AUTOMATING BRIDGE INSPECTION PROCEDURES: REAL-TIME UAS-BASED DETECTION AND TRACKING OF CONCRETE BRIDGE ELEMENT

A Thesis

by

EMILY ELIZABETH MILLER

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Stephanie Paal
Committee Members,	Luciana R. Barroso
	Julian Kang
Head of Department,	Robin Authenrieth

December 2018

Major Subject: Civil Engineering

ABSTRACT

Bridge inspections are necessary to maintain the safety, health, and welfare of the public. All bridges in the United States are federally mandated to undergo routine evaluations to confirm their structural integrity throughout their lifetime. The traditional process implements a bridge inspection team to conduct the inspection, heavily relying on visual measurements and subjective estimates of the existing state of the structure. Conducting unmanned automated bridge inspections would allow for a more efficient, accurate, and safer alternative to traditional bridge inspection procedures. Optimizing bridge inspections in this manner would enable frequent inspections in order to comprehensively monitor the health of bridges and quickly recognize minor problems which could be easily corrected before turning into more critical issues. In order to create an unmanned data acquisition procedure, unmanned aerial vehicles with high-resolution cameras will be employed to collect videos of the bridge under inspection. To automate a bridge inspection procedure employing machine learning methods, such as neural networks, and machine vision methods, such as Hough transform and Canny edge detection, will assist in identifying the entire beam. These methods along with future work in damage detection and assessment will be the main steps to create an unmanned automated bridge inspection.

DEDICATION

I would like to dedicate my Master's Thesis to my loving and supportive parents, Karen and Jim Miller.

ACKNOLEDMENTS

I would like to thank my committee chair, Dr. Stephanie Paal, and my committee members, Dr. Luciana R. Barroso and Dr. Julian Kang for their support and guidance throughout this process.

I would like to thank Michael Baker International for the contribution of donated video data collected by their unmanned aerial system. The video data collected was imperative to this research. I would especially like to thank Michael Baker International for remaining in contact with myself and Dr. Stephanie Paal throughout the progress of this research project.

I would like to thank the Miller family and the Sowden family for their support through this process. I would especially like to thank my mother, Karen Miller, and my father Jim Miller, for their constant help and support throughout my time at Texas A&M University. I would also like to thank my friends, coworkers, and superiors at BGE, Inc. for their words of encouragement and continued guidance. Special thanks to my friends, Ashlin Partin and Codi McKee, for their constant support, encouragement, and grace throughout the process of completing my thesis.

CONTRIBULORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professor Stephanie Paal and Luciana Barroso of the Department of Civil Engineering and Professor Julian Kang of the Department of Construction Science.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University Department of Civil Engineering and a fellowship from Peter C. Forester Foundation.

NOMENCLATURE

AASHTO	American Association of Highway and Transportation Operations
ANN	Artificial Neural Network
ASCE	American Society of Civil Engineers
BIRM	Bridge Inspector's Reference Manual
CFR	Code of Federal Regulations
CNN	Convolutional Neural Network
DoH	Determinate of the Hessian
DOT	Department of Transportation
EB	East Bound
FHWA	Federal Highway Administration
HIFLD	Homeland Infrastructure Foundation – Level Data
HPRC	High-Performance Research Computing
IDEA	Innovations Deserving Exploratory Analysis
IoU	Intersection-over-Union
K-NN	K-Nearest Neighbor
LoG	Laplacian of Gaussian
LR	Logistic Regression
MSER	Maximally Stable Extremal Regions
MCD	Maximum Cluster Dimensions
MSAC	M-estimator Sample Consensus

NB	Naïve Bayes
NBI	National Bridge Inventory
NCHRP	National Cooperative Highway Research Program
NHS	National Highway System
NICET	National Certification in Engineering Technologies
NMS	Non-maximum Suppression
OSHA	Occupational Safety and Health Administration
PCI	Prestressed Concrete Institute
PennDOT	Pennsylvania Department of Transportation
PMCD	Perpendicular Maximum Cluster Dimension
polyROI	Polygon Region of Interest
pROI	Proposed Region of Interest
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
ROI	Region of Interest
R-CNN	Region-based Convolutional Neural Network
SD	Structurally Deficient
SHM	Structural Health Monitoring
SIFT	Scale Invariant Feature Transform
SGD	Stochastic Gradient Descent
SGDM	Stochastic Gradient Descent with Momentum
SSD	Sum of Square Differences

SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
TMP	Traffic Management Plan
TVA	Tennessee Valley Authority
TxDOT	Texas Department of Transportation
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
U-BIROS	Ubiquitous Bridge Inspection Robot System
WB	West Bound

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOLEDMENTS iv

CONTRIBULORS AND FUNDING SOURCES v

NOMENCLATURE vi

TABLE OF CONTENTS ix

LIST OF FIGURES xii

LIST OF TABLES xvi

1. INTRODUCTION 1

1.1. Motivation 1
1.2. Research Objectives 7
1.3. Research Overview 8

2. LITERATURE REVIEW 10

2.1. A First Level Subheading 10
2.2. Concrete Beam Inspection 18
2.3. Unmanned Data Acquisition For Structural Health Monitoring 24
2.4. Structural Element Detection 28
2.5. Tracking Methods Used in Civil Engineering 34

3. BACKGROUND 37

3.1. Machine Vision Algorithms 37
3.1.1. Canny Edge Detector 37
3.1.2. Hough Transform 45
3.1.3. Speeded-Up Robust Features 47
3.1.4. Projective Transformation 56
3.2. Machine Learning Algorithms 59
3.2.1. Convolutional Neural Network 59

3.2.2. Region-based Convolutional Neural Network 69

4. METHODOLOGY 76

4.1. Overview 76
4.2. Dataset 79
4.3. Data Augmentation 85

5. OBJECT DETECTION 89

5.1. Prepare Training and Testing Set 90
5.2. Train and Implement a Region-based Convolutional Neural Network 92
5.3. Full Beam Detection (pROI to ROI) 98
5.3.1. Step one: Locate Beam Edges Within pROI 99
5.3.2. Extend pROI to Image Edges and Mask polyROI 103
5.3.3. Step Three: Locate Beam End and Establish Final ROI 107

6. TRACKING 110

6.1. Feature Extraction and Matching 112
6.2. Locate Full Beam in Subsequent Frame 114

7. IMPLEMENTATION AND RESULTS 116

7.1. Implementation 116
7.2. Results: R-CNN 117
7.2.1. Cross-validation of R-CNN 117
7.2.2. R-CNN Evaluation 120
7.2.3. Analysis of R-CNN Results via Alternate Approaches 127
7.2.4. Discussion of the Impact of Data Augmentation on Accuracy 129
7.3. Results: Full Beam (ROI) 130
7.3.1. Full Beam Detection Evaluation 130
7.3.2. Analysis of Full Beam Detection Results 133
7.4. Results: Tracking 134
7.4.1. Tracking Evaluation 134
7.4.2. Analysis of Tracking Results 139

8. CONCLUSIONS AND FUTURE WORK 140

8.1. Summary 140
8.2. Future Work 142
8.2.1. Limitations 142
8.2.2. Recommendations 142
8.2.3. Practical Implementations 144
8.3. Lessons Learned 145

REFERENCES 148

APPENDIX A Table A-1. performance of R-CNN on images in testing set. 160

Table A-2. Detailed results of tracking algorithm 161

Table A-3. Detailed results of R-CNN2 162

Table A-4. Detailed results of R-CNN3 163

APPENDIX B TRAINING THE R-CNN OBJECT DETECTOR 165

APPENDIX C FULL BEAM DETECTION STEP ONE 166

APPENDIX D FULL BEAM DETECTION STEP TWO 170

APPENDIX E FULL BEAM DETECTION STEP THREE 175

APPENDIX F TRACKING ALGORITHM 179

LIST OF FIGURES

Page

Figure 1.1. Bridge condition by functional classification (NBI, 2017). 2

Figure 1.2. Structurally deficient bridge count reported year 2013 through year 2017. ... 4

Figure 1.3. Bridge inspection team members inspecting tall structures (FHWA, 2012)... 6

Figure 1.4. Bridge inspection team implementing a traffic control plan (FHWA, 2012)..... 6

Figure 2.1. Total bridge count and structurally deficient bridge count based on the substructure (NBI, 2016)..... 21

Figure 2.2. Total bridge count and structurally deficient bridge count based on material type concrete and prestressed concrete (NBI, 2016)..... 22

Figure 2.3. AASHTO cross section of prestressed I beams (AASHTO, 2011). 23

Figure 3.1. Original image before a Gaussian filter is applied..... 39

Figure 3.2. Smoothed image ($\sigma = 4$)..... 39

Figure 3.3. The smoothed gradient of the original image in the horizontal direction. 41

Figure 3.4. The smoothed gradient of the original image along the vertical direction.... 42

Figure 3.5. Gradient magnitude of the original image..... 42

Figure 3.6. Original image after non maximum edge thinning. 44

Figure 3.7. Canny edge detection image result. 45

Figure 3.8. Left: original image, and Right: integral image..... 52

Figure 3.9. The structure of an RGB image. 60

Figure 3.10. Layer architecture of AlexNet CNN (Krizhevsky et al., 2012). 67

Figure 3.11. The detailed architecture of AlexNet CNN (Krizhevsky et al., 2012)..... 68

Figure 3.12. An example of comparing a ground-truth label (purple) and a candidate (yellow). 70

Figure 3.13. An example of determining the intersection (left) and union (right) for a ground truth and a candidate. 71

Figure 3.14. An example of a trained R CNN detector with application in the images prevalent to the work presented in this thesis..... 75

Figure 4.1. Overview of bridge beam detection and tracking methodology..... 78

Figure 4.2. Map of Pennsylvania locating Monroe County in red. 80

Figure 4.3. Bridge detail: SR 80 (LR 794)..... 80

Figure 4.4. Bridge detail: I 80 WB (LR1009) and I 80 EB (LR1009)..... 81

Figure 4.5. Manually labeled regions of interest for positive object classes. 85

Figure 4.6. Upper left corner crop example. 86

Figure 4.7. Four corner crop (all colors but white) and middle section crop (white) example. 87

Figure 4.8. Samples of different data augmentation techniques performed on the image dataset. 88

Figure 5.1. Overview of the proposed method for structural element detection..... 90

Figure 5.2. Training set image total after data augmentation..... 92

Figure 5.3. R CNN object localization output, showing pROIs..... 98

Figure 5.4. Lines detected via Canny + Hough Transform, Left: Canny edge detector without adaptive threshold; Right: Canny edge detector with adaptive threshold..... 102

Figure 5.5. Grayscale pROI image with superimposed detected beam flange edges.... 102

Figure 5.6. Example of proposed beam flange edges from all pROI detected in the image..... 103

Figure 5.7. Example of paired line segments extended to image border..... 104

Figure 5.8. Example polyROI of extended lines without detected joints. 104

Figure 5.9. Example of extended lines without acknowledged detected joints. 106

Figure 5.10. Example of polyROI of extended lines without acknowledged detected joints. 106

Figure 5.11. Example of polyROI extended lines with acknowledged detected joints. 107

Figure 5.12. Example of the final ROI fused with the original image. 109

Figure 6.1. Proposed method of structural element tracking. 111

Figure 6.2. Masked image key points (Left); subsequent image key points (Right). The matched features are shown by the yellow lines. 113

Figure 6.3. The original image containing only the beam flange (gray) projected onto the subsequent image (red). 115

Figure 7.1. DJI Inspirte 2 with Zenmuse X5s sensor. 116

Figure 7.2. One instance of 5 fold cross validation procedure. 118

Figure 7.3. Example of manually masked beam flanges. 121

Figure 7.4. R CNN example output images with annotated pROIs: Example 1. 122

Figure 7.5. R CNN example output images with annotated pROIs: Example 2. 122

Figure 7.6. R CNN example output images with annotated pROIs: Example 3. 123

Figure 7.7. R CNN example output images with annotated pROIs: Example 4. 123

Figure 7.8. R CNN detection of beam flange: Example 1. 125

Figure 7.9. R CNN detection of beam flange: Example 2. 126

Figure 7.10. R CNN detection of beam flange: Example 3. 126

Figure 7.11. R CNN detection of beam flange: Example 4. 127

Figure 7.12. Full beam detection of beam flange: Example 1. 131

Figure 7.13. Full beam detection of beam flange: Example 2. 132

Figure 7.14. Full beam detection of beam flange: Example 3. 132

Figure 7.15. Full beam detection of beam flange: Example 4. 133

Figure 7.16. Full beam detection of beam flange. 134

Figure 7.17. Detailed results of original and subsequent image key point comparison. 136

Figure 7.18. Matched and inlier point pairs for the original and subsequent images.... 137

Figure 7.19. Original image key points (red circles) mapped (yellow line) to the subsequent image's key points (green exes): Example 1..... 138

Figure 7.20. Original image key points (red circles) mapped (yellow line) to the subsequent image's key points (green exes): Example 2..... 139

LIST OF TABLES

Page

Table 1.1. Total count of bridges and number of structurally deficient bridges by time period. 4

Table 2.1. Team Leader Qualifications (CFR 650.309). 13

Table 2.2. Condition ratings for deck, superstructure, and substructure (FHWA, 2012). 16

Table 2.3. Kinds of Bridge Construction Material and/or Design (Farhey, 2010). 19

Table 2.4. Cost of recourse utilization of bridge inspection case study (Chan et al., 2015). 26

Table 4.1. Relevant bridge details extracted from HIFLD (HIFLD, 2017). 82

Table 4.2. Image dataset for ground truth labeled regions of interest. 84

Table 5.1. Image training set for ground truth labeled regions of interest. 91

Table 5.2. Image testing set ground truth labeled regions of interest. 91

Table 7.1. Classification error rate on partitioned testing set. 119

Table 7.2. Classification error rate on entire dataset. 119

Table 7.3. Confusion matrix for TP, FP, FN, and TN performance of R CNN. 124

Table 7.4. R-CNN training options and training set details. 128

Table 7.5. Confusion matrix of R CNN2. 129

Table 7.6. Confusion matrix of R CNN3. 129

1. INTRODUCTION

Bridge inspections are necessary to maintain current information regarding the structural state, capacity, and safety of all the nation's bridges. The Federal Highway Administration (FHWA) mandates that each bridge in the nation be inspected by a certified team of experts within a minimum of every two years. The inspection team visually evaluates and measures the condition of the bridge's structural components and assigns a level of service to the bridge as a whole. These findings are used to determine the functional classification of the bridge, calculate sufficiency ratings, and decide if any critical measures need to be taken. Implementing an unmanned automated bridge inspection procedure to replace the traditional procedure brings about the opportunity to address current issues in the existing approach. In order to automate inspections, algorithms identifying and tracking the critical structural elements need to be created. In this research, a tool is created to detect and track concrete beams within bridge spans in unmanned aerial system (UAS) imagery.

1.1. Motivation

Routine bridge inspections are crucial to verify or update the current functional category by comparing the existing condition to the condition documented in the initial or previous routine inspections. The state of every bridge in the United States is documented in the National Bridge Inventory (NBI) database which is constructed from data submitted to FHWA from all department of transportation agencies (local, state, federal). The bridge condition is based on the state of the deck, superstructure,

substructure, and culvert (if present), and it is defined as one of the following: (1) **Good,** if no more than minor problems are detected; (2) **Fair,** if all primary structural elements are sound but may have minor section loss, cracking, spalling, or scour; or (3) **Poor,** if advanced section loss, deterioration, spalling, or scour is detected (FHWA, 2012).

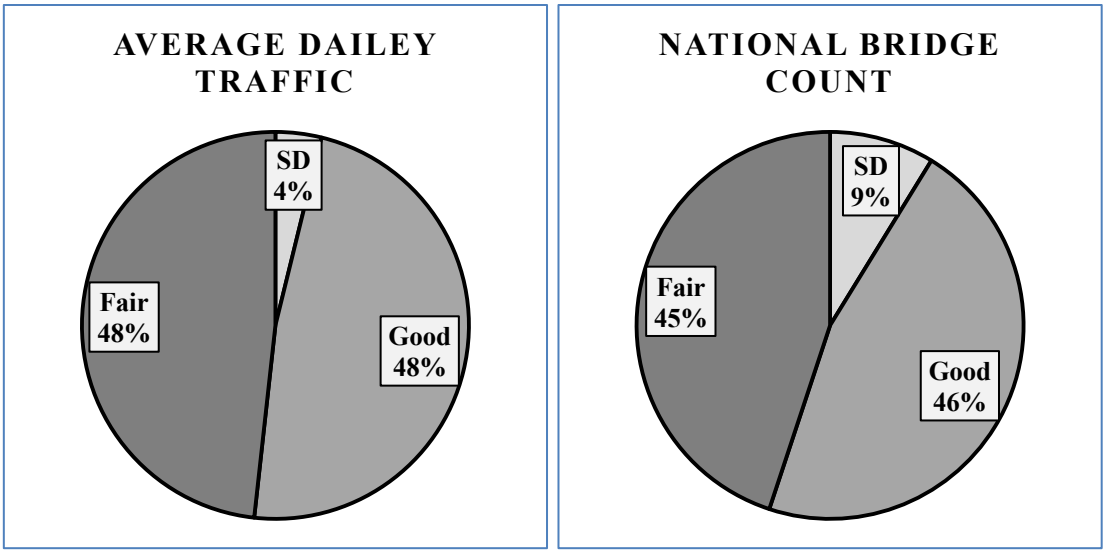

Figure 1.1. Bridge condition by functional classification (NBI, 2017).

The national bridge count and annual average daily traffic of bridges by functional classification, retrieved from the NBI database, is displayed in Figure 1.1 (NBI, 2017). In 2017, NBI documented a total count of 614,978 national bridges, and of these national bridges, 47,619 were considered in poor condition. Also documented 2017, the total annual average daily traffic on all national bridges was estimated at 4.5 billion trips, and 173 million of those daily trips were on bridges considered in **poor** condition (NBI, 2017).

The American Society of Civil Engineers (ASCE) publishes a report card grading the state of the nations' infrastructure for the previous year. The most recent ASCE infrastructure report card was published in 2017 regarding the state of the national infrastructure as of 2016. The ASCE reported that 56,007 (9.1%) of a total 614,387 bridges were considered structurally deficient in 2016, and every day there was an average of 188 million trips traveled across structurally deficient bridges (ASCE, 2017). Although both the percentage and number of structurally deficient bridges are decreasing (Figure 1.2), currently four in ten bridges are 50 years or older, and are reaching the end of their design life. The total count of bridges and the number of structurally deficient bridges itemized by age range are detailed in Table 1.1. As is evident in this table, the number of bridges 47+ years old that are structurally deficient account for nearly 78% of the total number of structurally deficient bridges.

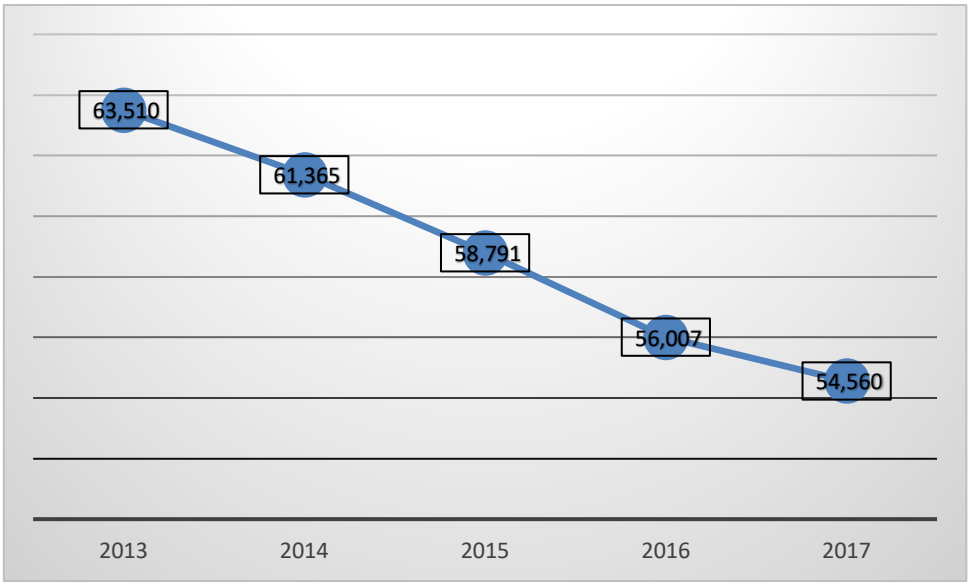

Figure 1.2. Structurally deficient bridge count reported year 2013 through year 2017.

Table 1.1. Total count of bridges and number of structurally deficient bridges by time period.

	2010-2016	2000-2009	1990-1999	1980-1989	1970-1979	1969 and earlier
All Bridges	38,038	71,475	81,410	78,279	82,129	427,367
SD	10	484	1,818	3,298	6,510	42,341

The backlog of bridge rehabilitation from the 2017 ASCE report card has been projected to cost approximately $123 billion, which has increased from $76 billion as of the 2013 ASCE infrastructure report card (ASCE, 2017; ASCE, 2013). Although the percent of structurally deficient bridges has seen a steady decrease, 10.5% in 2013 to 9.1% in 2067, the cost to rehabilitate these bridges has increased by $47 billion.

The National Cooperative Highway Research Program (NCHRP) Highway Innovations Deserving Exploratory Analysis (IDEA) program (2018) is focused on identifying ways

to advance the safety, maintenance, and management of the highway system. Highway IDEA Project 56 (NCHRP, 2000) concluded that the cost to implement traffic control and equipment allowing an inspector to obtain access needed for a detailed evaluation accounts for approximately 40-50% of the cost of the entire bridge inspection (Choset, 2000).

An audit of the Texas Department of Transportation (TxDOT) bridge inspection program (State Auditor's Office, 2009) analyzed the performance of five inspection teams inspecting 303 bridges. Of the 303 bridges, 203 (67%) were inspected within the mandated 24-month period, while 100 (33%) failed to meet this requirement. The audit recognized the main reason for the delayed bridge inspection was caused by a lack of available consulting engineers (92.8% of routine bridge inspections are contracted out to consulting engineers).

The required procedure of routine bridge inspections result in a time-consuming process. Inspection teams must prepare and plan all aspects of the inspection procedure, spend adequate time analyzing the structure, and create documentation of the overall inspection process and resulting decision. According to Highway IDEA Project 56 (Choset, 2000), approximately 40% of the total time invested in bridge inspections is allocated to the set-up and maneuvering of teams and equipment to inspect under-bridge components. Another time-consuming aspect is due to the risk associated with bridge inspections. Risk, such as scaling tall structures (Figure 1.3) and conducting inspections over roads with a high volume of traffic (Figure 1.4), requires the team to spend additional time planning and setting up bridge inspection and safety equipment, traffic

control plans, and safety precautions. In addition to the time required to plan and prepare a bridge inspection, there is a shortage of available consulting engineers.

Figure 1.3. Bridge inspection team members inspecting tall structures (FHWA, 2012).

Figure 1.4. Bridge inspection team implementing a traffic control plan (FHWA, 2012).

In a routine bridge inspection, various documents, such as, pictures and previous reports, aid in assessing the current bridge condition. Routine bridge inspections require photos of the as-built state, meaning the structure design as it is currently built (original design, change orders, maintenance), and the previous reports note the condition(s) of the structure elements. Photos remain essential to future bridge inspection comparisons.

Comparing the previous condition of the bridge to the current condition gauges the bridge's deterioration rate and can help to monitor damage propagation in order to effectively monitor for serious problems. The State Auditor Office (2009) concluded that four of the five district offices in Texas had up to a 6.5% error rate, and one had a 25% error rate. The report attributed the error to absent bridge inspection photographs on record.

1.2. Research Objectives

The objective of this research is to create a means of rapidly and automatically locating and tracking critical components (beams) in bridges in unmanned aerial system (UAS) imagery. This project is a step towards the ultimate goal of developing a fully automated unmanned bridge inspection procedure which involves detecting and tracking critical components as well as the damage to those components in addition to correlating this information to provide a real-time evaluation at the component and full system level. The proposed method implements a machine learning algorithm known as a region-based convolution neural network (R-CNN) to first identify a segment of the beam flange region. Then, specific machine vision algorithms are created to extend these segments to the full beam regions by way of identifying beam edges and consideration of typical beam properties. Finally, a feature-based tracking algorithm is implemented to locate the beam in subsequent images in order to assure all beams in a span are detected and to reduce the computational complexity of the overall approach, leading to a more rapid beam detection algorithm overall and one that is capable of being executed in the UAS video imagery.

1.3. Research Overview

Section 2 presents a literature review of the current practice of bridge inspections, recent research efforts towards improving the current procedure, as well as recent research efforts related to the more general challenges associated with structural element detection and object tracking methods in civil engineering. Section 3 provides necessary background information regarding the machine intelligence techniques that are implemented in the object detection portion of the research presented in this thesis. This includes machine vision techniques, such as the Canny edge detection method and Hough transform, and machine learning techniques, such as convolutional neural networks (CNNs) and region-based convolutional neural networks (R-CNNs).

Section 4 presents an overview of the research methodology and the dataset developed for and utilized in the construction, validation, and verification of the proposed methodology. Additionally, this section includes an overview of the data augmentation practices implemented to extend the dataset, accounting for variable environmental and image-related factors to provide ensure that the developed approach is robust.

Sections 5 and 6 detail the methodology developed in this work as introduced in Section 4. Section 5 presents a detailed description of the object detection algorithm of the overall method. This section details the process of preparing the dataset to be used to train the region-based convolutional neural network (R-CNN), the process of training and implementing an R-CNN, and the full beam detection algorithm developed. Section 6 reviews the developed object tracking algorithm of the overall method. The

procedure for the feature-based tracking method and the projective transformation matrix utilized to project the image from one frame to the next is explained in depth in this section.

In Section 7, the implementation and results of the methodology developed in this work are presented. The details regarding the programming languages, toolboxes, and platforms are presented first, and the results of each stage of the methodology follow. The performance of the object detection algorithm is presented using the mean average precision metric for the R-CNN and the accuracy metric for the full beam detection algorithm. The tracking results are presented in the form of the number of key features matched in subsequent images. This number is an adequate performance metric as it describes the number of points needed to create the transformation matrix.

Section 8 concludes the description of the research with a summary overview of the process, the overall beam detection and tracking methodology, and a discussion of the findings. The full potential for implementation of these algorithms and the overall methodology in unmanned automated bridge inspections is discussed, and needs for future work are identified.

2. LITERATURE REVIEW

2.1. A First Level Subheading

Bridge inspectors perform an imperative role in maintaining the safety and reliability of critical infrastructure. The Federal Highway Administration (FHWA) requires bridge inspections performed by an inspection team to adequately gauge the current state of all bridges nationwide. Bridge inspections require a certified and highly skilled inspection team to visually assess and measure the condition of the structural components. In conjunction, the inspection team appraises the existing level of service the bridge provides (FHWA, 2012). The findings from these inspections provide an indication of the functional classification of the bridge, a calculation of the sufficiency rating, and inform either the frequency of following inspections or whether or not the bridge requires critical measures.

In the *Bridge Inspector's Reference Manual* (BIRM) (FHWA, 2012), FHWA defines seven categorical bridge inspection types: (1) Initial – provide Structure Inventory and Appraisal data and a baseline of structural conditions and identify existing problems; (2) Routine – verify or update the current functional category by comparing the current condition to the condition documented in the initial or previous routine inspections; (3) Damage – unscheduled inspection to assess structural damage; (4) In-Depth – identify deficiencies of one or more members above or below the water level not readily evident in routine inspections (2); (5) Fracture Critical – important for steel tension members; (6) Underwater – inspection of underwater portion of elements when

necessary—typically requires diving or other procedures; and (7) Special – used to monitor known or suspected deficiencies (FHWA, 2012). The BIRM Topic 4 provides a detailed procedure for performing structural inventory, condition and appraisal, record keeping and documentation, critical findings, and inspection reports, and in subsequent Topics, the procedure and standards for inspecting material, structure type, etc. have been standardized (FHWA, 2012). As the procedure and standards of bridge inspections are explicitly stated, this will allow for the traditional approach to be directly replaced with an automated inspection with ease. Routine bridge inspections serve as the use-case in the work presented in this thesis; however, the approach developed with this research could be adapted and augmented to address all bridge inspection types. Currently, the BIRM mandates routine inspections on every bridge to be performed within 24-months of the last inspection. FHWA mandates the inspection frequency may be reduced depending on the bridge age, traffic demands, and recorded deficiencies or increased to 48 months if inspection results show that the bridge is low-risk.

Bridge inspections are conducted by a team consisting of a Program Manager, a Team Leader, and one or more Inspectors. The *Code of Federal Regulations* (CFR 650.309) details minimum standards for each member of the program. The Program Manager must be a licensed Professional Engineer and complete a FHWA approved bridge inspection training course. In order to be a Team Leader, the inspector should have the same licensure and course completion as the Program Manager or must satisfy other qualifications listed in the BIRM (Table 2.1) (FHWA, 2012). The team manager is tasked with organizing, preparing, and completing the bridge inspection, and

must remain on-site for the entirety of the bridge inspection. Inspectors are trained on-site by the team leader and other skilled inspectors and are not explicitly required (although it is highly recommended) to seek specific bridge inspector training (FHWA, 2012). Responsibilities of a bridge inspection team include upholding the public safety and confidence, protecting public investment, and maintaining precise bridge records. The protection of public safety and confidence remains a bridge inspectors' most important responsibility. To do this, bridge inspectors must accurately and carefully survey and record existing bridges and report all findings through the correct channels. The building, rehabilitating, and maintaining of national bridges is ultimately publicly funded. Therefore, in order to protect the public's investment, inspectors must promptly identify concerns to mitigate potentially expensive and time-consuming repairs. An automated procedure would greatly increase the probability of detecting minor issues before they become critical and would sustain efficient, safe, and cost-effective monitoring of the repair as well as throughout the bridges' lifetime.

Table 2.1. Team Leader Qualifications (CFR 650.309).

1)	Have the qualifications specified for the Program Manager; or
2)	Have five years bridge inspection experience and successfully completed an FHWA-approved comprehensive bridge inspection training course; or
3)	certified as a Level III or IV Bridge Safety Inspector under the National Society of Professional Engineer's program for National Certification in Engineering Technologies (NICET) and successfully completed an FHWA-approved comprehensive bridge inspection training course, or
4)	Have the following: (i) A bachelor's degree in engineering from a college or university accredited by or determined as substantially equivalent by the Accreditation Board for Engineering and Technology; (ii) Successfully passed the National Council of Examiners for Engineering and Surveying Fundamentals of Engineering examination; (iii) Two years of bridge inspection experience; and (iv) Successfully completed an FHWA-approved comprehensive bridge inspection training course, or
5)	Have the following: (i) An associate's degree in engineering or engineering technology from a college or university accredited by or determined as substantially equivalent by the Accreditation Board for Engineering and Technology; (ii) Four years of bridge inspection experience; and (iii) Successfully completed an FHWA-approved comprehensive bridge inspection training course.

The current bridge inspection method consumes an extensive amount of time due to the set-up tasks required before the inspection, the maneuvering and repositioning of safety equipment, vehicles, and other instruments for a close-up inspection, and the necessary safety procedures executed by the inspectors. The FHWA BIRM (2012) states that bridge inspectors must be familiar with the bridge and its limitations; for example, some bridges may have time restrictions due to a high volume of daily traffic requiring a set time period when inspections can occur, and depending on the location, may require a traffic control plan. A traffic control plan includes the design, execution, and fiscal

expenditure to ensure the safety of the inspection team from oncoming traffic throughout the process (FHWA, 2012). Highway interchange structures, tall structures, and structures over waterways require a substantial amount of skilled maneuvering and repositioning of equipment by the inspectors while upholding mandated safety procedures in accordance with the Occupational Safety and Health Administration (OSHA, 1970). The inspection of bridge elements above a waterway may require a variety of safety precautions such as proper shoes, vests, hard-hats, eyewear, different harnesses and tie cables, inspection vehicles with an extendable manlift (requiring the inspector to secure the body harness to the bucket), and other safety items detailed by the inspectors in a preemptive safety plan. Therefore, the time commitment of a visual inspection is exacerbated by the additional requirements due to the human element.

According to FHWA, the current bridge inspection policy requires a bridge inspector to recognize, document, and alert bridge owners of critical deficiencies (Ad-Hoc Group, 2009). In a routine bridge inspection, documents, including photos of the as-built state and the previous condition(s) of the structure, aid in assessing the current bridge condition. Photos remain essential to future bridge inspection comparisons. Comparing the previous condition(s) of the bridge to the current condition of the bridge gauges the bridge's deterioration rate. The current reliance on visual data for documentation purposes alone implies the potential applicability of an automated image-based inspection procedure in general.

Routine bridge inspection procedures include an onsite team of trained professionals implementing a close range visual evaluation. The method involves

satisfying the BIRM's instructions to search for and measure listed damage types specific to different structural elements (FHWA, 2012). The findings from the visual inspection and measurements of the entire bridge are represented by the assigned functional classification. Specified in the BIRM, functional classifications are defined as the following: (1) Good or Better – all federal requirements are met; (2) Structurally Deficient – substantial wearing or damage has caused important load-carrying components creating poor or worse condition; and (3) Functionally Obsolete – clearance and/or roadway alignment, load-carrying capacity, or deck geometry do not meet the road system's criteria (FHWA, 2012). Functionally obsolete is not used as a classification for NBI bridge data published in 2016. Also, effective February 2017, bridge conditions are classified as Good, Fair, or Poor, but Structural Deficiency information remains published (FHWA, 2017). Following the BIRM for inspecting national bridges and their structural elements, a certain procedure, specified measurements and concerns, and documentation are required. To document the condition for different structural elements in a bridge inspection report, condition ratings are assigned. Each structural element or group of structural elements are assigned an Item number in the *Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges* (FHWA, 1995). The condition rating is a standard scoring system detailed in the *Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges* regulates the state of bridge evaluations (FHWA, 1995). Table 2.2 shows the value representation of the condition ratings for bridge superstructures. The condition rating code contains a description for each value.

Once an inspector sees a characteristic featured in a lower condition rating, the member will receive the value that represents the condition. Note that once advanced section loss is present in a superstructure, the member reaches poor condition.

Table 2.2. Condition ratings for deck, superstructure, and substructure (FHWA, 2012).

Code	Description
N	NOT APPLICABLE
9	EXCELLENT CONDITION
8	VERY GOOD CONDITION – No problems noted.
7	GOOD CONDITION – some minor problems.
6	SATISFACTORY CONDITION – structural elements show some minor deterioration.
5	FAIR CONDITION – all primary structural elements are sound but may have minor section loss, cracking, spalling or scour.
4	POOR CONDITION - advanced section loss, deterioration, spalling or scour.
3	SERIOUS CONDITION – loss of section, deterioration, spalling or scour have seriously affected primary structural components. Local failures are possible. Fatigue cracks in steel or shear cracks in concrete may be present.
2	CRITICAL CONDITION – advanced deterioration of primary structural concrete may be present or sour may have removed substructure support. Unless closely monitored it may be necessary to close the bridge until corrective action is taken.
1	"IMMENNT" FAILURE CONDITION- major deterioration or section loss present in critical structural components or obvious vertical or horizontal movement affecting structure stability. Bridge is closed to traffic but corrective action may put back in light service
0	FAILED CONDITION – out of service- beyond corrective action.

The Federal Highway Administration (FHWA) and the Department of Transportation (DOT) issued a Final Rule in the Federal Register (2017) to assess the

National Highway System (NHS) Bridge Condition by determining the lowest condition rating for Item 58 (Deck), Item 59 (Superstructure), Item 60 (Substructure), or Item 62 (Culvert) published by the National Bridge Inventory (NBI) (National Performance Management Measures, 2017). If the lowest condition rating is greater than or equal to seven, the bridge is classified as Good; if it is less than or equal to four, the classification is Poor. Bridges with their lowest condition ratings at five or six are classified as Fair. The condition ratings specified in Table 2.2 apply to deck, superstructure, and substructure, and the condition rating for culverts can be found in the *Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges* under Item 62 (FHWA, 1995).

The condition rating for beams is included in Item 59 (Superstructures) and uses a numerical condition rating from zero, representing failed conditions, to nine, representing excellent conditions. Manual documentation of cracks and other damage is a subjective form of evaluation. These types of manual evaluation procedures are plagued with uncertainty due to variation in inspector tendencies (e.g., different background and expertise, etc.). A conservative bridge inspector, recording the deficiencies as worse than the actual situation, can create issues with federal funding or inaccurate ground truths for comparisons in future inspections. However, proper utilization of this code has the potential to create a fair, standard, and consistent procedure to calculate an objective sufficiency rating to appropriately distribute federal funds and achieve accurate databanks such as the National Bridge Inventory (NBI) if properly quantified or made consistent through the use of an automated approach.

The State Auditor's Office (2009) reported that quality control and quality assurance programs at TxDOT contained informal execution and did not fully comply with federal requirements. Implementing official quality control and quality assurance procedures guarantees satisfaction of NBIS requirements and all bridge divisions participate in standard programs that objectively identify critical deficiencies (Ad-Hoc Group, 2009). *Quality control* pertains to procedures that maintain the integrity of the bridge inspection process. *Quality assurance* encompasses the process of verifying the accuracy of the quality control procedures. The lack of formality and consistency across districts leads to an inaccurate national record and inappropriate distribution of federal funds to the roadway system. Implementation of automated bridge inspections will provide a regimented, uniform procedure to check the quality of bridges, maintaining consistency across bridge types and individual bridge structures. To enforce quality control procedures, verification clauses will validate the automated procedure's legitimacy. For example, bridge inspectors could easily periodically check that they concur with the data output of the automated tool.

2.2. Concrete Beam Inspection

The Ad-Hoc Group (2009) claimed that inspection consistency and effectiveness could improve at the national level if inspector qualifications matched the bridge type, conditions, and complexity in a more consistent manner, improving both the efficiency of bridge inspections and associated bridge safety. Focusing on one bridge type and material also improves the efficiency of an automated bridge inspection. Another benefit to identifying a specific structural element is that each structural element of a bridge

undergoes different inspection procedures, and each bridge element has a specified inspection process. Prestressed beams, for example, are structural elements that are completely in compression and designed to withstand flexure. Due to this design, BIRM states that any cracks larger than 6mm are considered severe damage (FHWA, 2012). Flexure cracks can be seen in the middle of the span and at the bottom of the beam (Naaman, 2012). Therefore, locating the beam flange and the damage on this region is of utmost importance. Table 2.3 shows the increase in bridge counts across the country for different bridge types and prestressed concrete bridges had a significant increase in the fifteen years between the measurements as the average growth per year more than doubled that of any other material (Farhey, 2010). Concrete and prestressed/posttensioned construction material and/or design had an average growth increase of 1,238 and 2,893, respectively, while all other material decreased in growth, with the exception of Aluminum, wrought iron, or cast iron which only grew by 31 bridges (Farhey, 2010). Therefore, bridge inspections for prestressed bridges will become more frequent, as prestressed beams remain dominant in bridge design and make up a large percentage of the overall bridge stock.

Table 2.3. Kinds of Bridge Construction Material and/or Design (Farhey, 2010).

Code	Description	Bridge count 2006 (% from total)	SD bridge count 2006 (%)	Average Age	SD average age	Average growth per year	Growth in 15 years
1	Concrete simple span	247,618 (41.45%)	18,782 (7.59%)	41.94	61.80	1,238	18,576
2	concrete continuous						
3	steel simple span	189,494 (31.72%)	39,496 (20.85%)	44.05	62.27	-1,272	-19,087
4	steel continuous						
5	prestressed or posttensioned concrete-simple span	129,024 (21.60%)	4,635 (3.59%)	24.93	41.88	2,893	43,400
6	prestressed or posttensioned concrete-continuous						
7	wood or timber	27,610 (4.62%)	10,174 (36.85%)	45.50	52.35	-1,138	-17,063
8	masonry	1,798 (0.30%)	450 (25.03%)	93.79	99.80	-11	-161
9	Aluminum, wrought iron, or cast iron	1,402 (0.23%)	214 (15.26%)	34.39	96.57	31	458
10	other	497 (0.08%)	49 (9.86%)	31.05	70.38	-80	-1,204

The average structural deficiency age of prestressed bridges reaches about 42 years (Farhey, 2010). Because bridges that either match or exceed this age tend to have surface defects and other damages, their use will validate the effectiveness of the automated bridge inspection procedure for prestressed beams.

In 2016, 60% of the total bridges documented in the U.S. were considered to have a structural design and/or construction type involving beams and/or girders and 67% of the total documented bridges have been identified as concrete superstructure material (NBI, 2016). Compared to other bridge superstructure types and bridge materials, bridges involving beams/girders and the use of concrete material are more frequent than the use of other bridge types and bridge materials. A detailed breakdown of bridge structural design is seen in Figure 2.1, and a detailed breakdown of the concrete bridge material types is seen in Figure 2.2. 72% of deficient structures on record in 2016 had a structural design involving beams (NBI, 2016). Compared to the total 56,007 structurally deficient bridges based on material type documented in 2016, concrete bridges contribute to 19,535 (3.5%). This is only second to the steel bridges, which contribute a little over 29,000 (5.2%), but as seen previously, the use of this material is seeing a rapid decrease, while concrete and prestressed concrete is seeing a substantial rise in usage.

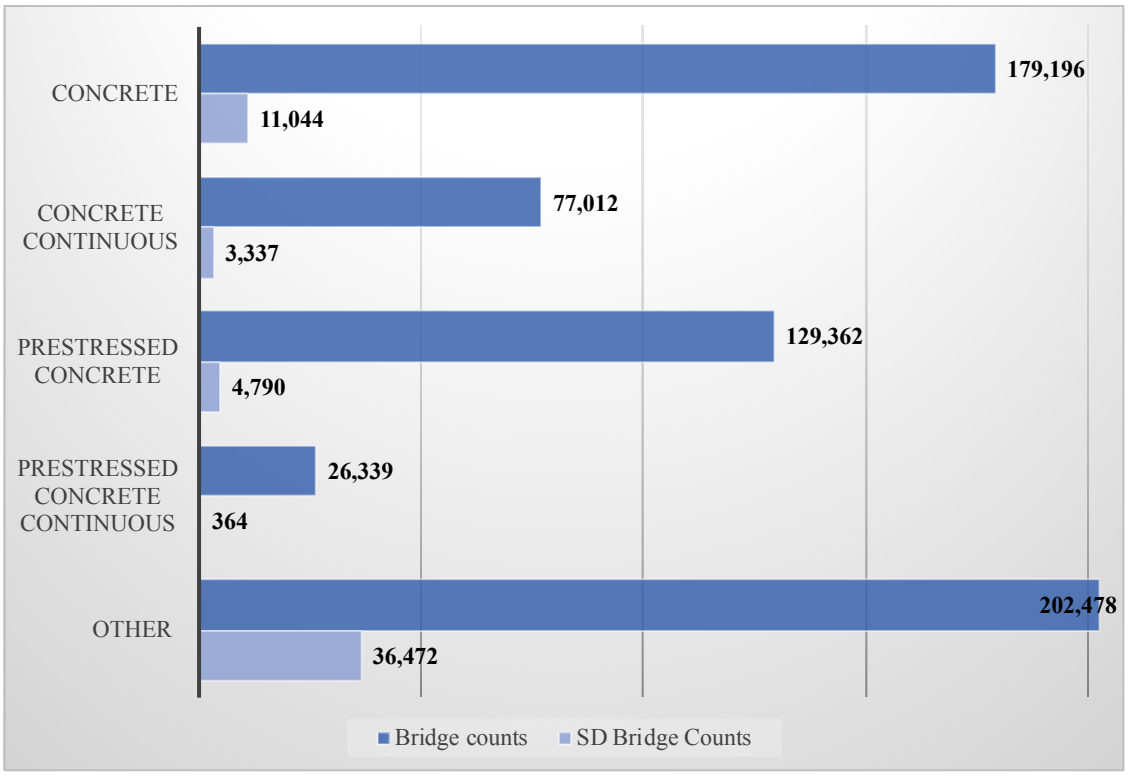

Figure 2.1. Total bridge count and structurally deficient bridge count based on the substructure (NBI, 2016).

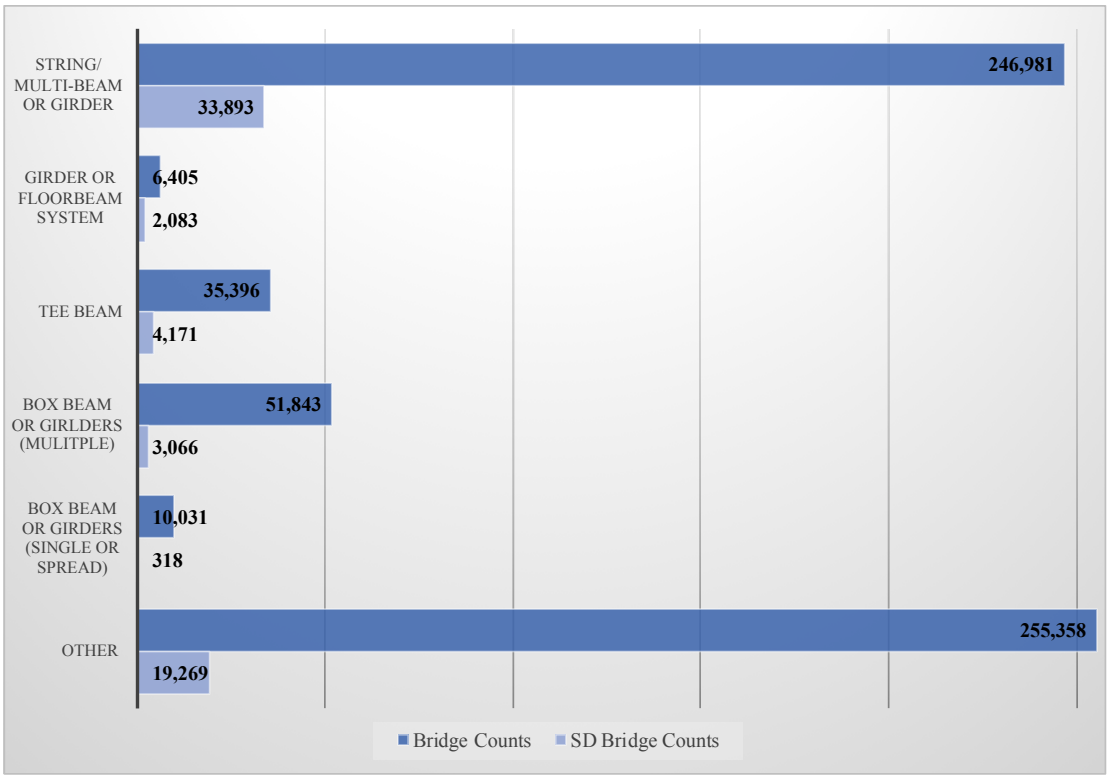

Figure 2.2. Total bridge count and structurally deficient bridge count based on material type concrete and prestressed concrete (NBI, 2016).

The BIRM details four different types of prestressed beams: prestressed double tee, prestressed I-beams, prestressed bub-tee, and prestressed box beams (FHWA, 2012). As for standard prestressed concrete elements, I-beams are the most commonly used beam section in Texas (Cox et al., 2007). The American Association of Highway and Transportation Operations (AASHTO) (Figure 2.3) specifies the most common I-beam shapes used in design for most state highway agencies (FHWA, 2012; PCI/AASHTO, 2011). The high demand for prestressed concrete I-beams bridge inspections will continue to increase, and with the struggle to complete bridge inspections on time, an augmented procedure will satisfy this quickly growing inspection

demand. Additionally, the work presented in this thesis can easily be validated for other concrete beam types and extended to automatically detect other component types of any material with ease. However, the commonality of these types of beams yields significant data for validation of preliminary automated procedures and demonstrates the wide applicability as mentioned.

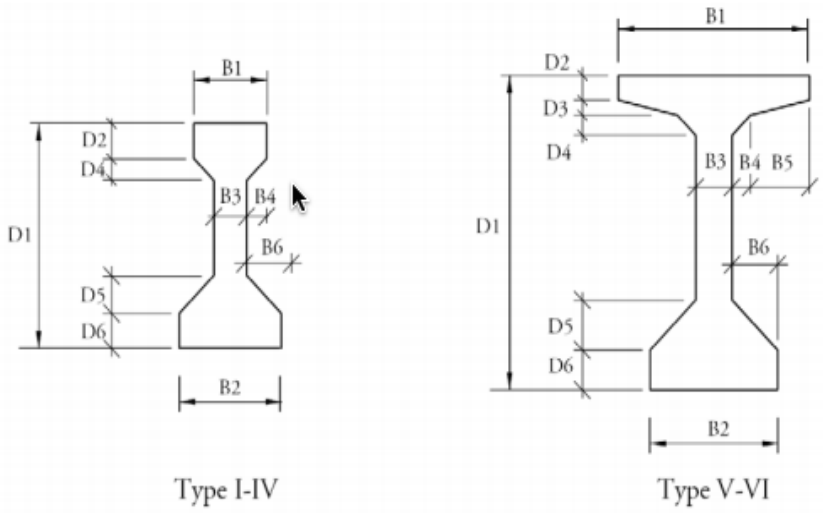

Figure 2.3. AASHTO cross section of prestressed I beams (AASHTO, 2011).

The *Bridge Inspector's Reference Manual* (BIRM) details a list of important considerations and common deficiencies when evaluating prestressed I-beams. Some of the common deficiencies include cracking, collision damage, overloading damage, and spalling. Along with other detail and design requirements, prestressed concrete I-beams were designed to remain completely in compression, detailed in AASHTO. Cracking caused by loss of compression (flexure cracking) indicates and a serious problem while other bridge members, such as bent caps, are more prone to cracks caused by shear, and therefore, classification of the superstructure condition rating is of utmost significance

(AASHTO, 2011). Also, a substantial amount of prestressed concrete bridge wear or loss of concrete sections is caused by traffic damage. These sections need quick repairs in order to avoid cascading complications, mainly the evolution and progression of steel corrosion. The more frequent the bridge inspection, the quicker the identification of damage and accordingly, the design, management, and administration of appropriate repair strategies. An unmanned procedure will allow the frequency necessary to more efficiently and robustly manage our nations' bridges in a comprehensive manner (from construction to regular maintenance to potentially extending the bridges expected lifetime and even to determining proper deconstruction strategies).

2.3. Unmanned Data Acquisition For Structural Health Monitoring

Many developments have taken place in terms of technology as well as programs to conduct different aspects of bridge inspections. Chan et al. (2015) compared the capabilities of aerial-based inspections to the current state of bridge inspections. In general, the overarching goal of UAS-based and/or automated inspections is often defined as either reducing the required time of an action or reducing the cost of the product or service for the client. Chan et al. (2015) conducted a cost-benefit analysis of implementing an aerial data collection system (UAS). The bridge used for this case study required under-bridge inspection, which required the use of a Traffic Management Plan (TMP) in order to designate the direction of traffic to ensure the safety of the road occupants as well as the bridge inspection team. The routine procedure (by the existing state-of-practice approach) resulted in a four-hour inspection consisting of traffic control for two hours and four hours for the actual under-bridge inspection. Additionally, the

inspection team was comprised of two inspectors and various items of inspection and safety equipment, such as safety gear and harnesses. The case study assumes the costs associated with the traditional manual bridge inspection resources as well as the reduced cost of the same operations if using a UAS purely for data collection (Table 2.4). An estimated $2,770 total cost savings was determined in this case study by the difference in the total cost of the manual inspection resources and the UAS-based inspection resources (not including the initial cost of a UAS). Additionally, in the case study, the assumed initial cost of $6,000 was estimated for a UAS, demonstrating that the UAS would pay itself off after just three uses. Hence, the manual inspection, merely augmented by a UAS as a data collection device in areas where access is difficult (under-bridge inspection) and to reduce the need for traffic control, delivers substantial savings. This case study does not address the cost of maintenance of the UAS nor does it include the financial savings which would accrue after decreasing the planning and preparation time no longer needed as a result of implementing intelligent UAS with built-in automated inspection procedures into bridge inspections. However, the case study does demonstrate a simple cost-benefit analysis which clearly illuminates the potential time and cost savings if bridge inspections were augmented by UAS in any form.

Table 2.4. Cost of recourse utilization of bridge inspection case study (Chan et al., 2015).

Inspection Disbursement	Manual Inspection	UAV-based Inspection
Traffic Control	$640	$320
Under-Bridge Unit	$2,000	-
Bridge Inspectors	$1,200	$750
Total	$3,840	$1,070

Khan et al. (2015) also conducted an experiment to gauge the feasibility of using UAS in bridge inspections at the *Center for Advanced Infrastructure and Transportation* at Rutgers University. They constructed a prototype of a concrete bridge to analyze two different types of UAS with two different cameras. The UAS systems were validated by comparison with mounted optical systems on a ground vehicle. The combination UAS-optical system showed the same delamination discovered by the portable cart system, confirming the feasibility of the UAS-optical system.

To address the shortcomings of on-site visual inspections carried out by human inspectors, several data acquisition techniques have been proposed as a substitution for onsite personnel. A visual monitoring system including a permanent on-site server and several cameras mounted in various locations allowing for the full view of the bridge was proposed to collect images for structural monitoring (Jahanshahi et al., 2013). However, this system requires the permanent installation of the bridge monitoring system on every bridge which is impractical considering there are more than 600,000 bridges comprising over 17,000 miles of roadway in the nation. Another proposed

solution was the use of a remote-controlled vision-based robot system (the "Ubiquitous Bridge Inspection Robot System (U-BIROS)") (Lee et al., 2011). U-BIROS consists of a large transporting vehicle and hydraulic transportation boom. The transporting vehicle must be close enough for the camera attached at the end of the boom to analyze the bridge, and the vehicle, due to its size, requires a large area to park. A traffic control plan is required to accommodate the proximity limitation and size of the system. These systems are reliant on expensive equipment, and the large vehicle must be anchored and stabilized in place. This set up can limit the use of the system on numerous bridges that would benefit from some sort of a semi-automated or a fully-automated inspection system (Chan et al., 2015).

In recent decades, UAS have been widely used in civil engineering for various applications. UAS have aided in data acquisition of disaster monitoring and assessment (Chou et al., 2010; Adams and Friedland, 2011), to develop a damage detection database on structures (Yeum and Dyke, 2015), and for construction site monitoring (Wen and Kang, 2014). Hallermann et al. (2015) implemented UAS as a cost-effective way to collect data on aging heritage structures to assist with structural health monitoring (SHM) by gathering aerial images to manually analyze the exterior damage and determine the condition of the structure. Ezequiel et al. (2014) employed a UAS for data acquisition, post-processing, and collaboration, providing information to be implemented into various UAS applications, mainly post-disaster applications. Specific UAS applications in bridge inspections have been investigated by many researchers recently (Lee et al., 2018; Hawken et al., 2017). These studies confirmed the versatility and

flexibility of UAS and the other benefits including safety, effectiveness, and accuracy. UAS paired with an off-site computer provides an inexpensive and versatile alternative for acquiring visual data, and because implementing a UAS replaces the need for on-site inspectors, the currently required set-up time lessens and the traffic control plan implemented to ensure inspector safety is no longer necessary. However, existing approaches have not focused on providing a real-time evaluation of the infrastructure. Research still needs to be performed to extract critical structural component and damage information to evaluate the state of the bridge automatically.

The implementation of an unmanned automated bridge inspection procedure will improve the transportation system by: (1) reducing traffic caused by inspection related traffic control plans; (2) improving maintenance on existing bridges by identifying damage earlier; and, (3) providing an inexpensive alternative allowing for financial reallocation to other aspects of transportation services. The benefits of exercising this method will allow for more time-efficient, cost-effective, safe, and accurate evaluations of bridge elements. Further, these benefits will lead to more frequent inspections with up-to-date and accurate bridge information for tactical actions such as distributing funds, determining the type of structural repair, and limiting the inspector's exposure to dangerous situations.

2.4. Structural Element Detection

As previously mentioned, the current bridge inspection procedures focus on determining an overall condition rating that is an accumulation of individual condition ratings, such as that for the superstructure. Additionally, a number of structural

components make up the overall superstructure, and certain components are more critical than others. Thus, if a fully-automated, real-time, robust UAS-based inspection procedure is to be created, critical structural components must first be detected in the UAS images. Additionally, identifying the specified structural element and localizing it in the image space allows for the rest of the image and video to be temporarily ignored. Extracting the structural elements from the image (a) simplify subsequent processes resulting in reduced computational complexity of the overall algorithm; and, (b) reduce the potential for loss in accuracy by removing unnecessary background data. These benefits could potentially reduce the time required for the inspection and increase the accuracy of the inspection results. Identification of the correct structural element in automated inspection algorithms removes unwanted complex scenes common in image backgrounds and exposes the complete bridge inspection process susceptible to false identification of damage.

Detection of structural elements in images or video frames has mostly relied on machine vision-based methods. Machine vision-based object detection methods are commonly divided into three categories: (1) color/texture-based methods, (2) shape-based methods, and (3) scale/affine-invariant feature-based methods. Feature-based methods will be reviewed in a later section. Neto et al. (2002) developed a color-based object detection method that first examined the image pixel color value to determine the edge of the element, and then, each pixel within the boundary was verified to satisfy the specified color value range. The image pixels that did not satisfy the threshold were removed and the remaining pixels within the boundary denote the

structural element. This method was developed based on the observation that a structural element should be described by a color value range versus a single value. This work requires a predetermined image pixel range and only detects the occurrence of structural elements in an image.

Color information retrieval was combined with a texture-based approach to represent a material's "signature" by Brilakis et al. (2006). In this work, the material signature was developed using a bottom-up clustering method to sector an image into regions. The mean and standard deviation of the cropped region's color and texture values are calculated, stored in a vector, and compared to samples retrieved from a material knowledge database. The Euclidean distance between the cropped region's signature and a material sample's signature is the metric examined to determine the material contained within the region. If the distance from the region's signature to a material sample's signature satisfies the predetermined distance threshold, then the material is said to be contained within the region. Brilakis and Soibelman (2008) augmented this material recognition method further to identify linear structural elements. For each region, the material recognition method determines the region's material and then the maximum cluster dimension (MCD) and the maximum dimension along the axis perpendicular to the MCD (PMCD) are calculated. The region is presumed to be column (beam) by (1) concluding the region is linear if the MCD exceeds the PMCD, (2) assuming the region's direction on the image plane is denoted by the tangent of the MCD edge points, and (3) classifying the region as a column (beam) if the computed direction is within 45° of the vertical (horizontal) image axis. Although this method

exploits valid assumptions regarding structural elements in an image, the method proves insufficient when a structural element is connected to another structural element of the same material as the connected elements are mistaken for a single structural element. Methods that rely exclusively on color and texture information often possess this limitation when attempting to detect structural elements (Zhu and Brilakis, 2010). The case study bridges in this work are all prestressed concrete bridges which are constructed with multiple connecting structural elements of the same material; therefore, color-based and texture-based detection methods are not suitable for this work.

Shape-based methods have been paired with material-based methods to perform object detection in order to overcome these limitations associated with color/texture-based methods. Zhu and Brilakis (2010) developed an algorithm to detect concrete columns in an image by considering two key characteristics of the columns: (1) concrete columns can be represented by a pair of long vertical lines that border one of the column's surfaces, and (2) the color and texture pattern on the column's surface appears uniform. The developed method depends on the assumptions that, if a line, made up of edge pixels, exceeds one-third the height of the image, the line is presumed "long," and a column's width-to-height ratio is smaller than one. In this work, to locate the pair of vertical long lines, edge detection and the Hough transform are employed, and the material recognition method is used to determine if the material between the line pair is concrete. Shape-based methods depend almost solely on accurate edge information which falters in complex scenes; therefore, an entirely shape-based algorithm will not be satisfactory in the application are presented in this work. Alternatively, the complexity

of the scene should be dealt with in order to directly and adequately implement a shape-based method.

At this point, machine vision, color-, texture-, and shape-based, methods alone have been proven insufficient for adequate structural element detection for components other than columns, and although, a combination of these approaches have proven successful in column detection methods detailed above, beam detection introduces new challenges. A shape-based approach tends to be the first step in previous element detection methods but is inadequate in complex scenes. Images captured containing beams located in a bridge span tend to be surrounded by significantly more complex shapes than that of columns.

More recently, the second rise of machine learning has led to the application of various algorithms in the realm of civil engineering. Machine learning methods such as neural networks have been applied to the civil engineering domain since 1989 (Adeli, 2001). One of the first applications of a neural network-based system to SHM was a network-based system for updating a baseline finite element model (Feng, 2004). This method used a limited number of sensors to measure vibration data caused by daily traffic on two bridges to create a baseline model for SHM applications. This method was further developed by Taffese and Sistonen (2017) to create a structural health prediction model using machine learning to assist in the durability and service-life assessment of reinforced concrete structures in general. However, visual data was not included in the data collection. As previously mentioned, visual damage, such as cracking, is a very reliable indicator of a serious problem for prestressed beams and a precursor of failure

(FHWA, 2012). Implementing a network system to analyze vibrations and create a baseline finite element model would not be adequate to evaluate a bridge's current condition as specified by the *Bridge Inspector's Reference Manual* (BIRM) (FHWA, 2012).

The algorithm previously mentioned, developed by Zhu and Brilakis (2010) to detect concrete columns also implemented an artificial neural network (ANN) model to recognize the concrete texture/material in between the vertical lines to aid in the overall detection procedure. This method does not consider variation in illumination, viewing angle, occlusion, and other discrepancies present when collecting video data using a UAS in the field. An automated color model-based concrete detection method has been explored by Son et al. (2011) in order to monitor construction progress. This work analyzed three different machine learning algorithms, a Gaussian mixture model, an ANN, and a support vector machine (SVM) model, as well as a hybrid ANN-SVM model in different color spaces. This method adequately identifies the concrete material and outputs a segmented image; however, the method lacks the ability to identify individual structural elements. Son et al. (2014) extended their work to improve the ability to detect common construction materials such as concrete, steel, and wood. The study considers the previously developed SVM and ANN algorithms, as well as a commercial version 4.5 (C4.5) advanced decision tree algorithm, a Naïve Bayes (NB) algorithm (a simple linear classifier), logistic regression (LR) (generalization of linear regression) and a K-Nearest Neighbor (K-NN) algorithm (instance-based learning approach). Although the algorithms are not developed to identify structural elements, the

ensemble classifier model (a model constructed with a set of individual classifiers, instead of a single classifier) was proven to be more accurate than the individual classifiers for the application.

In a study to implement bridge element detection, Narazaki et al. (2018) integrated scene classification and bridge component recognition using pixel classification with a multi-scale convolutional neural network (CNN). To train the CNN used in bridge component recognition the training set included bridge and non-bridge photos manually labeled into five different classes: non-bridge, columns, beams and slabs, others (structural), and others (nonstructural). The images used in the dataset were all augmented according to the techniques proposed by Farabet et al. (2013), which includes random resizing, cropping, rotation, flipping, and jitter. The proposed bridge component recognition method frequently predicted Beams & Slabs as nonstructural components, and thus, does not achieve the level of precision necessary for improving bridge inspection operations with accurate, robust object detection algorithms.

2.5. Tracking Methods Used in Civil Engineering

Once an object is detected in a video frame, it should be tracked throughout subsequent video frames to provide a computationally inexpensive approach which is also robust as, eventually, with tracking implemented, all beams in the bridge have a better probability of being detected in a single video frame. This will ensure every beam is detected and located throughout the automated bridge inspection, and provide only the most important information to an inspector or for further algorithmic operations. There have been various research efforts towards object tracking in civil engineering in recent

years. Most notably are the applications in construction performance and monitoring (Yang et al., 2015; Teizer, 2015; Zhu et al., 2016) and traffic surveillance (Beymer et al., 1997; Shao et al., 2016). Teizer and Vela (2015) explored the use of tracking workforce members on a construction site implementing four different methods including Bayesian segmentation and graph-cuts. Zhu et al. (2016) implemented high definition video cameras to monitor the workforce and equipment to improve job safety and monitor construction progress. Vision-based traffic monitoring has been implemented to count the number of cars through a segment of a roadway and to monitor vehicle speed (Wang et al., 2004). This method uses progressive background image generation, vanishing point calculations, and vehicle detection and tracking. Most implementations of tracking in civil engineering involve the motion of an object while the background remains stationary. However, within the realm of UAS-based bridge inspections, the desired object of detection and tracking (the bridge and its' components) is stationary, while the camera/sensor (UAS) is moving. Thus, the tracking methods presented in the literature are not necessarily entirely adequate for this application, and the method proposed in this work cannot depend on or relate to the motion (of the object or the sensor), but only on the matching of key features in subsequent video frames.

Feature-based tracking has been explored as a tool in civil engineering based research. Coifman et al. (1998) also used feature-based tracking, relying corner features, to track vehicles through the chaos of an intersection, and Mu et al. (2016) utilized scale invariant feature transform (SIFT) feature-based tracking algorithms to monitor vehicles on the highway contributing to traffic congestion. Feature detection is the process of

identifying key features in an image (Lindeberg, 2008-2009). Common key features identified are edges, corners (intersect points), blobs (regions of interest points), and ridges (Shi and Tomasi, 1994). Feature extraction is the process of computing a descriptor (feature vector) from the area pixels describing the key feature and assigning a key point to represent the location of the key feature (Linderberg, 2008-2009). Farman et al. (2016) reviewed several feature detectors in computer vision including FAST, Harris, Eigen, Maximally Stable Extremal Regions (MSER), and Speeded-Up Robust Features (SURF). The article compared literature on each of these detectors and concluded that the SURF detector was the most successful. The authors also reviewed the scale invariant feature transform (SIFT) algorithm which provides a detector and descriptor (Farman et al., 2016). The development of the SURF descriptor was developed by analyzing the capabilities and limitations of current descriptors, and due to the superiority of the SIFT descriptor, the SURF descriptor was developed using the SIFT descriptor as a base model (Farman et al., 2016). The SURF descriptor provides a computationally inexpensive alternative to the SIFT descriptor (Farman et al., 2016). Detail of SURF detector and descriptor can be found in Section 3.1

The benefit of feature-based tracking allows for the tracking of key points rather than the entire object. The SURF detector and descriptor will be utilized for tracking the beam flange in subsequent video frames in this work. Implementing this method will ensure that the part of the beam remaining in the subsequent images will be located.

3. BACKGROUND

In this section, a discussion of the existing machine vision, machine learning, and tracking algorithms adapted for this work are presented. The purpose of this section is to provide the background knowledge necessary to understand the development of the overall hybrid machine vision-machine learning methodology to detect and track prestressed concrete beams in UAS imagery. The augmentation and conglomeration of these algorithms and the specific methods developed in this work will be presented in Sections 4-6

3.1. Machine Vision Algorithms

3.1.1. Canny Edge Detector

The Canny edge detector was originally developed in 1986 to detect a wide range of edges in an image (Canny, 1986). This edge detection algorithm can be broken down into five main steps: (1) smooth; (2) find intensity gradient; (3) edge thinning; (4) apply double threshold; and (5) track edges by hysteresis (Zhou, 2011).

A Gaussian filter (Gaussian blur, Gaussian smoothing) blurs an image by a 2-D Gaussian function, (Eq. 3.1). A Gaussian filter reduces the image's high-frequency elements which reduces image noise and detail. Reducing noise increases the reliability of the edge detection algorithm to successfully identify true edges and disregard weak edges created by the noise (Shapiro and Stockman, 2001). The size of the Gaussian filter affects degree of smoothing; large filters result in an image with a large degree of smoothing and vice versa. Larger Gaussian filters substantially reduces image noise and

provide a quicker computational process compared to smaller Gaussian filters, but large Gaussian filters may cause uncertainty in exact edge localization (Praveen et al., 2016). Assigning a standard deviation value for the Gaussian distribution, σ, determines the size of the filter. Figure 3.1 and Figure 3.2 show an example of an original image and a smoothed image.

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{(Eq. 3.1)}$$

where G_σ is the Gaussian function, σ is the standard deviation of the Gaussian distribution, e is the exponential function, and x (y) is the horizontal (vertical) distance from the origin.

Figure 3.1. Original image before a Gaussian filter is applied.

Figure 3.2. Smoothed image ($\sigma = 4$).

The intensity gradient of an image is found by applying the derivative of the Gaussian filter (Eq. 3.2) (Deriche, 1987). The derivative of the Gaussian (with respect to coordinate direction) convolved with the smoothed image proves the image gradient in the horizontal, G_x, (Figure 3.3) and vertical direction, G_y, (Figure 3.4). The edge strength of an image pixel is represented by the gradient magnitude, G. The gradient magnitude is computed using the horizontal gradient value and vertical gradient value of the image pixel. Figure 3.5 shows the edge strengths of the image. In the gradient images, if the gradient magnitude is zero, the pixel is black, and if the gradient magnitude is 1, the pixel is white. Therefore, in Figure 3.3, Figure 3.4, and Figure 3.5, the weak edges are represented by the color range dark gray to black and the strong edges are represented by a lighter gray to white range. To calculate the edge strength the equation Eq. 3.4 is used.

$$G_x = \frac{\partial G_\sigma}{\partial x} \qquad \text{(Eq. 3.2)}$$

$$G_y = \frac{\partial G_\sigma}{\partial y} \qquad \text{(Eq. 3.3)}$$

where $G_x$and G_y are the first partial derivative of a Gaussian, and G_σ is the Gaussian function.

$$G = \sqrt{(G_x)^2 + (G_y)^2} \qquad \text{(Eq. 3.4)}$$

where G_x and G_y are the first derivative of the smoothed image in the horizontal and vertical direction, respectively, and G is the gradient magnitude, or edge strength, of the image.

Figure 3.3. The smoothed gradient of the original image in the horizontal direction.

Figure 3.4. The smoothed gradient of the original image along the vertical direction.

Figure 3.5. Gradient magnitude of the original image.

The gradient direction is also calculated in this step and will be used in the next step of the Canny edge detection method. The gradient direction, or edge direction, of each image pixel, is calculated using the equation in Eq. 3.5.

$$G_\theta = \tan^{-1}\frac{G_y}{G_x} \qquad \text{(Eq. 3.5)}$$

Where G_x and G_y are the first derivative of the smoothed image in the horizontal and vertical direction, respectively, and G_θ is the edge direction.

The next stage of the algorithm is edge thinning. Edge thinning occurs when the accurate location of each edge is retrieved. After applying a Gaussian filter and finding the intensity gradient magnitude and direction, the extracted edge pixels may have been blurred to make the edge appear several pixels thick. The inexact widening of the image's edges is relative to the size of the Gaussian filter and the specified parameters. In order to perform further operations, it is necessary to extract edges which are uniform in thickness (most often, one pixel thick). In this step, a nonmaximum suppression edge thinning technique suppressing the gradient values is implemented (Lindenberg, 1998). This is done by following the edge along the gradient in the edge direction and comparing the values perpendicular to the gradient. If the gradient values perpendicular to the current pixel is lower, the pixel value is set to zero (magnitude and direction). Therefore, all values will be set to zero except for the local maxima. The nonmaximum

suppression outputs all remaining edge pixels. Figure 3.6 shows the thinned edges after non-maxima suppression.

Figure 3.6. Original image after non maximum edge thinning.

Next, a double threshold is applied to the remaining edge pixels. Although non-maximum suppression adequately identifies the image edges, some remaining edges may remain due to noise. The double threshold contains a high and low threshold. Edge pixel values above the high threshold value are named strong edge pixels, and edge pixel values that are above the low threshold but below the high threshold are considered weak edge pixels. All edge pixel values below the low threshold value are discarded.

Finally, the edge pixels that have been identified as strong edge pixels are included in the final image. Any weak edge pixels included in the final image are

determined based on the proximity and characteristics of the surrounding (neighborhood) pixels. Edge pixels caused by noise are unconnected and edge pixels caused by edges are connected. After analyzing the neighborhood of each weak pixel, if the weak edge pixel is connected, the edge pixel is included. Figure 3.7 shows an example of a final binary image resulting from the Canny edge detector.

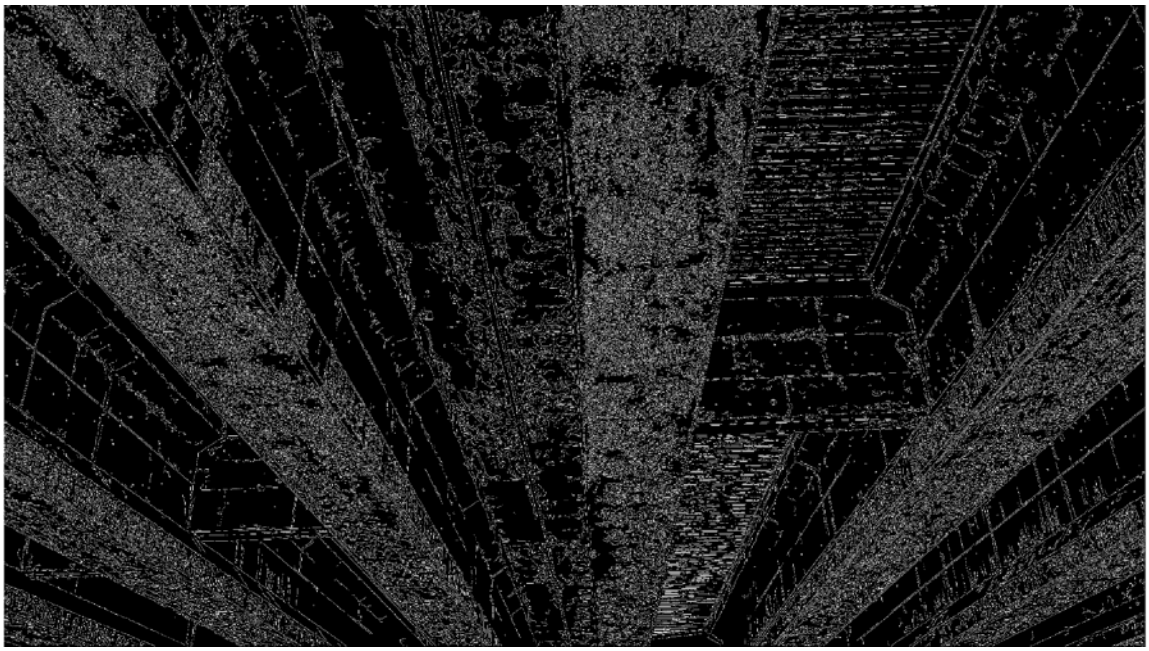

Figure 3.7. Canny edge detection image result.

3.1.2. Hough Transform

The Hough transform is a feature extracting algorithm that uses a voting procedure to determine the location of a specified shape (Ballard, 1987). The Hough transform analyzes a binary image, and for this research, the binary image with be the output of the Canny edge detector. The Hough transform for line detection groups

extracts edge features and merges each edge point into possible lines using this voting procedure (Fernandes, 2008). Straight lines can be described by Eq. 3.6, where for all points (x, y), there is a y-intercept *b* and slope *m* that uniquely defines the line; however, vertical lines pose an issue. Therefore, Hough transform detects lines using the parametric form in Eq. 3.7. The parameter space used to describe the lines is in Hesse normal form and each line can be represented by a unique point (r, θ).

$$y = mx + b \qquad \text{(Eq. 3.6)}$$

where (x, y) is a point on the line which has slope m and y-intercept b.

$$r = x\cos\theta + y\ \sin\theta \qquad \text{(Eq. 3.7)}$$

where r is the distance from the origin to the closest point on the line, x is the intercept on the x-axis, y is the intercept on the y-axis, and θ is the angle between the x-axis and the line from the origin and the closest point.

A parameter space matrix where the rows correspond to r and the columns correspond to θ is referred to as an accumulator array. The 2-D accumulator array stores each point (r, θ) in the appropriate accumulator cell, or bin (Fernandes et al., 2008). To determine the final accumulator array, the hyper-parameters rho-resolution, which

defines the spacing of the accumulator array bins corresponding to r, and theta, which determines the range of columns, corresponding to θ, are specified.

Once the Hough transform is completed, the number of points stored in each bin are counted. This value represents the number of points that create the line segment in the xy-plane. Bins with the greatest number of pairs represent potential lines in Hough space. The number of lines extracted is determined by identifying the Hough peaks. This requires setting the hyper-parameters number of peaks, which specify the number of identified peaks, peak threshold, which states the minimum number of points in a bin required to be considered a peak, and the suppression neighborhood, which is the area around each identified peak set to zero. The bins, representing the line (r, θ), that satisfy the Hough peak thresholds are extracted and used to predict the location of line segments (Ballard, 1987). To extract possible lines, the hyper-parameters fill gap, which specifies the maximum allowable distance where two-line segments, corresponding to the same (r, θ), can be combined into one-line segment, and minimum line length, which states the minimum length required to be considered a line segment.

Overall, the Hough transform requires the input of a binary image and the output is predicted lines that are represented by the line segment end points and the parameter space values, r and θ.

3.1.3. Speeded-Up Robust Features

The Speeded-Up Robust Features (SURF) consists of three main parts including the SURF detector, the SURF descriptor, and feature matching. The SURF detector is considered a blob detector (Bay et al., 2006). A blob detector finds areas of interest in an

image by comparing neighboring region's characteristics (e.g. brightness, coloring, etc.). These areas of interest, which vary in size, are called blobs, and the localization of a blob is referred to as the point of interest. To identify the points of interest, where each point has their own characteristic scale, blob detectors analyze image regions at different scales, and therefore blob detectors are scale-invariant.

3.1.3.1. SURF Detector

The SURF detector determines the location and scale of interest points by analyzing the approximated determinate of the Hessian matrix (Bay et al., 2008). The equations in this section have been retrieved from Bay et al. (2008) unless otherwise stated. For point $\boldsymbol{x} = (x, y)$ in an input image, the Hessian matrix (Eq. 3.8) is constructed and at an arbitrary scale σ. The Hessian matrix contains different 2-D Gaussian second-order derivatives convolved with the input image. The term $L_{xx}(\boldsymbol{x}, \sigma)$, for example is the convolution of the 2-D Gaussian second-order partial derivative with respect to *x*, and the input image (Eq. 3.9 and Eq. 3.10), and a similar calculation is made for $L_{xy}(\boldsymbol{x}, \sigma)$ and $L_{yy}(\boldsymbol{x}, \sigma)$. The determinate of the Hessian (DoH) matrix can be seen in equation (Eq. 3.11).

$$H(\boldsymbol{x},\sigma)=\begin{bmatrix}L_{xx}(\boldsymbol{x},\sigma) & L_{xy}(\boldsymbol{x},\sigma)\\ L_{xy}(\boldsymbol{x},\sigma) & L_{yy}(\boldsymbol{x},\sigma)\end{bmatrix} \quad \text{(Eq. 3.8)}$$

where $H(\boldsymbol{x},\sigma)$ is the Hessian matrix at point $\boldsymbol{x}$ at scale σ, and $L_{xx}(\boldsymbol{x},\sigma)$, $L_{yy}(\boldsymbol{x},\sigma)$, and $L_{xy}(\boldsymbol{x},\sigma)$ are the convolution of the second-order Gaussian derivatives and the image I at point $\boldsymbol{x}$.

$$L_{xx}(\boldsymbol{x},\sigma)=G_{xx}(\boldsymbol{x},\sigma)*I(\boldsymbol{x}) \quad \text{(Eq. 3.9)}$$

where $I(\boldsymbol{x})$ is the input image at point $\boldsymbol{x}$, and $G_{xx}(\boldsymbol{x},\sigma)$ (Eq. 3.10).

$$G_{xx}(\boldsymbol{x},\sigma)=\frac{\partial^2}{\partial x^2}G_\sigma \quad \text{(Eq. 3.10)}$$

where $G_{xx}(\boldsymbol{x},\sigma)$ is the Gaussian second-order partial derivative with respect to x and G_σ is the Gaussian function (Eq. 3.1).

$$DoH(\boldsymbol{x},\sigma)=L_{xx}(\boldsymbol{x},\sigma)\cdot L_{yy}(\boldsymbol{x},\sigma)-(L_{xy}(\boldsymbol{x},\sigma))^2 \quad \text{(Eq. 3.11)}$$

where $DoH(\boldsymbol{x},\sigma)$ is the determinate of the Hessian matrix at point $\boldsymbol{x}$ at scale σ, and $L_{xx}(\boldsymbol{x},\sigma)$, $L_{yy}(\boldsymbol{x},\sigma)$, and $L_{xy}(\boldsymbol{x},\sigma)$ are the convolution of the second-order Gaussian derivatives and the image I at point $\boldsymbol{x}$.

Gaussians are considered optimal in scale-space analysis, but computationally expensive (Lowe, 2004). To reduce this expense, Lowe (2004) approximated the Laplacian of Gaussian (LoG) with success (Bay et al., 2008). Therefore, SURF approximates the second-order Gaussian derivatives with box filters. Approximating the Gaussian derivatives provides a computationally inexpensive alternative, but an approximation error is introduced requiring a compensation term (Eq. 3.12 and Eq. 3.13) (Bay et al., 2008; Golub and Van Loan, 2012). With respect to the filter size, the response is normalized. At a location $\boldsymbol{x}$, the determinate of the approximate Hessian matrix (Eq. 3.14) symbolizes the blob responses. Points of interest, or blobs, are located where the approximate Hessian matrix determinate is at a maxima.

$$\beta = \frac{\left\|L_{xy}(\boldsymbol{x},\sigma)\right\|_F \cdot \left\|D_{yy}(\boldsymbol{x},\sigma)\right\|_F}{\left\|L_{yy}(\boldsymbol{x},\sigma)\right\|_F \cdot \left\|D_{xy}(\boldsymbol{x},\sigma)\right\|_F} \qquad \text{(Eq. 3.12)}$$

where $L_{yy}(\boldsymbol{x},\sigma)$, and $L_{xy}(\boldsymbol{x},\sigma)$ are the convolution of the second-order Gaussian derivatives and the image *I* at point $\boldsymbol{x}$, $D_{yy}(\boldsymbol{x},\sigma)$ and $D_{xy}(\boldsymbol{x},\sigma)$ are the convolution of the box filters and the image *I* at point $\boldsymbol{x}$, β is the compensation term, and $\|A(\boldsymbol{x},\sigma)\|_F$ is the Frobenius norm (Eq. 3.13).

$$\|A(\boldsymbol{x},\sigma)\|_F \equiv \sqrt{\sum_{x,y} |A(\boldsymbol{x},\sigma)|^2} \qquad \text{(Eq. 3.13)}$$

where $\|A(\boldsymbol{x},\sigma)\|_F$ is the Frobenius norm of filter $A(\boldsymbol{x},\sigma)$ at point $\boldsymbol{x}$ at scale σ.

$$det\left(H_{approx}\right) = D_{xx}(\boldsymbol{x},\sigma) \cdot D_{yy}(\boldsymbol{x},\sigma) - \left(\beta \cdot D_{xy}(\boldsymbol{x},\sigma)\right)^2 \qquad \text{(Eq. 3.14)}$$

where H_{approx} is the approximate Hessian matrix, $det\left(H_{approx}\right)$ is the determinate of the approximate Hessian matrix, $D_{xx}(\boldsymbol{x},\sigma)$, $D_{yy}(\boldsymbol{x},\sigma)$, and $D_{xy}(\boldsymbol{x},\sigma)$ are the convolution of the box filters and the image *I* at point $\boldsymbol{x}$, and β is the compensation term.

Box filters aid the processing speed of the convolution process because an integral image can be used (Simard et al., 1999; Viola and Jones, 2001). The integral image, *S*, is created by summing all intensity values in the original image *I* above and to the left of point $I(x,y)$ in order to compute the pixel value at $I_\Sigma(x,y)$ (Eq. 3.15). One example of this process is demonstrated in Figure 3.8 on the left side where the red area is the inclusive region where the intensity values are summed to compute the pixel value in the integral image located by a black do. An integral image is only computed once for any single original image. The sum of intensities within the area denoted by the purple box (Figure 3.8 right) is performed by adding and subtracting the pixel values located at A, B, C, and D of the integral image (Eq. 3.16).

$$I_{\Sigma}(x,y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j) \qquad \text{(Eq. 3.15)}$$

where $I_{\Sigma}(x,y)$ is the integral image value at location (x,y) and $I(i,j)$ is the input image where (i,j) denote the location of the input image within the rectangle region formed by the origin and (x,y).

$$\Sigma = I_{\Sigma}(A) + I_{\Sigma}(D) - I_{\Sigma}(C) - I_{\Sigma}(B) \qquad \text{(Eq. 3.16)}$$

where $I_{\Sigma}(A)$ is the value of the integral image at location A (similar location B, C, and D), and Σ is the sum of intensities within the rectangular region.

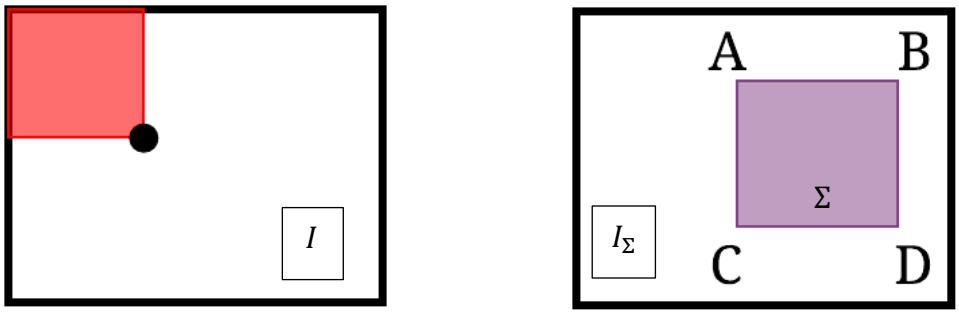

Figure 3.8. Left: original image, and Right: integral image.

The approximation of the Hessian matrix determinate by convolving an input image with the box filter and with use of the integral image. This allows various filter sizes to be used to mask an input image to develop the layers in the scale space. The

scale space is organized into octaves and each octave is segmented into scale levels. The local maxima of the approx. image DoH responses is retrieved by employing a Non-maximum suppression (NMS) search method developed by Neubeck and Van Gool (2006). Once a set of local maxima is located within each octave, the locations and scales are determined by interpolation (Brown and Lowe, 2002).

The hyper-parameters specified for the SURF detector include specifying the number of octaves, the number of scale levels per octave, a metric threshold which controls the number of strong blob responses, and Setting the number of octaves depends on the size of the image. If the image is large, larger filters are wanted to find the larger blobs. The number of scale levels per octave allow for controlling the number of blobs detected with the SURF detector. In order for the SURF detector to find more blobs, increasing the number of filters for a finer scale (Bay et al., 2008). The SURF detector locates blob features represented by a point of interest, and the output of the SURF detector is the position, scale, DoH magnitude, and the sign of the Laplacian (Eq. 3.17) for each point of interest.

$$sgn\{L_{xx}(\boldsymbol{x},\sigma) + L_{yy}(\boldsymbol{x},\sigma)\} = \begin{cases} +1 \Rightarrow light\ blob/dark\ background \\ -1 \Rightarrow dark\ blob/light\ background \end{cases} \quad \text{(Eq. 3.17)}$$

where $L_{xx}(\boldsymbol{x},\sigma)$, and $L_{yy}(\boldsymbol{x},\sigma)$ are the convolution of the second-order Gaussian derivatives and the input image at point $\boldsymbol{x}$ and sgn is the signum function.

3.1.3.2. SURF Descriptor

To describe the blob features identified by the SURF detector, the pixel information of the region around the interest point is summarized. First, the feature orientation is found by calculating the response of the Haar wavelet in the *x* and *y* direction within a circular region of the interest point. The region radius, 6*s*, and the wavelet size, 4*s* x 4*s*, are scale dependent, and therefore, at any scale, integral images can be used to quickly conduct the six filtering operations necessary for the *x* and *y* response (Han et al., 2010). The calculated wavelet responses are then weighted with a Gaussian centered at the interest point and plotted in a 2-D parameter space where the horizontal response strength and vertical response strength are represented as a point. The response values of the points located within a sliding orientation window (60°) are analyzed to locate the dominant orientation. Within each window, the summation of all horizontal and all vertical responses create a local orientation vector. The local vector lengths from each window are compared, and the direction of the vector with the longest length determines the orientation for that interest point (Eq. 3.18).

$$\theta\left(D_x, D_y\right) = \tan^{-1}\frac{D_y}{D_x} \qquad \text{(Eq. 3.18)}$$

where $\theta\left(D_x, D_y\right)$ is the dominate orientation direction and D_x and D_y are the sum of all horizontal and all vertical response strengths, respectively.

At each interest point, a 20*s* x 20*s* square region oriented along the orientation of the interest point is constructed around the feature. These regions are divided into 4x4 sub-regions, and the Haar wavelet response for each sub-region is computed at sample points. With respect to the interest point orientation, the Haar wavelet response in the horizontal, d_x, and vertical, d_y, direction are weighted with a Gaussian ($\sigma = 3.3s$) in order to increase robustness towards shape distortions and location uncertainties. Then for each sub-region the summation of the responses d_x and d_y are computed, $\sum d_x$ and $\sum d_y$, and the summation of the absolute value of the responses $|d_x|$ and $|d_y|$ are computed, $\sum|d_x|$ and $\sum|d_y|$. This forms the four entries of the sub-region feature vector and by concatenating all four sub-regions feature vectors, the descriptor vector, with a length of 64 is constructed. This is specific to setting the hyper-parameter SURF descriptor type to SURF-64 (Han et al., 2010).

3.1.3.3. Matching SURF descriptors

A byproduct of the SURF detector is the sign of the Laplacian which represents either bright blobs on dark backgrounds or vice versa. SURF offers the ability to only have to compare features descriptors from different images that have similar contrast; the matrix containing the feature descriptors from different images will be referred to as descriptor1 and descriptor2. Feature matching determines the correspondence between feature descriptors in different images. The exhaustive search method computes the pairwise distance between feature vectors in descriptor1 and descriptor2 to match descriptor1 to nearest neighbors in descriptor2. Feature vectors from descriptor1 and descriptor2 are normalized to unit vectors ($\boldsymbol{v_1}$and $\boldsymbol{v_2}$) using ℓ^2-norm (Eq. 3.19) and the

sum of square differences (SSD) (Eq. 3.20) between entries from the unit vectors ($v_1$and v_2**)** is computed and signifies the similarity of the two descriptors, score. For SSD, the perfect match value is zero.

$$\|x\|_2 = \sqrt{\sum_i x_i^2} \qquad \text{(Eq. 3.19)}$$

$$SSD(v_1, v_2) = \|v_1 - v_2\|_2^2 = \sum_i (v_{1i} - v_{2i})^2 \qquad \text{(Eq. 3.20)}$$

where $SSD(v_1, v_2)$ is the sum of square differences, v_1 and v_2 are entries from unit vectors.

A sorted matrix is created containing the score in ascending order and the corresponding matched feature points represented by an index pair. Specifying unique matches disregards matches where multiple feature points in descriptor1 match one feature point in descriptor2, and other hyper-parameters specified include the match threshold for selecting strong matches and the threshold ratio for rejecting ambiguous matches. The match threshold represents distance from a perfect match.

3.1.4. Projective Transformation

The geometric transformation matrix maps a subset of matched pairs from one image to the other image, and specifying a projective 2-D transformation allows flexibility for the image plane to tilt and register the image's possible misalignment. If an image pair produces a minimum of 4 match points represented by point1 and point2, from image1 and image2, respectively, then a projective transform can be computed.

The projective transform uses an M-estimator Sample Consensus (MSAC) algorithm to determine the subset of matched points, called inlier points, used to develop the projective transform model by excluding outlier points (Choi et al., 1997). To develop the a 2-D projective geometric transformation matrix, an initial 3x3 matrix identity matrix, T_0, MSAC is an iterative process which selects four matched point pairs to compute the projective transform matrix, T (Torr and Zisserman, 2000). To compute the projective transform, normalized points and normalized transform matrices used and created by translating the sample points so that their centroid is located at [0,0] and computing a scale to make the mean distance from the centroid equal to $\sqrt{2}$ (Torr and Zisserman, 1998). The points after the translation and scale are the normalized points, and the translation and scale are used to compute a matrix, the normalized matrix. It should be noted that the normalized points must satisfy equation Eq. 3.21 (Hartley and Zisserman, 2004).

$$\text{mean}\left(\sqrt{\sum |P_{norm}|^2}\right) = \sqrt{2} \qquad \text{(Eq. 3.21)}$$

where P_{norm} are the normal points.

A matrix is computed with the normalized match point pairs and undergoes singular value decomposition. This matrix is then denormalized using the normalized matrices computed earlier and the final matrix is a 2-D projective transformation matrix, T_i.

If T_{i-1} from the previous iteration has a better fitness (Eq. 21) than the current T_i, then T_i is replaced with T_{i-1}. The number of random trials (N) is updated (Eq. 3.22 and Eq. 3.23), and the entire process is repeated until the N trials are completed or the maximum number of trials is met.

$$\sum_i^N \min(d(u_i, v_i), t) \qquad \text{(Eq. 3.22)}$$

where $d(u_i, v_i)$ is a distance between the matched point pair u_i and v_i, t is the specified threshold maximum distance, and N is the total number of matched points.

$$N = \min\left(N_0, \frac{\log(1-p)}{\log(1-r^n)}\right) \qquad \text{(Eq. 3.23)}$$

where N is the total number of trials, p is a specified confidence parameter, N_0 is a specified number of random trials, and r is seen in Eq. 3.22 and can be described as the probability an inlier will be selected each time a point is selected.

$$r = \sum_{i}^{N} \frac{\text{sgn}(d(u_i, v_i), t)}{N} \quad \text{(Eq. 3.24)}$$

where r can be described as the probability an inlier will be selected for each point selection, t is the specified threshold maximum distance, $\text{sgn}(a, b) = 1$ if $a \leq b$ otherwise 0, $d(u_i, v_i)$ is a distance between the matched point pair u_i and v_i, and N is the number of trials.

Hyper-parameters specified include the maximum number of random trials, the confidence parameter, and the maximum distance, t, which is the allowable distance from a point to the projection of the corresponding point. The value determined for the maximum number of random trials N_0 is an input variable for equation Eq. 3.22 and the actual number of random trails is depends on the matched point pairs and the confidence parameter, p. The confidence parameter, p, is a percentage numeric scalar representing the confidence of finding the maximum number of inliers. Increasing p improves robustness at the cost of added computations.

3.2. Machine Learning Algorithms

3.2.1. Convolutional Neural Network

A convolutional neural network (CNN) is a deep-learning algorithm which utilizes a multi-layer architecture for classification, transfer learning, and feature detection (Lee, 2009). The original image pixel values are transformed by the CNN's

architecture to output the class score. The CNN architecture is created by combining layers into a specific sequence, and due to the explicit function of a CNN, layers contain neurons arranged in 3-dimensions. CNNs have the advantage of being specifically built for an image-based input. The input images must be an RGB image, creating an input volume: height, width, and depth. The height and width are equivalent to the image resolution, and the depth refers to the three-color channels, Red, Green, and Blue (Figure 3.9). The general CNN architecture is constructed of several layers such as a Convolutional Layer, Pooling Layer, and Fully Connected Layer and repetition of each of these layers (Guo et al., 2016).

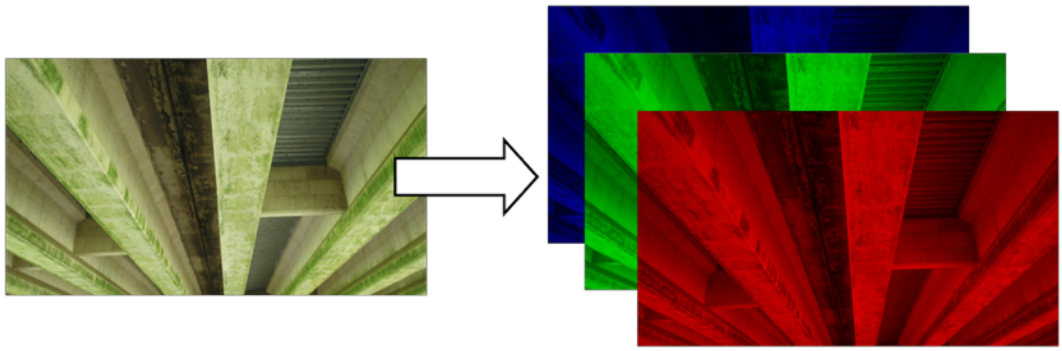

Figure 3.9. The structure of an RGB image.

The Convolutional Layer is tasked with extracting feature maps (activation maps) from the input image (Harley, 2015). The Convolutional Layer contains learnable filters that slide, or convolve, along the height and width of the input volume and compute the dot product at every position. The transformations from the Convolutional Layer are functions of the neuron parameters (weights and biases) and the input volume activations. The Convolutional Layer requires defining hyperparameters: zero-padding,

stride, and depth that govern the output volume and requires defining hyper-parameter: filter size (Karpathy, 2018d). Zero-padding adds zeros around the border of an image which allows control of the output volume's height and width. The hyper-parameter is specifying the size of the zero-padding. The hyper-parameter stride determines the number of pixels the filter moves after each convolution computation. The hyper-parameter depth specifies the number of filters used in each layer, and therefore, specifies the number of feature maps. The neurons along the depth of the feature map are connected only to the neurons in a local region. This neuron connectivity is called the receptive field, a hyper-parameter specifying the size of the filter (along the height and width). These filters create 2-dimensional feature maps that contain the response of the filters at each position (Dumoulin and Visin, 2016). The trained filters have learned to activate at certain key features (low-level and high-level features) depending on the input volume size and specified hyper-parameters. Once trained, the result is a Convolutional Layer comprised of trained filters (containing weights and biases) that are used to convolve over an input image and produce an output volume of 2-dimensional feature maps representing areas of likely key features (Deshpande, 2016).

An activation function (also called a non-linearity) is typically introduced after each Convolutional Layer to avoid the class scores, resulting in a linear function of the input values. This is the result of two matrices collapsing into a single matrix. One of the many activation functions commonly used is the Rectified Linear Unit (ReLU) where the function results in the activation thresholded at zero (Karpathy, 2018d). The

elementwise activation function is applied to the output volume comprised of feature maps.

The output volume of a Convolutional Layer is reduced in a Pooling Layer, often called down sampling (Karpathy, 2018d). One of the most common types of Pooling Layers is called a max pooling layer. Max pooling reduces the feature map (height and width) dimensions by choosing a value within the specified window (filter) size to represent the region. Therefore, rather than storing all the values in the window, the retrieved value represents the region. This reduces the number of parameters and computations while maintaining important information (Deshpande, 2016). The max pooling operation is applied separately to each feature map along the output volume depth. The hyper-parameters: window size (filter size) and the stride, must be specified. Advantages to reducing the input representations include: (1) reducing the number of parameters and computations, therefore, controlling overfitting; (2) results in a more manageable feature dimension; and, (3) makes the network invariant to small distortions, translations, and transformations present in the input image (Lee et al., 2009).

A Fully Connected Layer implies that every neuron from the preceding layer is connected to every neuron in the following layer (Harley, 2015). The Fully Connected Layer uses high-level features from the feature map created by the output volume from the previous layer to compute class scores and the class scores result in a vector that has a length equal to the number of classifications. The Fully Connected Layer searches for high values in the feature maps that represent high-level features and determine which features correlate to which classification. The ReLU activation function mentioned

earlier is common after Fully Connected Layers with the exception of the final Fully Connected Layer. For classification problems, the final Fully Connected Layer can utilize a Softmax classifier which implements a cross-entropy loss function. The Softmax classifier uses the class score vector from the Fully Connected Layer to generate probabilities for each specified class over all possible classes. The Softmax function (Eq. 3.25) takes the class score vector, z (Eq. 3.26), and returns a vector of normalized positive values between zero and one that sum to one (Karpathy, 2018a). This allows the Softmax classifier to utilize the cross-entropy loss and return each class score as unnormalized log probabilities (Eq. 3.27) (Karpathy, 2018a). The goal is to maximize the log-likelihood (for a loss function, minimize the negative log-likelihood of the correct class) (Awais, 2017).

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \qquad \text{(Eq. 3.25)}$$

where $\boldsymbol{z}$ (Eq. 3.25) is the unnormalized log probabilities of the class.

$$\mathrm{z} = \sum_{l=1}^{m} \boldsymbol{w^T x} + b \qquad \text{(Eq. 3.26)}$$

where $\boldsymbol{w}$ is the weight vector, $\boldsymbol{x}$ is the feature vector, and b is bias.

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \quad \text{(Eq. 3.27)}$$

where f_j is the jth element of the class score vector f, and L_i is the unnormalized log probabilities for each class score.

Other layers can be incorporated into the CNN architecture. For instance, layers that perform regularization and data loss. Regularization layers, such as Local Response Normalization Layers and Dropout layers perform data processing to control the capacity of the CNN (Karpathy, 2018c). The resulting regularization loss could be seen as a penalization for model complexity. Generalization is aided by Local Response Normalization (Slijepcevic, 2012). Dropout Layers reduce overfitting by only keeping a neuron active if the neuron probability satisfies the specified probability threshold, defined hyper-parameter. Otherwise, the neuron is set to zero (Srivastava et al., 2014). Loss Function Layers measure the prediction (class scores) compatibility with the labeled ground truth resulting in the data loss. An example of a Loss Function Layer is the Softmax classifier.

A CNN can be constructed with any number and order of layers. These layers undergo purely supervised learning to train a large, deep network. Supervised learning is any machine learning strategy where the labeled input is used to first train and build the model, rather than unsupervised or reinforcement learning. Additionally, all algorithms described in this work or implemented in previous research efforts described in this

thesis are supervised learning techniques. The parameters of each neuron of each layer are tuned using an iterative forward and backward pass method. Training a CNN can be broken down into four steps: (1) forward pass; (2) total loss function; (3) backward pass; and, (4) update parameters (Deshpande, 2016). The forward pass is when input images are propagated through the CNN architecture to create a fixed-length feature vector. Once the feature vectors are created, the predicted class is compared to the categorical labels of the images in the training data. The total loss is comprised of the data loss and regularization loss. The data loss is in the form of an average over the summation of every sample's data losses (Eq. 3.28) (Karpathy, 2018a). The data losses from Loss Function Layer(s) and regularization loss from Regularization Layer(s) construct the total loss function (Eq. 3.29) which quantify the quality of the current parameters (Awais, 2017).

$$L(\boldsymbol{w}) = \frac{1}{N}\sum_i L_i \qquad \text{(Eq. 3.28)}$$

where N is the amount of training data and L_i is the unnormalized log probabilities for each class score

$$E(\boldsymbol{w}) = L(\boldsymbol{w}) + R(\boldsymbol{w}) \qquad \text{(Eq. 3.29)}$$

where $L(\boldsymbol{w})$ is the data loss, $R(\boldsymbol{w})$ is the regularization loss, and $E(\boldsymbol{w})$ is the total loss.

The gradient of the total loss function is used during the backward pass training step. In order to minimize the loss, the weight contributing most to the loss needs to be

optimized. This is accomplished by solving for the gradient of the loss function to locate the steepest descent (Karpathy, 2018b). The steepest descent yields the best direction to change the weight vector. The gradient of the loss function is used when back-propagated through the network with stochastic gradient descent with momentum (SGDM) (Eq. 3.30) (Murphy, 2012). The SGDM updates the parameters, and this process is repeated until the total loss is minimized to within an acceptable error or a maximum number of epochs has been performed.

$$\boldsymbol{w}_{l+1} = \boldsymbol{w}_l - \alpha \nabla E(\boldsymbol{w}_l) + \gamma(\boldsymbol{w_l} - \boldsymbol{w_{l-1}}) \quad \text{(Eq. 3.30)}$$

where $\boldsymbol{w}$ is the parameter vector containing weights and biases (Note: $\boldsymbol{w_{l-1}}$ is the previous parameters, $\boldsymbol{w_l}$ is the current parameters, and $\boldsymbol{w_{l+1}}$ is the updated parameters), α is the learning rate, γ is the momentum factor, $E(\boldsymbol{w})$ is the loss function, and ∇ is the gradient function.

The robustness of the trained network depends on the classified image dataset used for training and the ability to construct an adequate architecture. The input dataset for a CNN must contain many images belonging to various categories. In this paper, the AlexNet CNN (Krizhevsky et al., 2012), which uses a subset of the ImageNet (Deng et al., 2009) dataset, is utilized to augment the dataset. The AlexNet CNN consists of roughly 10,000 different categories labeled throughout approximately 1.2 million images. These images also undergo data augmentation in order to reduce overfitting.

The AlexNet CNN main architecture consists of five Convolutional Layers and three Fully Connected Layers. Other layers in the AlexNet CNN include two Normalization Layers after the first and second Convolutional Layers, three Max Pooling Layers (the first and second layers are located after the first and second Convolutional Layers, respectively, and the third layer is after the last Convolutional Layer), two Dropout Layers after the first two Fully Connected Layers, and ReLU activation functions after each Convolutional Layer and the first two Fully Connected Layers. Finally, the Softmax classifier is applied after the final Fully Connected Layer. The 1000-way Softmax classifier wants to maximize the log-likelihood of the correct class (Krizhevsky et al., 2012). The AlexNet layers are shown clearly in Figure 3.10 and the detailed architecture is displayed in Figure 3.11.

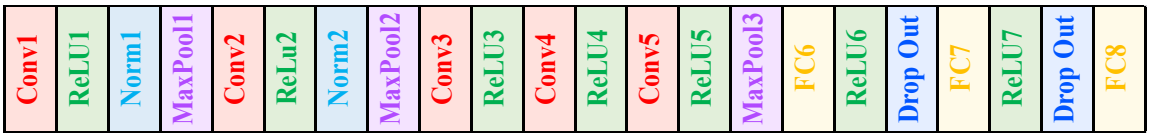

Figure 3.10. Layer architecture of AlexNet CNN (Krizhevsky et al., 2012).

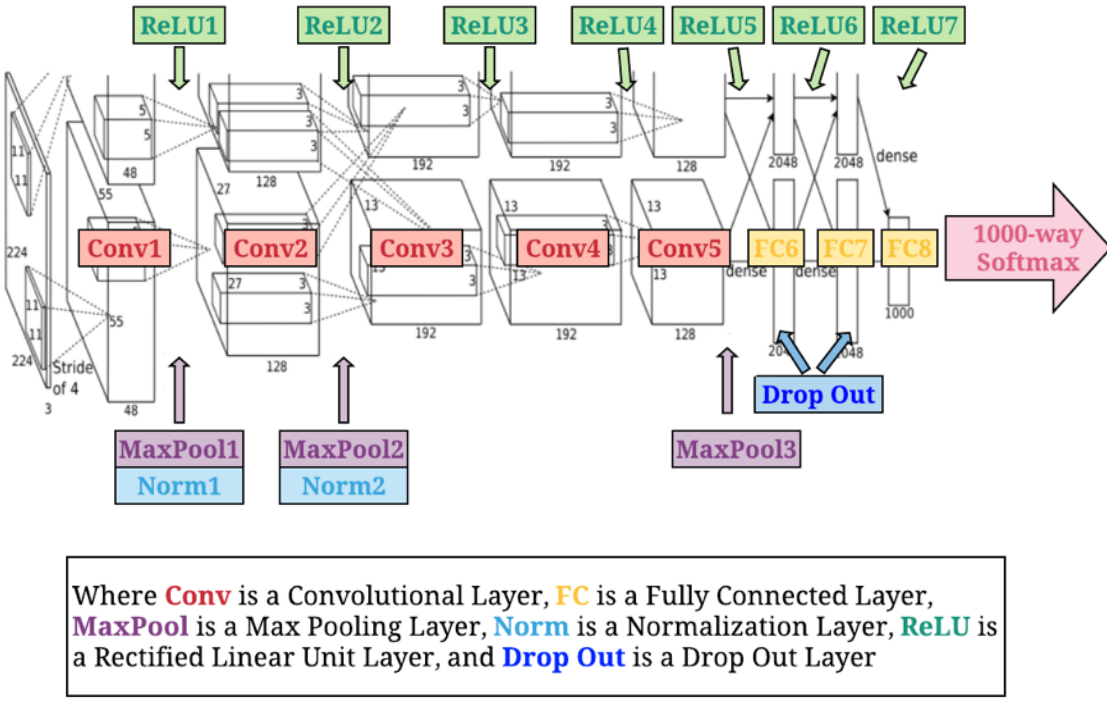

Figure 3.11. The detailed architecture of AlexNet CNN (Krizhevsky et al., 2012).

Krizhevsky et al. (2012) specify a zero-mean Gaussian distribution with a standard deviation of 0.01 to initialize the weights in each layer. Neuron biases were initialized to 1 in the second, fourth, and fifth Convolutional Layer, and the fully-connected hidden layers and the remaining layers are initialized to 0. The Dropout Layer hyper-parameter probability is established as $p = 0.5$.

Krizhevsky et al. (2012) assigned hyper-parameters for stochastic gradient descent with momentum. The momentum factor, γ, potentially reduces oscillation when converging by considering the previous iteration ($\boldsymbol{w}_{l-1}$) during the current gradient step ($\boldsymbol{w}_l$). In each iteration, a subset of the dataset, or mini-batch, is analyzed, and the number of iterations and the size of the subset can be specified and adjusted via

hyper-parameter tuning. For the AlexNet CNN, the training options are the following: batch size of 128, 90 total cycles (epochs) over the training data, a momentum of 0.9, and the initial learning rate of 0.01.

Convolutional neural networks (CNNs) can be used to categorize an entire image (classification) or apply a sliding window method to localize an object in an image (detection). However, to perform a detection task, it has been proven more efficient, accurate, and computationally inexpensive to implement a region-based CNN (R-CNN) (Sermanet et al., 2012). The application of an region-based convolutional neural network (R-CNN) addresses the limitations of the sliding window approach (Zhang et al., 2015).

3.2.2. Region-based Convolutional Neural Network

Region-based convolutional neural networks (R-CNNs) were developed to allow for the localization of multiple objects in an image (Girshick et al., 2014). R-CNN object detectors classify proposed regions using a CNN and a regression model. The traditional method for training an R-CNN is segmented into three stages. The first stage retrieves region proposals, the second stage fine-tunes a CNN, and the third stage creates a regression model.

The first stage retrieves region proposals. The R-CNN developed by Girshick et al. (2014) used a selective search method, developed by Uijlings et al. (2012). The selective search method outputs proposed regions possibly containing objects. This is done by identifying smaller objects, and depending on color spaces and other metrics, the small objects are combined with a hierarchical grouping. However, an alternative is locating region proposals using an EdgeBox method (Zitnick and Dollár, 2014). For

each input image, the EdgeBox method independently locates region (object) proposals called candidates. The candidates for each image are compared to the image's manually labeled ground-truth regions (Figure 3.12), and the intersection-over-union (IoU) is measured. To determine the IoU of the candidate and the ground-truth (Eq. 3.31), the area of overlap (intersection) and the area of union (union), are determined (Figure 3.13). If the IoU between the ground-truth label and the candidate satisfy a pre-defined positive overlap range threshold, then the region is considered a positive sample and vice versa for determining the negative samples. If the IoU does not satisfy the positive or negative threshold range, then the sample is disregarded. This process is completed for each manually labeled input image. The collected regions of the positive and negative samples, now referred to as training samples, are then cropped, and warped to serve as the input layer of the CNN structure (Uijlings et al., 2013). The output of the first stage is a collection of warped regions of training (positive and negative) samples.

Figure 3.12. An example of comparing a ground-truth label (purple) and a candidate (yellow).

Figure 3.13. An example of determining the intersection (left) and union (right) for a ground truth and a candidate.

$$IoU = \frac{Intersection}{Union} \quad \text{(Eq. 3.31)}$$

where $Intersection$ refers to the area of overlap, $Union$ is the area of union, and IoU is the intersection-over-union.

In the second stage, either a single warped region or a predetermined number of warped regions composing a mini-batch are used to fine-tune the convolutional neural network (CNN). The CNN used can either be a new CNN or an existing CNN. Using an existing CNN is called transfer learning, and is a preferred alternative to creating a new CNN, as the wealth of knowledge already embedded in the existing CNN is useful in helping to distinguish further classes in the new image set. Creating a robust and adequate new neural network requires a dataset containing millions of images divided into thousands of categories partnered with a well-planned CNN architecture. Transfer learning allows for the use of an existing, pre-trained CNN to be fine-tuned using the

warped regions, and various hyper-parameters, to adjust the existing parameters. The existing network is adapted to the new object detection task and allows for significantly fewer images in the focal dataset (Girshick et al., 2014). The AlexNet CNN, as previously mentioned, is a robust CNN that is adjusted in the second stage. In order to integrate the AlexNet CNN into the R-CNN, the architecture is altered in the final layers for the network to learn the new domain. More specifically, the AlexNet CNN uses a 1000-way Softmax classifier, with respect to the original 1000 categories and utilized in the final Fully Connected Layer, is replaced with the new value exclusive to the new dataset. Also, the output layer, which contained the name of each classification, is emptied so that the network learns the new category labels. Fine-tuning the CNN is done by an iterative forward and backward pass similar to the training of a CNN detailed in Section 3.2.1. Also, in this stage, feature vectors and their respective category are stored and used in following stage. The output of the second stage is a trained CNN specific to the new domain and a collection of feature vectors classified by the defined categories.

Finally, in the third stage, a bounding box regression model is developed. For each classification, a linear regression model is trained using the stored feature vectors with their predicted locations (retrieved from the training samples) and their actual locations (ground-truth). The data processed in this method is considered high-dimensional data and to reduce computing time, a least-squares regression method or regularized support vector machine (SVM) method. The predicted and actual location will be referred to as a training pair. The location of the training sample, S, and the ground-truth, G, are each denoted by the box [*x, y, width, height*] where the coordinate

(x, y) is the top left corner of the box. To train the bounding box regression model, the model learns four translation functions (f_x, f_y, f_w, and f_h) which map the location of the training sample, S, to the location of the ground-truth, G, and results in the proposed region, P (Eq. 3.32, Eq. 3.33, Eq. 3.34, and Eq. 3.35) (Girshick et al., 2014). The four functions (Eq. 3.36) include the feature vector and the weight coefficients.

$$P_x = S_w f_x(S) + S_x \quad \text{(Eq. 3.32)}$$

$$P_y = S_h f_x(S) + S_y \quad \text{(Eq. 3.33)}$$

$$P_w = S_w \, e^{f_w(S)} \quad \text{(Eq. 3.34)}$$

$$P_h = S_h \, e^{f_h(S)} \quad \text{(Eq. 3.35)}$$

where f_x and f_y are scale-invariant translations of the coordinates x and y respectively, f_w and f_h are log-space translations of the width and height, and e is the exponential function.

$$f(S) = \boldsymbol{w}^T \phi(S) \quad \text{(Eq. 3.36)}$$

where $\boldsymbol{w}$ are the weight coefficients and $\phi(S)$ is the feature vector for training sample, S.

The weight coefficients can be determined using a linear regression model with ridge regularization and stochastic gradient descent (SGD) to optimize (Zhu, 2011; Girshick et al., 2014; Hsieh et al., 2008; Xiao, 2010). The momentum term equal to zero

in SGDM (Eq. 3.30) resulted in SGD. The hyper-parameters used includes the regularization strength, the mini-batch size, and initializing the weights and biases. The bounding box regression model is computationally inexpensive but is exposed to over-fitting. Girshick et al. (2014) proved using high regularization reduced the chance of over-fitting. The bounding box regression model corrects the predicted window of the classified object to localize the detected object (Felzenszwalb et al., 2010). Once the third stage is finished, training the R-CNN is complete and results in a trained R-CNN detector specialized in classifying specific regions.

To determine the presence and location of an object in an image, a trained R-CNN detector (Figure 3.14) uses the retrieved region proposals, the trained CNN, and the trained bounding box regression model (Girshick et al., 2014). First, an image undergoes selective searching and each of the candidates are then warped for the CNN. Then, the CNN analyzes and classifies each predicted region as well as outputs a feature vector for use in the bounding box regression model. Finally, the classified region's feature vector is used to refine the proposed region and localize the object, and the regions classified into one of the specified categories is determined, meaning objects classified as *background* are not a specified output of the R-CNN. Overall, to use an R-CNN detector for object detection, the input is an image or video frame and the output is proposed regions of interest classified and located by a bounding box containing four elements, [*x, y, width, height*], where *x* and *y* are the coordinates of the top left corner of the bounding box.

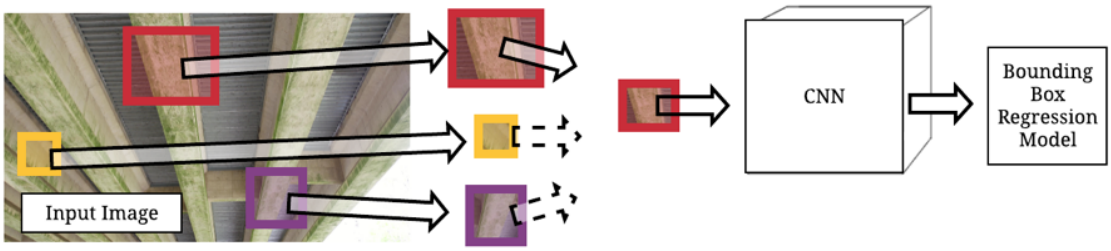

Figure 3.14. An example of a trained R CNN detector with application in the images prevalent to the work presented in this thesis.

4. METHODOLOGY

To work towards an unmanned automated bridge inspection approach, key critical components should be automatically detected and tracked in video data acquired with an unmanned aerial system (UAS) so that damage on these critical components can then be identified and evaluated accordingly. In this work, a methodology for automatically detecting and tracking prestressed beams in unmanned aerial system (UAS) imagery is presented. This methodology can be viewed as a framework for any bridge or material type, and can easily be augmented with existing damage detection algorithms (German et al., 2012; German et al., 2013) to provide comprehensive, quantitative, robust condition ratings for the superstructure and all other parts of the bridge. In this section, an overview of the proposed methodology for detection and tracking of prestressed beams in UAS images is first discussed. Then, the dataset used for the development and validation of the proposed methodology is presented. The dataset was augmented to ensure that the algorithms comprising the proposed methodology are robust and not prone to overfitting. The specific ways in which the dataset was augmented is discussed at the end of this section.

4.1. Overview

As previously mentioned, the proposed methodology is a hybrid machine learning-machine vision approach to detecting and tracking prestressed concrete beams in the UAS imagery. Due to the beams frequency, shape, and differing skews within one image, the use of machine learning or machine vision methods in isolation was not

successful in accurately detecting all beams. In this method, first, rectangular regions of interest were recognized as “beam flange” regions via a region-based convolutional neural network (R-CNN) trained with labeled images from the dataset described in Section 4.2 and Section 4.3; these regions are termed “proposed regions of interest” or “pROIs” as they do not represent the full beam or all beam pixels in a given image, but a region in the image where there is likely a section of a single beam. Following this, these pROIs were extended by machine vision algorithms which account for the distinct edges on either side of the beam flange as well as the nature of bridge design/construction (beam flanges have two edges which appear relatively parallel to one another, and beams span between girders or abutments in the same direction as the detected edges). At this stage, the beams which were identified as pROIs and further conform to the outlined standards were transformed into ROIs (fully recognized prestressed beam regions). Each prestressed beam ROI was then tracked throughout subsequent frames. This was achieved by implementing a feature-based tracking method employing Speeded-Up Robust Features (SURF) extraction to locate key features in each image and calculate the projective transform. The projective transform mapped the ROI from one frame to the next. Figure 4.1 shows an overview of the proposed methodology in bridge beam detection and tracking. The various stages in the methodology will be discussed in great detail in the following sections.

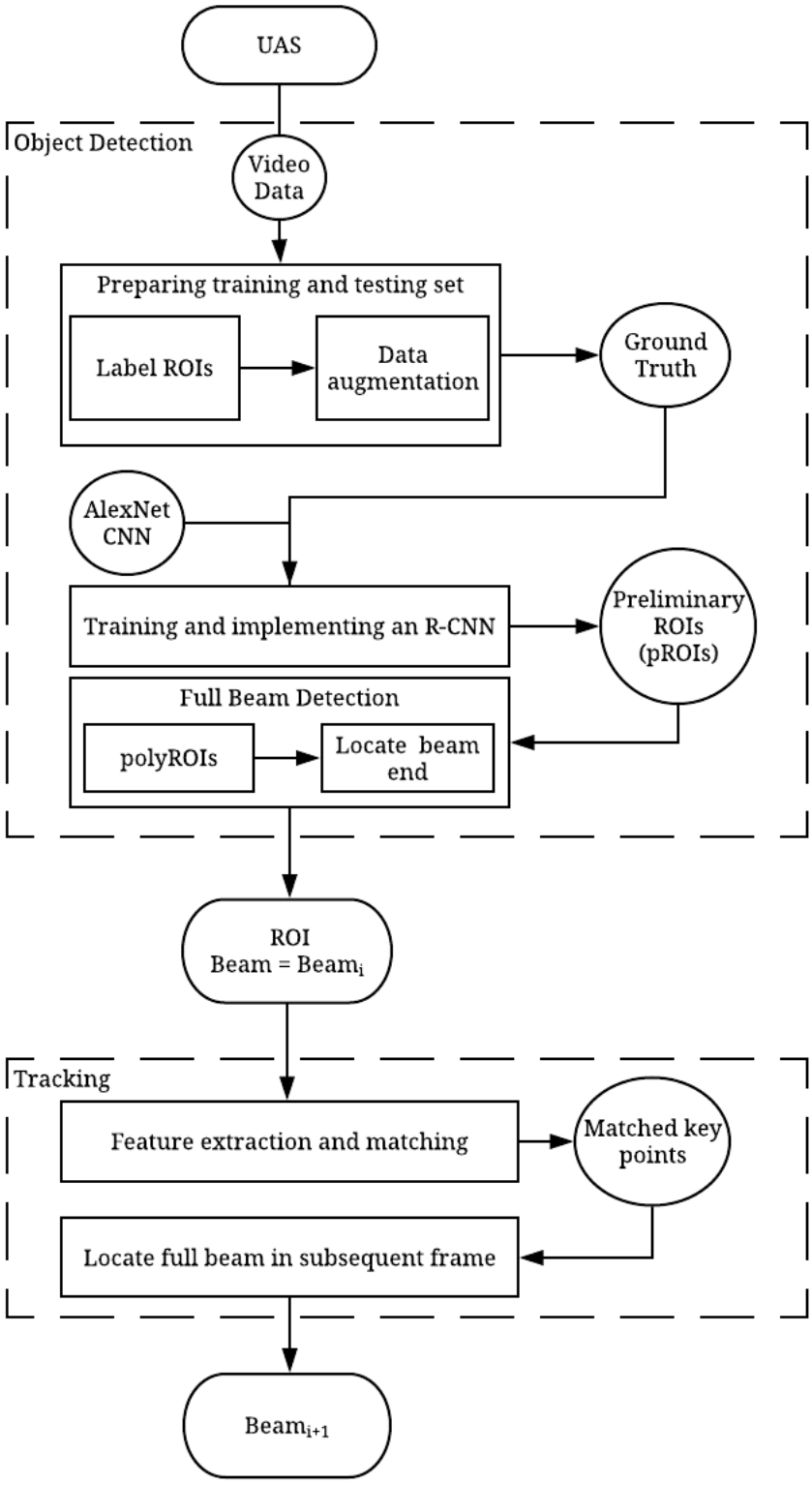

Figure 4.1. Overview of bridge beam detection and tracking methodology.

4.2. Dataset

An unmanned aerial system (UAS) equipped with a high-resolution camera was used to capture videos of three prestressed concrete multi-beam or girder bridges located in Monroe County, Pennsylvania (Figure 4.2). The UAS video of the three bridges totaled coverage of six spans, 50 individual I-beams, and over 4,700-ft of beams. Figure 4.3 and Figure 4.4 are aerial images of the three bridges. The structural information about the bridges was extracted from the Homeland Infrastructure Foundation – Level Data (HIFLD) and is displayed in Table 4.1 (HIFLD, 2017). All three bridges are owned and maintained by the state highway agency (Pennsylvania Department of Transportation (PennDOT)). It should be noted that in Table 4.1, the condition ratings are prescribed according to the Federal Highway Administration (FHWA) *Bridge Inspector's Reference Manual* (BIRM) as the following: (8) Very good condition – no problems noted; (7) Good condition – some minor problems noted; (6) Satisfactory condition – structural elements show some minor deterioration; and, (5) Fair condition – all primary structural elements are sound but may have minor section loss, cracking, spalling or scour (FHWA, 2012). Further explanation of the condition ratings and data items can be found in the FHWA *Recording and Coding Guide for the Structure Inventory and Appraisal of the Nations Bridges* (FHWA, 1995) and in Section 1 and Section 2.

Figure 4.2. Map of Pennsylvania locating Monroe County in red.

Figure 4.3. Bridge detail: SR 80 (LR 794).

Figure 4.4. Bridge detail: I 80 WB (LR1009) and I 80 EB (LR1009).

Table 4.1. Relevant bridge details extracted from HIFLD (HIFLD, 2017).

Bridge Detail	SR 80 (LR 794)	1-80 WB (LR1009)	1-80 EB (LR1009)
Year built or refurbished	1959 (1986)	2006	2006
Number of lanes	4	2	2
Average Daily Traffic (Number of vehicles)	67929	9351	11099
Type of service	Highway	Highway	Highway
Under bridge service type	Waterway	Waterway	Waterway
Main structure type	Multi-beam girder, prestressed concrete	Multi-beam girder, prestressed concrete	Multi-beam girder, prestressed concrete
Number of spans/beams per span	4 spans/ 10 beams	1 span/ 5 beams	1 span/ 5 beams
Total length of spans	323.2 ft.	149.0 ft.	146.0 ft.
Inspection type	C-routine/crane	N/A	N/A
Structural condition: Current (previous)	5 (5)	7 (7)	7 (7)
Superstructure condition: Current (previous)	6 (6)	8 (8)	8 (8)
Superstructure condition rating	Satisfactory	Very Good	Very Good
Sufficiency Rating	62.8	95.1	95.1

An unmanned aerial system (UAS) was used to collect a total of eleven videos surveying the listed three bridges. The video resolution was 3840x2160. The videos collected provided many different characteristics of beams. The videos captured multiple bridge spans from varying distances. The inspection occurred during weather conditions fluctuating from overcast, partially overcast, and sunny, exposing the bridges to different lighting conditions. The bridges recorded were built in 1959 (Bridge 1) and 2006

(Bridge 2 and Bridge 3). The 47 year age difference results in beams with a wide range of environmental exposure, and both bridges were built over water also adding to the varying exposure.

The database used for training, testing, and validation of the methodology presented in this paper was assembled from these videos. Frames were extracted from the videos such that the total number of images originally used to construct the training and testing sets was 1,760. From this set, a subset of 178 images that displayed beam segments was created for the development and testing of the methodology presented herein. These images vary in the amount of exposure, the condition of lighting, and the distance between the sensor (UAS) and beams. There are 176 individual appearances of beams in the dataset. As the methodology presented in this work involves the creation of a supervised machine learning algorithm (R-CNN), various categories of image regions present in these images were labeled in each image to establish a labeled training set. These categories consist of the positive object classes (the object of the detection task: *beam flange*, *beam* (sections of the beam that are not the flange)*, concrete that is not beam* (e.g., columns, abutments, supports)*, not concrete,* and *joint*) which are useful in training the algorithm to better distinguish boundaries between the object of the detection class and all other objects which may be present in the image. The instances where beam flanges are located by the R-CNN is used rather than the entire beam because every beam web present in an image cannot be seen by the UAS. The beam web of the beam directly above the UAS and on the far sides of the image cannot be seen, but the beam flange can be seen regardless of its location in the image. One example of the

region label annotation for a bridge image is displayed in Figure 4.5. In total, 7,129 bounding boxes were labeled and stored as the coordinates of the object's location in the form: [*x y width height*]. Details regarding the ground-truth labels for each object in the dataset (number of bounding boxes and the total area of pixels) can be found in Table 4.2.

Table 4.2. Image dataset for ground truth labeled regions of interest.

ROI Class	Beam	Beam Flange	Concrete Not Beam	Joint	Not Concrete
Bounding Boxes	870	4205	811	349	894
Total Area (million pixels)	258	640	911	934	1117

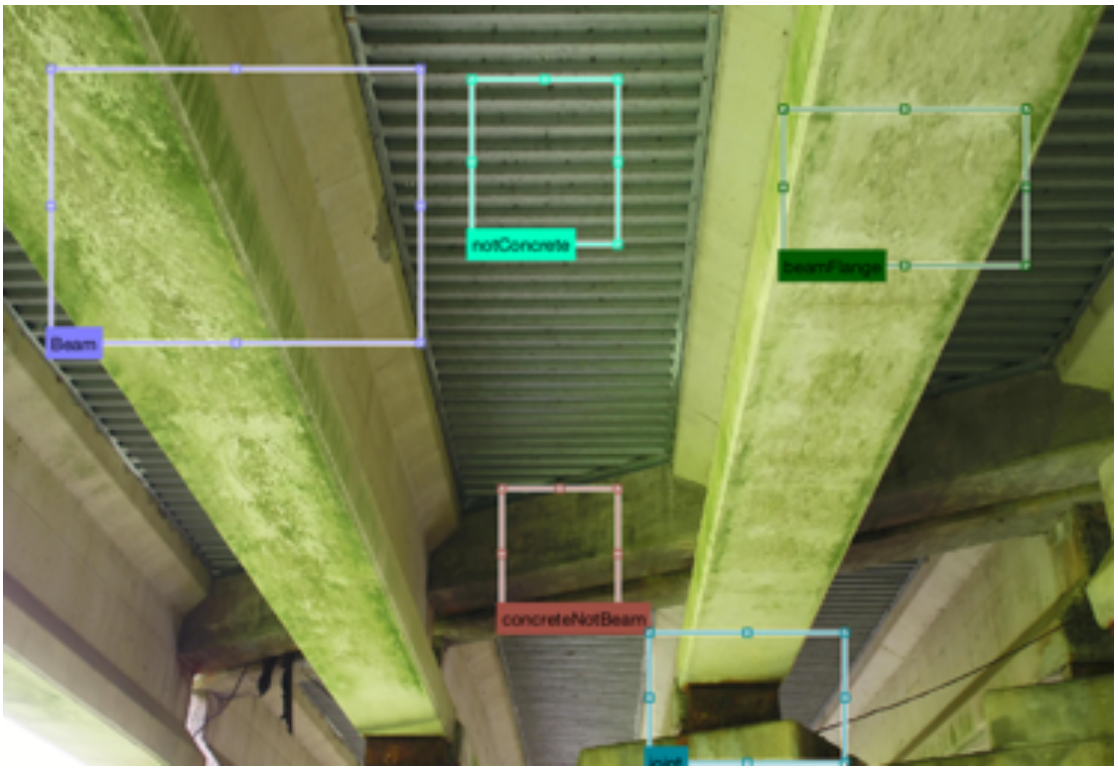

Figure 4.5. Manually labeled regions of interest for positive object classes.

4.3. Data Augmentation

Data augmentation has previously been proven important to the development of a successful neural network (Krizhevsky et al., 2012; Howard, 2013; Wu et al., 2015). A successful network is minimally affected by variance due to illumination, relative object placement, or lens distortion. As mentioned, the data used for this work was collected under varying environmental conditions; however, the dataset was further augmented to ensure a robust, generalizable approach to automated beam detection and tracking. The data augmentation procedures used in this paper are congruent with those described in Krizhevsky et al. (2012) and Wu et al. (2015). Krizhevsky et al. (2012) implemented a four-corner crop and a middle section crop image extraction, and Wu et al. (2015)

implemented horizontal reflection. For this dataset, the original images were horizontally reflected, and then the original and horizontally flipped images were implemented in a four-corner crop and middle-section crop. The images created from the four-corner and middle section crop were created by extracting patches that were approximately 65% of the original size. Figure 4.6 shows one instance of the four-corner crop and Figure 4.7 shows the entire image undergoing the four-corner crop and middle-section crop. Therefore, the 3840x2160 images generated five cropped images of size 2496x1404.

Figure 4.6. Upper left corner crop example.

Figure 4.7. Four corner crop (all colors but white) and middle section crop (white) example.

Then, the 12 images (original (1), horizontally flipped (1), four-corner crop (8), middle section crop (2)) each underwent further data augmentation. Color casting was applied to various channels of the RGB image by randomly generating a Boolean value for each channel and randomly shifting the range values of the channel. Horizontal and vertical stretching and two random rotations was applied to each image for lens distortion invariance. Figure 4.8 displays examples of various effects of data augmentation on images in the dataset.

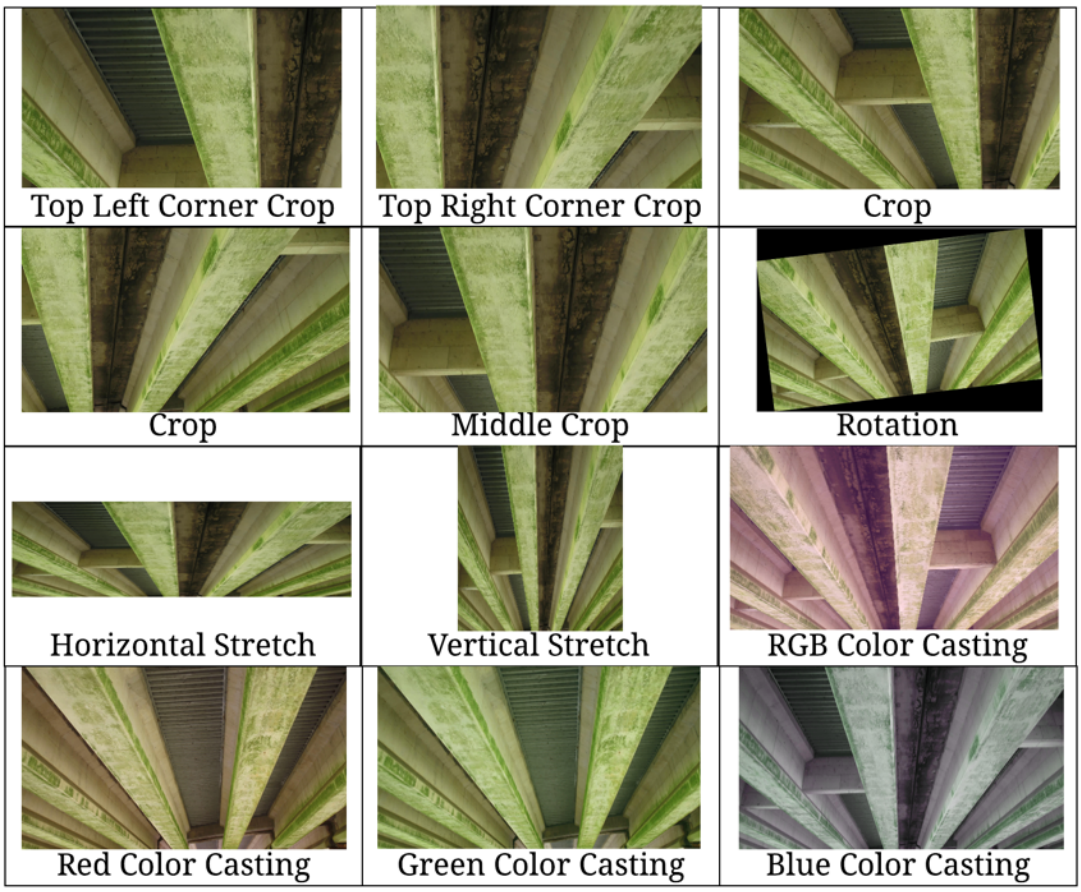

Figure 4.8. Samples of different data augmentation techniques performed on the image dataset.

For training, testing, and validation of the R-CNN, the dataset was broken down into training, testing, and validation subsets. There was zero overlap between the training and testing sets. Data augmentation was applied only to the images in the training set (Krizhevsky et al., 2012; Wu et al., 2015). The construction of all three datasets will be discussed in more detail in the following section, and the details of the overall algorithmic implementation will be presented in Section 7.

5. OBJECT DETECTION

In this section, the first step in the proposed methodology in prestressed concrete beam detection and tracking is presented: object (prestressed concrete beam) detection. The object detection approach is composed of a machine learning algorithm (region-based convolutional neural network (R-CNN)) and machine vision algorithms (edge detection, relative properties) to uniquely identify the location of prestressed beams in UAS imagery. Figure 5.1 shows the detailed overview of the proposed method for object detection developed for beam detection in UAS imagery.

The proposed method consisted of first labeling the ground-truth for each image in the dataset and then, splitting the images into training and testing sets. The training set with ground-truth labels was used to train the R-CNN, and then, the R-CNN detector was employed to locate the pROIs (rectangular regions that likely contain beam flanges). The last step was to extend these pROIs to cover the full length and width of the beam via various machine vision algorithms including the Canny edge detector (Canny, 1986) and the Hough transform (Ballard, 1987). Each of these steps will be discussed in detail in the following sections.

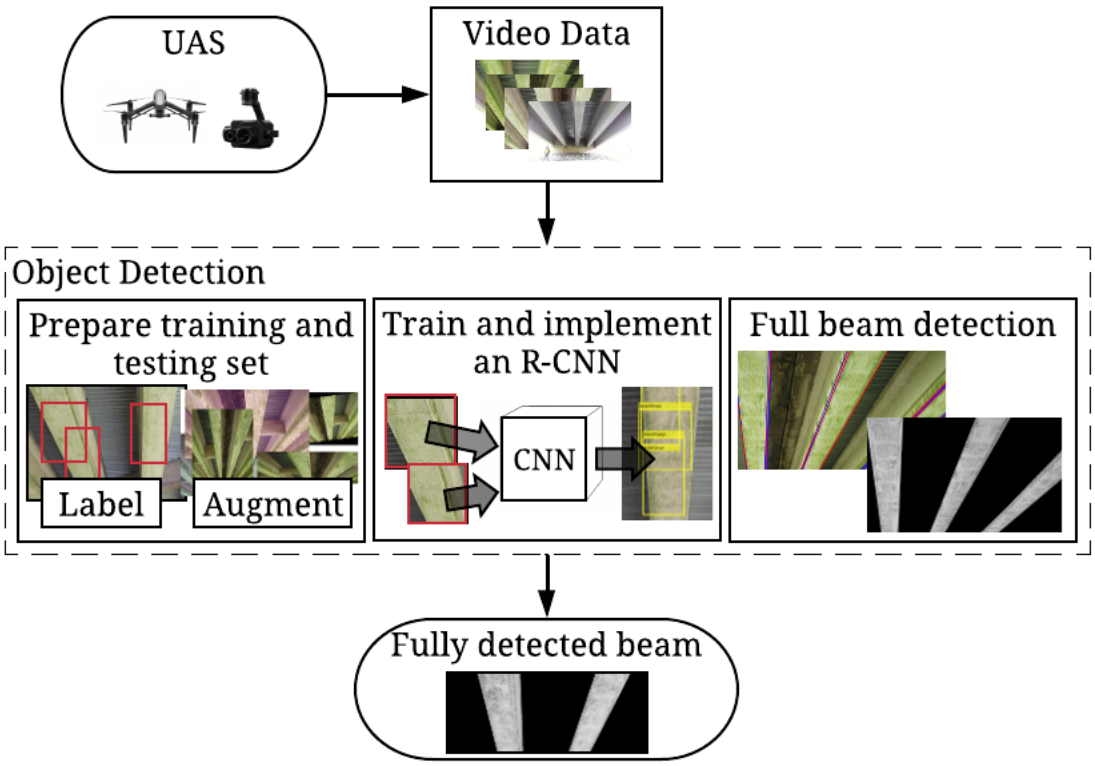

Figure 5.1. Overview of the proposed method for structural element detection.

5.1. Prepare Training and Testing Set

The dataset described in the Section 4 was used for training and validation of the R-CNN model as well as for testing of the R-CNN and overall object detection and tracking algorithms presented in this work. As mentioned previously, the video data retrieved by the unmanned aerial system (UAS) was used to create 1,760 individual images, of which 178 were used to identify and label 7,129 bounding boxes classifying image regions as ground-truth regions of interest (*beam flange* vs. all other classes). The training to testing ratio established in this work was approximately 8:2 (precisely 8.3:1.7) with 148 images randomly chosen to represent the training set, and the remaining 30 images set aside for testing. The quantity and pixel area of the labeled

bounding boxes, with respective category labels, are shown in Table 5.1 and Table 5.2. Table 4.2, in Section 4, contains the total quantity and pixel area for the 178 images in the dataset as they pertain to each of the positive object classes. Table 5.1 and Table 5.2 detail the training set, comprised of 83% of the 178 images, and the testing set, containing the remaining 17% of the dataset, respectively.

Table 5.1. Image training set for ground truth labeled regions of interest.

ROI Class	Beam	Beam Flange	Concrete Not Beam	Joint	Not Concrete
Bounding Boxes	729	3522	672	289	729
Total Area (million pixels)	215	535	761	781	930

Table 5.2. Image testing set ground truth labeled regions of interest.

ROI Class	Beam	Beam Flange	Concrete Not Beam	Joint	Not Concrete
Bounding Boxes	141	683	139	60	165
Total Area (million pixels)	43	105	149	153	186

The training to testing ratio of exactly 8.3:1.7 was selected because the training to testing ratio of the number of bounding boxes and pixel area for each label, regardless of data shuffle, consistently matched this ratio. When a 9:1 or 8:2 ratio was chosen, the

number of bounding boxes and pixel areas typically resulted in a ratio near 8.3:1.7. Therefore, the number of images to complement this ratio was chosen.

Data augmentation was applied only to the randomly selected images which make up the training set (Krizhevsky et al., 2012; Wu et al., 2015). The distribution of images for each type of data augmentation is shown in Figure 5.2. The total number of images after data augmentation including the original training set is 12,094.

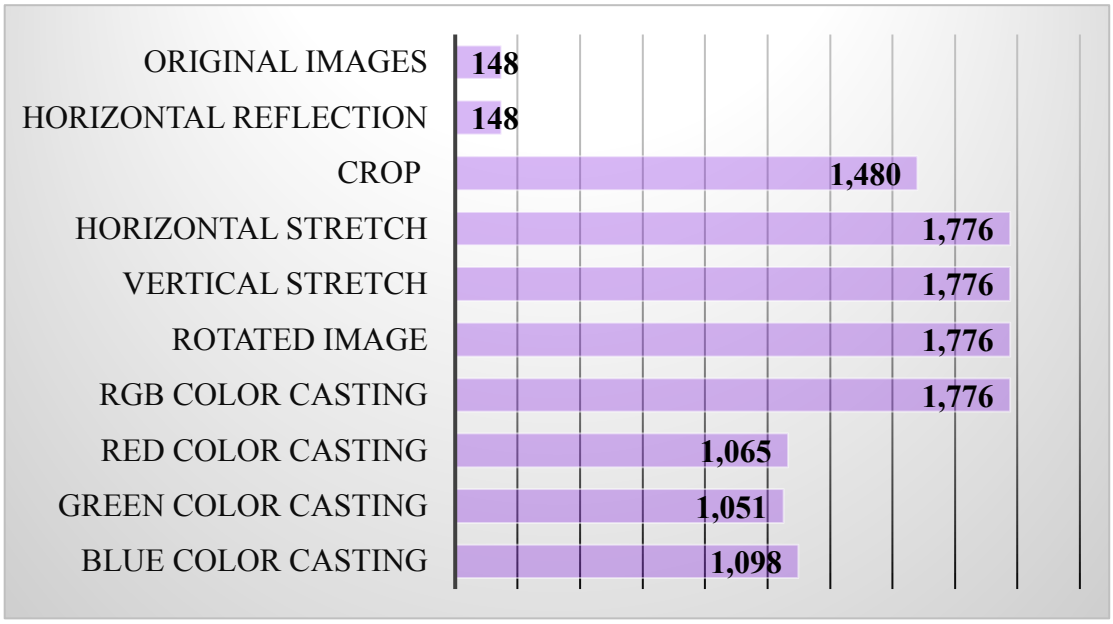

Figure 5.2. Training set image total after data augmentation.

5.2. Train and Implement a Region-based Convolutional Neural Network

A region-based convolutional neural network R-CNN detector was used to classify regions in the image according to five classifications: *beam flange, beam, concrete not beam, not concrete,* and *joint*. The trained R-CNN detector analyzed each input image and predicted rectangular sections, or regions of interest (ROIs), containing

objects classified into one of the five categories. The output of the R-CNN detector was a table containing the ROIs classification label and bounding box [*x, y, width, height*]. The specific interest of this work was the *beam flange* class. Regions labeled *beam flange* were termed the proposed regions of interest, or pROI*,* due to the pROI only containing a segment of the beam flange and not the entire beam. To train the R-CNN, the training set with manually labeled categories (ground-truth), the defined training options (hyper-parameters), and the AlexNet CNN with a modified architecture were used. Note, the hyper-parameters explicitly stated in this research were determined using hyper-parameter tuning by first setting the hyper-parameters to coincide with common practices and then adjusting or changing the hyper-parameter to increase the model's accuracy and/or to reduce computational expense. The training of the R-CNN detector that was used in this research is explained below, however, more detail on the background of a CNN and R-CNN can be found in Section 3.2.1 and Section 3.2.2.

The R-CNN was trained on five regions of interest (ROIs) classifications using the 12,094 training images. The training set used to develop the R-CNN detector is detailed in Section 5.1. To review, the first stage uses the EdgeBoxes algorithm to retrieve the candidates. The candidates were compared to the ground-truth to determine the intersection-over-union (IoU) value, and the result was the collection of warped regions representing positive and negative samples. The hyper-parameters explicitly determined in this stage were the number of extracted candidates and the IoU positive and negative sample threshold. The maximum number of candidates extracted in this research was set to 2000 regions. The IoU thresholds called positive overlap range and

negative overlap range were determined and the candidates with an IoU value (Eq. that satisfied either threshold were stored as positive and negative samples. The positive overlap range was between 0.5 and 1, and the negative overlap range was between 0 and 0.3. The candidates that met the positive and negative overlap range were then considered positive and negative samples, respectively, and if the IoU value did not satisfy these thresholds, the sample was discarded. This value range used for overlap ranges retrieved similar quantities of positive samples and negative samples which suggested the positive object classifications were well represented. The positive and negative samples, now termed training samples, were warped to satisfy the input layer of the AlexNet CNN [227x227x3], and used as the input image into the CNN.

In the second stage of training, the warped regions from the previous stage were used to modify the parameters the layers of the AlexNet CNN. As previously mentioned, transfer learning was utilized via the AlexNet CNN with an adjusted architecture. The final layers of AlexNet CNN were altered in this work in order to implement the network into the developed R-CNN. The CNN was tasked to learn six classifications (5 object classes: *beam flange, beam*, *concrete not beam*, *not concrete*, and *joint*, and 1 *background* class). The AlexNet CNN final Fully Connected layer was modified to a 6-way Softmax classifier and the output layer was emptied to learn the domain-specific classifications. The hyper-parameters specified by Krizhevsky et al. (2012) such as, zero-mean Gaussian distribution with a standard deviation of 0.01 to initialize the weights in each layer, neuron biases were initialized to 1 in the second, fourth, and fifth

Convolutional Layer, the Fully-Connected hidden layers and the remaining layers are initialized to 0, and the Dropout Layer probability $p = 0.5$, were preserved.

Fine-tuning the CNN was broken down into four steps: (1) forward pass; (2) total loss function; (3) backward pass; and, (4) weight update. The hyper-parameters that were determined for the fine-tuning process include the mini-batch options and optimization algorithm options. The specified mini-batch options were the mini-batch size set at 128, the maximum number of epochs set at 20, and the training data directed to shuffle once before training. During back propagation, the optimization algorithm options were the hyper-parameter values and methods specified. The optimization method chosen was the Stochastic Gradient Descent with Momentum (SGDM) optimizer; recall equation Eq. 3.30:

$$\boldsymbol{w}_{l+1} = \boldsymbol{w}_l - \alpha \nabla E(\boldsymbol{w}_l) + \gamma(\boldsymbol{w_l} - \boldsymbol{w_{l-1}}) \qquad \text{(Eq. 3.30)}$$

where $\boldsymbol{w}$ is the parameter vector containing weights and biases (Note: $\boldsymbol{w_{l-1}}$ is the previous parameters, $\boldsymbol{w_l}$ is the current parameters, and $\boldsymbol{w_{l+1}}$ is the updated parameters), α is the learning rate, γ is the momentum factor, $E(\boldsymbol{w})$ is the loss function, and ∇ is the gradient function.

The hyper-parameters specified for SGDM were the initial learning rate, the learning rate schedule, the momentum, and the gradient threshold. To avoid adjusting the pre-trained network parameters too quickly, the initial learning rate value was set at

0.001 ($1/10^{th}$ of the pre-trained rate), and the learning rate schedule was specified the continued use of the initial learning rate. The hyper-parameter momentum was set to 0.9 which allowed significant contribution of the previous parameter update on the subsequent iteration. The training loss output indicated that the training was stable and not going to incorrectly diverge, and therefore the gradient threshold was not needed. Once the second stage was completed, the new CNN was developed and the feature vectors representing each category were used in the next stage.

In the third and final stage, a bounding box regression model was comprised of five linear regression models (one for each positive object class) that were trained using the corresponding feature vectors from the last Convolutional Layer with their predicted locations (retrieved from the training sample information in stage one) and their actual locations (ground-truth). Each linear regression model for each object class was trained by learning four translation functions mapping the predicted location to the ground-truth location. An in-depth review of a bounding box regression model can be found in Section 3.2.2. Recall the translation functions (Eq. 3.16). The hyper-parameters specified here were the regularization constant was set to equal 1000, the mini-batch size was set at 128, and the initial weights and biases were initialized to zero. The weights were updated iteratively using a forward and backward pass with the mini-batches and went through the training samples once, or one epoch.

At this point, the trained R-CNN was used to perform the initial phase of the beam detection method: *beam flange* detection.

The trained R-CNN detector was implemented to detect all proposed ROIs containing a segment of *beam flange*. The proposed ROI(s) are represented by bounding boxes [*x y width height*], label (classification), and score (measure of confidence). Figure 5.3 shows one example of this output with each bounding box corresponding to the categorical label *beam flange*. This figure does not show the other classified regions found using the R-CNN such as *beam*, *concrete not beam*, *not concrete*, and *joint*, but these regions are identified as well by the R-CNN. The bounding box coordinates for each proposed ROI were used in the following section to identify the full beam. The bounding box coordinates were cropped in order to create a less complex region of interest ROI that was utilized in the following stages of the overall methodology. The region of interest output from this section is referred to as the proposed region of interest (pROI).

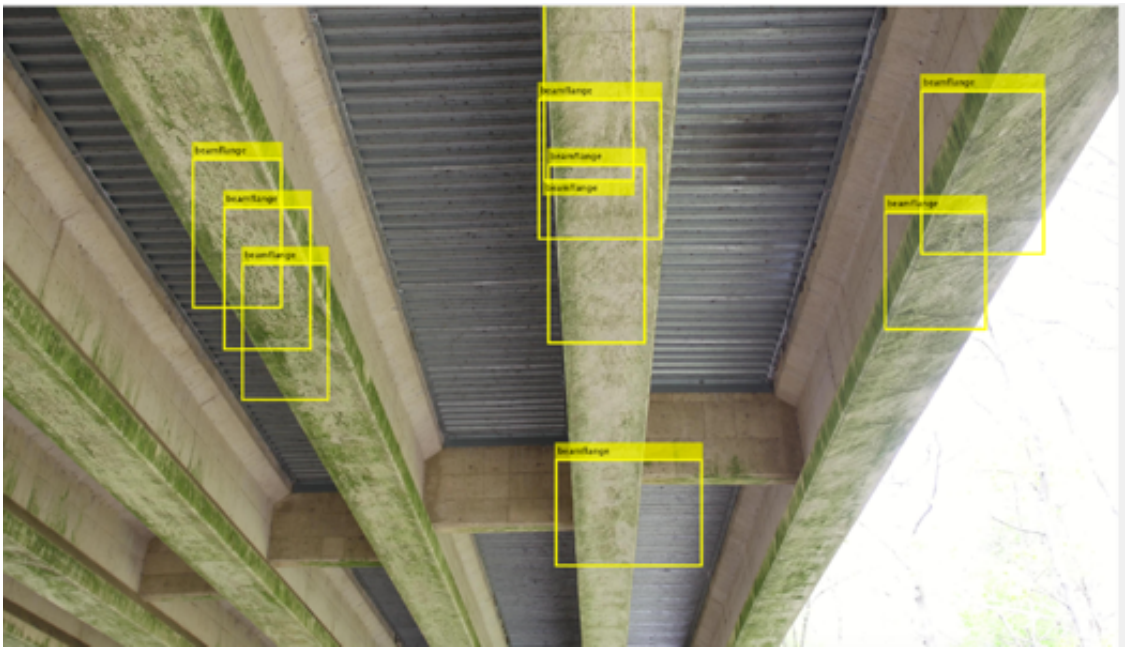

Figure 5.3. R CNN object localization output, showing pROIs.

5.3. Full Beam Detection (pROI to ROI)

Once the proposed region of interests (pROIs) were detected via the R-CNN, the best regions for each beam were extended to identify the full region of interest: the entire beam flange. This phase contains several intermediate steps which were designed specifically for the object of interest as they were defined according to key visual characteristics of bridge beams. The full beam detection algorithm is composed of three main steps (with sub-steps and intermediate checks to ensure accuracy):

Step one: Locate beam edges within pROI

Step two: Extend pROI to image edges and mask polyROI

Step three: Locate beam end and establish final ROI

5.3.1. Step one: Locate Beam Edges Within pROI

In Step one, each pROI was analyzed to locate both edges of the beam via an adaptive Canny edge detection technique and the Hough Transform. The edges of the beam located in this step were stored as a pair of line segments and delivered to Step two. The key visual characteristic assumed in this step was that each edge of the beam was able to be represented by two straight, semi-parallel line segments located near or within the boundary of the pROI.

To find these line segments, each bounding box proposed by the R-CNN was extracted to allow for analysis of a simple (small, most likely only containing a section of beam and minimal background pixels) image, rather than a complex image. Every extracted pROI was analyzed using an adaptive threshold-based Canny edge detector to identify edges and the Hough transform to locate the line segments possibly representing beam edges.

The adaptive threshold-based Canny edge detector employed the Canny edge detector to determine the high and low threshold of the edge strengths, adjusted this threshold, and implemented the new threshold when reapplying the Canny edge detector. To review, the Canny edge detector has five main steps: (1) smooth; (2) find intensity gradient; (3) edge thinning; (4) apply double threshold; and (5) track edges by hysteresis. Further detail on the Canny edge detector can be reviewed in Section 3.1.1. The hyper-parameter constant throughout the adaptive Canny edge detector was the Gaussian distribution set to $\sqrt{1.2}$ (Eq. 3.1). First, the Canny edge detector was applied to the extracted pROI image to retrieve the initial strong edge and weak edge values; the initial

double threshold was chosen heuristically by the Canny detection algorithm. Second, the strong and weak strength values and the range of these values were analyzed and adjusted as needed by increasing or decreasing the range values. Third, the adjusted double threshold range was applied when the Canny edge detector was used again on the extracted pROI image. By adjusting these values, the Canny edge detection was able to accurately retrieve the edges of the beam flange. The final output was a binary image, or white and black pixels, locating the edges in the extracted pROI.

The binary image was evaluated using the Hough transform in order to merge individual edge pixels into linear line segments. The Hough transform employed a voting method that determined the location of line segment. This was done by analyzing the accumulator array, determining peaks, and extracting line segments. A detail review of Hough transform is found in Section 3.1.2. For the accumulator array, the hyper-parameters specified were the rho-resolution, which was set to one, and theta, which was set to the range -70 to 70. When determining the peaks, the hyper-parameters specified were the number of peaks, the peak threshold, and the suppression neighborhood. The number of peaks was set to 1, and the peak threshold depended on the largest number of points within one of the accumulator array's bins and stated the minimum number of points in a bin was required to be 70% of that value to be considered a peak. The suppression neighborhood size was determined by determining the size of the accumulator array, decreasing the values by 2%, and choosing an odd value equal to or greater than that value. When the lines were extracted, the hyper-parameters specified were the fill gap and minimum line length. The fill gap

between two line segments was required to be less than 70% of the bounding box width, and the minimum line length was required to be 40% of the bounding box width. These greedy hyper-parameters were used to find one edge of the beam flange, and the identified edge provided information to find the second edge. Again, based on the assumption that the edges of the beam flange were two straight, relatively collinear lines. The angle of the second edge was explicitly stated based on the angle extracted from the first edge, and if two collinear line segments cannot be identified within the pROI, the pROI was expanded in the direction perpendicular to the detected line segment a number of pixels relative to the size of the bounding box of the pROI until a collinear line segment was found. These line segments were treated as line pairs.

Figure 5.4 shows extracted pROI binary images. On the left, the result from the Canny edge detector without the adaptive threshold followed by the Hough Transform, and, on the right, the result from the Canny edge detector with the adaptive threshold followed by the Hough Transform. Figure 5.5 shows the original gray-scale image with the superimposed line output from the described method. Figure 5.6 shows the location of all the stored line segments correlating with beam flange edges. A combination of blue and red lines indicates a line pair.

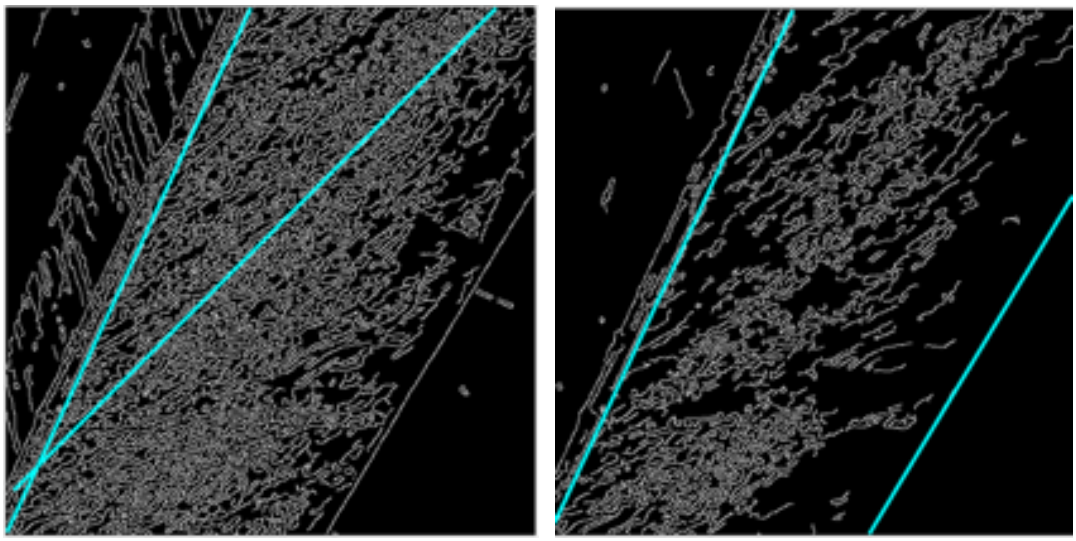

Figure 5.4. Lines detected via Canny + Hough Transform, Left: Canny edge detector without adaptive threshold; Right: Canny edge detector with adaptive threshold.

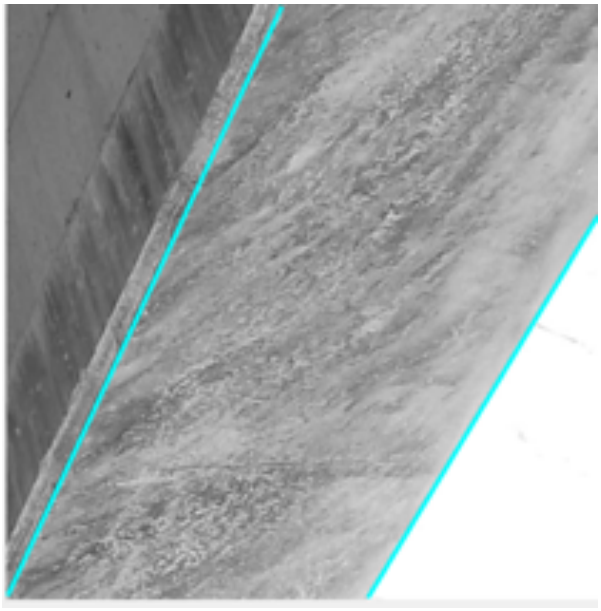

Figure 5.5. Grayscale pROI image with superimposed detected beam flange edges.

Figure 5.6. Example of proposed beam flange edges from all pROI detected in the image.

5.3.2. Extend pROI to Image Edges and Mask polyROI

Once beam edges were detected for a given pROI, the collinear line segments and any joints detected by the R-CNN were used to make a masked image where only the full beams were seen. The collinear line segments (Figure 5.6) are extended to the border of the image (Figure 5.7). If the R-CNN did not detect any *joints* in the input image, then the endpoints of these extended lines were used to create the polygon mask seen in Figure 5.8. At this stage, the pROI was transformed to a polygon shape, similar to that of the beam flange, and thus, was called a polyROI. In many cases, the polyROI is actually the full beam (ROI); however, it may be the case that the beam ends before the edge of the image. This will be dealt with in Step three.

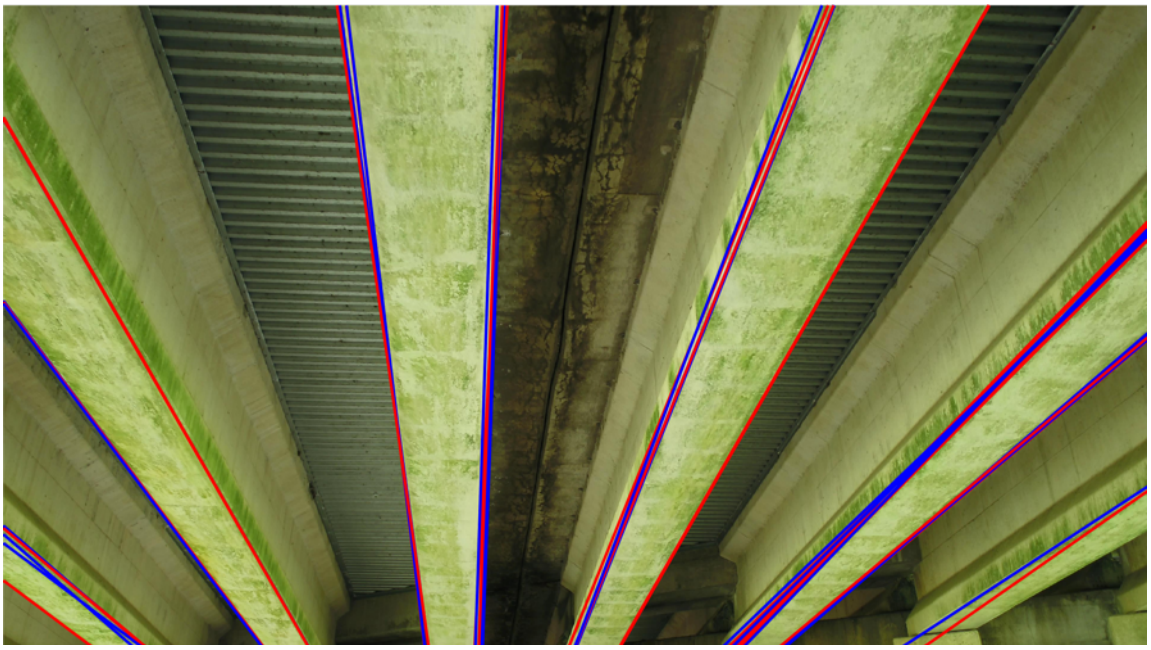

Figure 5.7. Example of paired line segments extended to image border.

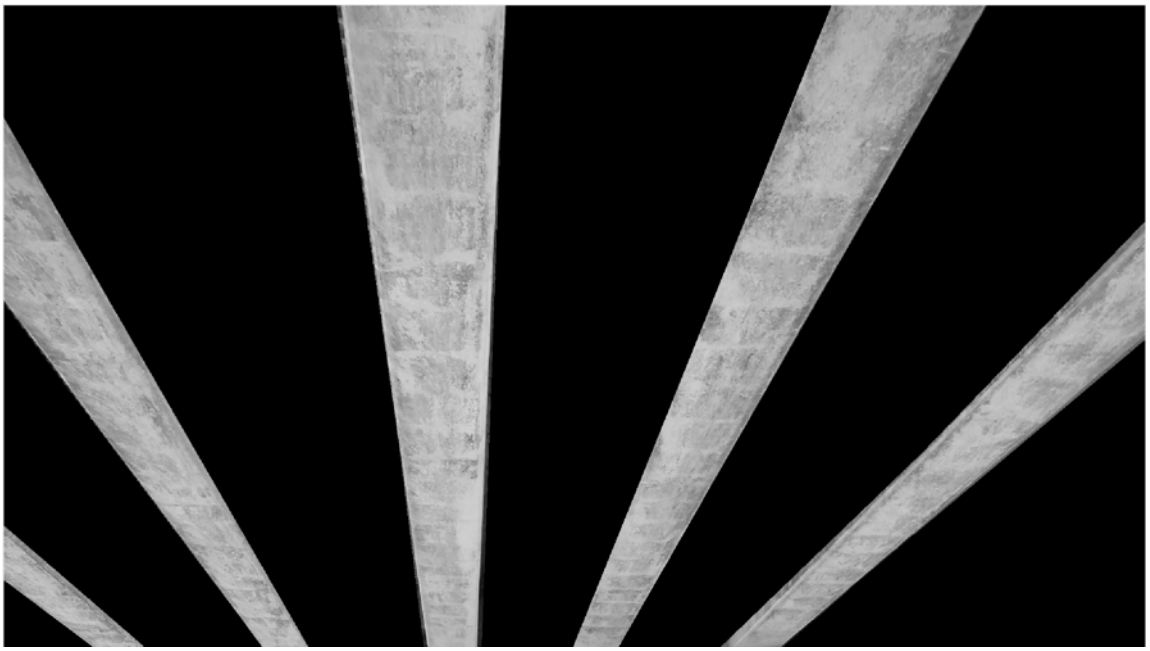

Figure 5.8. Example polyROI of extended lines without detected joints.

Recall: one of the five positive object classes defined and labeled in the images used to train the R-CNN was *joint*. At this stage, the image regions classified by the R-CNN as *joints* were utilized to further distinguish the shape of the beams. If *joints* were detected in the input image via the R-CNN, the information of the *joint's* bounding box was used to identify an approximate location of the end of the beam. In this way, domain specific knowledge about the general construction of bridge elements was used to improve the overall detection and tracking algorithm. The approximate location was achieved by retrieving the bottom boundary line of the bounding box. It should be noted that by using this as the boundary for the beam flange, the actual beam edge remained inside the masked polygon ROI. The location of the exact beam edge was found in Step three detailed in the next section. The reason for applying this bottom boundary when beam ends are present in an image was because the resulting masked image from the extended lines alone sometimes creates an inaccurate mask. Figure 5.9 shows the outcome of extending the collinear line segments to the border of the image. When the collinear lines intersected, the masked region created one object rather than multiple objects. In other words, masking the extended lines produced less proposed regions than beams found by the R-CNN (Figure 5.10). Figure 5.11 shows the outcome of masking an image after applying the bottom boundary when a joint was detected by the R-CNN.

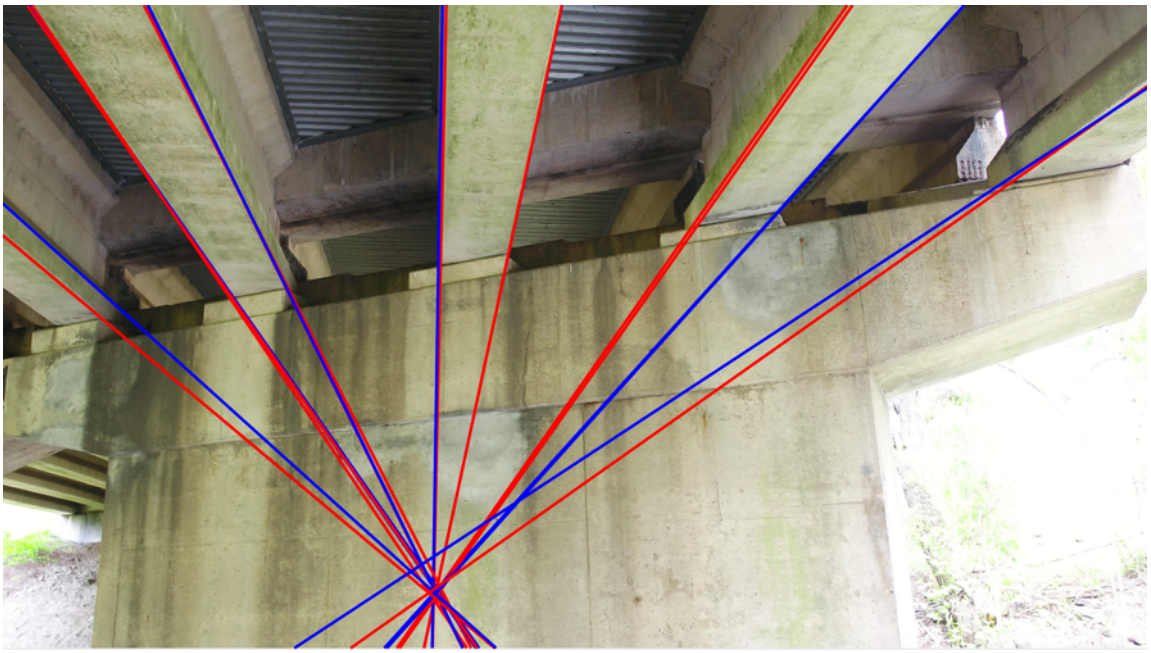

Figure 5.9. Example of extended lines without acknowledged detected joints.

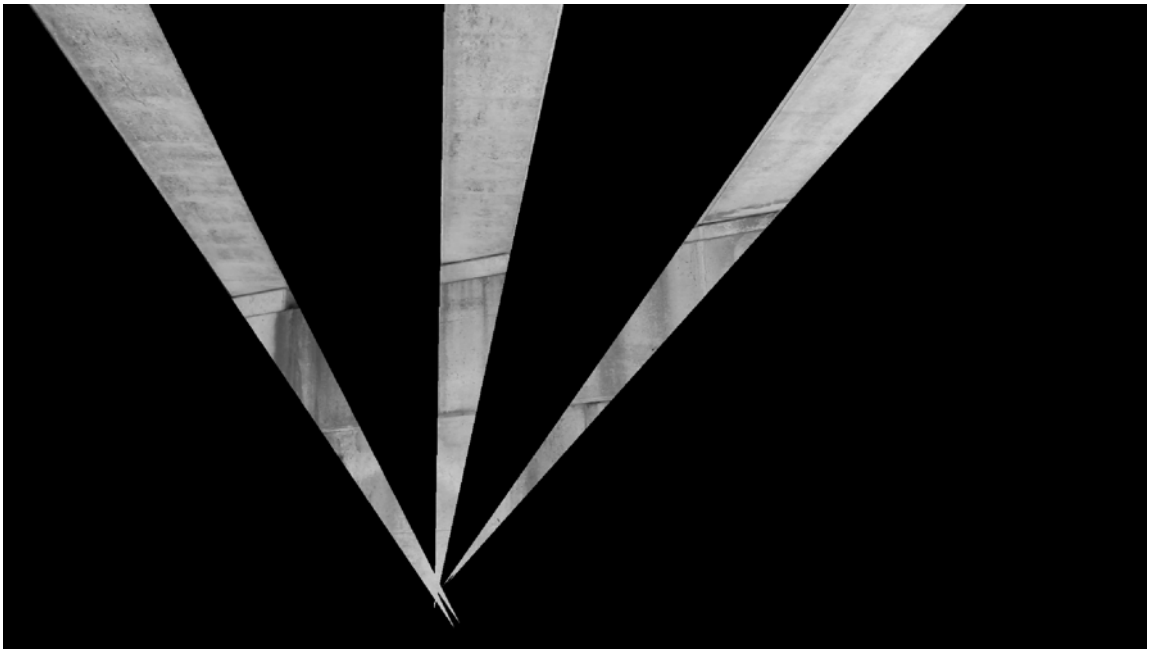

Figure 5.10. Example of polyROI of extended lines without acknowledged detected joints.

Figure 5.11. Example of polyROI extended lines with acknowledged detected joints.

Additionally in Step two, a check was performed regarding the likelihood that the two detected, extended lines were actually wide enough apart to contain a large portion of the beam flange. In Figure 5.6, some of the blue and red paired lines were not identified as a *beam flange* and therefore are not masked (Figure 5.7). This check can also be seen in Figure 5.9, Figure 5.8, Figure 5.10, and Figure 5.11. The masked polyROI was used in Step three to finalize the full beam ROI.

5.3.3. Step Three: Locate Beam End and Establish Final ROI

In Step Three, once the polyROIs were identified, the masked image containing only the full beam flange, termed the ROI, was the final result of the object detection algorithm. If a *joint* was not identified via the R-CNN, then the polyROI became the ROI. If the R-CNN detected *joints* in the image, then an accurate location for the end of the beam was needed. This was done by analyzing each beam within the polyROI and

the bounding box associated with the *joint* was cropped and analyzed. Analysis of the cropped *joint* bounding box was similar to the process of analyzing the *beam flange* bounding box (from Step one). In the proposed methodology, it was assumed that the edge of the beam is nearly orthogonal to the flange edge. Therefore, the beam edge was identified with the same Canny edge detection with an adaptive thresholded algorithm that was previously mentioned. The angle threshold for the Hough Transform method was adapted to be relative to the beam edge angles: angles within a range of +-90º of those detected as *beam flange* edges are considered at this step. In practice, the angles of the *beam flange* edges were rotated between 80 and 90 degrees to implement a theta threshold on the Hough transformation. At this stage, the edges of these *beam flanges* were located and the masked image was updated. At this point, the masked image is designated as the ROI and the beam is considered fully detected. Figure 5.12 shows the final ROI fused with the original image.

Figure 5.12. Example of the final ROI fused with the original image.

6. TRACKING

To successfully identify and track prestressed concrete beams in unmanned aerial system (UAS) imagery, the use of a region-based convolutional neural network (R-CNN) alone for real-time object tracking was insufficient for two reasons: (1) the R-CNN detection process lacks rapid or near-real-time output of ROIs due to the computational complexity of the algorithm; and (2) the ROIs localized a beam segment rather than the complete beam (the subsequent machine vision operations described in Section 5.3 were required to provide the full result). Although the second limitation was addressed with the machine vision operations developed, the first limitation needed to be minimized. The lack of a rapid or a near real-time output was addressed by utilizing a tracking algorithm. In this way, rather than using the R-CNN and full beam detection method to detect beams in every video frame, recognized beams were tracked in subsequent frames. The R-CNN and full beam detection algorithm were instead applied to a smaller subset of data in these subsequent frames. Utilizing the tracking algorithm provided additional benefits including: (1) the continued tracking of beams throughout each video frame which increased the success of identifying all bridge beams; and (2) the tracking algorithm provided an opportunity for storing global location in future damage detection algorithms.

The tracking algorithm developed in this work is summarized in Figure 6.1. The tracking algorithm developed in this research was developed knowing that the desired object to be tracked is stationary while the camera was moving, which is contrary to

most conventional tracking problems, as discussed in Section 2.5. The proposed methodology consists of two primary steps: (1) feature extraction and matching; and (2) developing a projective transformation to locate fully recognized beams in subsequent frames. The goal of feature extraction in this work was to identify key points in every image within the previously detected regions (ROIs from the object detection procedure (Section 5)). The neighborhood of each interest point was represented as a feature vector, and the descriptor vectors were matched and used to map the transformation of one image to a subsequent images. More detail regarding each of these steps is provided in the following sections.

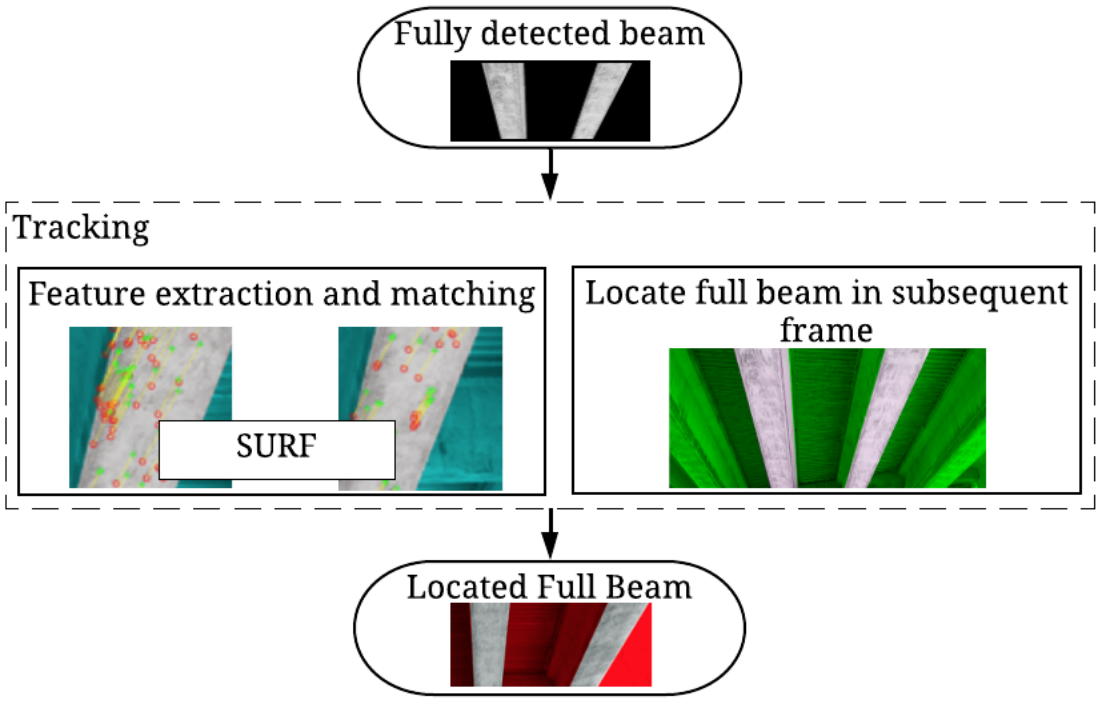

Figure 6.1. Proposed method of structural element tracking.

6.1. Feature Extraction and Matching

From this point on, the fully detected beam (ROI) segment(s) from the object detection algorithm (Section 5), is referred to as the original image or the masked image. The original image, was said to occur at time i ($frame_i$), and the subsequent image was at some time $i + 1$ ($frame_{i+1}$). The Speeded-Up Robust Features (SURF) detector and descriptor were used in the proposed tracking algorithm. The SURF is a scale-invariant and rotation-invariant feature detector and descriptor. Since these images were collected by a UAS, the feature extraction of the masked image and the subsequent image needed to allow for a change in scale and rotation, and thus, the SURF algorithm was highly applicable to this specific problem. The details of this algorithm were presented in Section 3.1. To detect features in each of these images, the (SURF) detector was used to locate SURF features, or blobs, and provides the location of the features, called SURF points. The metric threshold was specified to 500 limit the strong feature threshold and control the number of blobs considered important in each image. Other hyperparameters set were the number of octaves, which was set to three, and the number of scale levels per octave, which was set to four. Setting the number of octaves to three (maximum number was four) was due to the large image sizes used in this research that required larger filters to be used to find blob features. The number of scale levels per octave was set to four in order to retrieve enough features from the image but not use too fine of a filter where undesired features were retrieved. The hyper-parameter values were determined by hyperparameter-tuning.

The SURF points were used in feature extraction to derive the descriptors. The descriptor extraction method was the SURF descriptor. The feature vectors were then used in feature matching. Once key points and features were extracted from each image, features from the original image were matched with features in the subsequent image. The feature matching algorithm was specified to be unique, meaning that two features from the original image could not match one feature in the subsequent image and vice versa. Also, the max ratio was specified to 0.6 in order to ignore ambiguous matches. The feature matching output was the matched feature locations of the original image and the subsequent image. These are considered valid points, and in this work are called the key points. Figure 6.2 shows the masked image key points ($frame_i$) matched to key points in the subsequent image ($frame_{i+1}$). The yellow line represents the transform for one image pair.

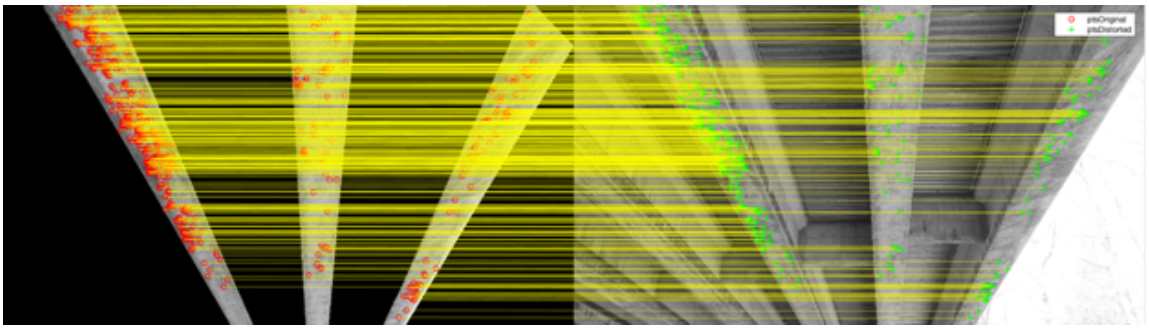

Figure 6.2. Masked image key points (Left); subsequent image key points (Right). The matched features are shown by the yellow lines.

6.2. Locate Full Beam in Subsequent Frame

Once unique point pairs were identified, the geometric transformation matrix describing the transformation from one image to the next was calculated. Due to the uncertainty of the movement of the UAS, the transformation matrix allowed for any possible transformation. A projective transform was chosen to allow flexibility for the image plane to tilt and register the image's possible misalignment. The projective transform uses an M-estimator Sample Consensus (MSAC) algorithm to map the inlier points from the masked image to the inlier points of the subsequent image. Recall, inlier points are point pairs that are within the collection of points that create the transformation matrix. An iterative process was used to find the inliers in the MSAC algorithm. In this work, a maximum of 1000 iterations was performed for a single image pair. Additionally, the maximum distance between the location of the original key point and the matched subsequent point was specified as 1.5 pixels. This proved to ensure robustness of the algorithm (only very well-matched points are chosen). The original image was then transformed and projected onto the subsequent image to locate the beam flange ROI in the new image. Figure 6.3 shows the beam flange in the original image (in gray) transformed to track the location of the beam flanges in the new image (red).

Figure 6.3. The original image containing only the beam flange (gray) projected onto the subsequent image (red).

7. IMPLEMENTATION AND RESULTS

7.1. Implementation

The complete unmanned aerial system (UAS) implemented to collect the data used in this research was made up of an unmanned aerial vehicle (UAV) and a removable high-resolution camera (sensor). The UAV was a DJI Inspire 2, augmented with the Zenmuse X5s sensor (Figure 7.1).

Figure 7.1. DJI Inspirte 2 with Zenmuse X5s sensor.

The method presented in this paper was implemented in MATLAB 2017b and MATLAB 2018a. Training of the R-CNN and the cross-validation operations which will be discussed in this section was conducted with the advanced computing resources provided by the Texas A&M High-Performance Research Computing (HPRC) center. The training time for the region-based convolutional neural network (R-CNN) on the 148 image training set was approximately 40 hours, and for the 12,094 images, the training time was approximately 139 hours. Although the R-CNN training time is high, this process was only implemented to create the R-CNN detector used in the object detection process. Once an adequate R-CNN was created, the process was not repeated

unless continued training was needed. The R-CNN can be re-trained at any time to account for newly available data with very little effort and this will only improve the overall algorithm. This can be done offsite at any time, and the trained algorithm can then be implemented in the field with a lower computational expense. For other portions of the research, a MacBook Pro with 2.5 GHz Intel Core i7 and 16 GB 1600 MHz DDR3 memory was used to develop and evaluate the algorithms in this methodology. Various MATLAB Toolboxes were employed as necessary in order to implement the novel detection and tracking methods, including the following: Image Processing Toolbox™, Parallel Computing Toolbox™, Statistics and Machine Learning Toolbox™, Neural Network Toolbox™, and Computer Vision System Toolbox™.

7.2. Results: R-CNN

7.2.1. Cross-validation of R-CNN

Cross-validation was used to successfully describe the predictive capability of neural networks (Refaeilzadeh et al., 2009). The robustness of the R-CNN, with the provided training set and experimentally concluded hyperparameter values, was estimated using a k-fold cross-validation method. In the k-fold cross-validation, the entire training set was randomly partitioned into k-equal subsections, or folds, that were mutually exclusive. These k-folds were used alternatively for new training and testing sets. The choice of the number of folds was determined to ensure testing set was large enough (decreasing the number of folds) to allow for fine-grained measurements of the performance. However, the number of folds needed to be large enough to adequately measure the generalization of the full data. For this paper, a 5-fold cross-validation

procedure was implemented. The choice of a 5-fold cross-validation was concluded from the results and discussions of Markatou et al. (2005), Kohavi (1995), and Refaeilzadeh et al. (2009). Markatou et al. (2005) claimed the variance estimator between 4-fold, 5-fold, and 10-fold cross-validation was not noticeably different.

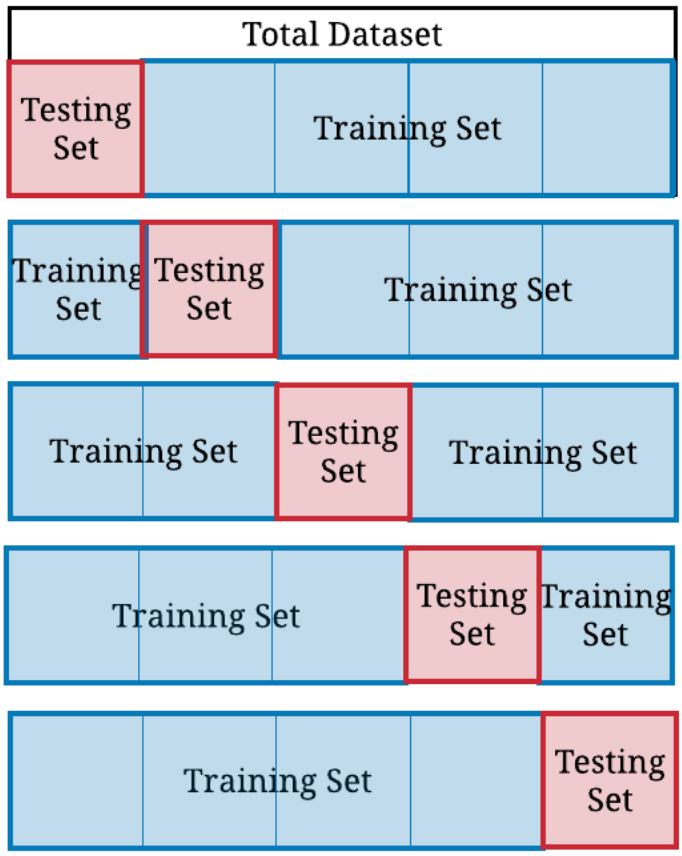

Figure 7.2. One instance of 5 fold cross validation procedure.

The R-CNN was trained five times using four folds to train the network and the fifth fold to test the network at each iteration. Figure 7.2 details one instance of the five training sets created for the 5-fold cross-validation procedure. For all five portions, the

misidentification error rate was calculated to describe the out-of-sample object detection prediction accuracy of the network. The detailed values for each iteration in the 5-fold cross-validation process are displayed in Table 7.1. Another predictive measurement was conducted by each of the five trained R-CNN's being tested with the entire dataset. This represents the true error rate or the classifier's error rate [misclassification/total classification]. This was done for two reasons: (1) The entire training set provides a better representation of the whole population; and (2) Since the same test set was tested on each R-CNN, sample variation was eliminated (Zhang et al., 2009). The detailed results from this process are demonstrated in Table 7.2. The classification error rate, e (Eq. 7.1), was calculated as the number of misclassified instances over the total number of classified instances and was used to measure the predictive capability of the R-CNN in this work.

$$e = \frac{FP + FN}{TP + FP + FN + TN} \qquad \text{(Eq. 7.1)}$$

where FP is false positive, FN is false negative, TP is true positive, and TN is true negative.

Table 7.1. Classification error rate on partitioned testing set.

Fold	1	2	3	4	5
Error	29%	22%	24%	25%	24%

Table 7.2. Classification error rate on entire dataset.

Fold	1	2	3	4	5
Error	16%	15%	17%	19%	21%

7.2.2. R-CNN Evaluation

Object detection algorithms were measured by the algorithms' ability to correctly locate objects in an image. Various commonly used performance metrics in this field include precision, recall, and accuracy, of which the first and last were used to evaluate the performance of the various algorithms developed in this work. The precision performance metric (Eq. 7.2) is defined as the fraction of relevant classifications (correctly classified regions or pixels) over all classified instances, and accuracy (Eq. 7.3) is a measure of all correctly classified regions (positively or negatively) over all the potentially classified regions. The performance of the object detection algorithm was evaluated using these metrics at the pROI stage (after the R-CNN implementation) and at the final ROI stage (after all machine vision operations).

$$precision = \frac{TP}{TP + FP} \qquad \text{(Eq. 7.2)}$$

where FP is false positive and TP is true positive.

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \qquad \text{(Eq. 7.3)}$$

where FP is false positive, FN is false negative,

TP is true positive, and TN is true negative.

The R-CNN accuracy evaluated in this section was the accuracy of the R-CNN trained with augmented data. Figure 7.3 shows an example of an original image with

manually covered *beam flange* regions. These regions represent the ground-truth (pixels which are actually *beam flange* pixels). The flange surface was masked in red to indicate all accepted *beam flange* locations (ground-truth values). Figure 7.4, Figure 7.5, Figure 7.6, and Figure 7.7 shows the R-CNN output image with annotation of the *beam flange* confidence score of each proposed region or bounding box. The 30 images set aside as the testing set are used here to evaluate the performance of this algorithm.

Figure 7.3. Example of manually masked beam flanges.

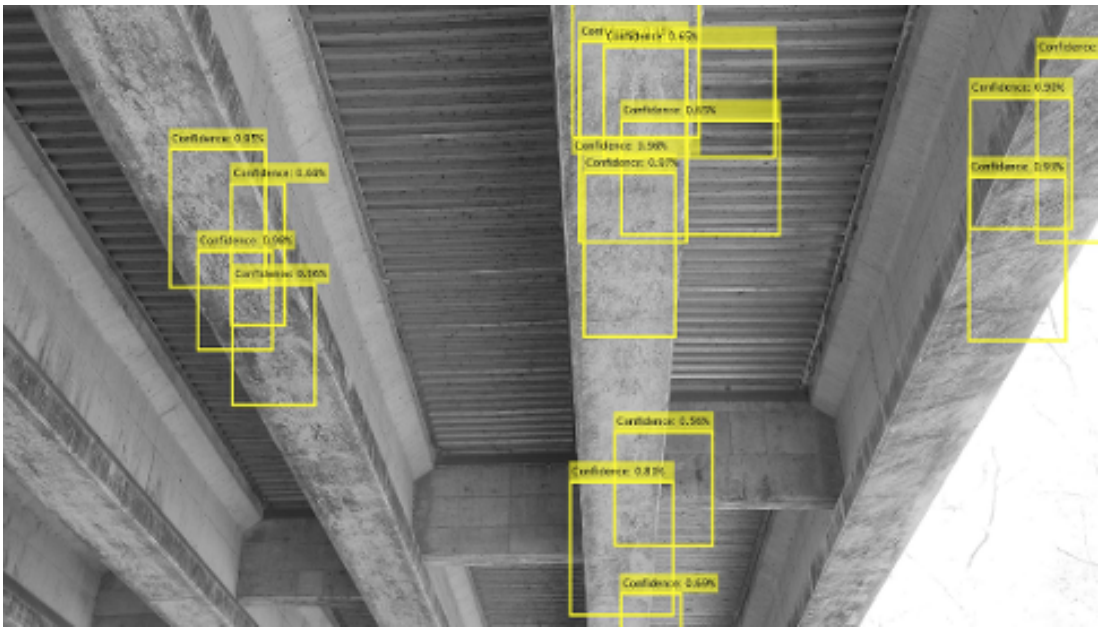

Figure 7.4. R CNN example output images with annotated pROIs: Example 1

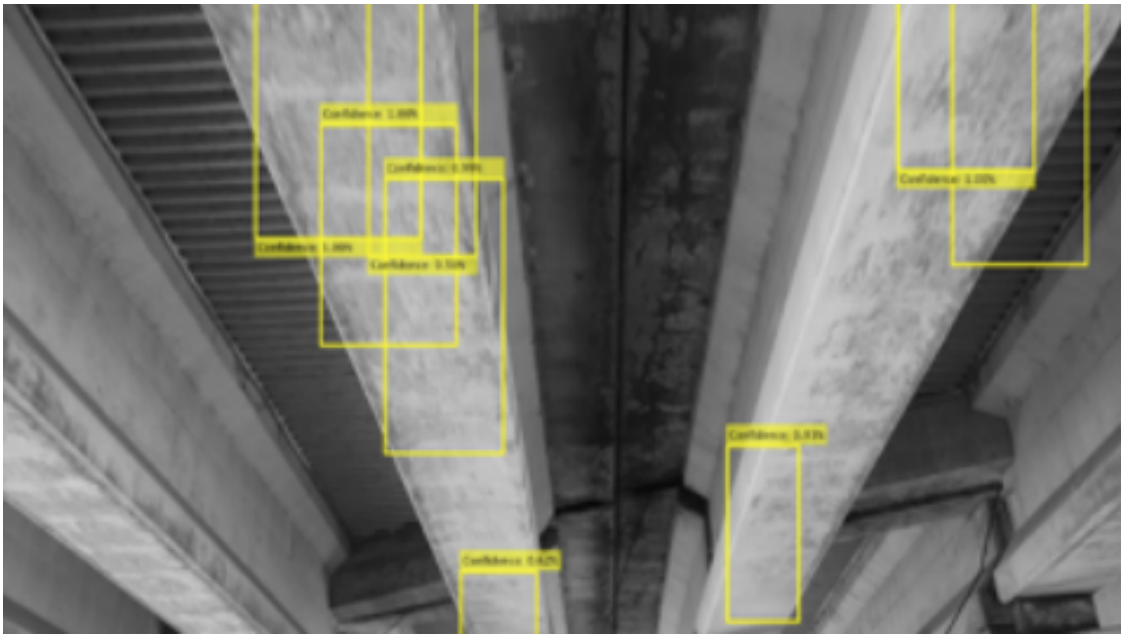

Figure 7.5. R CNN example output images with annotated pROIs: Example 2.

Figure 7.6. R CNN example output images with annotated pROIs: Example 3.

Figure 7.7. R CNN example output images with annotated pROIs: Example 4.

The midpoints and corner points of each proposed region (Figure 7.4) from the R-CNN algorithm are extracted and compared with the manually masked region from the ground truth image. If one of these points (corner or midpoints) is located within the manually masked region, the beam is considered to be detected by the R-CNN. Each bounding box with the label *beam flange* is evaluated in this way. If the bounding box correctly identifies the *beam flange*, it is considered a true positive (TP). If the bounding box does not identify the *beam flange*, then it is considered a false positive (FP). Each *beam flange* may only be identified once, meaning if there is more than one bounding box identifying a single *beam flange*, this only concludes to one TP, however, each box that incorrectly identifies beams is counted. The false negatives (FN) are the instances of actual beams that are not identified by the R-CNN. True negatives (TN) are the instances when the R-CNN correctly identifies other objects that are not *beam flange*. The true positive (TP), false positive (FP), false negative (FN) and true negative (TN) results from the R-CNN detector are detailed in a confusion matrix (Table 7.3). The detailed results for all 30 images can be found in Appendix A-1.

Table 7.3. Confusion matrix for TP, FP, FN, and TN performance of R CNN.

		Actual	
		Beam Flange	Not Beam Flange
Predicted	Beam Flange	35	10
	Not Beam Flange	93	251

Figure 7.8, Figure 7.9, Figure 7.10, and Figure 7.11 show the midpoints or corner point of the bounding box to verify that the proposed region is in fact detecting *beam flange*. The performance is quantified by the precision metric (Eq. 7.2). The precision of the R-CNN detector for the 30 images is calculated as 77.8%.

Figure 7.8. R CNN detection of beam flange: Example 1.

Figure 7.9. R CNN detection of beam flange: Example 2.

Figure 7.10. R CNN detection of beam flange: Example 3.

Figure 7.11. R CNN detection of beam flange: Example 4.

7.2.3. Analysis of R-CNN Results via Alternate Approaches

This R-CNN training method was originally used on the training set without any augmented data. The total number of training images in the original dataset was 148 and the augmented dataset is 12,094. This R-CNN trained with the augmented data is called "R-CNN1" in Table 7.4 and the R-CNN trained on the original training set is called "R-CNN2". The hyperparameter values were determined by previously trained R-CNN's and typical practices mentioned in Section 3.4. One of the most notable previously trained R-CNN's was trained with the same hyperparameters as the current practice except a mini-batch size of 64 was used, and the training set contained only 91 images. This R-CNN is called "R-CNN3" in Table 7.4. An outline of the training options and

details regarding the training set for each of these three R-CNN's are detailed in Table 7.4.

Table 7.4. R-CNN training options and training set details.

	R-CNN 1	R-CNN 2	R-CNN 3
Training Set	12094	148	91
Epoch	20	20	20
Mini-batch	128	128	64
Momentum	0.9	0.9	0.9
Learning rate	0.001	0.001	0.001
Positive IoU	[0.5, 1]	[0.5, 1]	[0.5, 1]
Negative IoU	[0, 0.3]	[0, 0.3]	[0, 0.3]

The two R-CNNs trained without augmented data, detailed in Table 7.4, varied according to the number of training images utilized and the mini-batch size. Recall that the R-CNN with a mini-batch size of 128 and training set size of 148 will be called R-CNN2, and the R-CNN with the mini-batch size of 64 and 91 training images will be called R-CNN3. The details of the true positive (TP), false positive (FP), false negative (FN), and true positive (TP) values are detailed in the form of confusion matrices (Table 5 (R-CNN1), Table 7.5 (R-CNN2), Table 7.6 (R-CNN3)). The precision of R-CNN2 is calculated as 75.0%. The confusion matrix is seen in Table 7.5 and the detail of the testing image output of TP, FP, FN, and TN is in Appendix A-3. The precision of R-CNN3 is 69.2%. The confusion matrix is seen in Table 7.6, and the detail of the testing image output of TP, FP, FN, and TN is in Appendix A-4.

Table 7.5. Confusion matrix of R CNN2.

		Actual	
		Beam Flange	Not Beam Flange
Predicted	Beam Flange	39	13
	Not Beam Flange	89	339

Table 7.6. Confusion matrix of R CNN3.

		Actual	
		Beam Flange	Not Beam Flange
Predicted	Beam Flange	45	20
	Not Beam Flange	69	72

Adding 57 images and increasing the mini-batch size from 64 to 128 increased the precision of the R-CNN detector by 5.8%. Adding more images to the training set increases the amount of relevant data for the network to learn. Larger mini-batch sizes result in estimates with a more accurate error gradient, but slow convergence, resulting in a longer training time. Since a larger size is more accurate, the mini-batch size of 128 will be used for the final R-CNN trained with augmented data.

7.2.4. Discussion of the Impact of Data Augmentation on Accuracy

The 148 training images and training options used to train R-CNN2 were applied to the R-CNN trained with data augmentation. The precision of the R-CNN with data

augmentation, called R-CNN3, is 77.8%. The increase of precision by 2.8%. Data augmentation is applied to a training set to increase the size of the training set and reduce overfitting. Increasing the size of the training set reaffirms the relevant data. The decrease in the number of incorrectly identified beam flanges is a result of applying data augmentation. The augmented dataset taught the network to rely less on high-frequency features, this can be seen by R-CNN2's false positives were mainly rip-rap and columns. The assumption is these were chosen due to the similarity in obvious features, such as color. The mainly chosen false positives in R-CNN1 was the concrete bar between beam flanges. Example of false positives can be found in Appendix A-5.

7.3. Results: Full Beam (ROI)

7.3.1. Full Beam Detection Evaluation

Results of the validation of the full beam detection algorithm are demonstrated in Figure 7.12, Figure 7.13, Figure 7.14, and Figure 7.15. These images show the true positive (pink), false positive (green), false negative (purple) and true negative (none) regions. In this work, true positive (TP) regions are the locations where the manually drawn mask pixels (Figure 7.3) are correctly identified by the masked final image given by the full beam detection algorithm. The false positive (FP) region is the region of pixels that was identified by the full beam detection algorithm but was not manually masked, and therefore, the algorithm incorrectly detected pixels as beam flange. Regions containing actual beam flange that the algorithm failed to identify are the false negative (FN) regions. True negative (TN) regions are the pixels not identified as full beam by the full beam detection algorithm and also were not manually masked. The performance of

the proposed method in full beam detection is validated by evaluating all 178 manually labeled regions (using the testing and training sets) to provide a performance measure that was independent of the performance of the R-CNN object detector developed in this work. The precision of full beam detection based on pixel area is calculated as 82.3%, and the accuracy is calculated as 94.1%.

Figure 7.12. Full beam detection of beam flange: Example 1.

Figure 7.13. Full beam detection of beam flange: Example 2.

Figure 7.14. Full beam detection of beam flange: Example 3.

Figure 7.15. Full beam detection of beam flange: Example 4.

7.3.2. Analysis of Full Beam Detection Results

As mentioned in Section 5, identifying the joint with the R-CNN and implementing an approximate location of the edge of the beam resulted in an accuracy of 94.1%, and without implementing this step, the accuracy was 88.4%. Applying the approximate location of the end of beam helped maintain the accurate location of the beam flange. In Figure 7.16, the FP (green) identification should actually be classified as a TP (pink) result. In Figure 7.16, one of the beams is not identified, but it should be noted that full beam detection is one of many steps to ensure the success of full identification of all the beam flanges in a span. The tracking algorithm has been employed to increase the success in the detecting all beam flanges.

Figure 7.16. Full beam detection of beam flange.

7.4. Results: Tracking

7.4.1. Tracking Evaluation

The 30 test set images and their succeeding images are used to evaluate the success of the tracking algorithm. Figure 7.17 and Figure 7.18 show the number of key points extracted from the original images and the subsequent images, the number of matched points, and the number of inlier points. For the original image ($frame_i$), the number of key points is extracted only from the identified full beam regions in the original image, resulting in an average of 2,185 key points per image (Figure 7.17). The entire subsequent image ($frame_{i+1}$) undergoes feature extraction resulting in an average of 4,600 key points (Figure 7.18). The succeeding image averages more than double the original image's key points because the entire succeeding image is used in feature extraction while the original image provides only the identified beam flanges for

feature extraction. The original image key points are compared to the subsequent image key points and an average of 390 matched points is identified per image (Figure 7.18). From the matched points, an average of 300 inlier points are identified per image pair (Figure 7.18). A detailed breakdown of the results for all 30 images can be found in Appendix A-2.

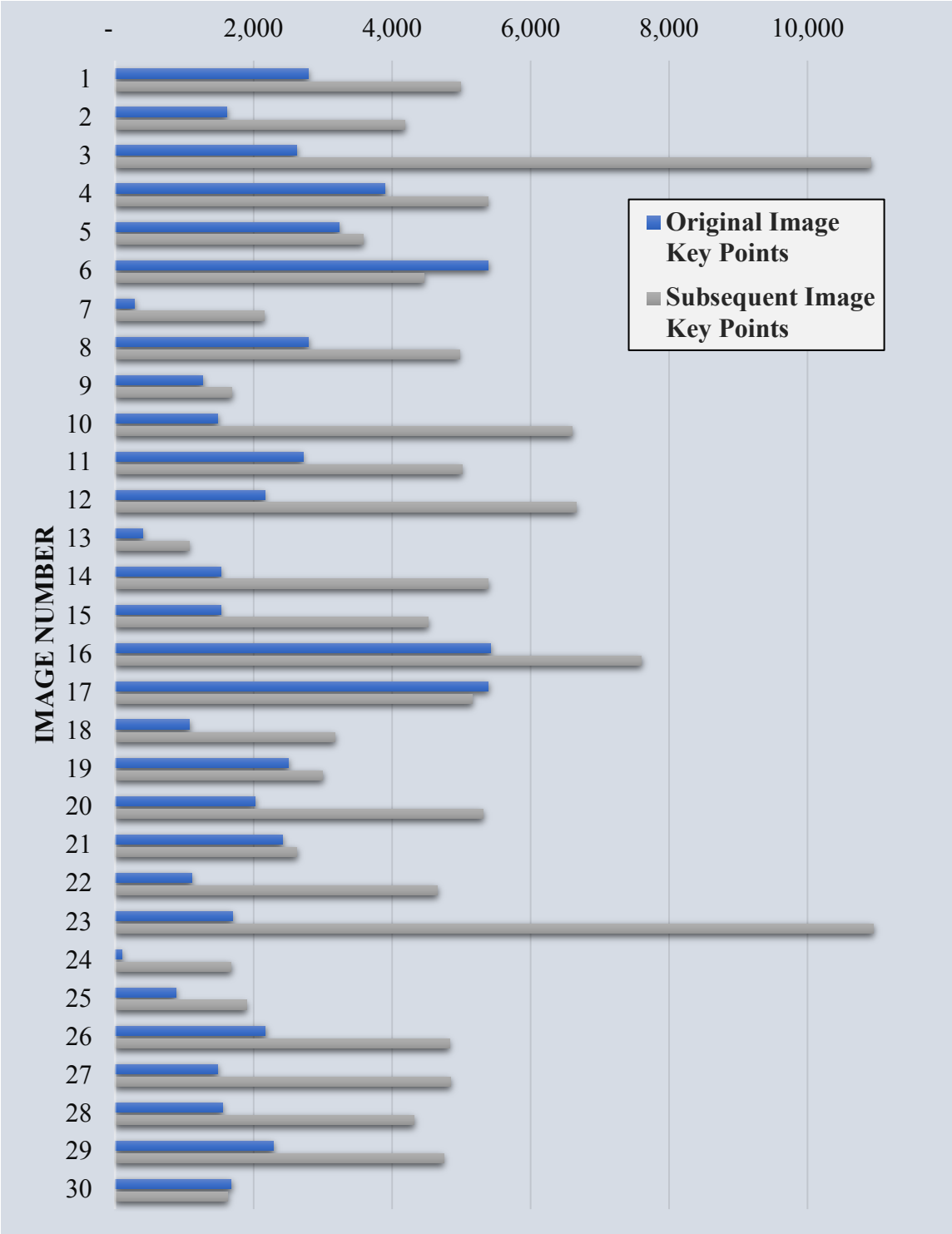

Figure 7.17. Detailed results of original and subsequent image key point comparison.

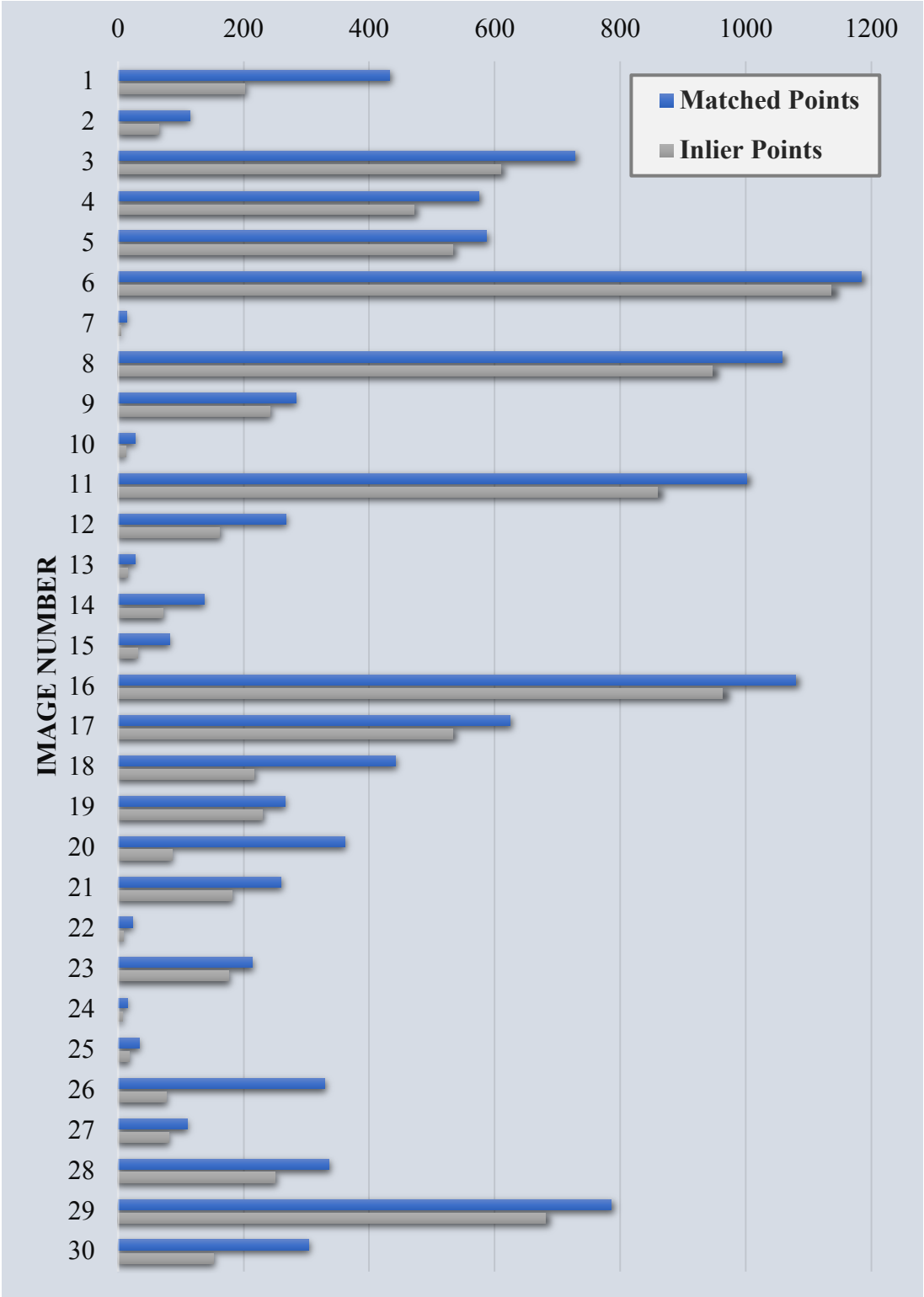

Figure 7.18. Matched and inlier point pairs for the original and subsequent images.

Figure 7.19 and Figure 7.20 show the original image's full beam flange identification (gray) projected onto the subsequent image (cyan) and the original image's key points (red circles) mapped (yellow line) to the subsequent images key points (green exes).

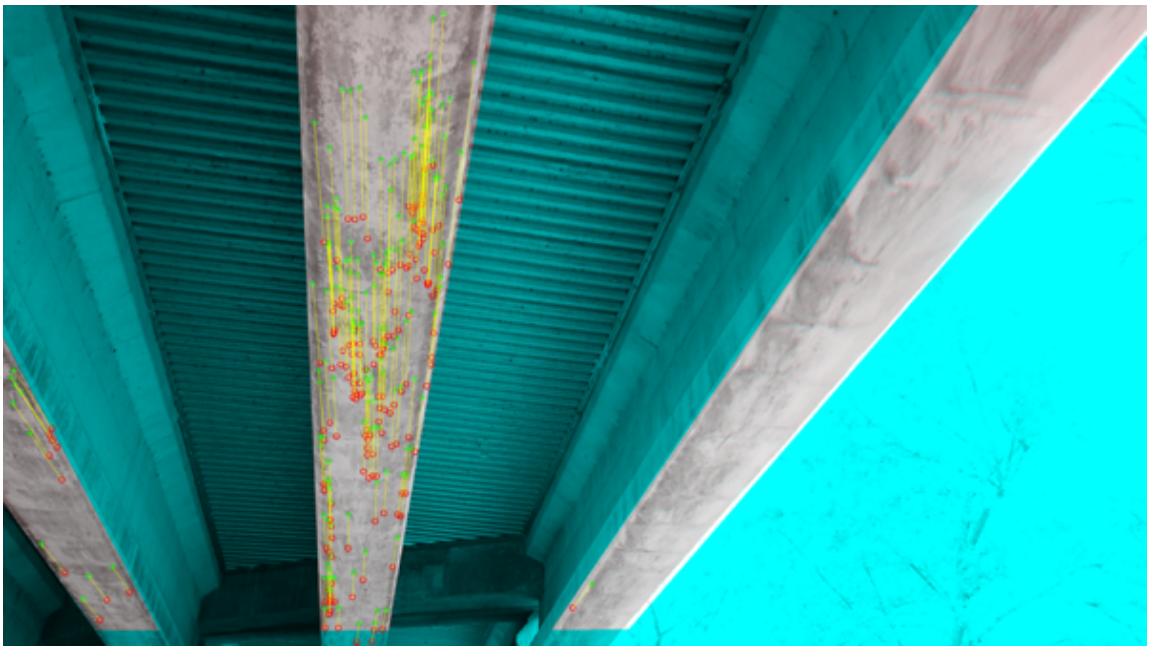

Figure 7.19. Original image key points (red circles) mapped (yellow line) to the subsequent image's key points (green exes): Example 1.

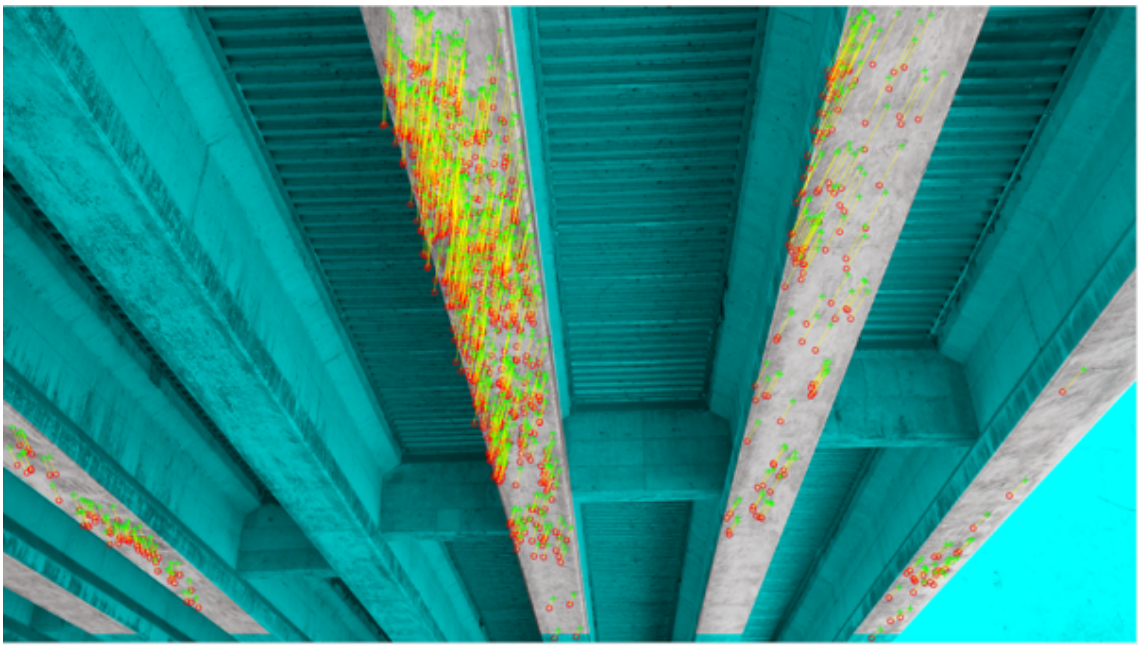

Figure 7.20. Original image key points (red circles) mapped (yellow line) to the subsequent image's key points (green exes): Example 2.

7.4.2. Analysis of Tracking Results

As previously mentioned in Section 6, inlier points are used to determine the adequacy of the tracking algorithm. In order to estimate the projected transform, a minimum of four inlier points must be identified, and from the 30 image pairs evaluated, the average number of inlier points substantially surpasses this minimum value. However, one image pair resulted in the minimum four inlier points. Images providing ten or fewer inlier points appeared to be the result of the original image providing only one beam flange for feature extraction. Therefore, one beam flange being identified does not provide a large number of inlier points but does provide a sufficient amount of points to create the necessary transformation matrix to perform the tracking method proposed in this research.

8. CONCLUSIONS AND FUTURE WORK

8.1. Summary

The current bridge inspection procedure requires a bridge inspection team to manually examine and determine the bridges condition. The current method is a subjective form of measurement, a time-consuming procedure, and exposes the inspection team to risks. The bridge inspection team must be qualified and due to the extreme level of training and expertise necessary to perform these inspection procedures in conjunction with their time-consuming nature, alternate methods of inspection procedures should be considered which are accurate, reliable, and more hands-off. As of 2016, 9.1% of national bridges are structurally deficient and require $123 billion dollars to solve this issue (ASCE, 2017). As of 2017, over 173 million trips are made on structurally deficient bridges daily (NBI, 2016). Routine inspections are conducted to periodically examine bridges to report the state of the bridge and document the current state of the bridge. Early detection of damages can help reduce the percentage of structurally deficient bridges.

Implementing an automated unmanned bridge inspection method will lessen and potentially eradicate the drawbacks of the current bridge inspection procedure. An unmanned method allows bridge inspections to be performed more frequently which increases the probability of early damage detection. Early detection reduces maintenance cost and avoids severe damage from delayed repairs which could lower the investment

cost of eradicating structurally deficient bridges and create opportunities to avoid bridges becoming structurally deficient.

The progress made in improving the current bridge inspection procedure typically involves implementing technologically advanced devices to acquire data. A common device for data acquisition is an unmanned aerial system (UAS), which is comprised of an unmanned aerial vehicle and a high-resolution camera. This system provides valuable qualities for easily and efficiently retrieving bridge data for analyzing the condition of the structural element. In this study, a UAS is employed to retrieve bridge span imagery for various steps within the proposed method. The imagery collected with the UAS was used to train a region-based convolutional neural network and evaluate the performance of the proposed novel method.

In this research, the proposed method accomplishes detection and tracking of a structural element, specifically a concrete beam, in high-resolution imagery. The proposed method was created by combining machine vision and machine learning algorithms, such as Hough transform, Canny edge detection, a region-based convolutional neural network (R-CNN), and the Speeded-Up Robust Features (SURF) extraction algorithm to detect and track concrete beams in an image.

The acquired bridge data from the UAS was used in evaluating he performance of the trained R-CNN and the performance of the developed full beam detection algorithm. The evaluation proved successful in identifying beam flanges in an image. However, tracking the detected beam flanges throughout the data aids in locating all bridge beams.

8.2. Future Work

8.2.1. Limitations

The first limitation identified was the limited exposure. The proposed procedure of beam detection and tracking was developed using three prestressed I-beam bridges located in the same county. Although the bridges significantly varied in age, exposure, and other characteristics, the region-based convolutional neural network has only been exposed to the key features of three bridges.

The second limitation identified was the viewpoint restriction. The proposed procedure identifies can identify beams from only under the bridge span. This research did not expand into beam identification from other viewpoints of the bridge.

The third limitation identified was the requirement for a beam flange to be present in the image for the algorithm to work. The proposed method uses the beam flange as the region of interest and not the web, or side, of the beam, and this algorithm has not developed a method for locating the web of the beam.

8.2.2. Recommendations

To answer the limited exposure, this research recommends expanding the dataset. To increase the robustness of the region-based convolutional neural network (R-CNN) used in this research, the R-CNN should continue training using concrete beam bridge imagery and research developments in hyperparameter tuning. Images of other types materials, structural components, bridges, etc. should not be added or should be added with caution to the dataset for continued training. The complexity of the model has the potential to decrease the performance of the object detector. However, the R-CNN is

classifying the beam flanges based on the output score of the beam flange category compared to the other categories. Therefore, increasing the number of labels of an existing category, adding a category if the object is constantly present, and/or splitting a category into two categories could adequately expose the R-CNN and increase the performance. Specifically for this research's R-CNN, the category *concrete not beam* contained different structural elements of concrete, such as columns, rip-rap, bridge deck, etc., that appeared frequently within the image and therefore, partitioning *concrete not beam* into these identified elements could increase the performance of the R-CNN.

The viewpoint restriction limitation was integrated into the full beam detection algorithm. Images captured under the bridge span contain multiple beams while the side of the bridge captures one beam. Assuming that identifying the full beam from the side would only find one beam an did appear applicable to interior beams or a beam wouldn't the other viewpoint containing beams was a skewed angle. The skewed angle produces beams with complex shapes relative to the side of the beam or the beam flange. Also, applying future damage detection algorithms and measuring the damage on the extracted beam at that angle could produce incorrect data or require complex calculations. Therefore, disregarding the findings at this angle and measuring the damage at a better angle should be considered.

The proposed method was developed with the idea that one of a beam's key characteristics in an image is the beam flange appears as a simple shape with long straight edges. This assumption proved useful and effective in identifying the beam while remaining computationally inexpensive. If the side of the beam were to be

detected, depending on the shape of the concrete beam viewed in the image, locating the limits of the beam element become more difficult. For instance, an I-beam when viewed from the side in an image appears to have six lines, unless the depth of the web was tall and then another line due to the seam, or the shape of the I-beam isn't adequately captured due to illumination and lines are lost. However, locating the beam web is necessary for a complete unmanned automated operation. Once the beam flange is located, the assumption of a beam web on either side of the edges and utilizing the tracking algorithm could assist in locating the sides of the beam.

8.2.3. Practical Implementations

The proposed method developed a tool to identify beams in an image or video and track the detected beam throughout the imagery. This research focuses on the importance of this tool in order to develop an unmanned automated bridge inspection procedure.

Utilizing the developed object detection and tracking process allows for a targeted approach to implementing a damage detection algorithm specific to the detected bridge element. Damage detection algorithms analyzing an image of a bridge are exposed to noise and are susceptible to false identification of damage or the damage detection algorithm may correctly identifying damage on an background item not belonging to the bridge.

The proposed tool is also a vital step in an unmanned automated bridge inspections because, although the damage detection algorithm successfully identifies damage to a bridge element, the functional classification computation requires knowing

the element type in order to determine the conditional rating of the superstructure. The *Bridge Inspector's Reference Manual* dictates how each element should be inspected, and therefore, the identification of structural elements is critical in order to implement future comprehensive structural component condition evaluation algorithms.

A required step in the bridge inspection process requires adequate documentation of the current state of the bridge. The documentation required includes a written report of the findings, notes on significant discoveries that either need to be monitored or immediately addressed, and ample photos depicting the condition of the bridge for: (1) use at future inspections to determine the rate of deterioration and identifying significant changes to bridge structure; and, (2) to collect information on the bridge's performance and submit the findings to the national bridge inventory and other databases for continued research on the performance of bridges. Employing a UAS allows a more flexible, accurate, and safer alternative to the current state of practice.

The proposed method's tracking component has the potential aid the bridge inspectors to know exact locations of detected damage. Visual inspection of extensive damage may be required to determine the severity of the damage before completion of the bridge inspection. Also, knowing the exact location of the damage will contribute to an efficient process of planning the necessary set-up and repair of the damage.

8.3. Lessons Learned

For this research, many region-based convolutional neural networks (R-CNNs) were trained in order to achieve a robust and accurate model. From these attempts, there

were a few lessons learned that increased the accuracy not previously stated within this Thesis.

The first lesson learned was techniques for a successful ground-truth labeling method. The ground-truth labeling method that achieved the greatest accuracy contained ground-truth labels that varied in size and shape, and the ground-truth labels for one classification overlapped each other. Placing labels of various sizes and shapes on top of each other generated the most positive and negative samples and allowed for greedy intersection-over-union thresholds. This was due to more candidates being correctly identified as positive and negative samples and due to the increase in data, the accuracy increased. Also, adding the category label *joint* was concentered later to increase the accuracy of the R-CNN. The first few R-CNN's trained labeled the joint as *beam flange* and adding in the category *joint* eliminated this false positive. Recommendations for continuation of this research within labeling would be to remove the category *not concrete* because it's essentially background and further categorize *concrete not beam.*

The second lesson learned was choosing a CNN. The first few R-CNNs trained, the CNN used was trained on a significantly smaller dataset and contained less categories. This CNN also had a smaller architecture and therefore less weights and biases to update. Fine-tuning the CNN was less computationally expensive, however, the lack of accuracy signified the need for a more robust CNN.

The third lesson learned involved hyper-parameter tuning. Conducting cross-validation to assist with hyper-parameter tuning was extremely computationally expensive when performed on large datasets. For this work, the first few attempts to

hyper-parameter tune were performed on the dataset that was not augmented in order to decrease computational time and quickly identify effects of adjusting particular hyper-parameters. Also, when identifying the effects of adjusting hyper-parameter values, max epochs was set to 5 and the same training images were used in order to analyze the difference different adjustments provided while limiting computational expense. Once hyper-parameter tuning the last R-CNNs, the hyper-parameter values used were as specified in Section 7.2.

More specifically, certain hyper-parameter tuning strategies were developed within this thesis. Strategies involving the CNN and R-CNN, the mini-batch size and max epochs had the largest effects on the accuracy of the R-CNN. A mini-batch size of 128 paired with a max epoch of 10 performed better than a mini-batch size of 64 paired with a max epoch of 10. Increasing the max epochs from 10 to 20 also increased accuracy and computational expense. When adjusting the stochastic gradient descent with momentum hyper-parameter momentum from 0.5 to 0.9 slightly increased the accuracy but required a significant increase in training time. Again, once hyper-parameter tuning the last R-CNNs, the hyper-parameter values used were as specified in Section 7.2.

REFERENCES

Adams, M. S., Friedland, J. C., (2011). A survey of unmanned areal vehicle (UAV) usage for imagery collection in disaster research and management. 9th int. workshop on remote sensing for disaster response, Stanford, CA.

Adeli, H. & Jiang, X. (2009), Intelligent Infrastructure— Neural Networks, Wavelets, and Chaos Theory for Intel- ligent Transportation Systems and Smart Structures, CRC Press, Taylor & Francis, Boca Raton, FL.

Adeli, H., (2001). Neural Networks in Civil Engineering: 1989 – 2000, Computer-Aided Civil and Infrastructure Engineering. Vol. 16 pp. 126-142.

American Society of Civil Engineers (ASCE), (2013). 2013 Report Card for America's Infrastructure. Retrieved from https://www.sec.gov/Archives/edgar/data/1534155/000153415517000004/ex1031bridgesreport.pdf

American Society of Civil Engineers (ASCE), (2017). 2017 Infrastructure Report Card. Retrieved from http://www.infrastructurereportcard.org/cat-item/bridges/

ASCE/SEI-AASHTO Ad-Hoc Group, (2009). White Paper on Bridge Inspection and Rating. Journal of Bridge Engineers, 14(1), 1-5.

Awais, M. (2017). Lecture: Deep neural networks [PDF]. Retrieved from https://udrc.eng.ed.ac.uk/summer-school-slides-2017-and-lecture-notes

Awrangjeb, M. Effective Generation and Update of a Building Map Database Through Automatic Building Change Detection from LiDAR Point Cloud Data. Remote Sens. 2015, 7, 14119-14150.

Ballard, D. H. (1987). Generalizing the Hough transform to detect arbitrary shapes. In Readings in computer vision (pp. 714-725).

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-Up Robust Features (SURF). Computer vision and image understanding, 110(3), 346-359.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. In European conference on computer vision (pp. 404-417). Springer, Berlin, Heidelberg.

Beymer, D., McLauchlan, P., Coifman, B., & Malik, J. (1997, June). A real-time computer vision system for measuring traffic parameters. In Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on (pp. 495-501). IEEE.

Bishop, C. M. Pattern Recognition and Machine Learning. Springer, New York, NY, 2006.

Bridge Condition by Functional Classification Length 2017. (2017). U.S. Department of Transportation Federal Highway Administration (Bridges & Structures) Retrieved from https://www.fhwa.dot.gov/

Brilakis, I. and Soibelman, I., (2008). Shape-Based Retrieval of Construction site photographs, journal of computing in civil engineering, ASCE, Vol. 22, No. 1 pp. 14-22

Brilakis, I., German, S. and Zhu, Z. (2011). "Visual pattern recognition models for remote sensing of civil infrastructure", Journal of Computing in Civil Engineering, ASCE, Vol. 25, No. 5, pp. 388–393.

Brilakis, I., Soibelman, L., Shinagawa, Y., (2006). Construction site image retrieval based on material cluster recognition, Advanced Engineering Informatics, Vol. 20, No. 4. pp. 443-452.

Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6), 679-698.

Cha, Y.-J., Choi, W. & Bu¨yu¨ko¨ztu¨rk, O. (2017), Deep learning-based crack damage detection using convolutional neural networks, Computer-Aided Civil and Infrastructure Engineering, 32, 361–78.

Chan, B., Guan, H., Jo, J., & Blumenstein, M., (2015, August). Towards UAV-based Bridge Inspection Systems: A Review and an Application Perspective. Structural Monitoring and Maintenance, 2(3), 283-300.

Choset, H., (2000), Bridge Inspection with Serpentine Robots, Highway IDA Project 56, Washington D.C., USA.

Chou, T.-Y., Yeh, M.-L., Chen, Y.-C., Chen, Y.-H., (2010). Unmanned Areial Vehicle Data Axquisition for Damage Assesment in Hurricane Events. Technical Commission VII Symposium 2010.

Coifman, B., Beymer, D., McLauchlan, P., & Malik, J. (1998). A real-time computer vision system for vehicle tracking and traffic surveillance. Transportation Research Part C: Emerging Technologies, 6(4), 271-288.

Cox, R., Hohmann, D., Eskridge, A., Hyzak, M., Freeby, G., Wolf, L., Merrill, B., & Holt, J., (2007, Spring). Concrete Bridges in Texas. ASPIRE The Concrete Bridge Magazine, 43-45. Illinois: Precast/Pre-stressed Concrete Institute.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). Ieee.

Deriche, R. (1987). Using Canny's criteria to derive a recursively implemented optimal edge detector. International journal of computer vision, 1(2), 167-187.

Deshpande, A. (2016, July 20). A Beginner's Guide to Understanding Convolutional Neural Networks. Retrieved from https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.

Everingham, M., Van Gool, L., Williams, C.K.I. et al. Int J Comput Vis (2010) 88: 303. https://doi.org/10.1007/s11263-009-0275-4

Ezequiel, C. A. F. et al., "UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development," 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, 2014, pp. 274-283.

Ezequiel, C. A. F., Cua, M., Libatique, N. C., Tangonan, G. L., Alampay, R., Labuguen, R. T., ... & Loreto, A. B. (2014, May). UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development. In Unmanned Aircraft Systems (ICUAS), 2014 International Conference on (pp. 274-283). IEEE.

Fang, W., Ding, L., Zhong, B., Love, P. E., & Luo, H. (2018). Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach. Advanced Engineering Informatics, 37, 139-149.

Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2013). Learning hierarchical features for scene labeling. IEEE transactions on pattern analysis and machine intelligence, 35(8), 1915-1929.

Farhey, Daniel, (2010). Performance of Bridge Materials by Structural Deficiency Analysis. Journal of Performance of Constructed Facilities, 24(4), 345-352. Virginia: American Society of Civil Engineers.

Fathi, H., Dai, F., Lourakis, M., (2015) Automated as-built 3D Reconstruction of Civil Infrastructure Using Computer vision: Achievements, opportunities, and challenges, Advanced Engineering Informatics, Vol. 29, No. 2, pp. 149-161.

Federal Highway Administration (FHWA). (1995). Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges (Report No. FHWA-PD-96-001). Washington DC: United States Department of Transportation.

Federal Highway Administration (FHWA). (2012). Bridge Inspection Manual (FHWA Publication No. NHI 12-050). Washington DC: United States Department of Transportation.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence, 32(9), 1627-1645.

Feng, M. Q., Kim, D. K., Yi, J. H., & Chen, Y. (2004). Baseline models for bridge performance monitoring. Journal of Engineering Mechanics, 130(5), 562-569.

Fernandes, L.A., Olivera, M.M., Real-time line detection through an improved hough transform voting scheme, pattern recognition 41(1) (2008) 299-314.

German, S., Brilakis, I., & DesRoches, R. (2012). Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments. Advanced Engineering Informatics, 26(4), 846-858.

German, S., Jeon, J. S., Zhu, Z., Bearman, C., Brilakis, I., DesRoches, R., & Lowes, L. (2013). Machine vision-enhanced postearthquake inspection. Journal of Computing in Civil Engineering, 27(6), 622-634.

Girshick, R., J. Donahue, T. Darrell, and J. Malik. (2014) "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation."Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014, pp. 580–587.

Golub, G. H., & Van Loan, C. F. (2012). Matrix computations (Vol. 3). JHU Press.

Guo, Y. et al. (2016). "Deep learning for visual understanding: A review." Neurocomputing, 187 (2016): 27-48.

Hallermann, N., Morgenthal, G., & Rodehorst, V. (2015). Vision-based monitoring of heritage monuments: Unmanned Aerial Systems (UAS) for detailed inspection and high-accuracy survey of structures. WIT Transactions on The Built Environment, 153, 621-632.

Hallermann, N., Morgenthal, G., Rodehorst, V., 'Vision-based monitoring of heritage monuments – Unmanned Aerial Systems (UAS) for detailed inspection and high-accurate survey of structures', Proceedings of STREMAH 2015, pp 621-632, 2015.

Han, B., Wang, Y., & Jia, X. (2010, August). Fast calculating feature point's main orientation in SURF algorithm. In Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on (Vol. 6, pp. 165-168). IEEE.

Harley, A. W. (2015, December). An interactive node-link visualization of convolutional neural networks. In International Symposium on Visual Computing (pp. 867-877). Springer, Cham.

Hartley, R. and Zisserman, A. (2004) Multiple View Geometry in Computer Vision. NY: Cambridge University Press.

Hawken, R., Nguyen, T., Ivanyi, J., (2017). Bridges: Condition Inspections Using Unmanned Aerial Vehicles: a Trial Project, presented at Bridges: Connecting Communities: Austroads Bridge Conference, 2-6 April 2017, Melbourne, Victoria.

Homeland infrastructure Foundation-Level Data (HIFLD). (2017). National Bridge Inventory (NBI) Bridges [Data file]. Retrieved from https://hifld-geoplatform.opendata.arcgis.com/datasets/national-bridge-inventory-nbi-bridges/data

Hoskere, V., Narazaki, Y., Hoang, T., & Spencer Jr, B. (2018). Vision-based Structural Inspection using Multiscale Deep Convolutional Neural Networks. arXiv preprint arXiv:1805.01055.

Howard, A.G. Some improvements on deep convolu- tional neural network based image classification. CoRR, abs/1312.5402, 2013.

Hsieh, C. J., Chang, K. W., Lin, C. J., Keerthi, S. S., & Sundararajan, S. (2008, July). A dual coordinate descent method for large-scale linear SVM. In Proceedings of the 25th international conference on Machine learning (pp. 408-415). ACM.

Jahanshahi, M., Masri, S., Padgett, C., Sukhatme, G., (2013). An Innovative Methodology for Detection and Quantification of Cracks Through Incoporation of Depth Perctption. Machine Vision and Applications, No. 24, Vol. 2 pp. 227-241.

Jia, J., & Tang, C. K. (2008). Image stitching using structure deformation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(4), 617-631.

Jin-Hwan Lee, Sung-Sik Yoon, In-Ho Kim, Hyung-Jo Jung, "Diagnosis of crack damage on structures based on image processing techniques and R-CNN using unmanned aerial vehicle (UAV)," Proc. SPIE 10598, Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2018, 1059811 (27 March 2018);

Kamijo, S., Matsushita, Y., Ikeuchi, K., & Sakauchi, M. (2000). Traffic monitoring and accident detection at intersections. IEEE transactions on Intelligent transportation systems, 1(2), 108-118.

Karpathy, A. (2018a). CS231n: Convolutional Neural Networks for Visual Recognition, Module 1: Neural Networks, Linear classification: Support Vector Machine, Softmax [website]. Retrieved from http://cs231n.github.io/

Karpathy, A. (2018b). CS231n: Convolutional Neural Networks for Visual Recognition, Module 1: Neural Networks, Optimization: Stochastic Gradient Descent [website]. Retrieved from http://cs231n.github.io/

Karpathy, A. (2018c). CS231n: Convolutional Neural Networks for Visual Recognition, Module 1: Neural Networks, Setting up the Data and the Loss [website]. Retrieved from http://cs231n.github.io/

Karpathy, A. (2018d). CS231n: Convolutional Neural Networks for Visual Recognition, Module 2: Convolutional Neural Networks, Convolutional Neural Networks: Architectures, Convolution / Pooling Layers [website]. Retrieved from http://cs231n.github.io/

Kim, H., Kim, H., Hong, Y. W., & Byun, H. (2017). Detecting Construction Equipment Using a Region-Based Fully Convolutional Network and Transfer Learning. Journal of Computing in Civil Engineering, 32(2), 04017082.

Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Ijcai(Vol. 14, No. 2, pp. 1137-1145).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." Advances in neural information processing systems. 2012.

Lee, B. J., Shin, D. H., Seo, J. W., Jung, J. D. Lee, J., Y. (2011), Intellegent Bridge Inpsection Using Remote Controlled Robot and Image Processing Technique, Proceedings of Internatinal Association for Autormation and Robotics in Construction, Seoul, Korea, June 29-July 2, 1426-31

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009, June). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th annual international conference on machine learning (pp. 609-616). ACM.

Lee, Jin-Hwan, Yoon, Sung-Sik, Kim, In-Ho, Jung, Hyung-Jo "Diagnosis of crack damage on structures based on image processing techniques and R-CNN using unmanned aerial vehicle (UAV)," Proc. SPIE 10598, Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2018, 1059811 (27 March 2018);

Lejeune, E., Luna, B., Josipovic, G., Rivera, J., & Whittaker, A., (2015). Automated Detection and Measurement of Cracks in Reinforced Concrete Components. ACI Structural Journal, 112(3), 397-406.

Lindeberg, T. (1998). Edge detection and ridge detection with automatic scale selection. International Journal of Computer Vision, 30(2), 117-156.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

Mader, D; Blaskow, R; Westfeld, P; H-G Maas. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, suppl. W3; Gottingen Vol. XL, Iss. 3, : 335-341. Gottingen: Copernicus GmbH. (2015)

Markatou, M., Tian, H., Biswas, S., and Hripcsak, G. (2005). Analysis of variance of cross-validation estimators of the generalization error., J. Mach. Learn. Res., 6:1127–1168 (electronic)

Mu, K., Hui, F., & Zhao, X. (2016). Multiple Vehicle Detection and Tracking in Highway Traffic Surveillance Video Based on SIFT Feature Matching. Journal of Information Processing Systems, 12(2).

Murphy, K. P. Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge, Massachusetts, 2012.

Naaman, A. E. (2012) Prestressed Concrete Construction. Ann Arbor, Michigan: Techno Press 3000.

Narazaki, Y., Hoskere, V., Hoang, T. A., & Spencer Jr, B. F. (2018). Automated Vision-based Bridge Component Extraction Using Multiscale Convolutional Neural Networks. arXiv preprint arXiv:1805.06042.

Narazaki, Y., Hoskere, V., Hoang, T. A., & Spencer, B. F. (2018). Vision-based automated bridge component recognition integrated with high-level scene understanding. arXiv preprint arXiv:1805.06041.

National Bridges. (2016). The National Bridge Inventory (NBI) Database -2016 [Data File]. Retrieved from http://nationalbridges.com/

National Performance Management Measures; Assessing Pavement Conditions for the National Highway Performance Program and Bridge Condition for National Highway Performance Program, 88 Fed. Reg. 11 (final rule Jan 18, 2017) (to be codified at 23 C.F.R. pts. 490)

Neubeck, A., & Van Gool, L. (2006, August). Efficient non-maximum suppression. In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on (Vol. 3, pp. 850-855). IEEE.

O'Byrne, M., Schoefs, F., Ghosh, B. & Pakrashi, V. (2013), Texture analysis based damage detection of ageing infrastructural elements, Computer-Aided Civil and Infra- structure Engineering, 28(3), 162–77.

Paal, S. G., Brilakis, I., & DesRoches, R. (2014). Automated measurement of concrete spalling through reinforcement detection.

Paal, S. G., Jeon, J. S., Brilakis, I., & DesRoches, R. (2014). Automated damage index estimation of reinforced concrete columns for post-earthquake evaluations. Journal of Structural Engineering, 141(9), 04014228.

Praveen, K. S., Hamarnath, G., Babu, K. P., Sreenivasulu, M., & Sudhakar, K. (2016). Implementation Of Image Sharpening And Smoothing Using Filters. International Journal of Scientific Engineering and Applied Science, 2(1), 7-14.

Precast Pre-stressed Concrete Institute and American Association of State Highway Transportation Officials (PCI/AASHTO), (2011). ASSHTO I-Beams. PCI Bridge Design Manual. Retrieved from http://www.pci.org/design_resources/transportation_engineering_resources/bridge_design/

Rashidi, A., Sigari, M. H., Maghiar, M., & Citrin, D. (2016). An analogy between various machine-learning techniques for detecting construction materials in digital images. KSCE Journal of Civil Engineering, 20(4), 1178-1188.

Refaeilzadeh P., Tang L., Liu H. (2009) Cross-Validation. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA

Rivera, J., Josipovic, G., Lejeune, E., Luna, B., & Whittaker, A., (2015). Automated Detection and Measurement of Cracks in Reinforced Concrete Components. ACI Structural Journal, 112(3), 397-406.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

Shapiro, L. G. & Stockman, G. C: "Computer Vision", page 137, 150. Prentice Hall, 2001

Simard, P., Bottou, L., Haffner, P., & LeCun, Y. (1999). Boxlets: a fast convolution algorithm for signal processing and neural networks. In Advances in Neural Information Processing Systems (pp. 571-577).

Simonyan, K. & Zisserman, A. (2014), Very deep convolu- tional networks for large-scale image recognition, in Pro- ceedings of the International Conference on Learning Rep- resentations (ICLR), San Diego, CA, 7–9 May 2015.

Son, H., Kim, C., & Kim, C. (2011). Automated color model–based concrete detection in construction-site images by using machine learning algorithms. Journal of Computing in Civil Engineering, 26(3), 421-433

Son, H., Kim, C., Hwang, N., Kim, C., & Kang, Y. (2014). Classification of major construction materials in construction environments using ensemble classifiers. Advanced Engineering Informatics, 28(1), 1-10.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

State Auditor's Office. (2009, December). An Audit Report on The Department of Transportation's Bridge Inspection Program (Report No. 10-017). Texas: John Keel.

Teizer, J. (2015). Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites. Advanced Engineering Informatics, 29(2), 225–238.

Torr, P. H., & Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. Computer vision and image understanding, 78(1), 138-156.

Torr, P., & Zisserman, A. (1998, January). Robust computation and parametrization of multiple view relations. In Computer Vision, 1998. Sixth International Conference on (pp. 727-732). IEEE.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International journal of computer vision, 104(2), 154-171.

United States Department of Transportation. (2015). Deficient Bridges by Functional Classification Count [Data File]. Retrieved from https://www.fhwa.dot.gov/bridge/nbi/no10/fccount15.cfm

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on (Vol. 1, pp. I-I). IEEE.

Wang, J. M., Chung, Y. C., Lin, S. C., Chang, S. L., Cherng, S., & Chen, S. W. (2004, August). Vision-based traffic measurement system. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on(Vol. 4, pp. 360-363). IEEE.

Wen, M.-C., & Kang, S.-C., (2014), Augmented Reality and Unmanned Aerial Vehicle Assist in Construction Management, 2014 International Conference on Computing in Civil and Building Engineering

Woo, H., Ji, Y., Kono, H., Tamura, Y., Kuroda, Y., Sugano, T., ... & Asama, H. (2016, October). Lane-changing feature extraction using multisensor integration. In Control, Automation and Systems (ICCAS), 2016 16th International Conference on (pp. 1633-1636). IEEE.

Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015). Deep image: Scaling up image recognition. arXiv preprint arXiv:1501.02876.

Xiao, L. (2010). Dual averaging methods for regularized stochastic learning and online optimization. Journal of Machine Learning Research, 11(Oct), 2543-2596.

Yamaguchi, T. and Hashimoto, S. (2009). “Fast crack detection method for large-size concrete surface images using percolation-based image processing”, Machine Vision and Applications, Vol. 11, No. 5, pp. 797–809.

Yang, J, Park, M-W, Vela, P A, & Golparvar-Fard, M (2015). Construction performance monitoring via still images, time-lapse photos, and video streams: Now, tomorrow, and the future. Advanced Engineering Informatics, 29(2), 211–224.

Yeum, C. M. & Dyke, S. J. (2015), Vision-Based Automated Crack Detection for Bridge Inspection, Computer-Aided Civil and Infrastuture Engineering. 30(10), 759-70.

Yuan, Y., Emmanuel, S., Fang, Y, Lin, W. “Visual Object Tracking Based on Backward Model Validatoin.” IEEE Transactions on Circuts and Systems for Video Technology. Volume 24, Issue 11. pp. 1898-1910, November 2014

Zhang, F., B. Du and L. Zhang (2015). "Saliency-Guided Unsupervised Feature Learning for Scene Classification," in IEEE Transactions on Geoscience and Remote Sensing. 53(4): 2175-2184.

Zhang, G., Hu, M. Y., Patuwo, B. E., & Indro, D. C. (1999). Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. European journal of operational research, 116(1), 16-32.

Zhao, X., Dawson, D., Sarasua, W. A., & Birchfield, S. T. (2016). Automated traffic surveillance system with aerial camera arrays imagery: Macroscopic data collection with vehicle tracking. Journal of Computing in Civil Engineering, 31(3), 04016072.

Zhou, P., Ye, W., Xia, Y., & Wang, Q. (2011). An improved canny algorithm for edge detection. Journal of Computational Information Systems, 7(5), 1516-1523.

Zhu, Z. and Brilakis, I. (2010). “Concrete column recognition in images and videos”, Journal of Computing in Civil Engineering, ASCE, Vol. 24, No. 6, pp. 478–487

Zhu, Z., German, S. and Brilakis, I. (2010). Detection of large-scale concrete columns for automated bridge inspection, Automation in Construction, Vol. 19, No. 8, pp. 1047–1055.

Zhu, Z., German, S. and Brilakis, I. (2011). “Visual retrieval of concrete crack properties for automated post-earthquake structural safety evaluation”, Automation in Construction, Vol. 20, No. 7, pp. 874–883.

Zhu, Z., Ren, Z., and Chen, Z. (2016). “Visual tracking of construction jobsite workforce and equipment with particle filtering.” J. Comput. Civ. Eng., 30(6), 04016023.

Zitnick, C. L., & Dollár, P. (2014, September). Edge boxes: Locating object proposals from edges. In European conference on computer vision (pp. 391-405). Springer, Cham.

APPENDIX A

TABLE A-1. PERFORMANCE OF R-CNN ON IMAGES IN TESTING SET.

Image	TP	FP	FN	TN
1	0	0	3	3
2	0	0	4	9
3	0	0	7	9
4	2	5	3	9
5	2	0	5	8
6	2	1	4	12
7	1	2	5	13
8	1	0	2	12
9	2	1	2	12
10	1	0	2	10
11	1	0	2	1
12	3	0	2	0
13	0	0	2	4
14	0	0	3	8
15	0	0	3	11
16	2	0	4	2
17	3	0	2	10
18	0	0	4	6
19	1	0	2	5
20	3	0	2	10
21	2	0	1	15
22	1	0	4	14
23	0	0	7	9
24	0	0	3	9
25	2	0	3	9
26	0	0	5	13
27	0	0	2	8
28	2	0	1	3
29	2	0	1	2
30	2	1	3	15
Total	35	10	93	251

TABLE A-2. DETAILED RESULTS OF TRACKING ALGORITHM

Original Image Key Points	Subsequent image Key Points	Matched Points	Inlier Points
2796	4985	433	202
1613	4180	114	65
2620	10920	728	609
3899	5383	575	472
3237	3582	587	533
5393	4460	1184	1136
282	2149	14	4
2791	4973	1058	947
1271	1692	283	242
1486	6604	27	12
2715	5009	1002	859
2162	6664	268	162
402	1071	27	15
1527	5391	137	72
1526	4518	82	31
5421	7601	1079	963
5393	5157	624	533
1079	3173	442	217
2507	3002	266	230
2021	5312	361	86
2420	2625	260	181
1112	4659	23	9
1703	10948	214	176
105	1678	15	7
881	1902	34	18
2172	4831	329	77
1483	4841	110	81
1549	4319	336	250
2290	4746	785	681
1680	1629	304	152

TABLE A-3. DETAILED RESULTS OF R-CNN2

	TP	FP	FN	TN
1	0	0	3	5
2	0	0	4	13
3	0	0	7	8
4	1	0	4	24
5	2	0	5	9
6	2	0	4	9
7	2	10	4	26
8	1	1	2	15
9	2	0	2	10
10	1	0	2	20
11	2	0	1	0
12	2	0	3	1
13	2	0	0	8
14	2	0	1	11
15	0	0	3	15
16	2	0	4	0
17	3	0	2	9
18	0	0	4	5
19	2	0	1	11
20	3	0	2	18
21	2	0	1	16
22	1	0	4	21
23	0	0	7	6
24	0	2	3	13
25	2	0	3	10
26	1	0	4	22
27	0	0	2	12
28	2	0	1	0
29	2	0	1	1
30	0	0	5	21
Total	39	13	89	339

TABLE A-4. DETAILED RESULTS OF R-CNN3

	TP	FP	FN	TN
1	2	0	1	3
2	2	0	1	3
3	2	0	1	3
4	2	0	3	3
5	2	0	3	3
6	2	0	3	3
7	2	1	3	3
8	2	0	2	3
9	2	0	2	3
10	2	2	4	3
11	2	2	4	3
12	2	2	4	3
13	2	2	4	3
14	2	1	4	3
15	2	0	3	3
16	2	0	1	3
17	2	1	3	3
18	2	1	3	3
19	1	2	3	3
20	1	2	4	3
21	2	0	3	3
22	2	0	2	3
23	2	0	4	3
24	1	4	4	3
Total	45	20	69	72

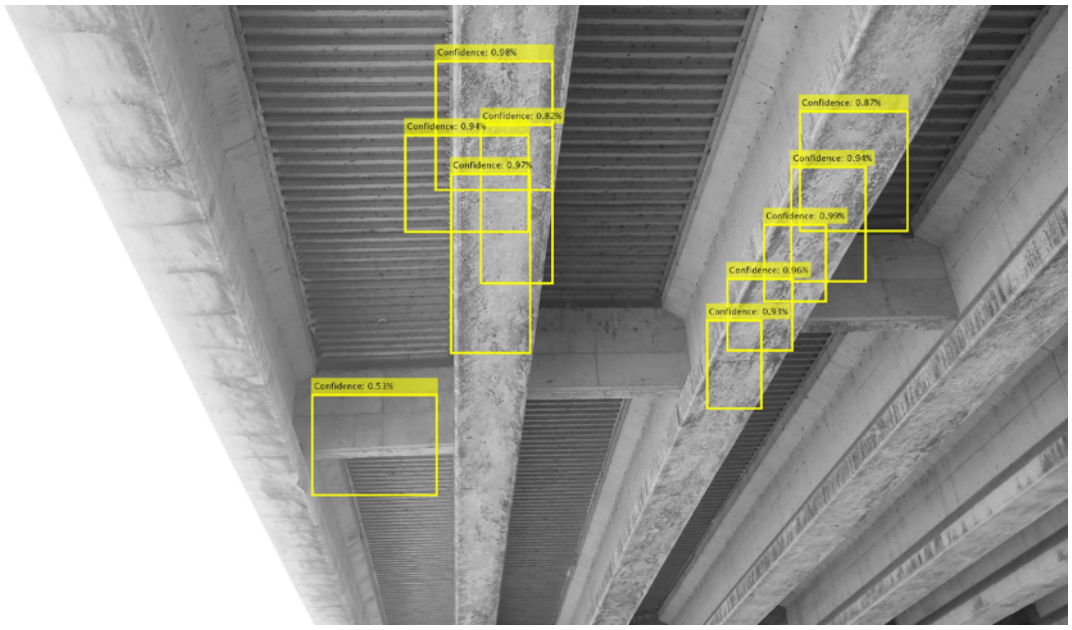

Figure A - 1. False positive example 1.

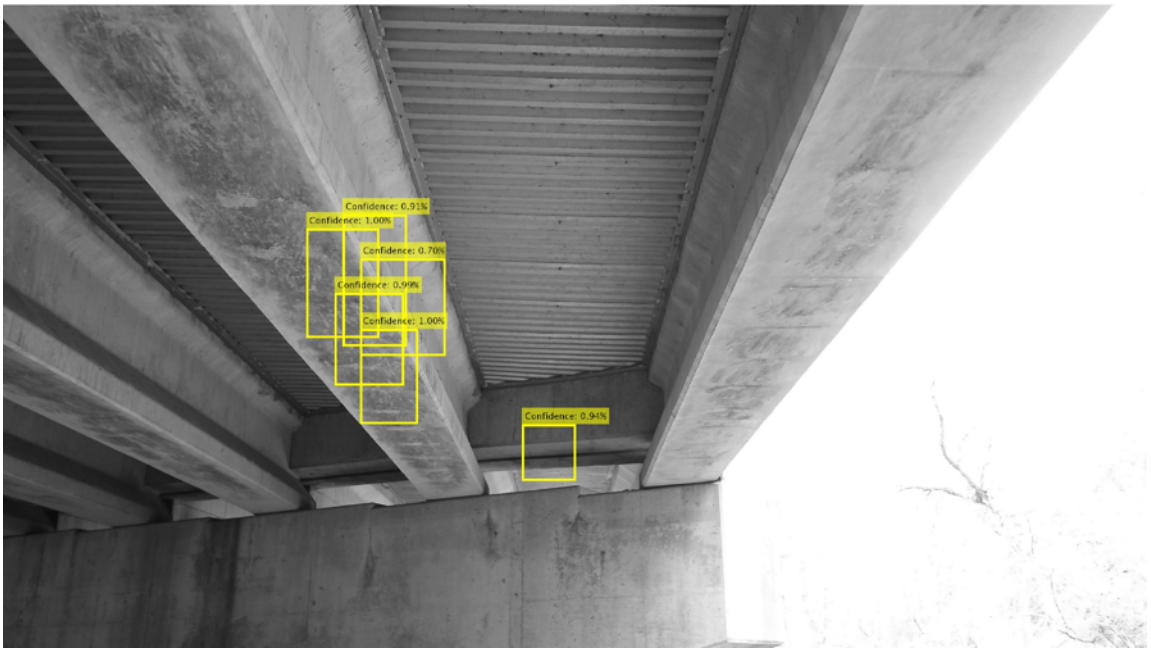

Figure A - 2. Beam flange false positive: Example 2

APPENDIX B

TRAINING THE R-CNN OBJECT DETECTOR

```
% training R-CNN detector on new domain
alex = alexnet;
layers = alex.Layers;
layers(23) = fullyConnectedLayer(6);
layers(25) = classificationLayer;

opts = trainingOptions('sgdm', 'InitialLearnRate',
0.001,'Momentum',0.9,...
    'MaxEpochs', 20, 'MiniBatchSize', 128);

% augemented data set
load 'beamTrain.mat' ;
beam.imageFilename = strcat(fullfile('dataset/',beam.imageFilename));

trainData = beam(:,:);

rcnn = trainRCNNObjectDetector(trainData, layers,
opts,'useParallel',true,...
        'NegativeOverlapRange',[0 0.3], 'PositiveOverlapRange',[0.5 1])
```

APPENDIX C

FULL BEAM DETECTION STEP ONE

```
% Step one: Locate beam edges within pROI
function [Col, Row, midpt] = elemEdgeDetectAllROI(testImage,bbox)

clearvars totlines threshold lines lines_new val out threshOut
[w ~] = size(bbox);
g = 0;
tHold = struct([]);

for l = 1:w
im = rgb2gray(testImage);
% get specified ROI dimensions
box = bbox(l,:);
% a = x, b = y, c = width, d = height
% Ax and Cx = a(x) , Ay and By = b(y)
% Bx and Dx = a+c  , Cy and Dy = b+d
[a,b,c,d,Ax,Ay,Bx,By,Cx,Cy,Dx,Dy,X,Y,x,y] = getROIDimensions(box, im);

% save midpoint of RCNN to verify (a) BBox midpoint is similar to other
% BBox and (b) if (a) is ture, make sure lines are on either side of
midpoint

[mid] = cornerAndMidPoints(Ay,Ax,c,d);

% disregard any bbox areas smaller than .25% of the overall image

area(l) = c*d;
areaIm = x*y;
percent(l,1) = area(l)/areaIm*100;
if (area(l)/areaIm*100) >= .02

% ------------------------------------------------------------------------
---
% ************************ Analyze ROI form BBox
************************
% Croping test image -> less complex image better for line detection

% Crop test image
image = im(Ay:Cy,Ax:Bx);

% edge detection of cropped image
% adaptive canny edge detection dependent on val

[BWnTH,threshOut] = edge(image,'Canny');
val = abs(threshOut(1,1)-threshOut(1,2));
[numThresh] = adaptThresh(val);
```

```
threshold = threshOut*numThresh;

if threshold(1,2) >= .99
    threshold(1,2) = .99;
elseif threshold(1,2) < .99
    threshold(1,2) = threshold(1,2);
end

tHold(l).val = val;
tHold(l).threshold = threshold;
tHold(l).threshOut = threshOut;
BW1 = edge(image,'Canny',threshold);

% find longest edge in cropped image to find left or right edge of the
beam
% P = 1 to choose one line
P = 1;
theta1 = -70;
theta2 = 70;

[lines] = linesByHoughTransform(BW1,d,P,theta1,theta2);

% figure, imshow(BW1), hold on
% for k = 1:length(lines)
%    xy3 = [lines(k).point1; lines(k).point2];
%    plot(xy3(:,1),xy3(:,2),'LineWidth',2,'Color','cyan');
%
% end
% hold off

% ------------------------------------------------------------------------
---
% ********************** If edge of beam found
************************

    if isempty(lines) == 0

        % Determine new ROI based on location of midpoint of previously
found line
        [image2,Ax,theta_new1,theta_new2] = expandROI(lines, im, Ax,
Ay, Cy, Bx, c, d, areaIm);

        BW = edge(image2,'Canny',threshold);
        P = 2;
        [lines_new] =
linesByHoughTransform(BW,d,P,theta_new1,theta_new2);
        if length(lines_new) > 2
            lines_new = lines_new(1,1:2);
        end
%       figure, imshow(BW), hold on
%       for k = 1:length(lines_new)
%           xy4 = [lines_new(k).point1; lines_new(k).point2];
%           plot(xy4(:,1),xy4(:,2),'LineWidth',2,'Color','cyan');
```

```
%       end
%       hold off

% ****************** Make sure new line was picked up
*********************
    [e f] = size(lines_new);

    if f > 1 % Two different lines found

        g = g+1;
        totlines(g,:) = [lines_new];
        Axx(g,:) = Ax;
        Ayy(g,:) = Ay;
        midpt(g,:) = mid;

% If new line not picked

    elseif f == 1 % only one line found

      % see if new line is same as threshold line

      if lines.theta ~= lines_new.theta % if not the same line save old
and new line
          g = g+1;
          % adjust lines

          lines.point1(1,1) = lines.point1(1,1)+abs(a-Ax);
          lines.point2(1,1) = lines.point2(1,1)+abs(a-Ax);

          lines_new = [lines lines_new];

          totlines(g,:) = [lines_new];
          midpt(g,:) = mid;
          Axx(g,:) = Ax;
          Ayy(g,:) = Ay;

      elseif lines.theta == lines_new.theta % if same line continue
          %continue
      end
    elseif f == 0
        %continue
    else
        disp('error in size of lines_new')
    end
    % if no lines are found then move onto next ROI

    elseif isempty(lines) == 1
        %continue
    else
        disp('error in isempty(lines)')
    end
end
```

```
end

%%
% if lines were found
if g > 0

line1 = totlines(:,1);
line2 = totlines(:,2);

figure, imshow(testImage), hold on
for k = 1:g
   xy1 = [line1(k).point1; line1(k).point2];

plot(Axx(k,1)+xy1(:,1),Ayy(k,1)+xy1(:,2),'LineWidth',6,'Color','red');
   xy2 = [line2(k).point1; line2(k).point2];

plot(Axx(k,1)+xy2(:,1),Ayy(k,1)+xy2(:,2),'LineWidth',6,'Color','blue');
end
hold off


for h = 1:g
% Columns
x1 = Axx(h)+line1(h).point1(1,1);
x2 = Axx(h)+line1(h).point2(1,1);
x3 = Axx(h)+line2(h).point2(1,1);
x4 = Axx(h)+line2(h).point1(1,1);

% row
y1 = Ayy(h)+line1(h).point1(1,2);
y2 = Ayy(h)+line1(h).point2(1,2);
y3 = Ayy(h)+line2(h).point2(1,2);
y4 = Ayy(h)+line2(h).point1(1,2);

% store first and last point of lines found
Col(h,:) = [x1 x2 x3 x4];
Row(h,:) = [y1 y2 y3 y4];
end

% ------------------------------------------------------------------------
---
% ********************** If edge of beam not found
*********************
elseif g == 0
    Row = [];
    Col = [];
    midpt = [0,0];
    %figure, imshow(im)
else
    disp('error with totlines')
end

end
```

APPENDIX D

FULL BEAM DETECTION STEP TWO

```
%% Step two: Extend pROI to image edges and mask polyROI
% find polyROI
function [polyROI,go,jgo,L,N] = proposedFullBeam(testImage, bbox, jbox)


[Col, Row, mid] = elemEdgeDetectAllROI(testImage,bbox);
jgo = 0;
im = rgb2gray(testImage);
[im1 im2] = size(im);
polyROI = zeros(im1,im2,'uint8');
if isempty(Col) == 0
    go = 1;


a = size(Col,1);
mask = zeros(im1,im2);


% ------------------------------------------------------------------------
----
% ****************** rough estimate of joint location
********************

num = 0;

%figure, imshow(testImage), hold on
if isempty(jbox) == 0
    jgo = 1;
   for rr = 1:size(jbox,1)
        % boubox = [a b c d];
        % X = [Ax Cx Dx Bx Ax];
        % Y = [Ay Cy Dy By Ay];
        jox = jbox(rr,:);
        [boubox,X,Y,mpt] = getJboxDimensions(jox,polyROI);
        Cy = Y(2); Dy = Y(3);
        Cx = X(2); Dx = X(3);
H = (Cx-Dx);
J = (Dy-Cy);
K = (Cy*Dx-Dy*Cx);
aLine = [J, H, K];
points(rr,:) = lineToBorderPoints(aLine,size(im));
        jointPoints(rr,:) = [Cx,Cy,Dx,Dy];
        %plot(jointPoints(rr,[1 3]),jointPoints(rr,[2 4]),'-r')
      %  line([points(1,1) points(1,3)], [points(1,2) points(1,4)],
'Color','red','LineWidth',3)
```

```
    end

elseif isempty(jbox) == 1
else
    disp('error with jbox')
end
%hold off

% ------------------------------------------------------------------------
----
% ************************* create polyROI mask
*************************

%figure, imshow(testImage), hold on
for i = 1:a

x1 = Col(i,1);
x2 = Col(i,2);
x3 = Col(i,3);
x4 = Col(i,4);

y1 = Row(i,1);
y2 = Row(i,2);
y3 = Row(i,3);
y4 = Row(i,4);

% Bx +Cy + D = 0
B=(x1-x2);
C=(y2-y1);
D=(y1*x2-y2*x1);

% EX + Fy + G = 0
E = (x4-x3);
F = (y3-y4);
G = (y4*x3-y3*x4);

% Creates lines
aLine1 = [C,B,D];
aLine2 = [F,E,G];

% Extends created lines all the way to the border
% Stores begin and end points
points1 = lineToBorderPoints(aLine1,size(im));
points2 = lineToBorderPoints(aLine2,size(im));
% line([points1(1,1) points1(1,3)], [points1(1,2) points1(1,4)],
'Color','red','LineWidth',3)
% line([points2(1,1) points2(1,3)], [points2(1,2) points2(1,4)],
'Color','blue','LineWidth',3)


 % analyze line pair for intersection

[inside,xint,yint] = isIntersect(points1,points2);
```

```
    if points2(1,1) >= points2(1,3)
        pt2x1 = points2(1,1);
        pt2x2 = points2(1,2);
        pt3x1 = points2(1,3);
        pt3x2 = points2(1,4);
    elseif points2(1,1) < points2(1,3)
        pt2x1 = points2(1,3);
        pt2x2 = points2(1,4);
        pt3x1 = points2(1,1);
        pt3x2 = points2(1,2);
    end

    if points1(1,3) >= points1(1,1)
        pt1x1 = points1(1,1);
        pt1x2 = points1(1,2);
        pt11x1 = points1(1,3);
        pt11x2 = points1(1,4);
    elseif points1(1,3) < points1(1,1)
        pt1x1 = points1(1,3);
        pt1x2 = points1(1,4);
        pt11x1 = points1(1,1);
        pt11x2 = points1(1,2);
    end


if inside == 1
    X1(i,:) = [pt1x1 xint xint pt3x1];
    X2(i,:) = [pt1x2 yint yint pt3x2];
elseif inside == 0

    X1(i,:) = [pt1x1 pt11x1  pt2x1 pt3x1];
    X2(i,:) = [pt1x2 pt11x2  pt2x2 pt3x2];
else
    disp('error with intersect')
end
pt1 = [X1(i,1) X2(i,1) X1(i,2) X2(i,2)];
pt2 = [X1(i,3) X2(i,3) X1(i,4) X2(i,4)];

% analyze line pair for intersection with joint bounding box
if jgo == 1
[jinside,jint] = isIntersectj(pt1,pt2,jointPoints);
%jint = [jxintersect1 jyintersect1 jxintersect2 jyintersect2];
for ii = 1:length(jinside)
    if jinside(ii) == 1
        X1(i,[2 3]) = jint(ii,[1 3]);
        X2(i,[2 3]) = jint(ii,[2 4]);
    elseif jinside(ii) == 0
        continue
    else
        disp('error with jintersect')
    end
end
end
```

```
mask_ii = poly2mask(X1(i,:),X2(i,:),im1,im2);

in = mask_ii(mid(i,1),mid(i,2));
if in == 1
    % creates interm mask within the two correlating lines (lines
within ROI)
    mask = mask + mask_ii;
elseif in == 0
    mask = mask;
else
    disp('error with interm mask');
end

end
%hold off
%%
% figure, imshow(insertShape(im,'polygon',[points1([3,1])
points1([4,2]) ...
%   points2([4,2]) points2([3,1])]));

% ------------------------------------------------------------------------
----
% ***************************** fill mask
*******************************

% fill mask between two lines
% This is the *Prelim* mask
I_fill = imfill(mask,'holes');
I = imbinarize(I_fill);
% B = row and column coordinates of boundary pixels
% L = 2D matrix of integers
% N = number of objects found
[B,L,N,A] = bwboundaries(I);

% Mask everything but full beam

for j = 1:im1
    for k = 1:im2
        if L(j,k) == 0
            polyROI(j,k) = 0;
        elseif L(j,k) ~= 0
            polyROI(j,k) = im(j,k);
        else
            disp('error in new image creation with mask')
        end
    end
end

figure, imshow(polyROI), %hold on
% plot(mid(:,2),mid(:,1),'xr')
% hold off
elseif isempty(Col) == 1
```

```
    go = 0;
    L = zeros(im1,im2);
   N = 0;
else
    go = 0;
    N = 0;
     L = zeros(im1,im2);
    disp('col error');
end


end
```

APPENDIX E

FULL BEAM DETECTION STEP THREE

```
%% Step three: Locate beam end and establish final ROI

% IF = Final Image
% RH = boundary of each region
% N = number of regions
% LH = masked regions
function [IF,NH,LH] = fullBeamDetection(testImage,bbox,jbox)

[polyROI,go,jgo,L0,N0] = proposedFullBeam(testImage,bbox,jbox);

if go == 1 && jgo == 1

[im1 im2] = size(polyROI);
pROI = zeros(im1, im2, 'uint8');
g = 0;
maskFinal = zeros(im1,im2);

[R,L,N,A] = bwboundaries(polyROI);


for r = 1:length(R)

    mask_pm = zeros(im1,im2);
    region = R{r,1};

    pm = poly2mask(region(:,2),region(:,1),im1,im2);
    mask_pm = mask_pm+pm;

    I_fill_r = imfill(mask_pm,'holes');
    I_r = imbinarize(I_fill_r);


    for j = 1:im1
        for k = 1:im2
            if I_r(j,k) == 0
                pROI(j,k) = 0;
            elseif I_r(j,k) ~= 0
                pROI(j,k) = polyROI(j,k);
            else
                disp('error in new image creation with mask')
            end
        end
    end

% find corner points
```

```
clearvars corners cx1 cx2 cx3 cx4 cy1 cy2 cy3 cy4 threshhold threshOut
val
% row col
[r1 c1 ~] = find(I_r(:,:)', 1, 'first');
[r2 c2 ~] = find(I_r(:,:)', 1, 'last');
[r3 c3 ~] = find(I_r(:,:), 1, 'first');
[r4 c4 ~] = find(I_r(:,:), 1, 'last');
% r or c == 2 -> 1
o1 = [r1 c1]; o2 = [r2 c2]; o3 = [c3 r3]; o4 = [c4 r4];
if o2 == o4
    [r4 c4 ~]= find(I_r(c2,:),1,'first');
    o4 = [c4 c2];
end

if o1 == o3
     [r3 c3 ~]= find(I_r(c1,:),1,'last');
     o3 = [c3 c1];

end


figure, imshow(pROI), hold on
plot(r1,c1,'ro',r2,c2,'go',o3(1),o3(2),'bo',o4(1),o4(2),'co')
hold off

for ii = 1:size(jbox,1)
% boubox = [a b c d];
% X = [Ax Cx Dx Bx Ax];
% Y = [Ay Cy Dy By Ay];

[boubox,X,Y,mpt] = getJboxDimensions(jbox(ii,:),pROI);
Ay = Y(1); Cy = Y(2); Ax = X(1); Bx = X(4);
l = abs(Ax-Bx);
image = pROI(Ay:Cy,Ax:Bx);

[BW,threshOut] = edge(image,'Canny');
val = abs(threshOut(1,1)-threshOut(1,2));
[numThresh] = adaptThresh(val);
threshold = threshOut*numThresh;

if threshold(1,2) >= .99
    threshold(1,2) = .99;
elseif threshold(1,2) < .99
    threshold(1,2) = threshold(1,2);
end

BW1 = edge(image,'Canny',threshold);

clearvars mask_r
% see if there are any horizontal lines meeting theta and l
[Hlines,point1,point2] = linesByHoughTransformH(BW1,1,l);

%if there is a line remask the image
```

```
    if isempty(Hlines) == 0
        g =1;
        p1 = [Ax+point1(1,1), Ay+point1(1,2)];
        p2 = [Ax+point2(1,1), Ay+point2(1,2)];
        P1 = [o1(1) p1(1,1) p2(1,1) o4(1)];
        P2 = [o1(2) p1(1,2) p2(1,2) o4(2)];
        pt(r,:) = [o1(1) o1(2) p1 p2 o4(1) o4(2)];
        mask_r = poly2mask(P1,P2,im1,im2);
        maskFinal = maskFinal + mask_r;

        figure, imshow(BW1), hold on
        xy3 = [Hlines.point1; Hlines.point2];
        plot(xy3(:,1),xy3(:,2),'LineWidth',2,'Color','cyan');
        hold off


    elseif isempty(Hlines) == 1
        pt(r,:) = [o1 o2 o3 o4];

    else
        disp('error in H')
    end
end
    if g == 0
        maskFinal = maskFinal + mask_pm;
    end
end



    I_fillH = imfill(maskFinal,'holes');
    IH = imbinarize(I_fillH);

    [RH,LH,NH,AH] = bwboundaries(IH);

%    Mask everything but full beam
    IF = testImage;
    for j = 1:im1
        for k = 1:im2
            if LH(j,k) == 0
                IF(j,k,:) = 0;
            elseif LH(j,k) ~= 0
                IF(j,k,:) = testImage(j,k,:);
            else
                disp('error in new image creation with mask')
            end

        end
    end

elseif go == 0 || jgo == 0
    IF = polyROI;
    LH = L0;
    NH = N0;
```

```
end
% figure, imshow(IF)
end
```

APPENDIX F

TRACKING ALGORITHM

```
% Track final ROI through to subsequent frame
function [projImage] = tracking(testImage,bbox,image2)
[I1,B,N,L] = fullBeamDetection(testImage,bbox);

if size(testImage,3) ~= 1
    I0 = rgb2gray(testImage);
elseif size(testImage,3) == 1
    I1 = I0;
end

if size(image2,3) ~= 1
    I2 = rgb2gray(image2);
elseif size(image2,3) ==1
    I2 = image2
end

sizeI = size(I1);

% [features0,valid_points0] = extractFeatures(I1,
detectSURFFeatures(I1));
% [features,valid_points] = extractFeatures(I2,
detectSURFFeatures(I2));
%
% [indexPairs, matchMetric] = matchFeatures(features0,features,
'Unique', true);
%
% matchedPoints0 = valid_points0(indexPairs(:,1),:);
% matchedPoints = valid_points(indexPairs(:,2),:);

R1 = normxcorr2(I1,I2)
[r1_D c1_D v1_D] = find(R1==max(R1(:)));

location1 = [c1_D r1_D Tc Tr];
outputImage1 = insertShape(I_01, 'rectangle', location1,'LineWidth',2);

[tform, inlierD, inlierO] = estimateGeometricTransform( matchedPoints0,
matchedPoints, 'projective');

outputView = imref2d(size(I2));
projImage  = imwarp(I1,tform,'OutputView',outputView);

end
```