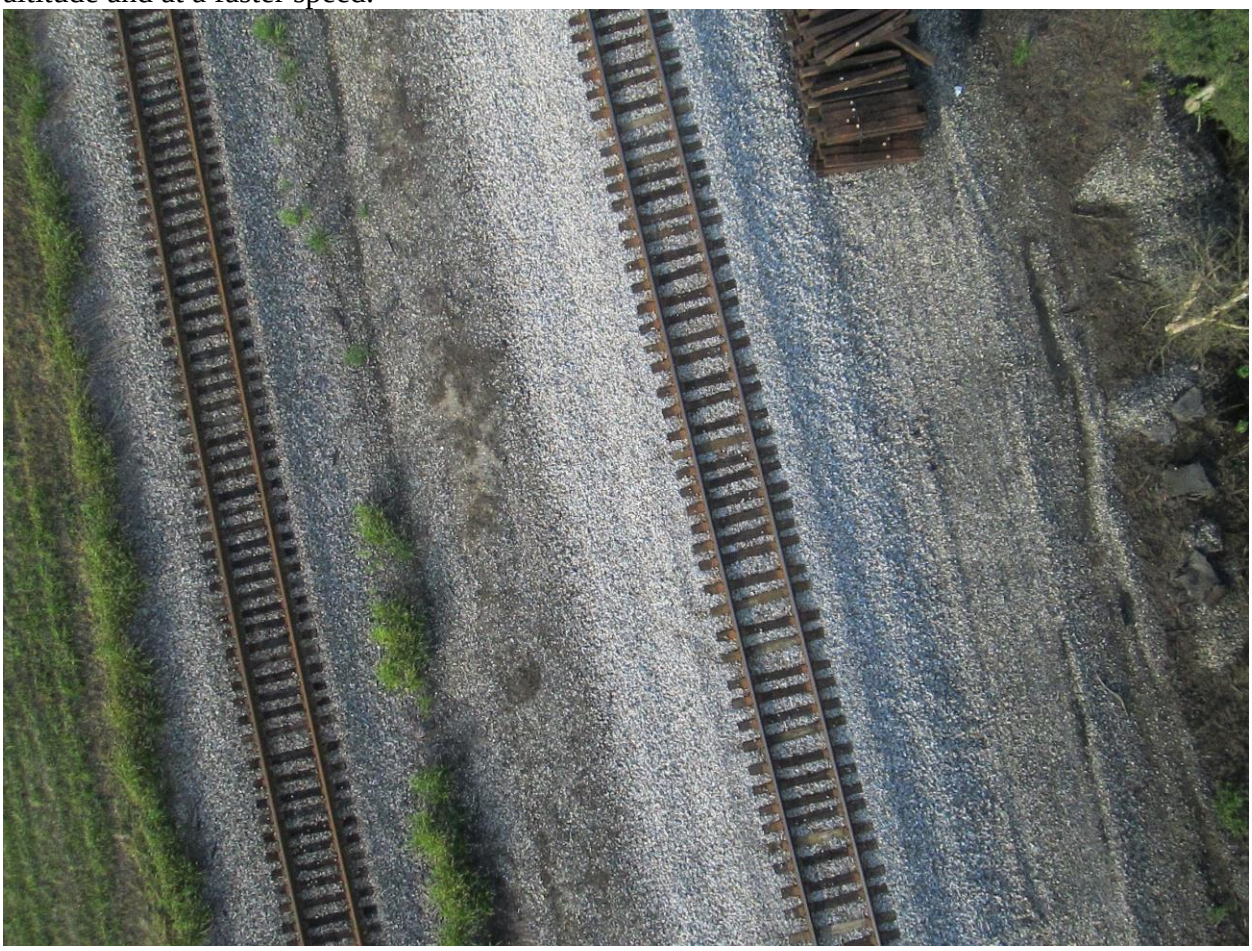

Figure 3.3 This is an example of an image taken with set 3 as outlined in Table 3.1

3.1.4 Image Set 4

Figure 3.4 is an example of an image in set 4. This set of images is very different from the first three sets. This is the first and only set that was taken by hand as opposed to from an aerial vehicle. These are images of a rail yard which has multiple walking bridges spanning the width of the yard. The pictures were taken by walking along the bridges. The tracks in these images range from fully functioning and used regularly to broken down and retired. Also, due to the acquisition method, there is perspective distortion which causes the rails to stray from parallel in the images.

Figure 3.4 This is an example of an image taken with set 4 as outlined in Table 3.1

3.1.5 Image Set 5

Figure 3.5 is an example of an image in set 5. This set of images is the most unique of all data sets. These were taken with a different vehicle/camera system at a different location and at a different resolution. As shown in Figure 3.5, parts of this set contain images of track on a bridge. This offers a unique challenge to algorithms because there are many lines that run parallel to the rail and have similar color to the rail.

Figure 3.5 This is an example of an image taken with set 5 as outlined in Table 3.1

3.1.6 Video

Figure 3.6 is a frame in a video sequence taken using a Go-Pro Hero 4-Silver camera. The camera was mounted on a 2-axis gimbal to an Iris+ Quadcopter. This video was taken over the same section of track that Image Set 1 was. The frame size of the video is 854x480 pixels, it was taken at 29 frames/second and there are a total of 707 frames. The video was taken to test how well the algorithms work frame-to-frame. This is useful, for example, to determine if the rail detection algorithm can stay with the rails over time.

Figure 3.7 is an example of an undistorted version of Figure 3.6. Removing the lens distortion of the video greatly decreases the complexity of the problem. This is a simple conversion, so it would be practical to undistort the images in real time.

Figure 3.6 This is an example of a frame taken from the rail video.

Figure 3.7 This is the image in Figure 3.6 with the lens distortion removed. This was accomplished using standard image calibration techniques.

3.2 Truth Masks for Library Images

It is essential that the location of important track features within the image library are known. This aids development and allows verification of the accuracy of algorithms to be possible. So, a method was developed to distinguish rail pixels from non-rail pixels. This was done manually by selecting the four corners of each rail and saving the information. This information was used to create a database of masks that correspond to the rail dataset. Figure 3.8 is an example of such a mask. The masks are the same size as the image it was made for. Then, each pixel is set to either zero or one: zero indicating that it is not a rail and one indicating that it is a rail. Then, rail pixels can easily be extracted by searching for any pixels with a label of one.

Masks were only created for rails because this work focuses on locating rails and locating defects on the rails. This same concept could be used for ties, tie plates or any other part of a track to be analyzed.

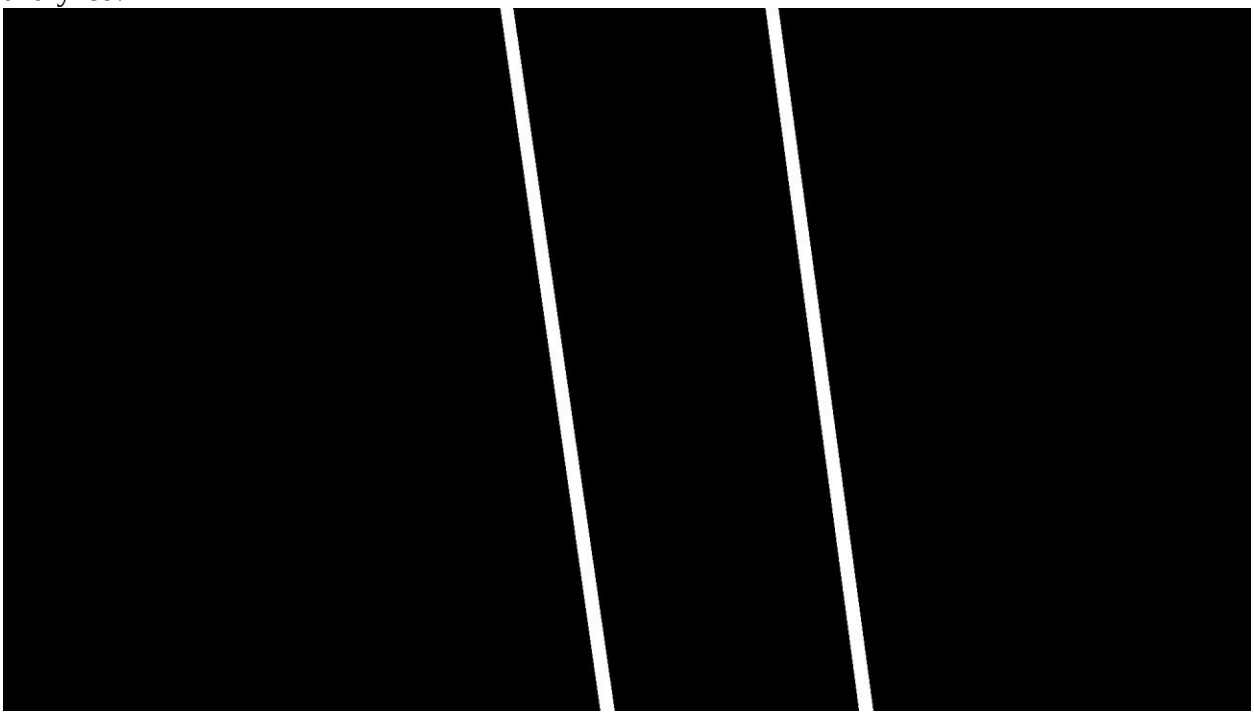

Figure 3.8 This is an example of a mask that was manually created to distinguish rail pixels from other pixels in the image. This mask corresponds to the image in set 1 shown in Figure 3.1.

Chapter 4 Mask Bank Rail Detection Technique

The mask bank technique is a method of locating railroad tracks in images taken from an aerial vehicle. The idea is to use previously known traits of a rail to locate rails in new images. This is done by analyzing specific portions of an image to see if its characteristics match that of a rail. The analyzed areas are the expected size and shape of a rail: rectangular, run from top to bottom of the image, certain width according to vehicle altitude. These pre-defined areas are stored in masks. So, the algorithm cycles through certain masks and compares each one to known rail characteristics. When a mask-area's features are very similar to the known characteristics, the algorithm concludes that the mask falls on top of a rail. This technique is explained in detail in the following sections.

4.1 How the Technique is Used

I'll begin explaining the technique by illustrating how it is used. There are two different phases when using the technique to locate rails: training and detection. The training phase is needed to calibrate the system to the specific traits of the rails being analyzed. The step-by-step process of how to train the system is detailed below:

1) Go to desired track location
2) Raise vehicle to approximately 10 meter altitude
3) Fly over ~40 meters of track and take at least 20 images
4) Retrieve vehicle and the images from the camera
5) Manually select the rail locations in the 20 images
6) Allow algorithm to calculate histograms and standard deviations using the true rail locations

This process creates a histogram which quantifies the type and amount of color that the rails contain and the average standard deviation of the rail pixel values. Both of these pieces of information are used as truth data in the detection process. Once the truth data has been created, the system is ready to be used. It has been shown to work using the following process:

1) Go to track location where training data was created within 2 months' time at the same time of day
2) Raise vehicle to approximately 10 meter altitude
3) Fly over ~80 meters of track and take images at least every 8 meters
4) Retrieve vehicle and images from the camera
5) Manually select rail locations in the first image
6) Allow algorithm to run on the images

So, the system is able to detect rails over an 80 meter span at least. Also the algorithm will work 2 months after training data creation as long as the weather and time of day are similar. This list also highlights the shortcomings of the algorithm. It has only been tested using these parameters: use within 2 months' time, 80 meter span, etc. This 80 meter span included the initial 40 meter training span. So, further testing would be needed to prove any further functionality. Also, currently the algorithm doesn't work in real time. During testing, the process of locating the

tracks was always slower than the frame rate of the camera. These upgrades would make the system even more useful.

4.2 Reasoning behind the Strategy

This technique is based upon masks for many different reasons: masks are a quick way to access image information, they are versatile and they allow the algorithm to be easily upgraded. Masks are essentially pointers that show the processer which pixels to look at. In this case, the algorithm was programmed in Matlab and the masks are stored as binary matrices. This method utilizes a Matlab technique called logical indexing which is the fastest way to retrieve information from a matrix. So, in Matlab you can use the following equation to access certain pixels from a matrix:

$$Desired\ Pixel\ Values = Image(mask) \qquad \textit{Equation 4.1}$$

where Image is an image matrix and mask is a set of desired pixels.

Masks are also very versatile. They can be made into any shape or size. This trait is used to take advantage of known traits of rails in images. Ideally, rails are straight and rectangular in the image. This occurs if the image is taken at a nadir point of view of the tracks and there is no lens distortion. Both of these can be controlled using hardware or software. Also, since the vehicle will be flying along the tracks it is safe to assume that the rails approximately run from top to bottom in the image.

The most significant aspect of this method is the upgradability. Currently, color profile and standard deviation features of rails are used. These two features have been found to be significant at this time, but ideally the algorithm would be able to detect any rail on any track in any location. The mask method makes adding other features extremely simple. The exact set of pixels that need to be analyzed is already distinguished from the rest of the image, so only one change would need to be made: calculate a different feature on those pixels.

4.3 Color Plane Analysis

The first step in designing the mask bank was to determine which features best distinguished rails from other areas of the image. One common distinguishing feature in computer vision is color. This study was an attempt to determine certain color traits that constitute a rail in general. If rails were found to be consistently made up of certain colors, then this information could be essential to locating a rail in an image. To the human eye, the rails tend to be a shade of silver and shiny in nature. This study attempts to determine these factors about rails in a more quantitative and scientific way.

4.3.1 Types of Color Information Collected and Method for Doing So

Masks were used to locate the color information for the rails. The method for obtaining these masks was explained in section 3.2. The masks were an easy way to distinguish ‘rail’ pixels from ‘not rail’. The goal was to determine as much information as possible, so information on many different types of color spaces was collected: RGB, HSV, Lab, XYZ and CMYK. The details of each color plane can be found in section 2.2.2.

This large quantity of color information was then organized into manageable chunks. This data was separated into 5 different sets, one for each image set in the image library and then placed the data into histograms. Then a histogram for each unique combination of color plane and image set was created: blue in image set 1, hue in image set 4, etc. Each histogram contained 255 bins, one for each possible value of color. The bins were filled in the following way:

- Locate a rail pixel in the desired image set and color plane
- Determine the value of the pixel, ex: 54
- Increment the corresponding histogram bin by 1, ex: increment bin 54 by 1.
- Repeat for every rail pixel in the image
- Normalize the histogram
- Average the histograms for all of the images in the desired set

Figure 4.1 shows how a histogram is laid out. Each bin on the x-axis corresponds to a possible pixel value: integers from 0 to 255. The height of each bar corresponds to how many pixels contain that value. This histogram design captures the overall color traits in an area; in this case, a rail. Once the histogram is created, it needs to be normalized. The purpose of normalization is to allow comparison between different size areas. The overall profile is kept while the number of pixels that created the profile is eliminated. Normalization was done using the following equation:

$$normVal = \frac{curVal - minVal}{maxVal - minVal} \qquad \textit{Equation 4.2}$$

where curVal is the current histogram value being normalized, minVal is the minimum y-value in the current histogram and maxVal is the minimum x-value in the current histogram. These values are displayed in Figure 4.1. Figure 4.2 is an example of a histogram plot after normalization.

The process resulted in 80 histograms for rails and 80 histograms for non-rails, given that 16 color planes and 5 image sets were analyzed. Figure 4.3 shows all 10 histograms plots for the red color plane. Appendix A: Color Plane Histograms shows all of the histogram plots for all of the color planes analyzed.

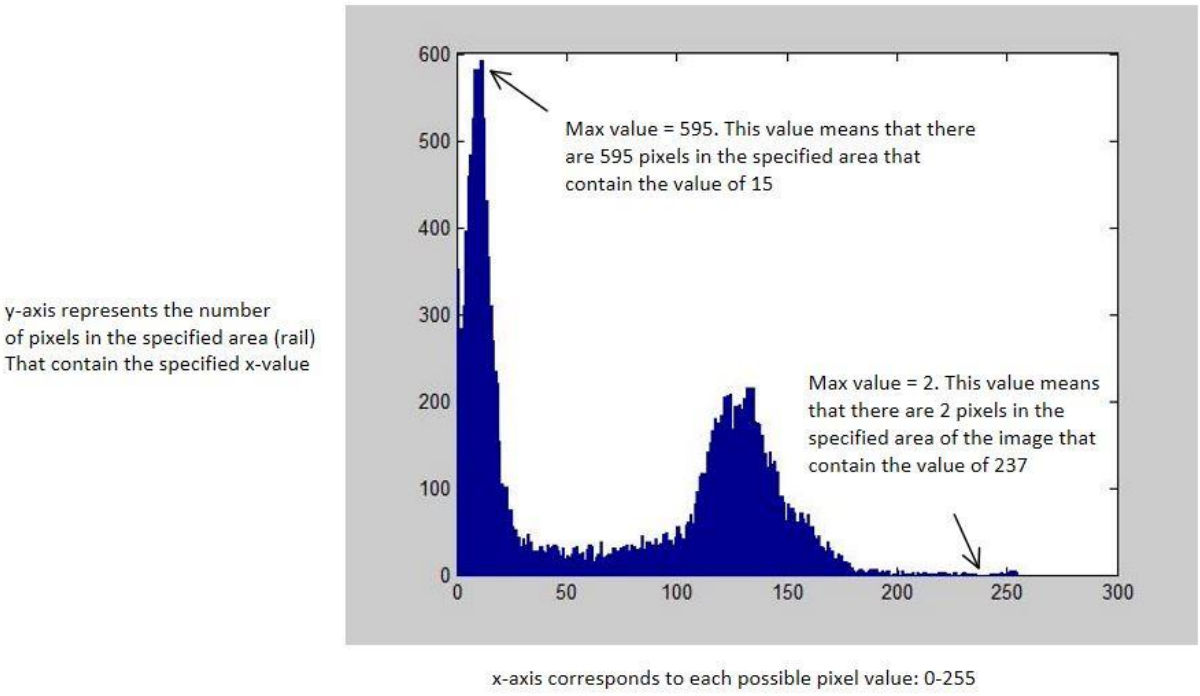

Figure 4.1 This plot is an example of a rail histogram before normalization. The max and min value pointed out would be used to normalize every column.

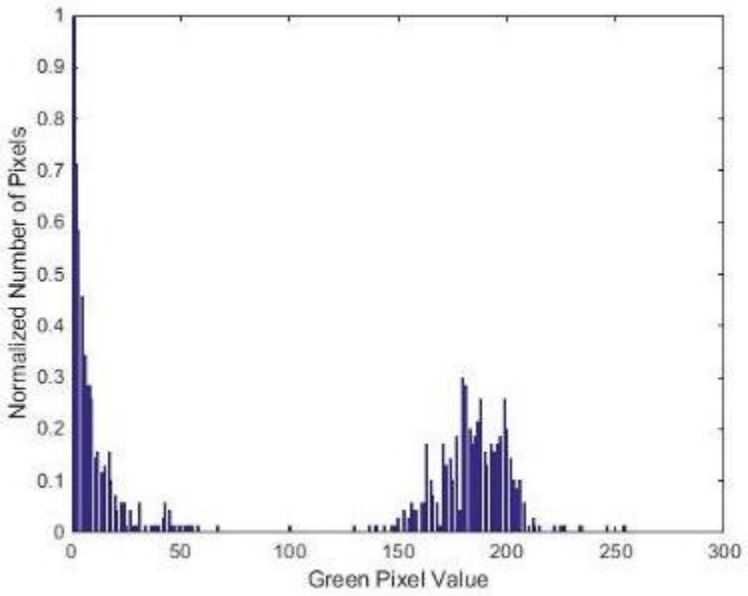

Figure 4.2 This is an example of a normalized histogram for the green color plane of a rail.

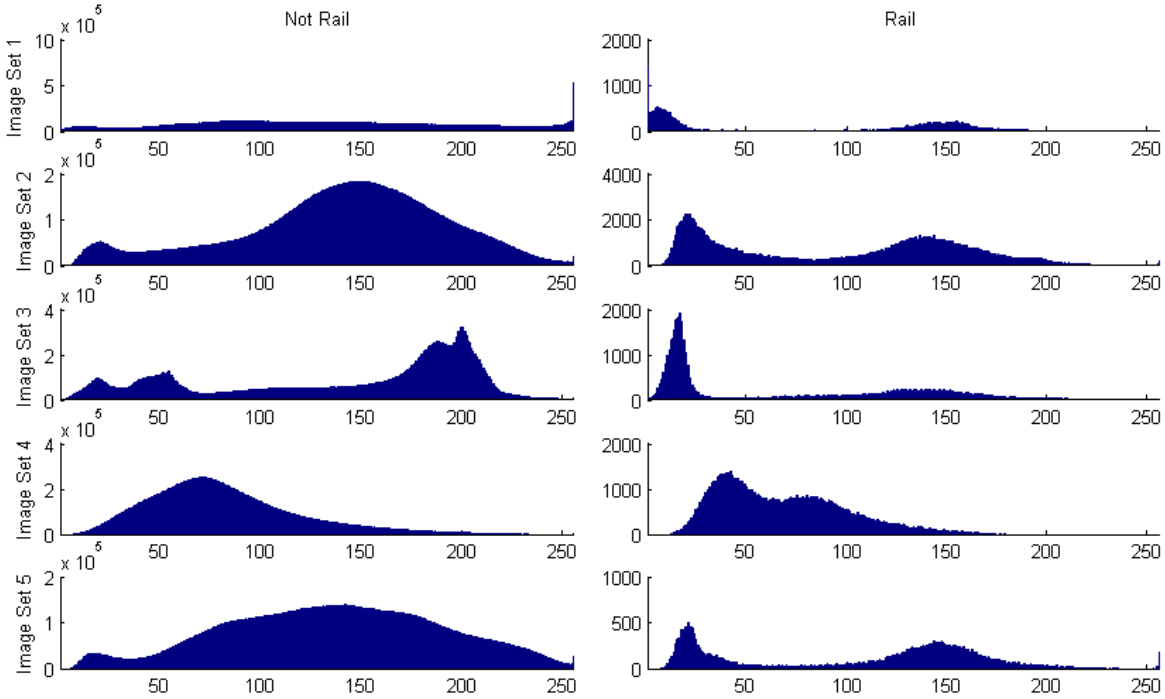

Figure 4.3 The histograms created using the red plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of red and the y axis represents the number of rail pixels that contain that value of red. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

4.3.2 Analyzing Color Information

The color histograms that were created were only analyzed visually. There are machine learning techniques, such as a Support Vector Machine (SVM), that could be used to determine important features more accurately. These techniques were not employed and are classified as future work that could be done.

Finding a threshold range that all rails fell within is ideal. If a range could be found, then it would be very simple to find the rails by thresholding out any pixels in the image that are outside that range. This would be ideal, but no such threshold was found. Instead, the sharpest consistent contrast between rails and the rest of the images was the overall profile of the R, G and B color planes. The histograms for the red color plane for rails and not rails in all image sets are shown in Figure 4.3. There is not a color range that distinguishes rails from the rest of the image, but the color profile that it produces is distinct and somewhat consistent across different image sets. These results were used in the mask bank technique which will be explained in detail.

4.4 Mask Bank Technique Details

4.4.1 Mask Bank Design

As explained initially, the mask bank technique uses a database of masks to locate rails in an image. The mask bank is made up of a large number of masks that represent possible rail locations. Figure 4.4 is an example of a mask in the mask bank. The bank contains masks that are ± 20º from vertical and all possible locations along the width of the image. These are the most likely rail locations, since the vehicle will be flying along the tracks. Each mask is a binary matrix that is the same size as the image being searched. Pixels that are a value of '1' in the bank are the potential rail location and pixels that are a value of '0' are disregarded. The algorithm works by comparing the traits of the current set of pixels that are a value of '1' to the traits that rails are known to have.

Figure 4.4 This is an example of one of the masks in the mask bank. The masks are created to be the same size as the rail image. It consists of all zero values with a band of ones.

4.4.2 Features Used

The traits/features that were used to locate rails were the red color profile and the standard deviation. The details of this decision can be found in section 4.6. The red color profile used was the average red color profile for the specific image set as explained in section 4.3. A histogram was found that characterized the profile of red rail pixels over the course of image set 1. The average standard deviation of the rail pixels was recorded at the same time. By then calculating the same traits for the mask pixels, the results could be compared to the true values. If the values were very similar, it's concluded that the set of pixels was a rail.

4.4.3 Determining Threshold Values

Ideally, the algorithm would be able to detect any rail in any image. To do this, the algorithm would have to have extensive knowledge of the traits of rails. This problem is complex due to a wide variability in brightness, weather, track material, surrounding environment etc. The training method explained in section 4.1 is the first step towards implementing this algorithm. After analyzing the results of the color plane study, it was decided that the red color plane would be the best for the analysis. Any color space other than RGB would require extra processing and the other color spaces did not present any advantage. So, the red color space was chosen.

This training method allows an operator to train on a small stretch of track and analyze a larger stretch of track on a later date. It has been shown to work within 2 months of training with similar weather conditions, time of day and hardware setup. Even with this training technique, the algorithm requires a calibration step to account for slight differences from day to day and flight to flight. Calibration is the reason why the operator must manually select the rail locations in the first image. Once these are selected, the algorithm calculates normalized histograms of the current day's rails as detailed in section 4.3.1. Then the calibration threshold values are created using the following Matlab equations:

$$redScore = \sum| trainingHist - calibrationHist| \quad \textit{Equation 4.3}$$

$$stdScore = std(railPixels) \quad \textit{Equation 4.4}$$

where trainingHist is the histogram calculated using the red color plane of the training data rails, calibrationHist is the histogram calculated using the red color plane of the calibration rails and railPixels are the red color plane pixels of the calibration rails. The difference is taken between the training data and the calibration data to determine the expected return value if a mask fell over a rail. This is a preliminary calculation used in the following threshold equations:

$$redThreshold = floor(redScore) + 2 \quad \textit{Equation 4.5}$$

$$stdUpperLimit = ceil(stdScore) + 2 \quad \textit{Equation 4.6}$$

$$stdLowerLimit = floor(stdScore) - 2 \quad \textit{Equation 4.7}$$

A factor of safety value of 2 is added to reduce the number of false negatives. This value was chosen through trial and error, so further research is necessary to refine this value.

The details of this process were determined while performing Video Testing as detailed in section 4.8. The process could be shown to work in this way because the video was taken 2 months after the training data. The first image in the GoPro video was the training image. First, the rail location in this image was selected manually. Then, the histogram corresponding to this data was compared with the training data histogram. During this process redScore was found to be 32.89 and redStd was found to be 14.95. So, redThreshold = 35, stdUpperLimit = 17 and stdLowerLimit = 12. Using these threshold values enabled the algorithm to locate the rails in the remaining 706 images in the video sequence. So, this method has been proven to work 2 months after the training data was created on a day with similar weather and at approximately the same time of day.

4.4.4 Locating Rails in New Images

Once the threshold values are created, they can be used to locate rails in successive images. This is done by cycling through masks and testing the mask pixels against the training data with the calibration threshold. The following IF statement is used to make the decision:

$$if \text{ (redScore < redThreshold \&\& stdScore < stdUpperLimit \&\& stdScore > stdLowerLimit)} \qquad \textit{Equation 4.8}$$

if the statement returns true, the mask is said to be in the same location as a rail, but if the statement returns false the area is considered a non-rail.

The number and location of the masks tested are dependent upon previous knowledge. If there is knowledge of the approximate location of a rail, then the entire image need not be searched. Rail location knowledge was utilized in the Video Testing section detailed in section 4.8. Theoretically, the rails should be in the same pixel location in every image because the vehicle would be flying exactly in-line with the tracks without turning or moving perpendicular to them. In the practical implementation inaccuracies from GPS and vehicle response time cause this to be false. Regardless, these inaccuracies would be small compared to the video's 30 fps speed. So, for this application, 36 masks were used that span across a 12 pixel span. This range correlates to a 2.4" span using the 10 meter altitude that the video was taken at. This was calculated using the following equation:

$$distance\ in\ pixels = \frac{rail\ width\ in\ pixels}{rail\ width\ in\ inches} * distance\ in\ inches \qquad \textit{Equation 4.9}$$

where rail width in pixels was approximately 17, rail width in inches was approximately 3.25 and distance in inches was 1. This is a simple proportion to correlate inches on the ground to pixels in inches. The input values were measured.

If there isn't any previous knowledge of rail location, then the entire image must be searched. This was done in sections 4.6 and 0. This was done using ~6,000 masks. These masks were more widely spaced than the previous example to be able to analyze a large area more quickly. This value was chosen arbitrarily. With more research, the relationship between mask spacing and area being searched could be determined.

4.5 Mask Bank Technique Assumptions

The concept behind this technique is to create a rail detector that is fast, robust and is easily modified. A few key assumptions were made to reduce the complexity of the problem: the rails are straight lines in the image, the rails are a consistent width in the image and across images, rails have similar feature traits image to image and rails run top to bottom in the image. Through testing and observation, these were found to be safe assumptions given certain requirements.

With regards to functioning tracks running in a straight line, the only factor that could cause the rails to be curved is lens distortion. Lens distortion can be easily removed at runtime with pre-calculated camera calibration information. Section 3.1.6 shows an image with lens distortion removed. With regards to curving tracks, the image must be on a small enough portion of the tracks to remain approximately straight. This can be done in a couple ways. The vehicle taking the picture can remain at a low altitude or a high altitude vehicle can be equipped with a zoom lens. Also, a non-zoomed image can be cropped to create 'straight' sections of track. The specifics of these limits were not determined and are classified at future work.

To keep the rails at an approximately consistent width in the image, the camera must be positioned at a nadir view of the ground and lens distortion must be eliminated. The camera can be kept at a nadir view with a standard 3-axis gimbal. These gimbals have become common place in the UAV community. Lens distortion was addressed previously.

To keep rails at consistent widths across images, the altitude/zoom of the camera must be kept constant. Standard UAV controllers have become accurate in terms of altitude due to on board barometers. These systems have been shown to be reliable at maintaining a certain altitude. The assumption that rails have similar feature traits image to image is the only risky assumption. The RGB color of rail pixels change between flights as shown in section 4.3.2. So, to some extent, this rail detection algorithm must be trained on a specific set of hardware, weather conditions, and track composition. This assumption was found to be valid on a similar set of tracks, with similar weather conditions and similar hardware.

Lastly, the assumption was made that the rails would run from top to bottom in the image. In its current state, the mask bank only contains masks that run from top to bottom in the image as opposed to coming from one side or the other. The assumption being valid is dependent on the quality of the vehicle controller. The assumption relies on the ability of the controller to be able to keep the tracks between the two sides of the images.

4.6 Algorithm Design for Accuracy

4.6.1 Initial Design

One key design feature for the mask bank technique is robustness. The algorithm had to be able to distinguish a rail from everything else in a reliable manner. The mask bank technique was designed so that the specific features analyzed could be changed. For example, the color of the current mask pixels could be compared with the known color of rails. Other possibilities include color patterns, texture, gradient magnitude or direction or more complex features.

The process for deciding which features to use began using the previous color plane work as detailed in section 4.3. This work showed that the rail's color profile was distinct from other parts of the image in certain color planes and spaces: all three planes in RGB, CMK in CMYK and L in Lab. So, the red plane was used. The red plane was chosen as opposed to the others because a conversion calculation was required for the other spaces.

To begin, an image from Image Set 1 was chosen. It can be seen in Figure 4.5. A series of masks were then run on this image and a histogram of pixels was created for each one. The score used to judge was based upon the following equation:

$$score = \sum | H_c - H_m | \qquad \textit{Equation 4.10}$$

where H_c is the histogram created from the current mask and H_m is the truth histogram that was found during testing. A score threshold of 13 was used, so any mask that returns a value less than 13 would be considered a rail. The results of this calculation can be seen in Figure 4.6. The white in this figure represents any area that the algorithm designated as rail and black represents not rail.

Obviously, this amount of accuracy is insufficient, so another feature was added: standard deviation. Calculating the standard deviation of an area of pixels in an image gives the rough magnitude of the texture or roughness of the area. The theory behind this decision is that a properly functioning rail is going to be very smooth and therefore the pixels will have a small standard deviation. In this way, the algorithm was modified so that the score must be less than 13 and the standard deviation must be less than 40 to be considered a rail. The results of this process are shown in Figure 4.7. This modification greatly increased the accuracy of the algorithm. Most of the remaining errors in Figure 4.7 are a result of the shadow next to the rail being picked up. To eliminate this, the standard deviation threshold was changed to a range. A standard rail should have some texture, just not very much. This change can be seen in Figure 4.8.

At this point the remaining errors were not completely dependent upon the type of features being calculated. Up to this point the mask bank being used contains masks at 15 pixel spacing between each other as opposed to every possible rail location (1 pixel spacing). In this case, the true rail location falls between the mask spacing, so each of the masks shown in Figure 4.8 all have approximately equal scores and standard deviations. So, a more accurate mask bank was created with a pixel spacing of 2. At the same time the formula for judging if a mask is a rail or not was slightly modified. Of the mask areas that are within the set thresholds, two masks are chosen which have the lowest color score. These modifications resulted in locating the two rails in the image as shown in Figure 4.9. This shows that the algorithm works in one case. This same process is tested more extensively in sections 4.6.2 and 4.8.

Figure 4.5 Image from Set 1

Figure 4.6 The results from running a mask bank on the image with a score threshold of < 13.

Figure 4.7 The results from running a mask bank on the image with a score threshold of < 13 and a standard deviation threshold of < 40.

Figure 4.8 The results from running a mask bank on the image with a score threshold of <13 and a standard deviation threshold of < 40 and > 38.

Figure 4.9 The results from running a mask bank on the image with a score threshold of <13 and a standard deviation threshold of < 40 and > 38 with a mask bank pixel increment of 2 and selecting the two with the lowest scores.

4.6.2 Image Set 1 Testing

After the functionality of the mask bank technique was designed in section 4.6.1, it was necessary to test it. This was done by scanning across all of the images in Image Set 1 using a mask bank with a pixel increment of 2. Then comparing the results with the truth masks that were created in section 3.2. The results of this process are detailed in Table 4.1. As shown, there were no missed rails and the accuracy of each hit is high.

Table 4.1 Table indicating the details of the mask bank testing results.

Image Set	Found Rails/ Total Rails	Mean Location Accuracy	Standard Deviation Location Accuracy	Mean Angle Accuracy	Standard Deviation Angle Accuracy
1: Rail = ~20 Pixels Wide	40/40	3.6 Pixels 0.59 Inches	2.5 Pixels 0.41 Inches	5 Pixels 0.27 Degrees	2.7 Pixels 0.14 Degrees

4.7 Algorithm Design for Speed

Most of the speed testing was done with a 15 pixel increment mask bank containing 6130 masks. This gives a general idea of the speed the final algorithm will run at. If increased accuracy is needed, a larger mask bank or more features will be required. Both of these accuracy improvements will also increase the processing time.

4.7.1 Original Implementation

Originally, each mask was created as needed. This was done during runtime because the entire mask bank was too big to fit into ram and be loaded into the Matlab environment. The 15 pixel spacing mask bank contained 6130 masks which span ± 20º from vertical and the width of the image. So, the algorithm worked as follows:

- Create mask
- Process mask pixels
- Delete mask
- Repeat

This algorithm was run for all of the masks in the mask bank on a single image and timed the results with Matlab 'Run and Time' feature. The aspects that took the most time are shown in Table 4.2. The algorithm took a total of about 39 minutes to run, which was too slow for this application. This spurred on further research to reduce computation time.

Table 4.2 This table shows the time results of the first iteration of the mask bank technique. The results shown are the functions that took the longest time to run.

Timed Function	Total Time	This function is used to…
Total time	2335 seconds	
inpolygon()	1570 seconds	create masks
meshgrid()	356.6 seconds	create masks
hist()	15.16 seconds	create histograms
std()	1.935 seconds	calculate standard deviation

4.7.2 Store Mask Bank on Hard Drive

It is important to note that most of the computation time in the first iteration was in mask creation (inpolygon() and meshgrid()). For this reason, a new implementation was made that used a pre-saved filter bank. Tips on implementing this process in Matlab were found at [49].

A matfile is a data type in Matlab that allows a user to access portions of the file. This data type was used to create a pre-process script to create the mask bank. This script created masks one at a time and loaded them into their respective position in the matfile. Then, when a mask was desired, the portion of the matfile could be loaded into the Matlab environment. The algorithm was modified to the following:

- Load 100 masks
- Process the 100 masks
- Delete the 100 masks
- Repeat for the next set of masks

Table 4.3 shows the results of timing this process for a single image. The total computation time was about 2.5 minutes, which is a significant improvement from 38.9 minutes.

Table 4.3 This table shows the time results of the second iteration of the mask bank technique. The results shown are the functions that took the longest time to run.

Timed Function	Total Time	This function is used to…
Total time	146.2 seconds	
matfile	33.42 seconds	obtain filters from the hard drive
hist()	14.93 seconds	create histograms
std()	1.837 seconds	calculate standard deviation

4.7.3 Narrow Search Area

The next technique used to reduce the computation time was to narrow the search area. Once the rails are found in the first image, there is a general idea of where the rails will be in the second image due to vehicle movement data and high frame rate. The accuracy of this knowledge is dependent on frame rate of the camera and the speed in which the vehicle is moving. So, a number of mask filters around the estimated location need to be analyzed. An arbitrary number of 30 was chosen. The final implementation might need more or less than 30, but this would give a general idea of how quick the algorithm would be. Table 4.4 show the timing results for this test. The total time of about 0.76 seconds is another significant improvement.

Table 4.4 This table shows the time results of the third iteration of the mask bank technique. The results shown are the functions that took the longest time to run.

Timed Function	Total Time	This function is used to…
Total time	0.761 seconds	
matfile	0.508 seconds	obtain the filters from the hard drive
imread	0.074 seconds	read the image to be searched
hist	0.066 seconds	create histograms
intersect	0.020 seconds	retrieve the desired set of masks

4.8 Video Testing

The purpose of testing the video set is twofold. First, it contains images from a different hardware setup and flight conditions from the set used to design the algorithm (Image Set 1), but on a connected portion of track. This will show that the algorithm is robust to changes in hardware and slight daylight conditions. Secondly, it will test the ability of the algorithm to follow tracks in a frame-by-frame manner.

4.8.1 Video Test

This test was conducted with a 2 pixel spacing mask bank. The 2 pixel spacing mask bank was used to obtain a high level of accuracy. Since this was a different hardware setup, the algorithm was calibrated with different thresholds. In this case the score threshold was < 35 and the standard deviation threshold was < 17 and > 12. This calibration data was determined using a

single image and would need to be completed for each hardware change. In this test, one rail was followed as a proof of concept. The same process would be completed in parallel for the second rail in the final implementation.

Table 4.5 shows the timing results of running the test on the Video Set. This table shows the speed results on a per-image basis, so it contains the average speed over all frames. The average speed was ~0.5 seconds which is 0.2 seconds quicker than the test conducted in section 0. The increase in average speed is most likely caused by the decrease in image size.

During this test, 640 of 707 rails were found, which constitutes approximately 90%. Rails are classified as found if over 50% of the mask is over top of the rail. Figure 4.10 is an example of a rail that was found and Figure 4.11 is an example of a rail that was not found. As shown in Figure 4.11, the algorithm classifies the shadow adjacent to the rail as the rail. All of the 67 rails that were classified incorrectly were misclassified in this way. The lower bound on the standard deviation is designed to correct for this error, but it did not in all cases. An interesting phenomenon was found when analyzing these errors. Out of the 707 frames, the errors did not begin until frame 600. This indicates that the fixed thresholds created with the original calibration became less accurate over time. In future implementations, this error can likely be fixed with a running threshold. The thresholds could be adjusted over time to accommodate slight changes in rail color and lighting.

Table 4.5 This table shows the average time to find the rail in each image in the Video Set. This set contains a total of 707 images.

Timed Function	Total Time	This function is used to…
Total time	0.532 seconds	
matfile	0.362 seconds	obtain the filters from the hard drive
hist	0.034 seconds	create histograms
imread	0.017 seconds	read the image to be searched
std	0.008 seconds	calculate standard deviation

Figure 4.10 Example of a rail in the Video Set test that was found

Figure 4.11 Example of a rail in the Video Set test that was not found

4.8.2 Real-Time Analysis

Through testing, the mask bank technique was shown to be fast, reliable and robust. If an approximate rail location was known, the technique consistently located a rail in ~0.6 seconds. If there is no previous knowledge as to where the rail is located, the technique ran in under 3 minutes. These are promising times, but they are insufficient for real-time operation. The video that was analyzed was taken at a frame rate of 29 fps, so the algorithm must be able to run at 29 fps (0.035 seconds per frame) to keep up real-time. So, with this current hardware setup, real-time rail detection/following could not be done on board the UAV; the video feed would have to be post-processed.

To allow real-time rail detection on the UAV, the frame rate of the camera and the speed of flight must be matched up with the speed of the algorithm. So, to simulate a slower frame rate, only portions of the Video Set were used: every other frame to simulate 15 fps, every third frame to simulate 10 fps, etc. The results of this study are detailed in Table 4.6. It can be seen that the algorithm runs slower as the video frame rate runs more slowly. This is because with a high frame rate, the algorithm can be more confident that the rail is in a similar location in the image. If the frame rate is low, the vehicle will have moved a greater distance and therefore the estimate of the rail location is more uncertain. As stated in Chapter 3, the vehicle speed was approximately 4 m/s when this video was taken. The last column of Table 4.6 shows the max speed the vehicle could move to allow the algorithm to keep up with the frame rate. Both of these speeds are impractical for fixed wing aircraft flight. Air speeds of 10 m/s minimum would be needed in the final implemented solution, so increases in algorithm speed will need to be made.

The theory behind the calculations in the last column of Table 4.6 is if the vehicle is flying more slowly, the movement of the rail image-to-image will be less and therefore the algorithm will speed up. This max speed calculation was made using the following equation:

$$max\ speed = \frac{Vehicle\ Speed}{fps_{Camera} \Big/ fps_{Alg}} \qquad \textit{Equation 4.11}$$

Another factor to consider in this study is risk. As the frame rate slows down, there is more risk of something being missed by the camera. So, the optimal frame rate is the quickest and also allows the vehicle to fly the fastest. From the results of the study, a clear cut best frame rate did not emerge. This study should be considered in the final application and a decision be made based on the final hardware and application.

Table 4.6 Results of the real-time analysis of the mask bank algorithm. The Video Set was modified to find the required frame rate for the algorithm to function properly. These speeds are significantly lower than the 10 m/s required for fixed wing flight. So, at this time images need to be post-processed.

Which Frames Used	Simulated Video Frame Rate (fps)	Algorithm Runtime Frame Rate (fps)	Max Speed Vehicle Could Fly at 10m Altitude (m/s)
All	29	1.88	0.26
Every Other	15	1.71	0.46
Every Third	10	1.47	0.59
Every Fourth	7.5	0.80	0.43
Every Fifth	6	0.92	0.61
Every Sixth	5	1.02	0.81
Every Tenth	3	0.65	0.87
Every Fifteenth	2	0.24	0.47
Every Thirtieth	1	0.23	0.93
Every Fortieth	0.75	0.15	0.81
Every Sixtieth	0.5	0.09	0.74

4.9 Future Work

There are many possible methods that could be employed to increase the computational speed of the algorithm. Currently, the algorithm was programmed and runs in the Matlab environment. It is well known that the Matlab environment should not be used if algorithm speed is a critical design factor. Speed could potentially increase by a factor of 10 if the algorithm was written in C++ or C and run on embedded hardware. The Matlab environment was chosen for its ease-of-use and familiarity. So, re-writing the algorithm in C would be worthwhile.

Another potential speed increase would be to utilize GPU technology. These processors are specially designed for parallel applications. The mask bank algorithm is currently running the same calculations on different image areas all in series. So, it's very well suited to be done in parallel.

Also, speed could be increased by refining the features analyzed in each mask area. Currently, standard deviation is being used as a feature. Calculating standard deviation requires an expensive squared operation, so using variance would accomplish the same goal with less processing. The color profile feature could be refined as well. A study could be done to determine which pixel values (0 – 255) best captured the overall traits of a rail. Then, the size of each histogram could be decreased to allow for faster computation.

4.10 Conclusion and Application

With the algorithm in its current state, a functioning rail detection vehicle could be implemented as detailed in section 4.1. This specific setup has been proven to work on an 80 meter section of track and could likely detect rails for a much further distance. This functionality is sufficient to prove the concept of a UAV rail detection system.

More research must be done to enable the rail detection system to work on a wider variety of rails. The algorithm is set up in such a way that features used in the detection can be easily changed depending upon the results of future research. For instance: if future research reveals that a certain feature is common throughout most rails, this feature can be added to the battery of features tested on each mask. Also, if more speed and less accuracy is needed, features can be removed from the algorithm just as easily.

The mask bank technique has shown to be fast, reliable and robust. This was shown through a battery of test and analysis. In an accuracy test, the algorithm located rails within an average of 0.59 inches and 0.27 degrees from the true rail location. In a speed test the algorithm was able to run at 1 fps with a video frame rate of 5 fps. Since the vehicle was flying at 4 m/s, theoretically a vehicle speed of 0.8 m/s would eliminate the frame rate disparity. These numbers give confidence that a real-time rail detection system could be implemented with further research.

Chapter 5 Defect Detection

The following sections detail methods used to detect defects and methods used to aid in the defect detection process.

5.1 Transforming the Rail Image

To aid in defect classification efforts, a process to transform any track image was created. This transformation re-orients the pixels in the image such that the rails are vertical and the tie orientation is maintained relative to the rails. This simplifies the defect classification process because traversal down the length of the rail can now be done by traversing a column of the matrix. Many other potential defect detection methods would be simplified as well such as: template matching, Hough transform or any other descriptor detector.

5.1.1 Process for Transforming the Rail Image

As detailed in section 2.2.3, four points are needed to classify the transformation. From previous work, the endpoints of the rails have been determined and they are shown in Figure 5.1. If all four of these points are used, an improper transformation results. As shown in Figure 5.2, the angle of the ties to the rails is not preserved.

To preserve the symmetry of the tracks, the line between the selected points must be perpendicular to the rails. Figure 5.3 shows an example of the proper transformation points. These points were selected by first analyzing the angle of the tracks relative to the image. If the tracks are tilted to the left, the bottom point of the left rail and the top point of the right rail is used and vice versa. The other two points are found by locating the line that passes through one of the endpoints and is perpendicular to the rail. Then, the intersection of this line and the rail is the desired point. Figure 5.4 shows an example of this process.

Using this process results in a properly transformed rail image as shown in Figure 5.5; the rails become vertical and the ties maintain their angle relative to the rails. This process was tested on over 100 rail images and did not result in a failure.

Figure 5.1 Rail image with rail line endpoints labeled. Each red dot illustrates an endpoint of a rail line.

Figure 5.2 Rail image that was improperly transformed. This transformation was done using the 4 endpoints as shown in Figure 5.1.

Figure 5.3 This is the same rail image shown in Figure 5.1 with the correct transformation points shown. Two points are endpoints of the rails and two points are at angles perpendicular to the rails and intersecting the endpoints.

Figure 5.4 This is an example of the transformation points being determined. The point in the blue circle is the intersection of the two red lines: the rail line and the line intersecting the right endpoint and perpendicular to the rail.

Figure 5.5 This is a track image transformed with the points shown in Figure 5.3. These points result in an accurate transformation.

5.1.2 Maintaining Information after Transforming the Rail Image

As explained previously, the rail locations must first be known to be able to transform the image. Once the image is transformed it is essential that the location information be preserved. To do this, the same homography matrix used in the image transformation can be used on the rail endpoints. The following process was used:

- Convert points to homogeneous coordinates
- Multiply homography by the points desired
- Convert points back to inhomogeneous coordinates
- Add an x and y shift factor to the x and y points respectively

The shifting factor is needed because the image is resized in the transformation process. The shifting factors account for the increase in image size.

5.1.3 Current Usefulness of this Technique

The current process is to orient the entire image so that the tracks are vertical and the ties are approximately horizontal. This process is not optimal in terms of speed. Ideally, only the portion of the image that's needed for analysis would be transformed. In this case, only the rails should be transformed and saved in their own matrix. This should be adjusted in further algorithm implementations.

The process of transforming rails does not necessarily increase the processing speed of the algorithm. It's possible, in some cases that transforming rails would increase the computational speed, but the real purpose of the process is to decrease the complexity of any future calculations. Any calculations would be simple to do with a square matrix. If the rails weren't transformed, a mask would need to be used to keep track of the exact boundaries of the rail.

The true advantage of the transformation has not yet been realized at this time, but this will simplify future analysis of the rail. In the future, templates could be stored of previously located and defined defects. For the templates to be compared against an unknown defect, either the unknown defect or the template will need to be transformed. It is easier to store a database of properly oriented templates than to store templates at unique angles and transform them each time they are used. Also, computational savings starts to become apparent when more and more templates are used. If many types of defects are being searched for, the following tradeoff is made: transform the rail once vs. transform every defect template.

5.2 Gradient Method for Defect Detection on Rail Head

Gradients are the backbone of method to detect defects in the rail head. The theory behind the research is that a pristine rail head would be free of any notable changes in gradient. So, if a change is gradient is detected, it gives confidence that it is either an anomaly on the rail surface or an object laying on the rail. This is assuming that there are no shadows cast over the rail. Shadows were considered a special case for this study and excluded. The following study was conducted assuming that the rail locations were known in the image.

5.2.1 Obtaining Rail Gradient Information

The first step that taken to obtain the rail gradient information was to transform the track image, so the rails became vertical. This process is explained in section 5.1. This was done to simplify any future steps. Transforming the rails to a vertical orientation allows easy traversal of the rail pixels along the rows and columns of the matrix. Figure 5.6 is an example of one such transformed rail. The rail location information was transformed as well and is also shown.

Once the image was transformed, it was trivial to extract the rail information. The rows and columns of the known rail locations can be extracted and saved in a separate M x N matrix, where M is the height of the image and N is the width of the rail. An extracted rail is shown in Figure 5.7. Black artifacts are located at the top and bottom of the extracted rail due to the transformation process. This is not a problem and will, in fact, aid in the analysis of the rails.

The next step taken was to calculate the vertical gradient on each pixel location of the extracted rail. The vertical gradient is used to detect horizontal changes in pixel intensity as explained in section 2.2.1. The reason that a vertical gradient is uses is explained in Figure 5.8. A large vertical gradient typically is caused by a defect in the tracks. This is opposed to a large horizontal gradient which usually doesn't indicate a defect. This results in gradient information the same size as the rail: M x N. The gradient information across each row was averaged to reduce the number of false positives and simplify the analysis. This creates a single column of gradient values representing the track: M x 1. Plots similar to Figure 5.9 and Figure 5.10 can then be

created to analyze the gradient along the length of the rail; these plots represent the left and the right rail of Figure 5.6 respectively. This is the gradient information used to determine if a defect is present in the rail or not.

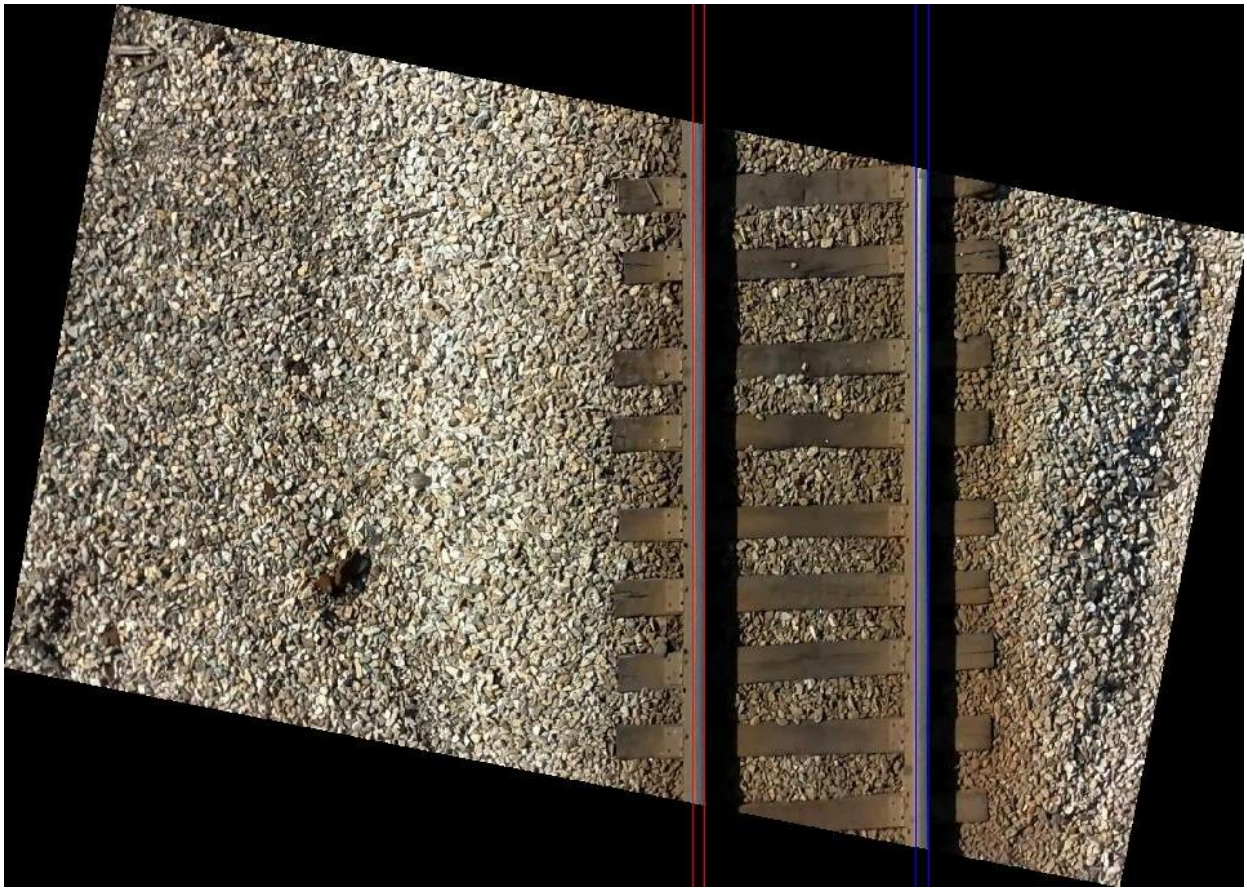

Figure 5.6 This is a transformed image of a rail. This image was transformed using manually picked rail edge points. The red lines represent the bounds of the left rail and the blue lines represent the bounds of the right rail. The area bounded by the lines is used in the defect detector.

Figure 5.7 An extracted rail from a transformed image. This rail contains no obstructions or defects. This is the data that is used to detect anomalies on the rail head.

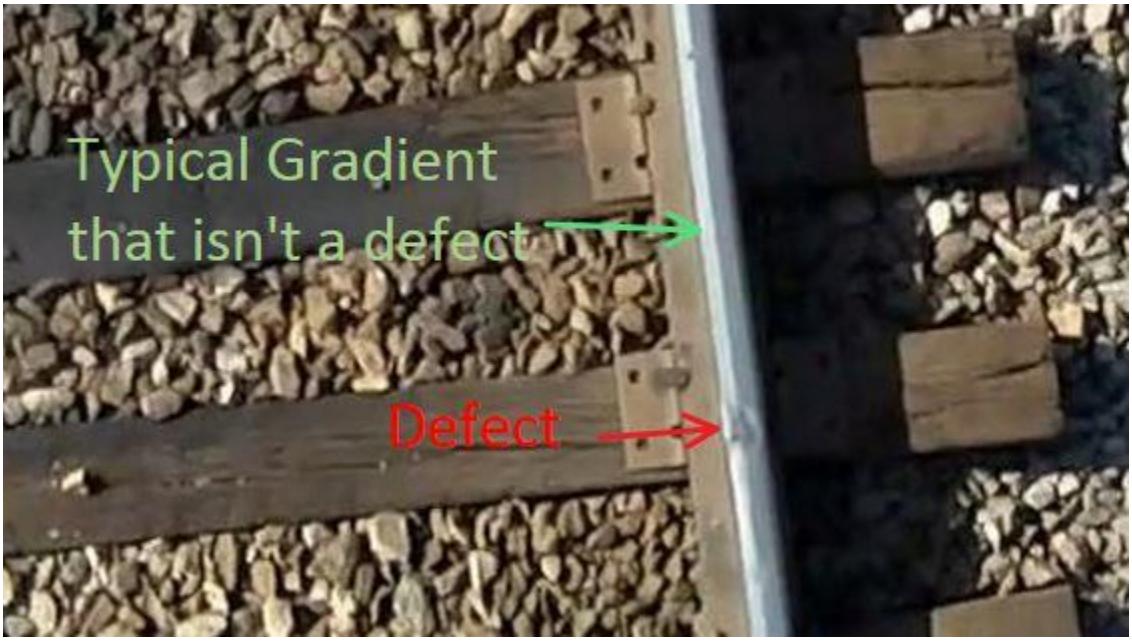

Figure 5.8 This figure illustrates the difference between a typical defect gradient and the gradient of a properly functioning rail.

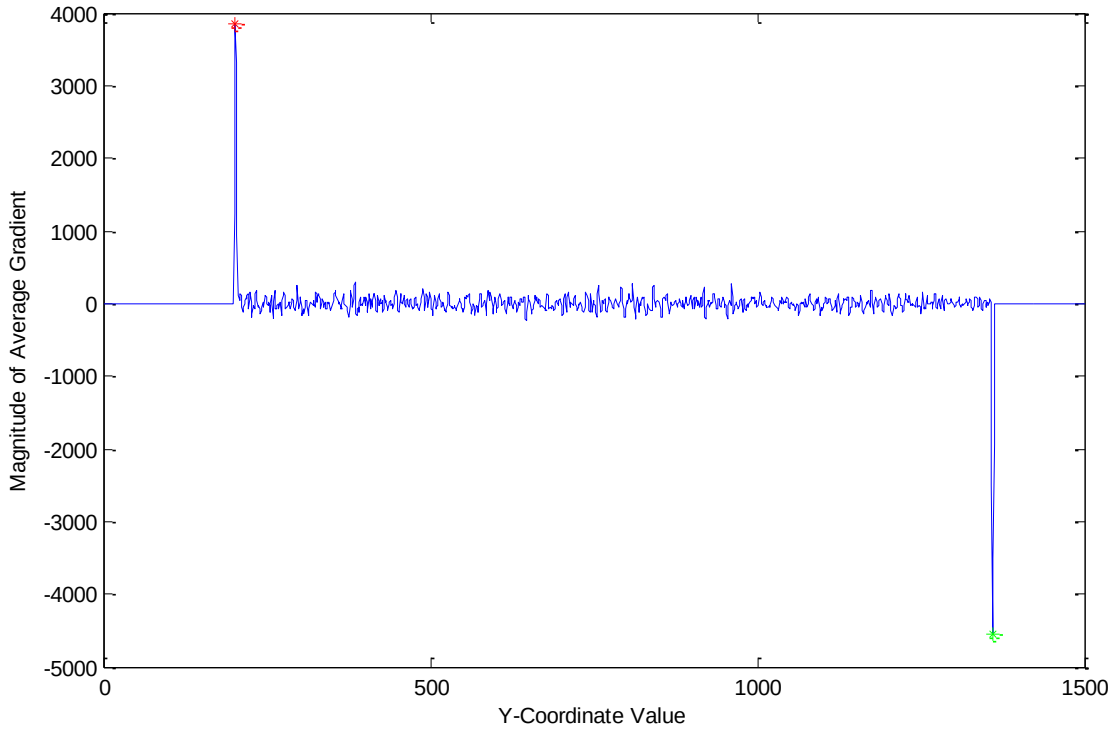

Figure 5.9 This is a plot of the average vertical gradient value of the pixels within the given rail bounds of the left rail in Figure 5.6. The x-axis runs from top to bottom along the rail. The peaks on both ends of the plot are the beginning and ending of the rail on the transformed image. The data in-between the two peaks is the average vertical gradient on the rail surface. There are no significant gradient values on the rail surface because there are no defects in this rail. A red star indicates a significant peak and a green star indicates a significant valley.

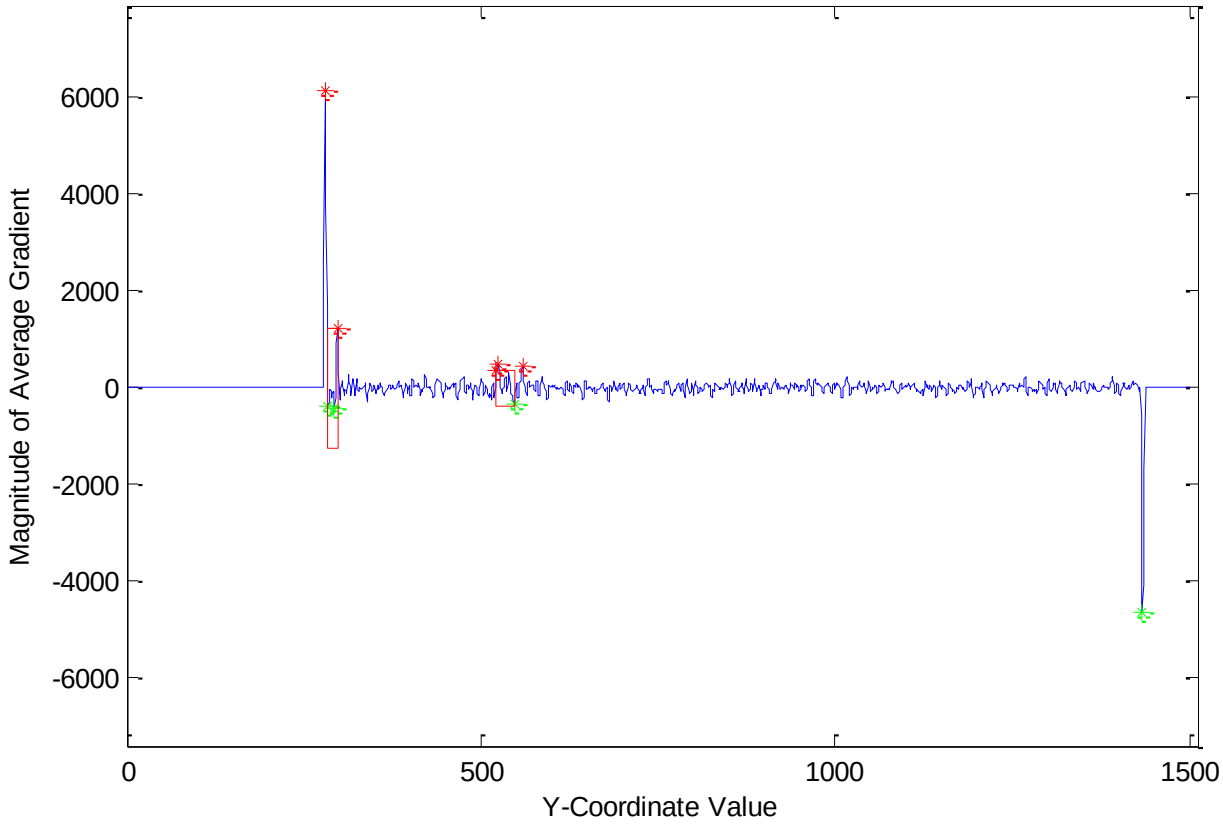

Figure 5.10 This is a plot of the average vertical gradient value of the pixels within the given rail bounds of the right rail in Figure 5.6. The x-axis runs from top to bottom along the rail. The peaks on both ends of the plot are the beginning and ending of the rail on the transformed image. The data in-between the two peaks is the average vertical gradient on the rail surface. There are significant gradient values on the rail surface because there are defects in this rail. A red star indicates a significant peak and a green star indicates a significant valley. Since these peaks are found on the rail surface they are due to an anomaly. The two red boxes are the locations the algorithm believes the anomalies are located.

5.2.2 Analyzing Rail Gradient Information

The first step in analyzing the gradient plots shown in Figure 5.9 and Figure 5.10 is to locate the peaks. A code written by Carlos Adrian Vargas Aguilera was used which is designed to locate all extrema on a 2D plot. This code can be found at [50]. This code returns all of the local extrema across the whole plot which is not exactly the desired output. The significant gradient values need to be distinguished from the noise in the plot due to the rail texture. So, significant gradient values are classified according to the following equation:

$$Defect \geq Std(Data) * 2 \qquad \textit{Equation 5.1}$$

where Data is the gradient values. In this way, the algorithm automatically determines a threshold value that will eliminate the noise of the rail texture. This leaves the data outliers as

shown with the green and red stars in Figure 5.9 and Figure 5.10. A summary of the defect-detection process is shown below:

1) Begin with a vertically oriented image of a rail. The length of the rail should run along the Y-axis.
2) Calculate the Y gradient for the entire rail area. Using the gradient in this direction will specifically look for discrepancies running perpendicular to the length of the rail. It is typical for a rail head to have lines running along the length of the rail, but a horizontal line is likely to be a defect.
3) Sum the results along the x-axis. This is done to smooth out any noise.
4) Take the standard deviation of the previously calculated data.

If one of the values calculated in step 3 is larger than 2 times the standard deviation, it is considered a defect. Once this process is complete, there is still more analysis to be done. The two large peaks at both ends of the plots are not defects, but instead are the start and end of the rail. These are the large gradients between the black artifacts created by the image transformation and where the rail pixels begin. If the rails have been located properly and the transformation done properly, these sharp peaks should always be similar in magnitude and opposite in sign. So, these attributes are used as a check to verify that the process is working properly. Then, any remaining outlying peaks are classified as anomalies. Figure 5.11 is a zoomed-in version of the gradient of the right rail of Figure 5.6 and it shows how the defects were classified. The suspected defects are boxed in red. Figure 5.12 shows these same defect locations boxed in the image. Both of the detected defects are minor flat spots in the rail and are difficult to detect with the human eye. This proves the validity of the system and the defect detection concept. A more detailed analysis of this detection method was performed alongside the Camera Resolution Study described in Chapter 6.

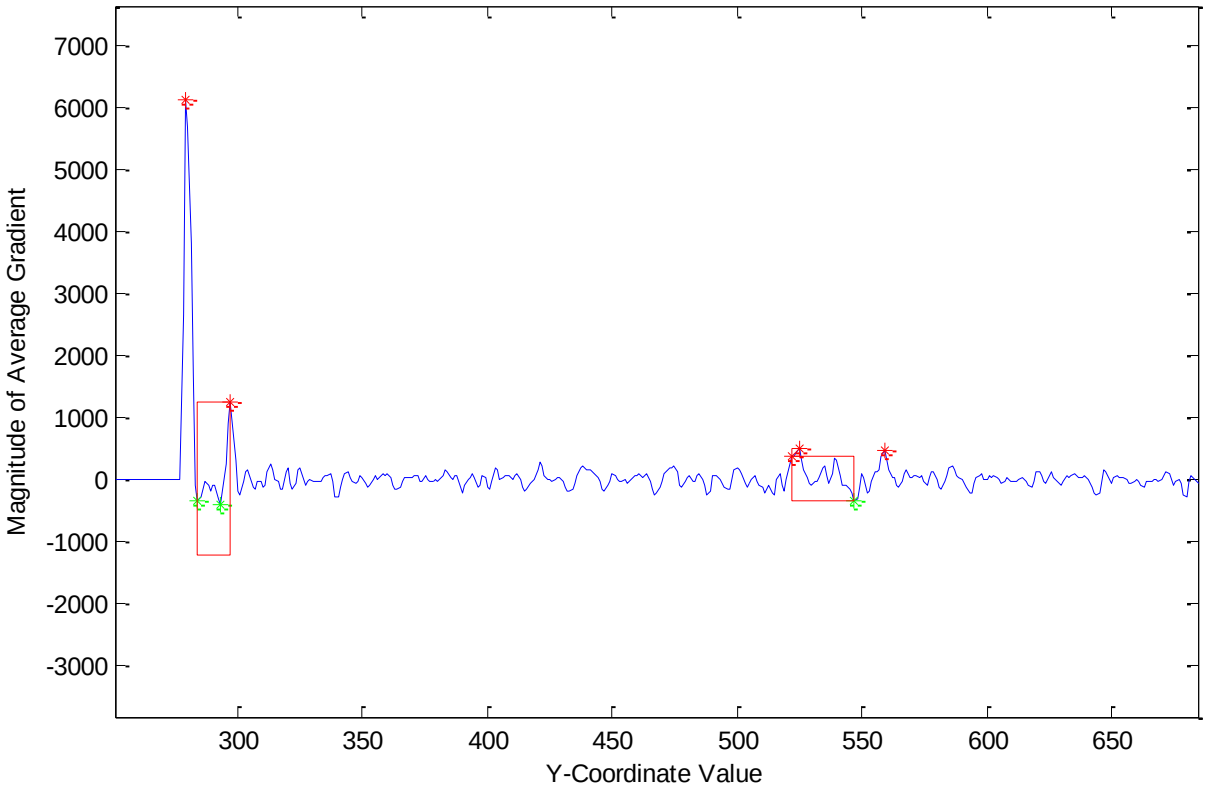

Figure 5.11 This is a zoomed-in version of Figure 5.10.

Figure 5.12 This is a zoomed-in version of Figure 5.6 with defects boxed in red as classified using the gradient defect detection method.

5.2.3 Flaws with the Gradient Method

One flaw with this method is that the image must be free from all distortion. If the rails are not vertical on the image an error shown in Figure 5.13 may occur. The left rail in this image is not vertical and therefore part of the rail is missed. Luckily, this phenomenon can be easily detected. Figure 5.14 shows the gradient analysis of the left rail in Figure 5.13. As shown in this figure, the difference in magnitude between the first peak and the last peak becomes very large when this distortion error occurs. So, by verifying that the magnitude of the first and last peak is similar, the system can be confident that the rail was properly located.

To prevent this issue from happening in the final system, a gimbaled camera should be used. The gimbal will guarantee that the camera stays at a nadir view to the ground while the aircraft is maneuvering. Also, the camera images should be calibrated and undistorted to remove any lens distortion.

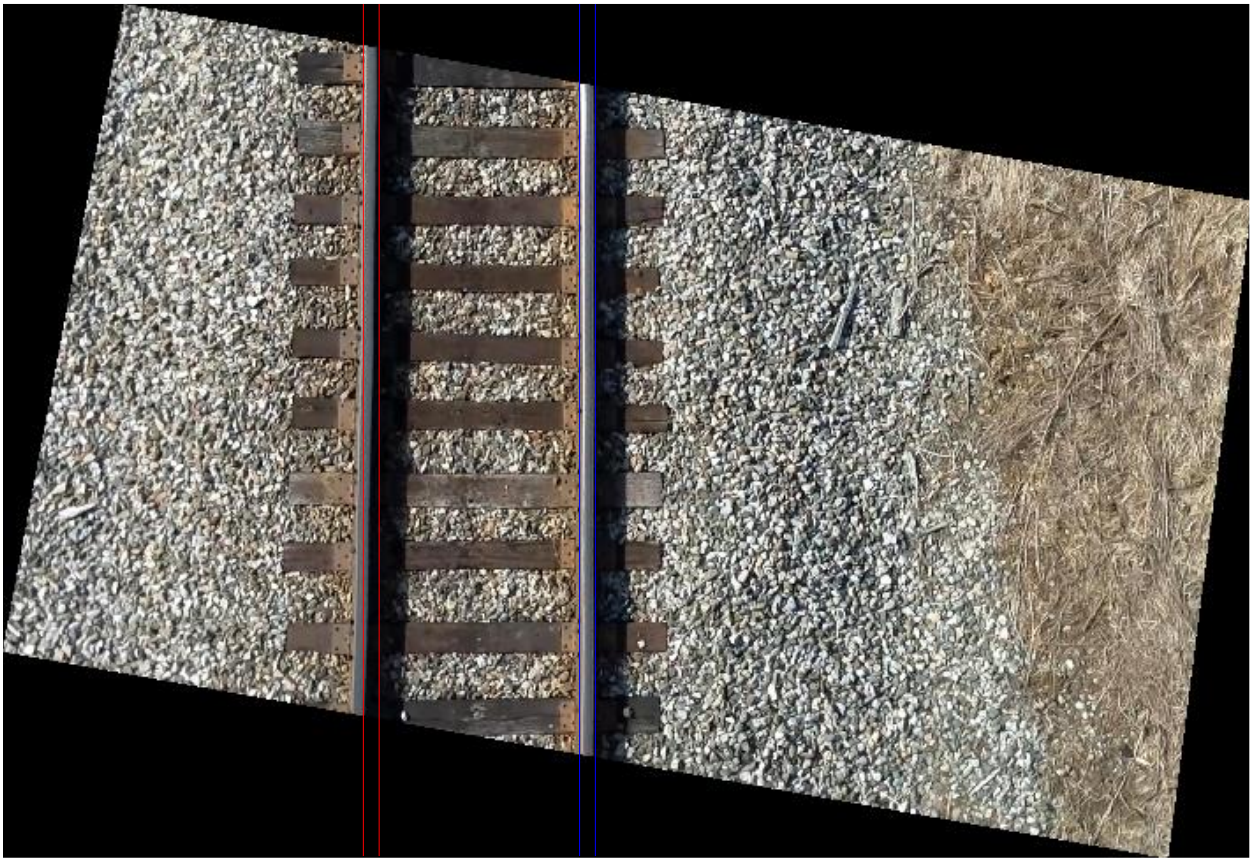

Figure 5.13 This image is the result of transforming a rail image. An error in the transformation or distortion in the image caused the left rail to curve out of the search radius.

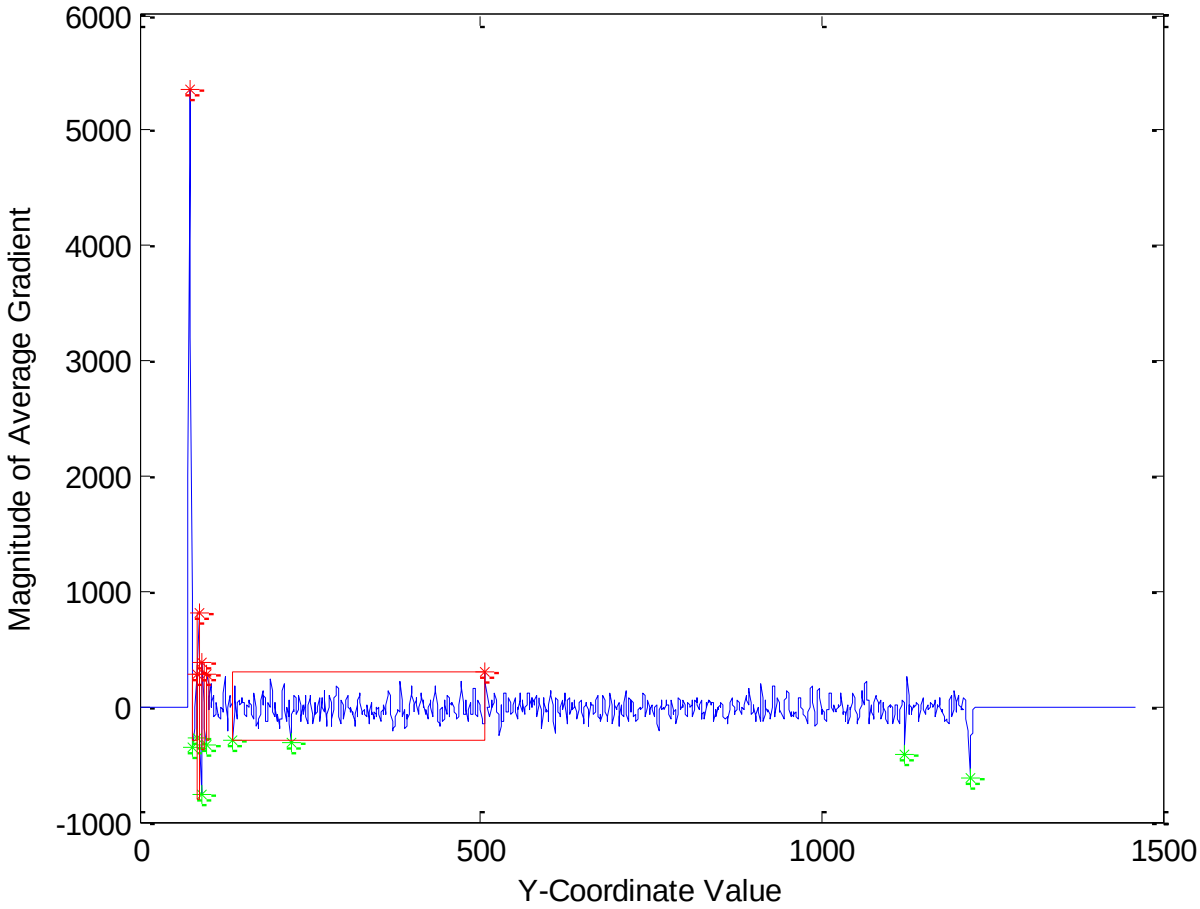

Figure 5.14 The gradient analysis of the left rail in Figure 5.13. The distortion error found in this image shows up as a large difference in the gradient magnitude when comparing the first to the last peak. Detecting this phenomenon indicates that an error has occurred.

5.2.4 Locating Vegetation on Tracks

The NDVI method explained in section 2.3.1 is the key to detecting vegetation on tracks. This is a common method that uses light wavelength to determine the health of plants. In this case, it can also determine if green plants are present or not.

5.2.4.1 Creating the Near-Infrared Camera

For this study, it was decided to create a custom near-infrared (NIR) camera as opposed to purchasing one. A cheap near-infrared camera costs approximately $2000 which was double the cost of the entire UAV system. So, instead a Canon A810 was modified to be used as a near-infrared camera. The modification process is detailed in [51]. Standard point and shoot cameras are equipped with an infrared filter to obtain properly colored images. To create a low cost NIR camera, the infrared filter was removed and a Roscolux #2007 blue filter was added. This modification effectively changes the blue values in the image to NIR information. Figure 5.15 is an example image taken from the modified camera.

This modification changes the blue pixel values to be NIR because the IR filter was replaced by the blue filter. So, in Equation 2.5, blue pixel values are put into NIR and use the red values are put into VIS. Figure 5.16 is an example of this calculation. The warmer the color in the image, the more likely the pixel represents vegetation. Two blue filters were also tried in an attempt to obtain better results. Figure 5.17 shows the results of this effort. This more effectively blocked the blue wavelengths, so the results became more predominant. Three filters would not fit into the camera cavity, so two filters were used.

Figure 5.15 Image taken from a Canon A810 with the infrared filter removed and a Roscolux #2007 blue filter added.

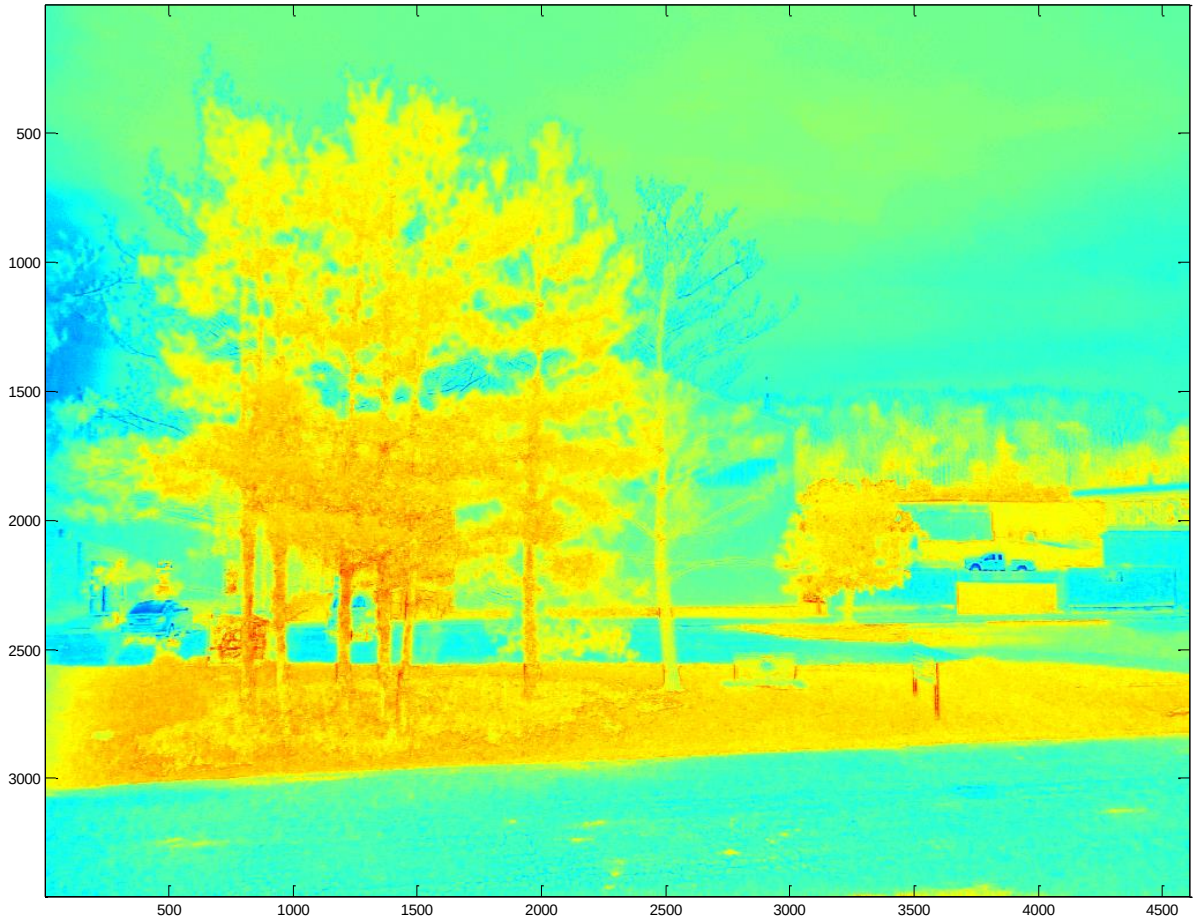

Figure 5.16 NDVI calculation applied to Figure 5.15. This image was taken using one Roscolux #2007 blue filter. The warmer the color in the image, the more likely it is vegetation.

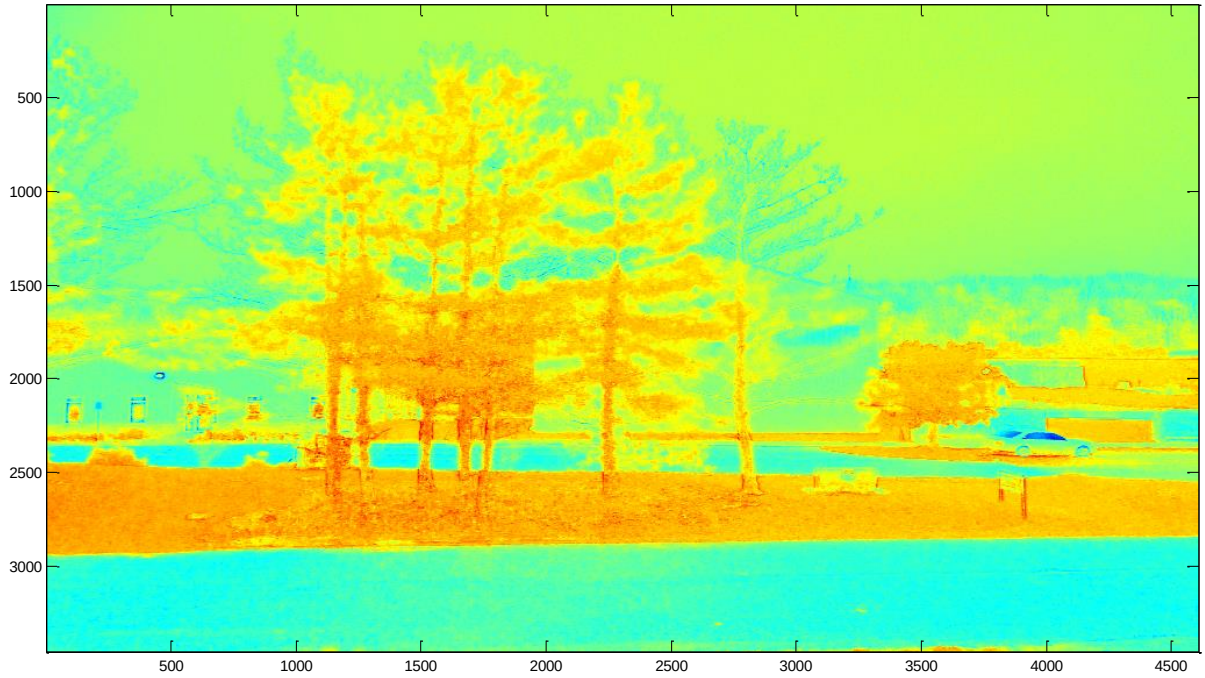

Figure 5.17 NDVI calculation applied to an image taken with two Roscolux #2007 blue filters. This image was taken using one Roscolux #2007 blue filter. The warmer the color in the image, the more likely it is vegetation. It can be seen that the vegetation is more prominent in this image as opposed to Figure 5.16.

5.2.4.2 Using the Near-Infrared Camera to Analyze Track Vegetation

Ideally, a series of images of track would be taken with the NIR camera and analyzed: some track with vegetation on it and some without. Due to FAA flight restrictions, images of vegetation on tracks were not able to be obtained. For this reason, no concrete conclusions could be made on how well the method will work for this application. On the other hand, it can be determined whether or not the method will work to some extent or not work at all.

So, to analyze this method, some images of track were taken with the NIR camera. One such image is shown in Figure 5.18. This image was taken by flying a Rascal with the NIR camera mounted in a nadir view to the fuselage. On the left side and the middle of the image are areas of vegetation that were growing near the tracks. Both of these areas are boxed in red. The tracks in these images did not contain vegetation on them.

The NDVI calculation detailed in section 5.2.4.1 was then performed on the image. Figure 5.19 is the result. In this image, the warmer the pixel, the more likely it is to be live vegetation. There are two boxed areas in the figure: one is track and the other is vegetation. To determine the validity of this process, the value of these two areas were considered. If there is a notable difference in the average pixel value of these areas, it shows the process is plausible for determining vegetation on track. The area containing vegetation had an average NDVI of 0.84 and the area containing track had an average NDVI of 0.63. This is a notable difference and shows that this process has promise.

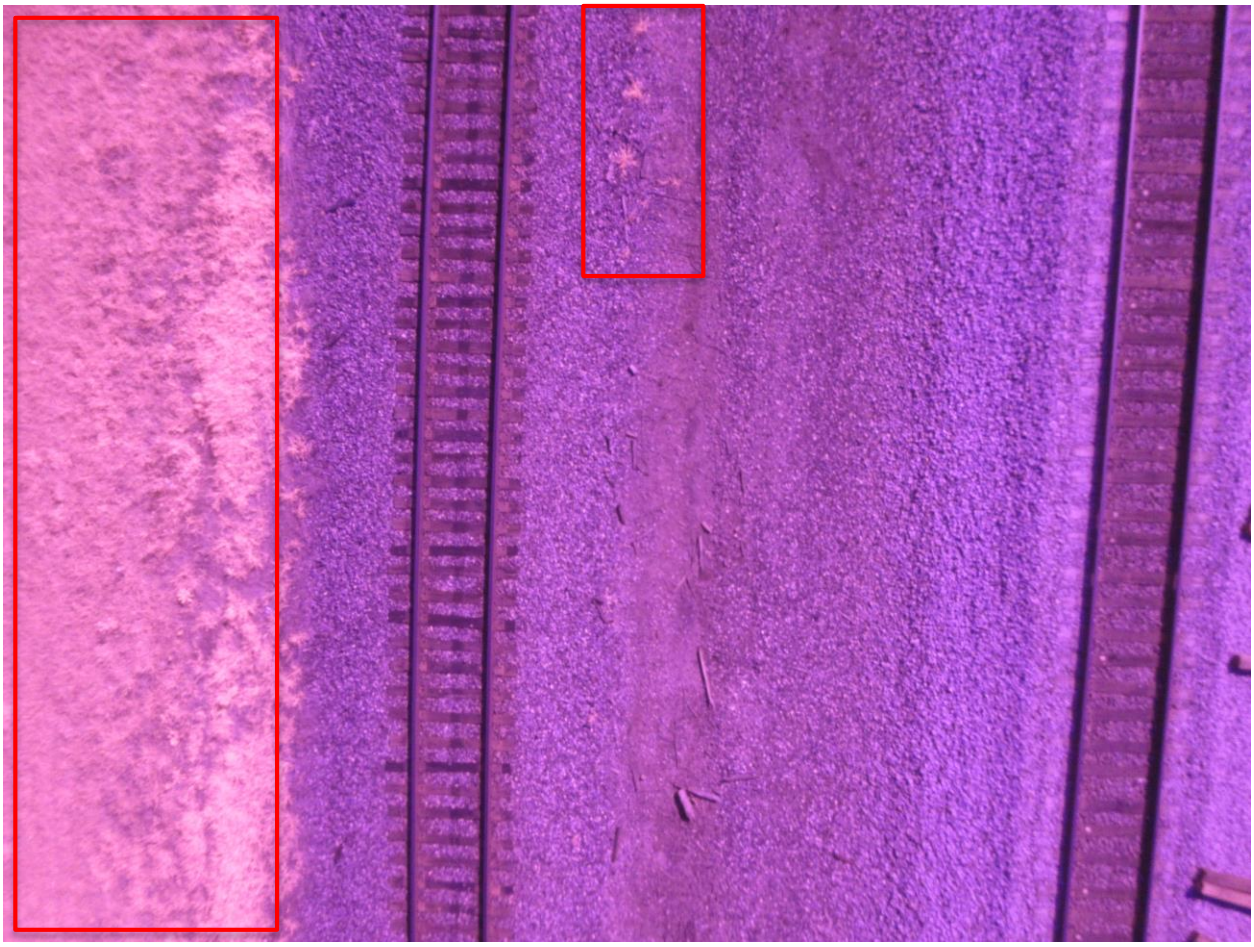

Figure 5.18 Image taken with the modified Canon A810. The image contains two patches of vegetation which are boxed in red. The rails do not contain vegetation.

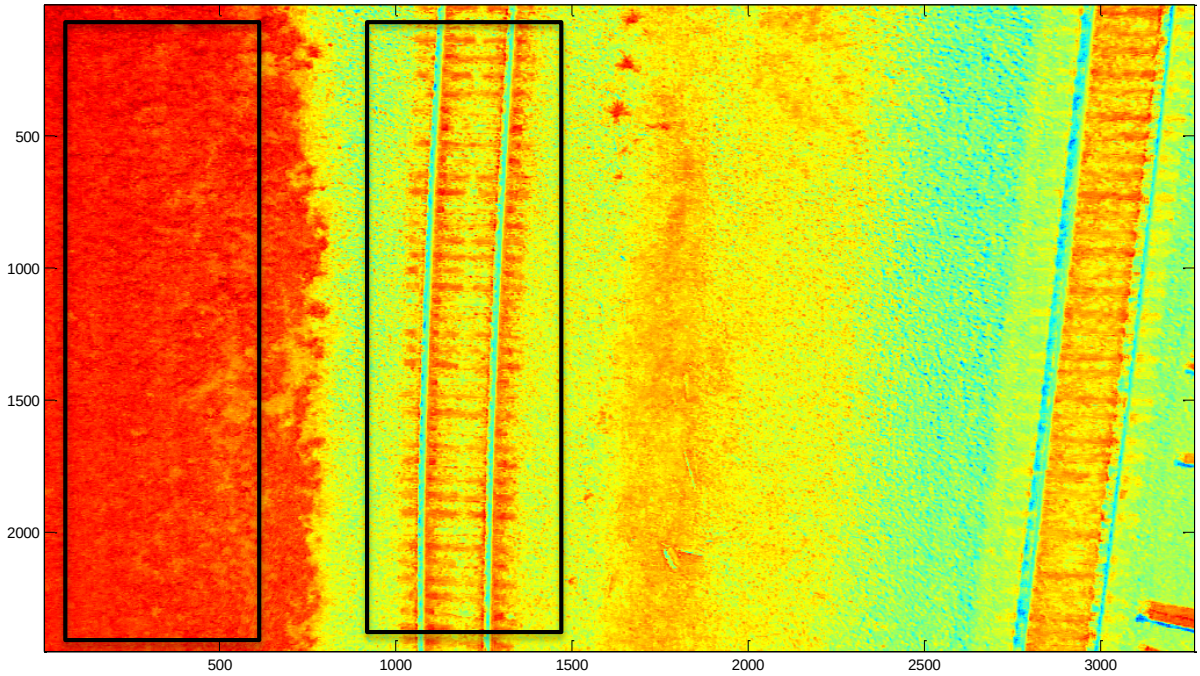

Figure 5.19 NDVI calculation applied to Figure 5.18. The warmer the color, the more likely the pixel is to representing vegetation. The rail ties show up somewhat as live vegetation because of the organic nature of the wood. The black boxes represent the two areas that were compared.

Chapter 6 Camera Resolution Study

It is very important to determine an optimal camera resolution for rail defect detection. As resolution is increased, processing time increases. On the other hand, if the resolution is too low, defects on the rail could be missed. So, a detailed analysis was conducted to determine the required resolution for the defect detection method outlined in section 5.2.

6.1 Data Set Information

6.1.1 Obtaining Test Image and Adding Defects

This study was conducted to determine what camera resolution is required to detect a 1" defect in the rail head. The basis of the study was a 2MP image that was taken at nadir view from a quadcopter flying at a 10 meter altitude. This image was then warped so that both rails were positioned vertically on the image in the way explained in section 5.2. Then, to determine the size of a 1" defect, there must be a correlation from pixels to inches. To do this, the width of the rail was measured to be 3 ¼" and the width of the rail in the image to be ~17 pixels wide. So, simple proportions show a 1" defect corresponds to ~5 pixels in the original resolution image. Using these results, a 1" defect was modeled by putting 5x5 pixel discolorations into the image at different shades and locations. Figure 6.1 shows this warped image with defects added.

The details of the added defects are can be found in Table 6.3. Most defects were 1"x1", but some were modeled across the entire rail width and two tree branches were added across the rail. Each defect that was added was unique in color, location or size. Based on these factors, each defect was ranked on its difficulty to be detected: easy, medium, hard, very hard. The difficulty rankings are based on size and how similar the defect is to the color of the rail. Examples of each of these difficulty levels can be seen in Figure 6.2.

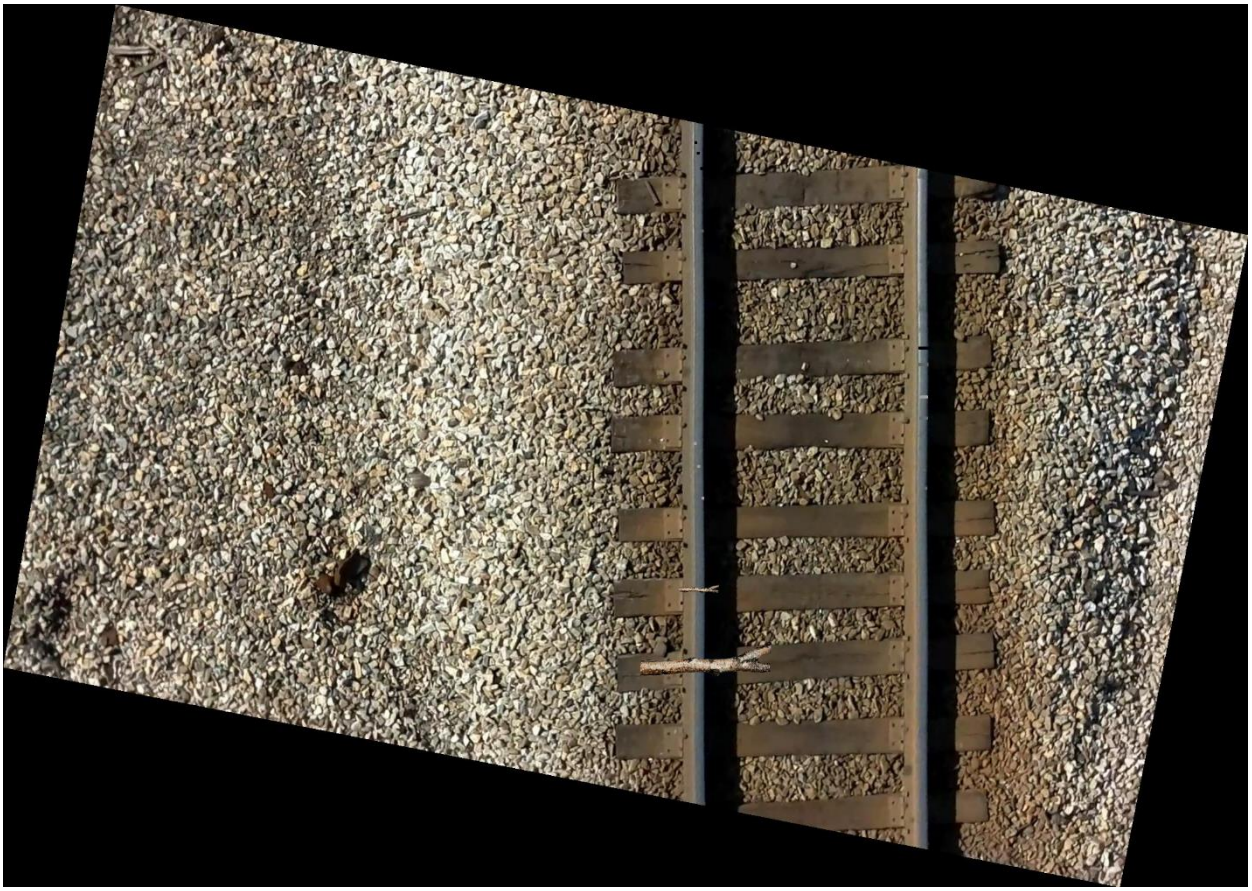

Figure 6.1 Image used in defect detection resolution study. A total of 20 defects are on the rails. A total of 18 defects have been doctored into the image; each defect has a unique color, location and size.

Table 6.1 Table outlining the details of all defects in the image that was used in the study.

Defect	Rail	Color [R, G, B]	Location (x, y)	Location on Rail	Height	Width	Difficulty
1	Left	[0, 0, 0]	1176, 234	Left	5	5	Easy
2	Left	[0, 0, 0]	1180, 252	Middle	5	5	Easy
3	Left	[0, 0, 0]	1186, 278	Right	5	5	Easy
4	Left	[96, 96, 96]	1176, 306	Left	5	5	Hard
5	Left	[96, 96, 96]	1181, 352	Middle	5	5	Hard
6	Left	[96, 96, 96]	1187, 384	Right	5	5	Hard
7	Left	[160, 160, 160]	1175, 745	Left	5	5	Medium
8	Left	[160, 160, 160]	1181, 784	Middle	5	5	Medium
9	Left	[160, 160, 160]	1188, 838	Right	5	5	Medium
10	Left	Brown Branch	1178, 993	Whole	5	Past Rail	Easy
11	Left	Brown Branch	1177, 1113	Whole	23	Past Rail	Easy
12	Right	Natural Flat Spot	1555, 288	Whole	10	Rail	Hard
13	Right	Natural Flat Spot	1555, 512	Whole	50	Rail	Hard
14	Right	[0, 0, 0]	1557, 583	Whole	5	Rail	Easy
15	Right	[96, 96, 96]	1557, 611	Whole	5	Rail	Medium
16	Right	[160, 160, 160]	1557, 666	Whole	5	Rail	Medium
17	Right	[97, 108, 112]	1558, 702	Whole	5	Rail	Hard
18	Right	[97, 108, 112]	1559, 753	Left	5	5	Very Hard
19	Right	[97, 108, 112]	1563, 790	Middle	5	5	Very Hard
20	Right	[97, 108, 112]	1567, 821	Right	5	5	Hard

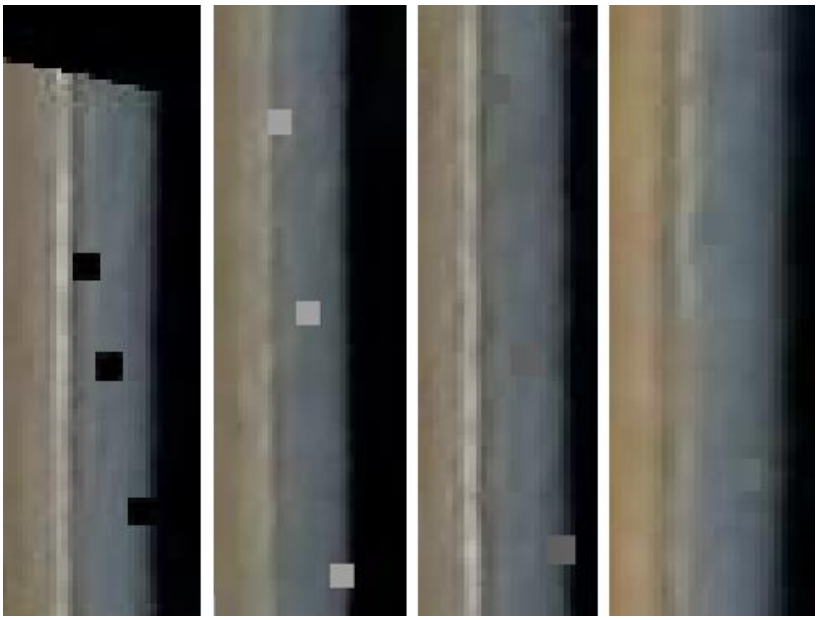

Figure 6.2 This figure shows zoomed-in examples of the 4 different difficulty levels of the defects in the test image. From left to right: easy, medium, hard and very hard. The defects were placed into categories based on size and how similar the color of defect was to the color of the rail.

6.1.2 Down-Sampling the Defect Image

To perform the resolution study, the image shown in Figure 6.1 was down-sampled multiple times. The image resolutions range from ~2MP to ~0.1MP. These images were designed to cover a wide range of resolutions to thoroughly test the requirements of the system. Table 6.2 shows the details of the images used in the resolution study.

Table 6.2 Table outlining the different resolutions used in the study. Image 1 is the original image, Image 2 is a down-sampled version of Image 1, etc. These cover a wide range of resolutions to thoroughly test the requirements of the system.

Image	Image Size (pixels)	Approximate Defect Size	Approximate Rail Width
Image 1	2116x1500	5x5 pixels, 1”x1”	17 pixels, 3 ¼”
Image 2	1411x1000	3x3 pixels, 1”x1”	11 pixels, 3 ¼”
Image 3	1058X750	2.5x2.5 pixels, 1”x1”	9 pixels, 3 ¼”
Image 4	706x500	2x2 pixels, 1”x1”	5 pixels, 3 ¼”
Image 5	353x250	1x1 pixels, 1”x1”	3 pixels, 3 ¼”

6.2 Results

Defect detection, as outlined in section 5.2, was performed on each of the images outlined in Table 6.2. The detection results are shown in Table 6.3. Each defect is listed as ‘Found’, ‘Partial’ or ‘Missed’ which corresponds to how well the defect was located. Text is highlighted red in the table if the image performed worse than the previous resolution image and green if it performed better.

Figure 6.3 is a summary of the test results. The plot shows rail width (resolution) vs. number of defects found. Only easy and medium defects were included in the plot because they are the most applicable to this work. This plot shows that the algorithm was able to detect all defects when using the original resolution image, but the accuracy degraded as resolution decreased.

Table 6.3 This table is an outline of the details of which defects were located by the defect detector: 'Found' means that both sides of the defect were detected, 'Partial' means one side of the defect was detected and 'Missed' means that neither side of the defect was detected. Green text represents the current image performed better than the image of higher resolution. Red text represents the current image performed worse than the image of higher resolution.

Defect	Difficulty	Image 1	Image 2	Image 3	Image 4	Image 5
1	Easy	Found	Found	Found	Found	Found
2	Easy	Found	Found	Found	Found	Found
3	Easy	Found	Found	Found	Partial	Missed
4	Hard	Missed	Missed	Missed	Missed	Missed
5	Hard	Missed	Missed	Missed	Missed	Missed
6	Hard	Found	Found	Found	Missed	Missed
7	Medium	Found	Found	Partial	Missed	Missed
8	Medium	Found	Found	Found	Missed	Missed
9	Medium	Found	Found	Found	Partial	Found
10	Easy	Found	Found	Found	Found	Found
11	Easy	Found	Partial	Found	Partial	Found
12	Hard	Found	Partial	Partial	Partial	Partial
13	Hard	Missed	Missed	Missed	Missed	Missed
14	Easy	Found	Partial	Found	Found	Found
15	Medium	Found	Missed	Missed	Missed	Missed
16	Medium	Found	Missed	Partial	Partial	Partial
17	Hard	Missed	Missed	Missed	Missed	Missed
18	Very Hard	Missed	Missed	Missed	Missed	Missed
19	Very Hard	Missed	Missed	Missed	Missed	Missed
20	Hard	Found	Missed	Found	Found	Missed

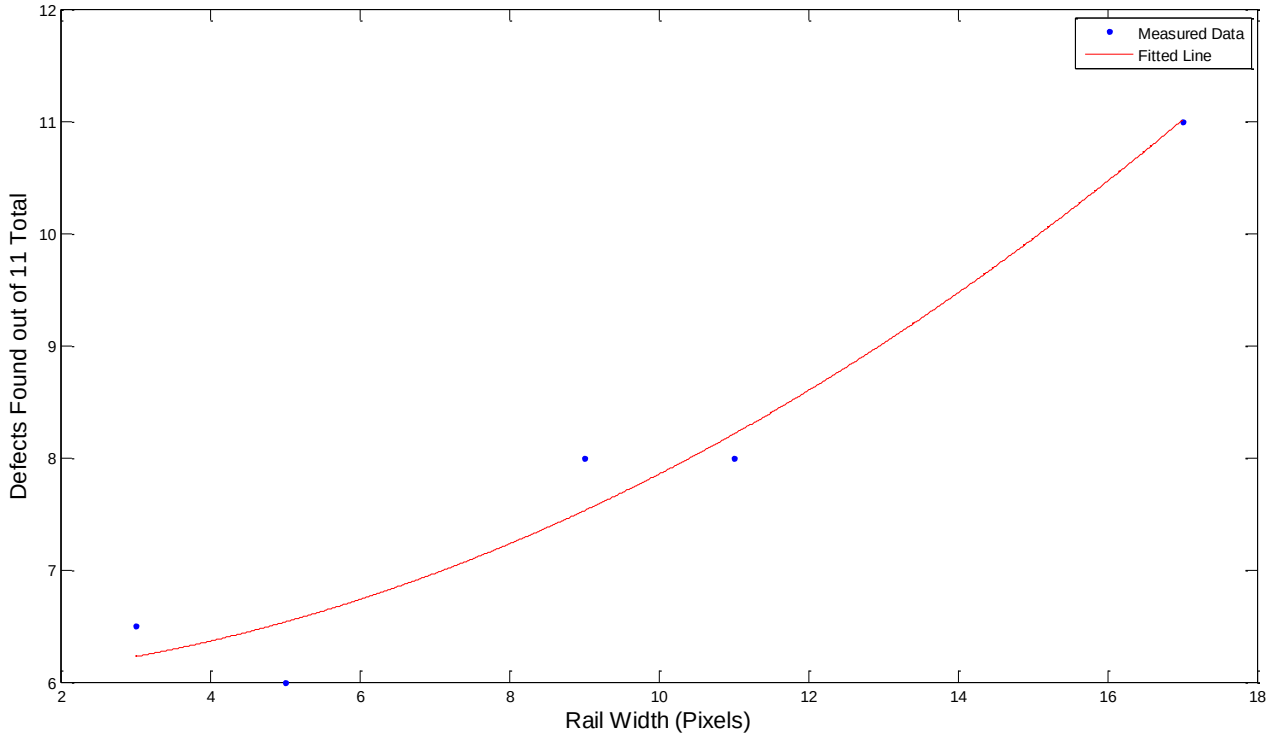

Figure 6.3 Plot of defects found vs. rail width for this detection method. For this plot, only the 'easy' and 'medium' defects were included because they are the most pertinent to this work.

6.3 Conclusions

Some valuable information can be extracted from this study. First, the ability to detect a defect is more dependent on how visibly prominent the defect is than how large it is. The defects that were very prominent (easy) were much more likely to be found regardless of resolution than that of discrete defects (hard or very hard).

Next, all 'easy' and 'medium' defects were located at the full resolution, but as resolution decreased, defects were missed. So, having a rail that is at least 17 pixels wide is needed to robustly detect 1" square defects with the current detection method. This estimate can be extrapolated to apply to any detection method, but the estimate would be less accurate. This would occur because the resolution required is dependent on type of detect and detection method.

It is important to note a curious behavior that showed up in the study. It is intriguing that the same defect would be detected in a lower resolution image, but not a higher resolution image. The exact reasoning for this is uncertain, but it could be illustrating that the ability to detect a specific defect is case dependent. It was concluded that a factor of safety should be included when deciding upon the resolution of the image.

The final conclusion of this study is that in order to consistently and robustly locate 1"x1" defects with this technique, the rail must be at least 17 pixels wide in the image. This conclusion is only true for this specific defect size and detection algorithm. This resolution was achieved by flying at a 10 meter altitude and taking a 2 MP image.

Chapter 7 Conclusions and Future Work

This research has shown that creating a rail detection and health monitoring UAV is an achievable goal. This paper has shown the details of a prototype rail detection algorithm which was able to detect rails with an average accuracy of 0.59 inches and 0.27 degrees from the true rail locations. This rail detection algorithm was able to run at a speed of 1.8 fps.

In terms of defects, a gradient-based defect detection algorithm was created. This algorithm is based on the premise that a properly functioning rail will not contain significant gradient changes. This algorithm was tested and shown to reliably detect easily visible defects at least 1" square. Easily visible is a somewhat subjective measure. Defects that were not easily visible were detected as well, but not in a reliable manner. Larger defects were reliably detected as well. For these results to remain true with the current algorithm, a rail width of at least 17 pixels must be maintained.

Other periphery research was completed to be used by further researchers. An extensive image library was created with many different hardware setups, weather conditions and track locations. These will be necessary when designing the rail detection algorithm to be robust to all situations. Also, NDVI was shown to be a promising way to detect vegetation on tracks.

A logical next step in implementing this final system is creating a hardware platform with a computer on-board to complete real-time calculations. The algorithms can then be tuned and optimized to this specific hardware. This will allow for any residual errors to be brought to the surface and taken care of. Another logical step would be to adjust the features used in the rail detection algorithm. This could be refined by doing a more detailed study of features of rails as opposed to other objects in an image.

In terms of defect detection, a logical next step would be to test the algorithm on real defects. This would require a library of images containing defects and manually classifying the defects. Then training could be done on specific real-life defects to allow for more detailed defect classification.

All of these facts give confidence that a complete rail detection and health monitoring UAV can be implemented.

Bibliography

[1] [Online]. Available: http://safetydata.fra.dot.gov/OfficeofSafety/Default.aspx. [Accessed 13 10 2014].

[2] "Federal Railroad Administration Issues Final Review to Improve Rail Inspections," 24 1 2014. [Online]. Available: http://www.fra.dot.gov/eLib/details/L04921. [Accessed 13 10 2014].

[3] "Fact Sheet – FAA UAS Test Site Program," 30 12 2013. [Online]. Available: https://www.faa.gov/news/fact_sheets/news_story.cfm?newsid=15575. [Accessed 13 10 2014].

[4] "GPS Accuracy and Error Sources," Mio, 2014. [Online]. Available: http://www.mio.com/technology-gps-accuracy.htm. [Accessed 1 1 2015].

[5] FRA, "Track Inspection Time Study," [Online]. Available: http://permanent.access.gpo.gov/gpo26599/TrackInspectionTimeStudyFR62911.pdf. [Accessed 30 October 2014].

[6] F. R. Administration, "Track Safety Standards," 24 January 2014. [Online]. Available: www.fra.dot.gov/Elib/Document/3546. [Accessed 30 October 2014].

[7] "Geismar Filus X27," Geismar, [Online]. Available: http://www.geismar.com/en/electronic-measurement/197-filus-x27.html. [Accessed 3 11 2014].

[8] "Ultrasonic Rail Testing," Herzog Services, Inc. , 2014. [Online]. Available: http://herzogservices.com/services2/ultrasonic-rail-testing/. [Accessed 3 11 2014].

[9] "RDM - Ultrasonic Flaw Detectors and Non-Destructive Testing Systems," Railway-Technology, 2014. [Online]. Available: http://www.railway-technology.com/contractors/track/npp-rdm/. [Accessed 3 11 2014].

[10] "Rail Inspection The Eddy Current Solution," Hocking, [Online]. Available: http://www.krautkramer.com.au/Eddy%20Current%20Rail%20Inspection.pdf. [Accessed 3 11 2014].

[11] "Background on Pulsed Eddy Current," NDT Resource Center, [Online]. Available: https://www.nde-ed.org/EducationResources/CommunityCollege/EddyCurrents/AdvancedTechniques/background.htm. [Accessed 5 11 2014].

[12] "Pulsed Eddy Current," TUV Rheinland, [Online]. Available: http://www.tuv.com/en/corporate/business_customers/materials_testing_and_inspection/advanced_ndt/pulsed_eddy_current/pulsed_eddy_current.html. [Accessed 5 11 2014].

[13] "Rail Inspection," Wikipedia, 25 10 2014. [Online]. Available: http://en.wikipedia.org/wiki/Rail_inspection. [Accessed 5 11 2014].

[14] "RailScan Lite Track Geometry Measurement System," Ensco Rail, [Online]. Available: http://www.ensco.com/userfiles/file/Products_Services_PDF/07_Rail/13.0037-Rail-Scan-Lite-Track-Geometry-Measurement-System-ENSCO-Rail.pdf. [Accessed 6 11 2014].

[15] "Track Inspection Vehicle Design," Ensco, [Online]. Available: http://www.ensco.com/products-services/rail-technologies/track-inspection-systems/track-inspection-vehicle-design.htm. [Accessed 5 11 2014].

[16] "Sperry Rail Service," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Sperry_Rail_Service. [Accessed 5 11 2014].

[17] "Trimming the Cost of Track Inspection," Mermec, 11 2010. [Online]. Available: http://www.mermecgroup.com/news-e-events/media-coverage/471/1/trimming-the-cost-of-track-inspection.php. [Accessed 5 11 2014].

[18] S. V. Sawadisavi, "Machine-Vision Inspection of Railroad," 2009. [Online]. Available: http://pdf.aminer.org/000/346/445/a_machine_vision_system_for_inspecting_bearings.pdf. [Accessed 19 1 2015].

[19] U. o. Illinois, "Advancements in Railroad Track Inspection Using Machine-Vision Techniques," 2009. [Online]. Available: https://www.arema.org/files/library/2009_Conference_Proceedings/Advancements_in_Railroad_Track_Inspection_Using_Machine-Vision_Technology.pdf. [Accessed 19 1 2015].

[20] "Machine Vision Inspection of Railroad Track," USDOT Region V Regional University Transportation Center, 10 1 2011. [Online]. Available: http://www.purdue.edu/discoverypark/nextrans/assets/pdfs/Year%202%20Final%20Reports/Final%20Report%20028.pdf. [Accessed 19 1 2015].

[21] Mermec Inc., "Track Surface Inspection System," MERMEC Inc., [Online]. Available: http://www.progressiverailroading.com/railproducts/product/Track-Surface-Inspection-System-TSIS--1888. [Accessed 19 1 2015].

[22] "Aurora," GRex, [Online]. Available: http://www.georgetownrail.com/Rail-Solutions/Track-Inspection/Aurora. [Accessed 19 1 2015].

[23] D. M. Shah, "Automated Visual Inspection/Detection of Railroad Track," 07 2010. [Online]. Available: http://www.dot.state.fl.us/research-center/Completed_Proj/Summary_PTO/FDOT_BD550-08_rpt.pdf. [Accessed 19 1 2015].

[24] P. Babenko, "Visual Inspection of Railroad Tracks," 2006. [Online]. Available: http://crcv.ucf.edu/papers/theses/Babenko_Pavel.pdf. [Accessed 19 1 2015].

[25] "SurfView," Beena Vision, [Online]. Available: http://www.beenavision.com/products_surfview.html. [Accessed 19 1 2014].

[26] "Gradient," Wikipedia, 24 12 2014. [Online]. Available: http://en.wikipedia.org/wiki/Gradient. [Accessed 14 1 2015].

[27] B. Dubrovin and S. N. A.T. Fomenko, in *Modern Geometry -- Methods and Applications: Part I: The Geometry of Surfaces, Transformation Groups, and Fields* , Springer, 1991, pp. 14 - 17.

[28] "Image Gradient," Wikipedia, 24 09 2014. [Online]. Available: http://en.wikipedia.org/wiki/Image_gradient. [Accessed 14 1 2015].

[29] R. Gonzalez and R. Woods, in *Digital Image Processing*, Pearson Education, pp. 165-68.

[30] "CIE 1931 Color Space," Wikipedia, 12 09 2014. [Online]. Available: http://en.wikipedia.org/wiki/CIE_1931_color_space. [Accessed 15 1 2015].

[31] W. D. Wright, "A re-determination of the trichromatic coefficients of the spectral colours," Transactions of the Optical Society, 1928, pp. 141-164.

[32] J. Guild, in *The colorimetric properties of the spectrum*, JSTOR, 1932, pp. 149-187.

[33] B. a. S. Susstrunk, "Standard RGB Color Spaces," 09 07 2011. [Online]. Available: http://infoscience.epfl.ch/record/34089/files/SusstrunkBS99.pdf?version=3. [Accessed 15 1 2015].

[34] C. A. Poynton, Digital Video and HDTV: Algorithms and Interfaces, 2003.

[35] "RGB Color Space," Wikipedia, 13 12 2014. [Online]. Available: http://en.wikipedia.org/wiki/RGB_color_space. [Accessed 15 1 2015].

[36] "HSL and HSV," Wikipedia, 2 1 2015. [Online]. Available: http://en.wikipedia.org/wiki/HSL_and_HSV. [Accessed 15 1 2015].

[37] M. K. Agoston, in *Computer Graphics and Geometric Modeling: Implementation and Algorithms*, London, Springer, 2005, pp. 300-306.

[38] "Lab Color Space," Wikipedia, 17 12 2014. [Online]. Available: http://en.wikipedia.org/wiki/Lab_color_space. [Accessed 1 15 2015].

[39] G. Hoffmann, "CIELab Color Space," [Online]. Available: http://docs-hoffmann.de/cielab03022003.pdf. [Accessed 15 1 2015].

[40] "List of Color Spaces and their Uses," Wikipedia, 14 12 2014. [Online]. Available: http://en.wikipedia.org/wiki/List_of_color_spaces_and_their_uses#CMYK_2. [Accessed 15 1 2015].

[41] "CMYK Color Model," Wikipedia, 14 10 2014. [Online]. Available: http://en.wikipedia.org/wiki/CMYK_color_model. [Accessed 15 1 2015].

[42] "Homography (Computer Vision)," Wikipedia, 4 11 2014. [Online]. Available: http://en.wikipedia.org/wiki/Homography_%28computer_vision%29. [Accessed 15 1 2015].

[43] E. Dubrofsky, "Homography Estimation," 05 2009. [Online]. Available: https://www.cs.ubc.ca/grads/resources/thesis/May09/Dubrofsky_Elan.pdf. [Accessed 15 1 2015].

[44] "Homogeneous Coordinates," Wolfram, [Online]. Available: http://mathworld.wolfram.com/HomogeneousCoordinates.html. [Accessed 15 1 2015].

[45] "Homogeneous Coordinates," Wikipedia, 1 12 2014. [Online]. Available: http://en.wikipedia.org/wiki/Homogeneous_coordinates. [Accessed 15 1 2015].

[46] "Measuring Vegetation (NDVI & EVI)," NASA, [Online]. Available: http://earthobservatory.nasa.gov/Features/MeasuringVegetation/measuring_vegetation_2.php. [Accessed 7 1 2015].

[47] "NDVI, the Foundation for Remote Sensing Phenology," USGS, 6 1 2011. [Online]. Available: http://phenology.cr.usgs.gov/ndvi_foundation.php. [Accessed 7 1 2015].

[48] "NDVI History," 15 09 2014. [Online]. Available: https://www.maxmax.com/ndv_historyi.htm. [Accessed 7 1 2015].

[49] "How to Store Large Datasets," Matlab Tips, 26 09 2012. [Online]. Available: http://www.matlabtips.com/how-to-store-large-datasets/. [Accessed 22 1 2015].

[50] C. A. V. Aguilera, "Matlab Central," 11 4 2007. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/12275-extrema-m--extrema2-m. [Accessed 8 1 2015].

[51] "Superblue," Public Lab, 20 4 2013. [Online]. Available: http://publiclab.org/notes/cfastie/04-20-2013/superblue. [Accessed 7 1 2015].

[52] "Hough," Mathworks, [Online]. Available: http://www.mathworks.com/help/images/ref/hough.html. [Accessed 15 1 2015].

[53] "Hough Transform," Wikipedia, 14 12 2014. [Online]. Available: http://en.wikipedia.org/wiki/Hough_transform. [Accessed 15 1 2015].

Appendix A: Color Plane Histograms

A1: RGB

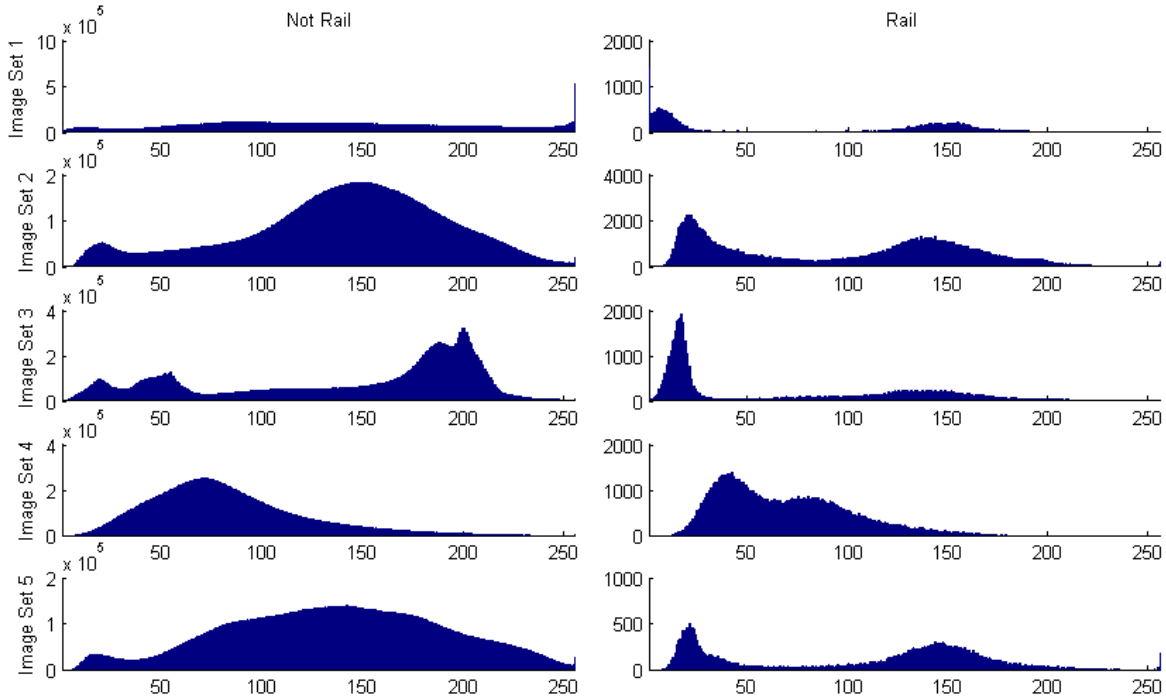

Figure A.1 The histograms created using the red plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of red and the y axis represents the number of rail pixels that contain that value of red. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

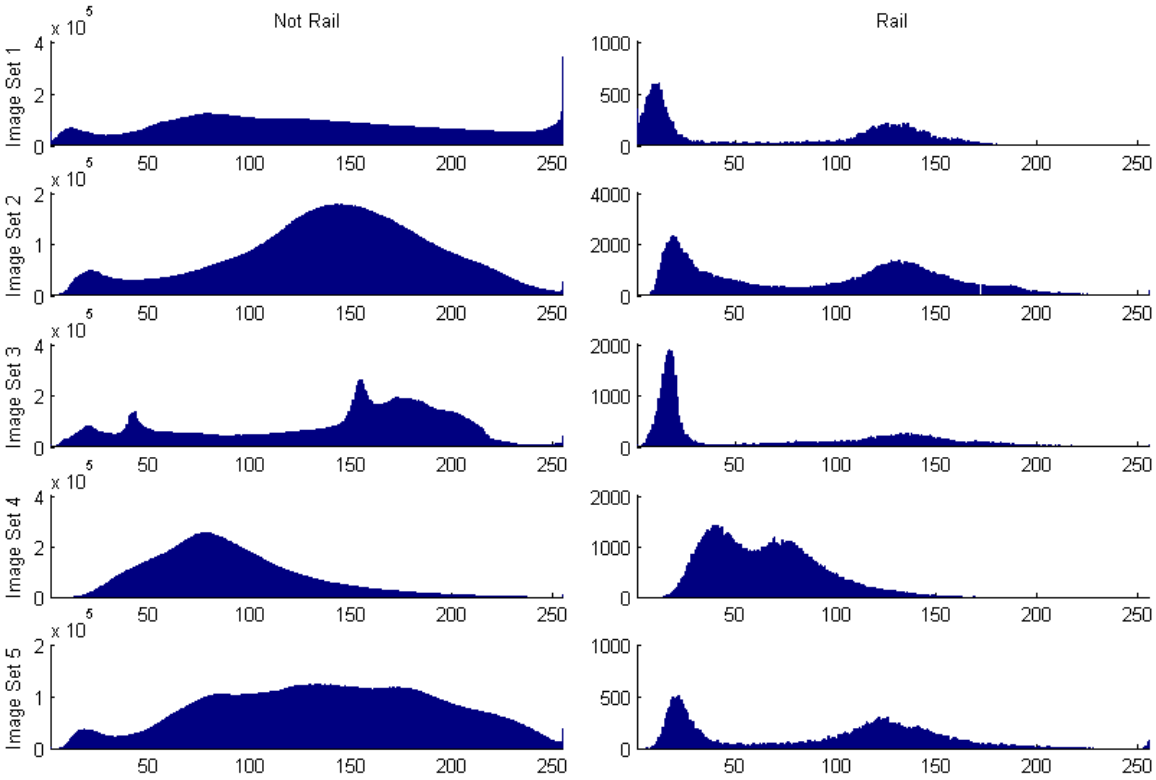

Figure A.2 The histograms created using the green plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of green and the y axis represents the number of rail pixels that contain that value of green. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

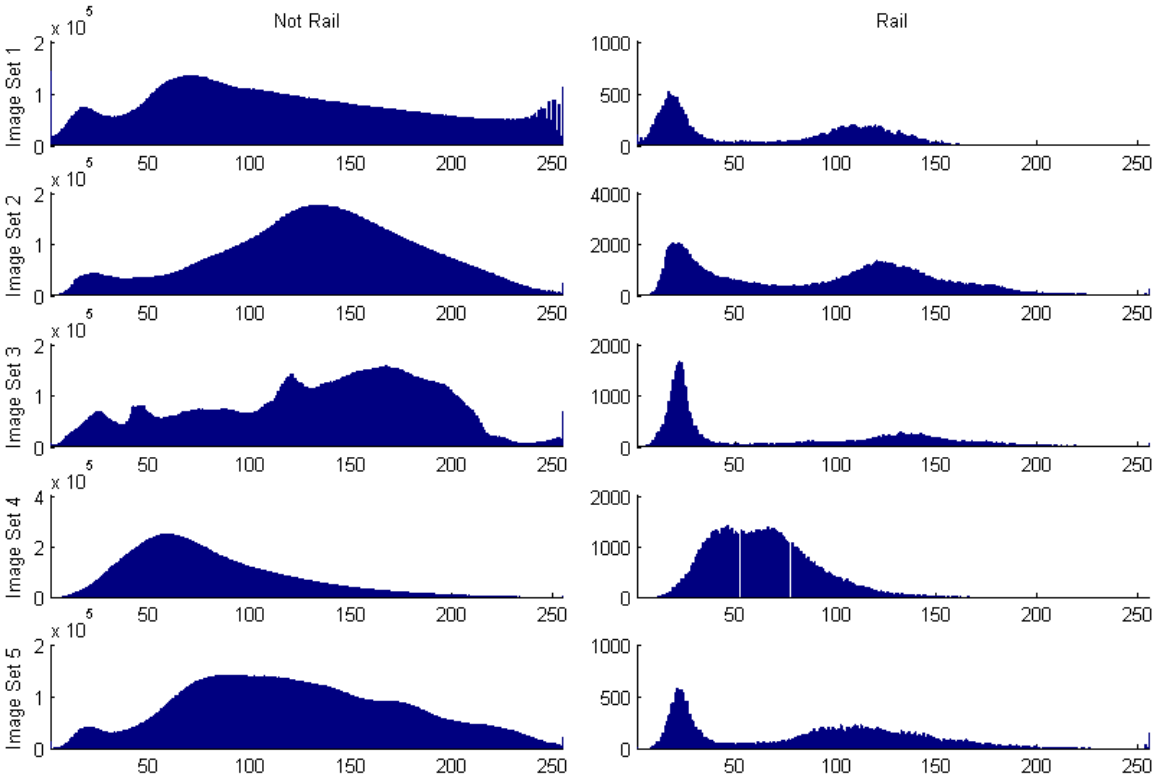

Figure A.3 The histograms created using the blue plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of blue and the y axis represents the number of rail pixels that contain that value of blue. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

A2: HSV

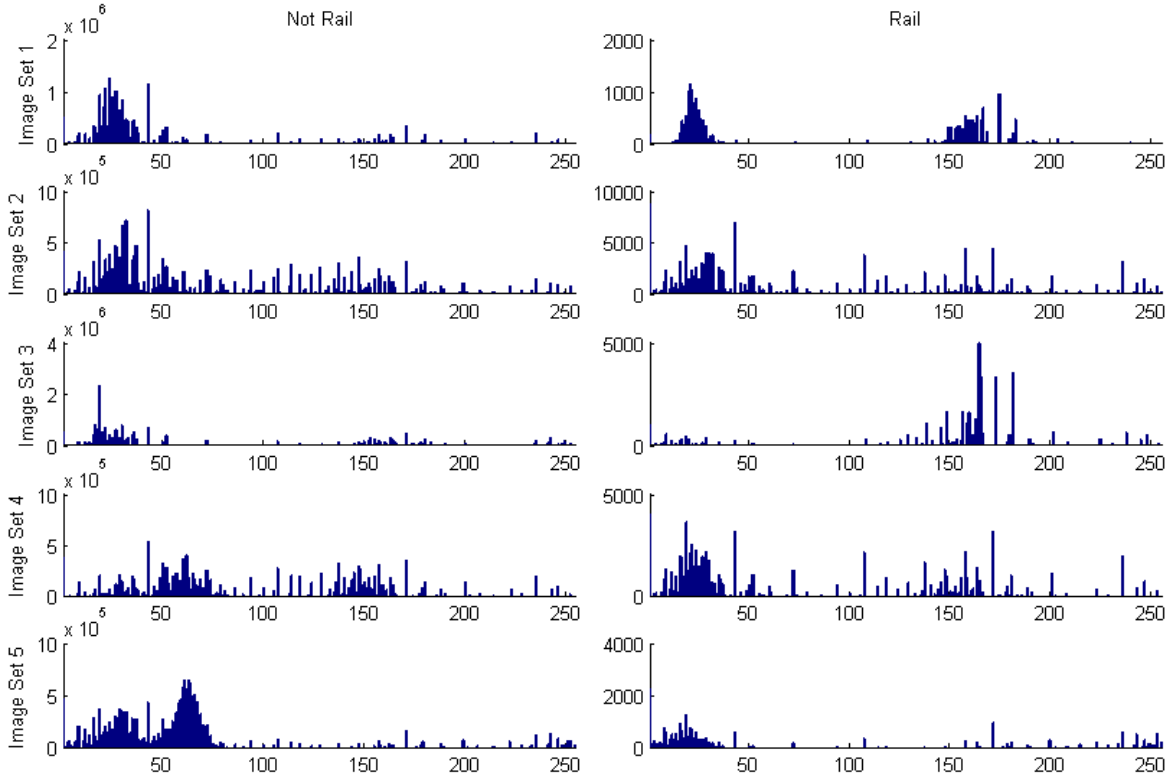

Figure A.4 The histograms created using the hue plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of hue and the y axis represents the number of rail pixels that contain that value of hue. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

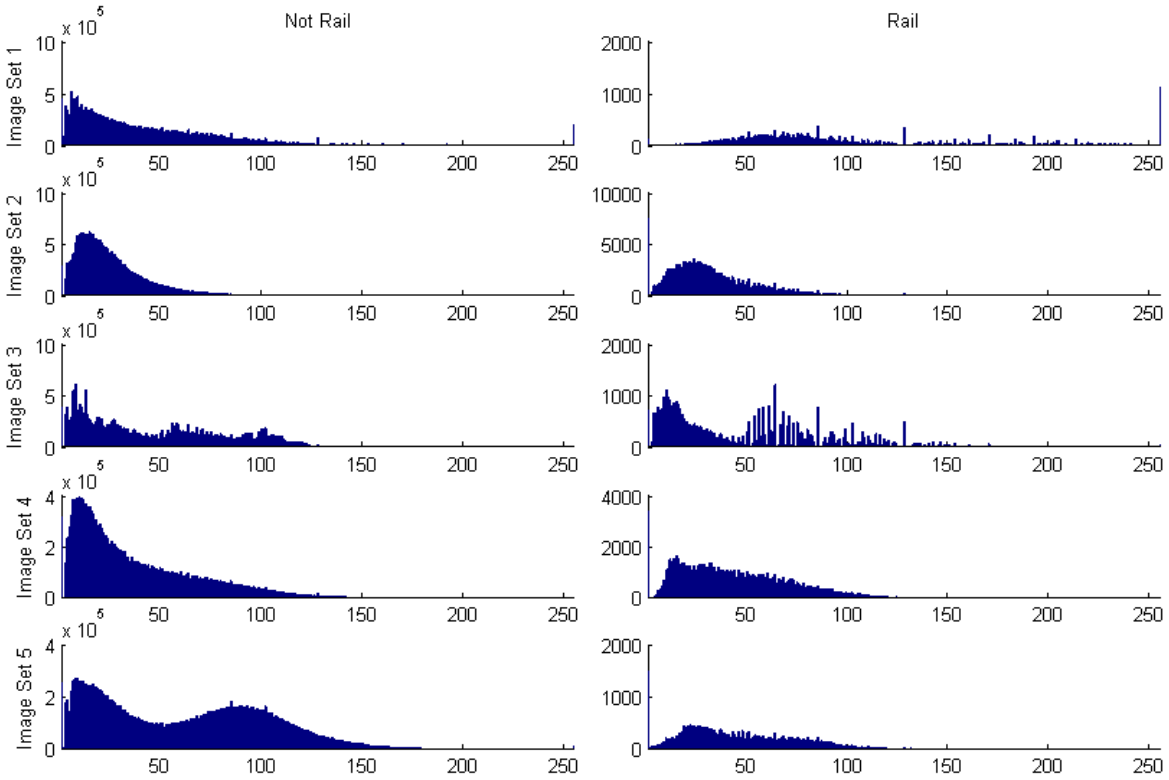

Figure A.5 The histograms created using the saturation plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of saturation and the y axis represents the number of rail pixels that contain that value of saturation. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

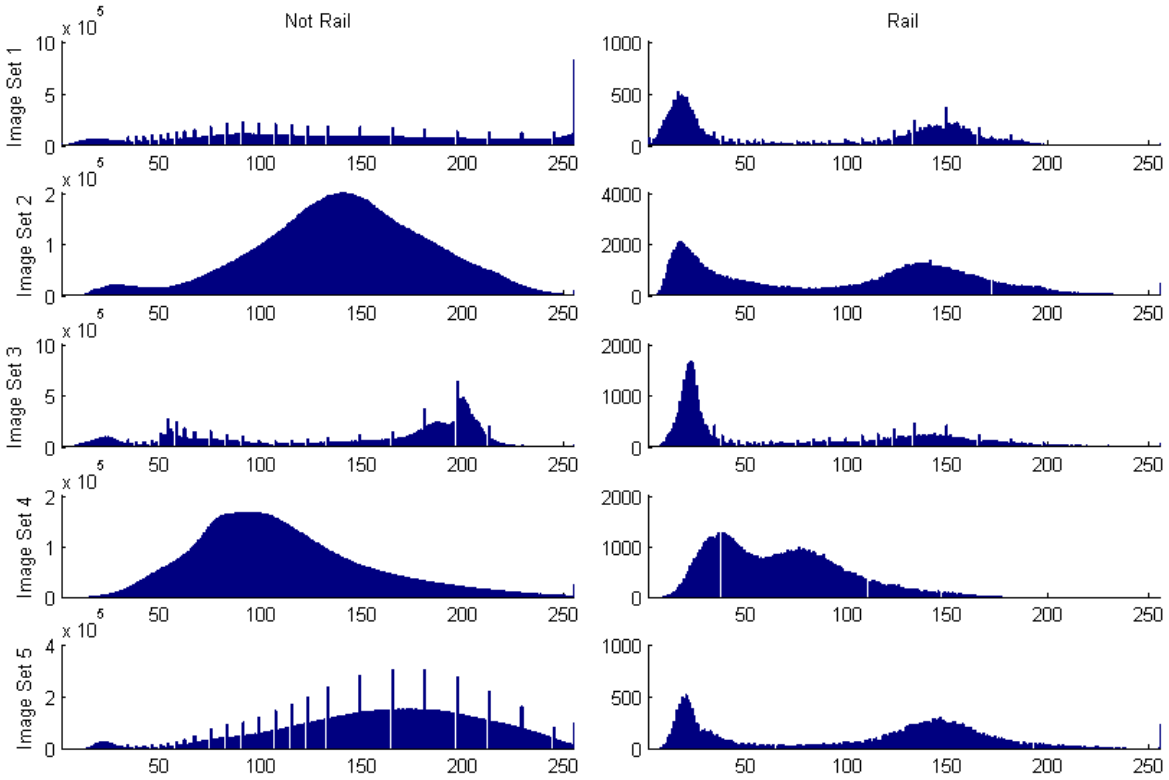

Figure A.6 The histograms created using the value plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of value and the y axis represents the number of rail pixels that contain that value of value. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

A3: Lab

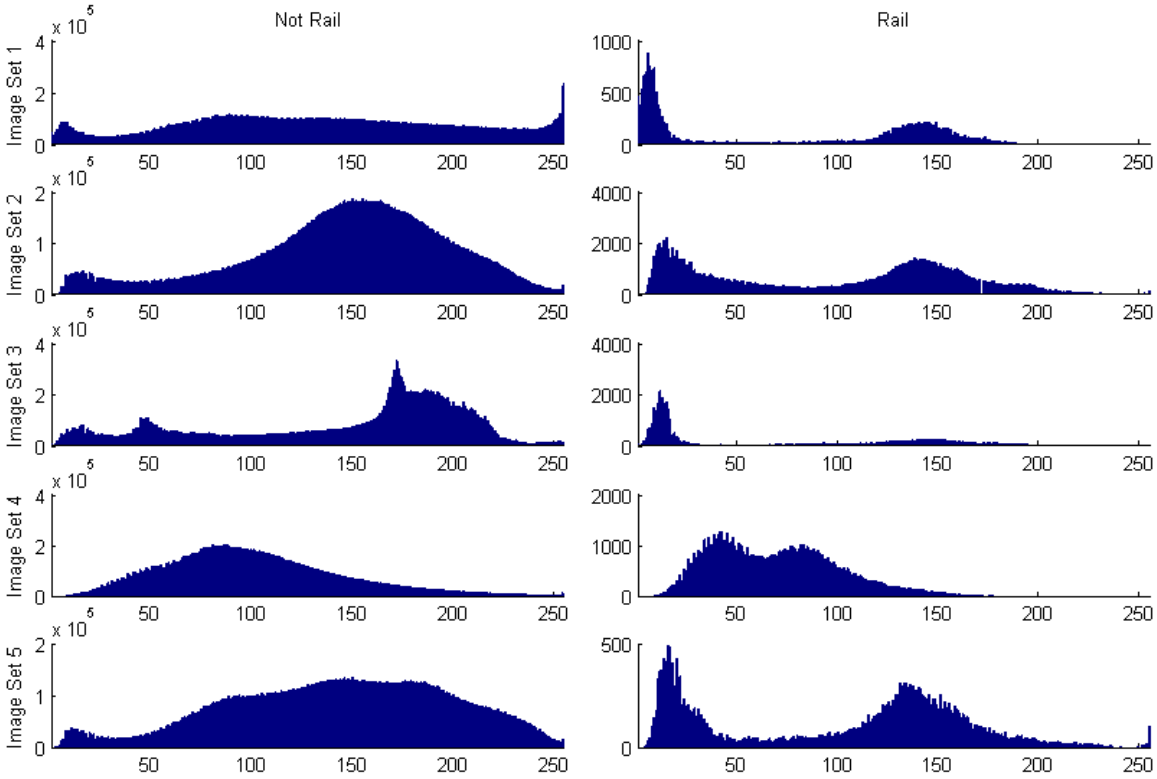

Figure A.7 The histograms created using the light plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of light and the y axis represents the number of rail pixels that contain that value of light. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

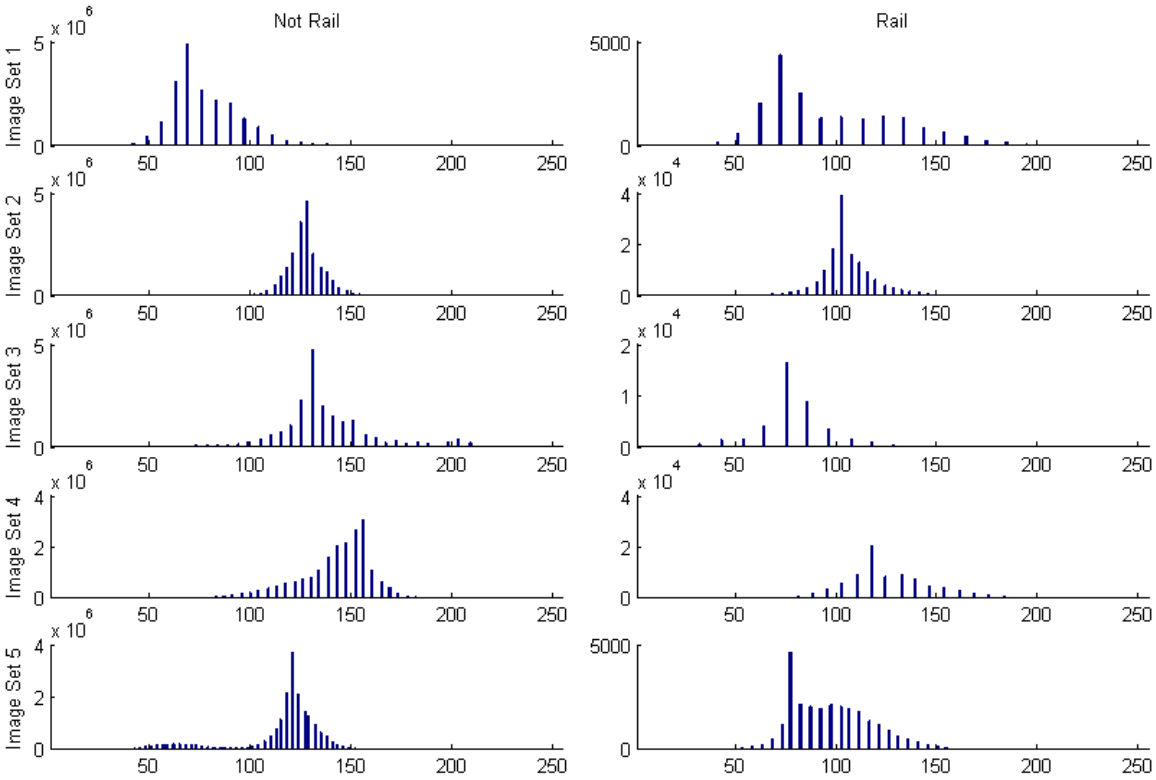

Figure A.8 The histograms created using the aColor plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of aColor and the y axis represents the number of rail pixels that contain that value of aColor. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

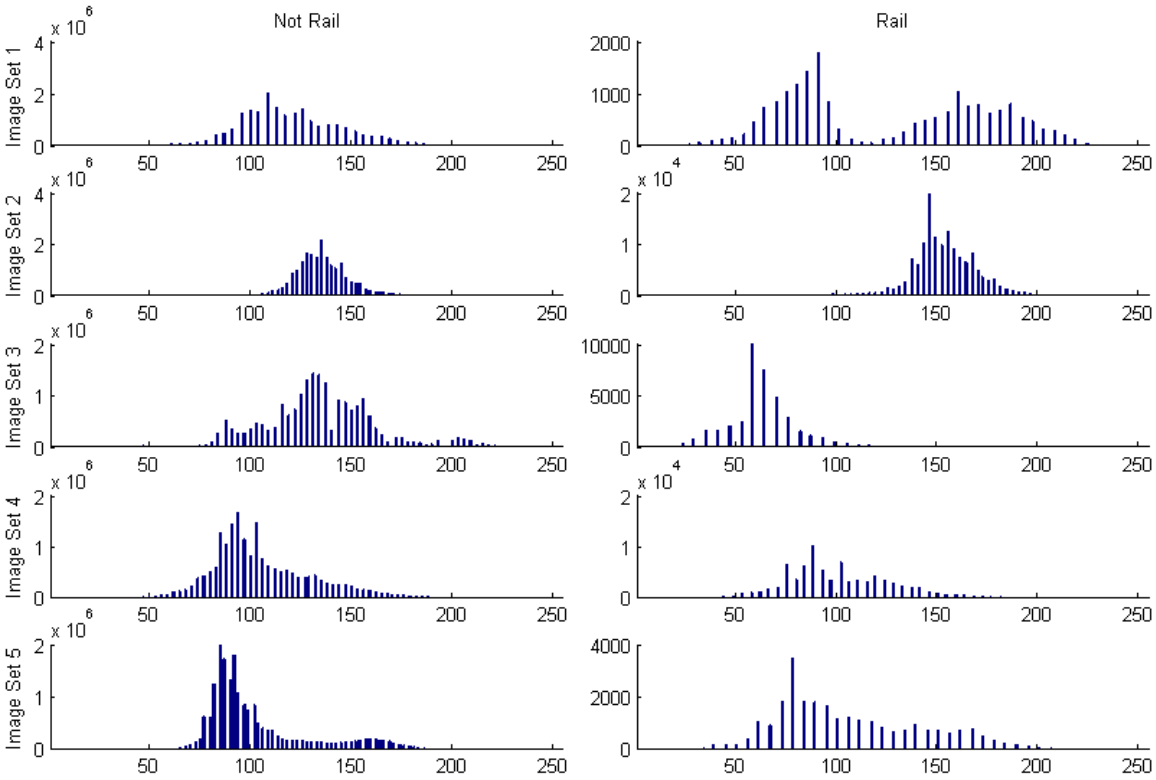

Figure A.9 The histograms created using the bColor plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of bColor and the y axis represents the number of rail pixels that contain that value of bColor. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

A4: CMYK

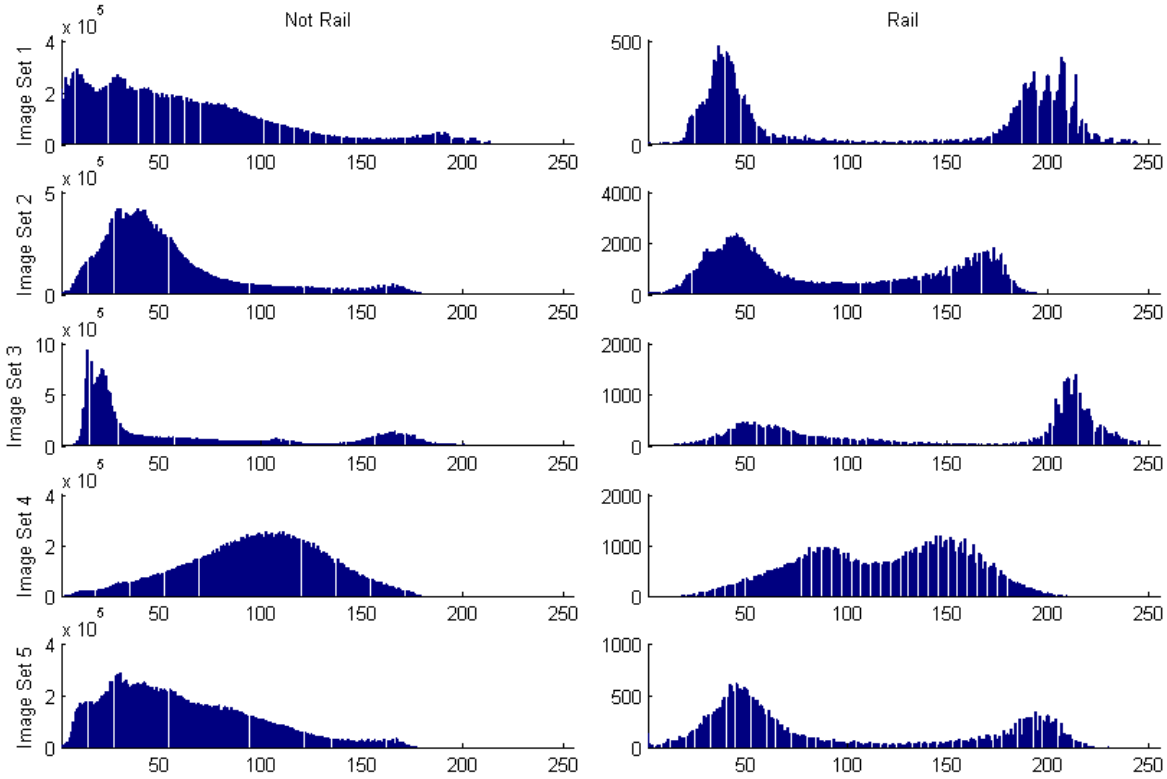

Figure A.10 The histograms created using the cyan plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of cyan and the y axis represents the number of rail pixels that contain that value of cyan. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

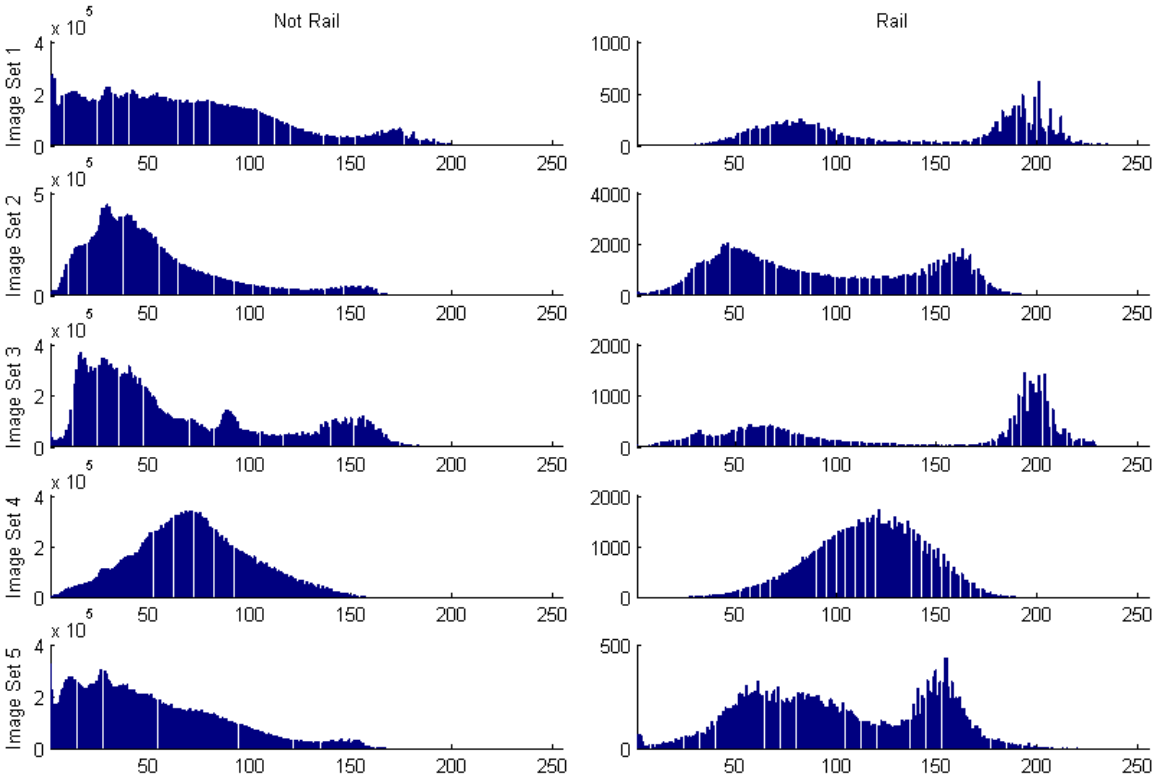

Figure A.11 The histograms created using the magenta plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of magenta and the y axis represents the number of rail pixels that contain that value of magenta. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

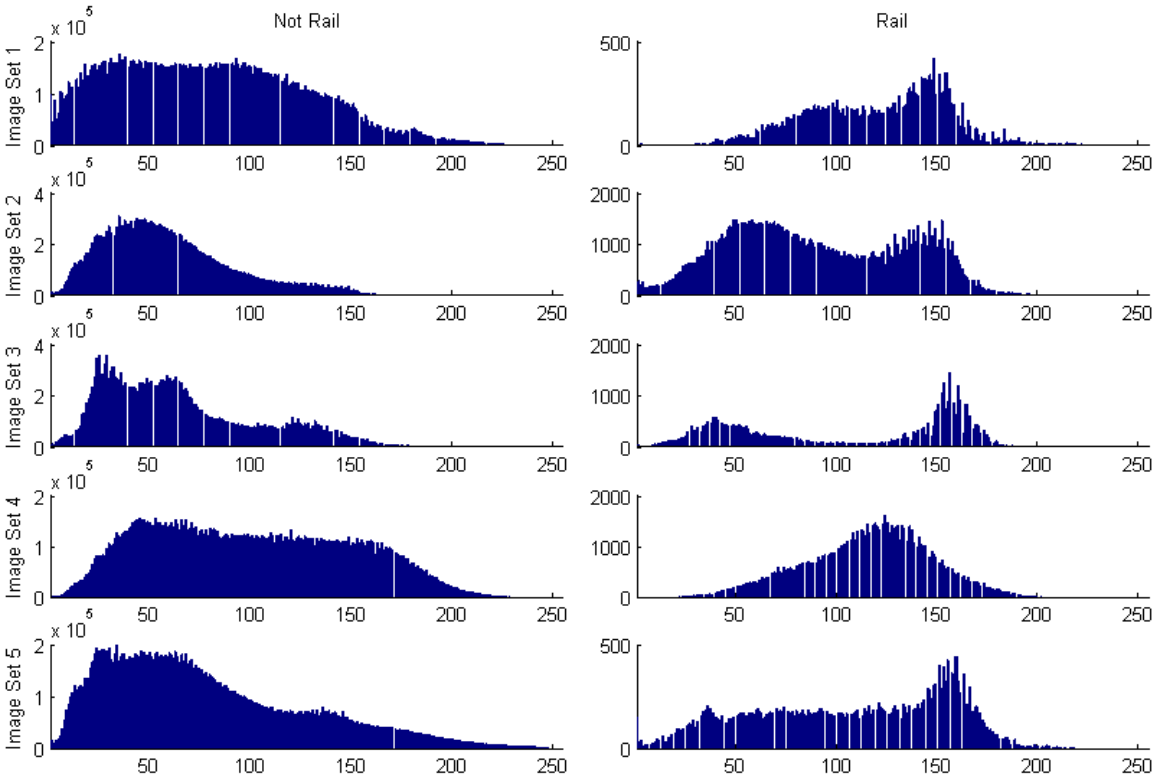

Figure A.12 The histograms created using the yellow plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of yellow and the y axis represents the number of rail pixels that contain that value of yellow. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.

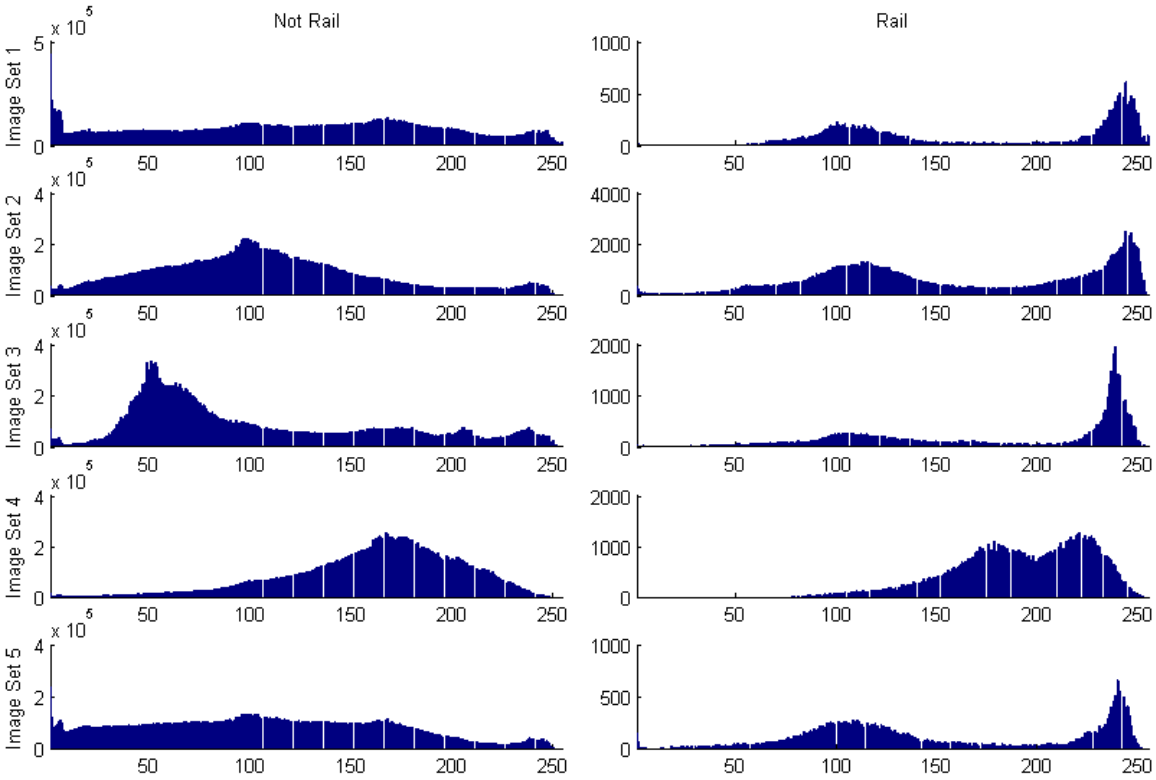

Figure A.13 The histograms created using the key plane of all sets of images. The left column of plots corresponds to the pixels that do not represent rails and the right column of plots corresponds to rail pixels of each image set. Each unit along the x axis represents a value of key and the y axis represents the number of rail pixels that contain that value of key. A histogram was created using each image in the set. These histograms were averaged to create the histograms shown.