

---

Mechanical Engineering Theses

Mechanical Engineering

---

Summer 8-12-2019

## Acute Angle Repositioning in Mobile C-Arm Using Image Processing and Deep Learning

Armin Yazdanshenas  
*University of Texas at Tyler*

Follow this and additional works at: [https://scholarworks.uttyler.edu/me\\_grad](https://scholarworks.uttyler.edu/me_grad)



Part of the [Mechanical Engineering Commons](#)

---

### Recommended Citation

Yazdanshenas, Armin, "Acute Angle Repositioning in Mobile C-Arm Using Image Processing and Deep Learning" (2019). *Mechanical Engineering Theses*. Paper 8.

<http://hdl.handle.net/10950/1861>

This Thesis is brought to you for free and open access by the Mechanical Engineering at Scholar Works at UT Tyler. It has been accepted for inclusion in Mechanical Engineering Theses by an authorized administrator of Scholar Works at UT Tyler. For more information, please contact [tgullings@uttyler.edu](mailto:tgullings@uttyler.edu).

ACUTE ANGLE REPOSITIONING IN MOBILE C-ARM USING IMAGE  
PROCESSING AND DEEP LEARNING

by

ARMIN YAZDANSHENAS

A thesis submitted in partial fulfillment  
of the requirements for the degree of  
Master of Science in Mechanical Engineering  
Department of Mechanical Engineering

Chung Hyun Goh, Ph.D., Committee Chair

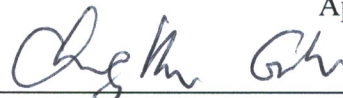
College of Engineering

The University of Texas at Tyler  
June 2019

The University of Texas at Tyler  
Tyler, Texas

This is to certify that the Master's Thesis of  
  
ARMIN YAZDANSHENAS  
  
has been approved for the thesis requirement on  
June 17<sup>th</sup>  
for the Master of Mechanical Engineering degree

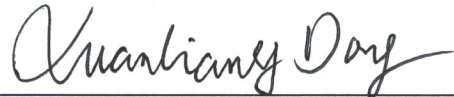
Approvals:



Thesis Chair: Chung Hyun Goh, Ph.D.



Member: Fredericka Brown, Ph.D.



Member: X. Neil Dong, Ph.D.



Chair, Department of Mechanical Engineering

  
FOR JK

Dean, College of Engineering

© Copyright by ARMIN YAZDANSHENAS 2019  
All rights reserved

## **Acknowledgments**

I would like to express my appreciation and gratitude to my thesis advisor Dr. Chung Hyun Goh for his unmatched support, continuous encouragement, and ceaseless guidance throughout the completion of this thesis.

I want to thank Dr. Fredericka Brown and Dr. X. Neil Dong for being irreplaceable members of my thesis committee and for their kind support and willingness to assist me in my endeavors. I would also like to thank the Chair of the Department of Mechanical Engineering Dr. Nael Barakat, the Associate Dean of the College of Engineering Dr. Michael McGinnis, and the Dean of the College of Engineering Dr. Javier Kypuros for allowing me to finish my thesis in a timely manner.

Finally, I would like to express my utmost appreciation to my brother and research partner Alireza Yazdanshenas for his immeasurable assistantship during my graduate studies, for the completion of this thesis would not be possible without his guidance. I would also like to acknowledge my parents for their unconditional support. I will always honor the sacrifices they have made to set me on a path worth venturing.

## Table of Content

List of Tables .....	iii
List of Figures .....	iv
List of Equations .....	vi
Abbreviations .....	vii
Nomenclature .....	vii
Abstract .....	viii
Chapter 1 Introduction .....	1
1.1 Background/Motivation .....	1
1.2 Problem Statement .....	2
1.3 Scope/Framework .....	2
1.4 Outline of Thesis .....	5
Chapter 2 Literature Review .....	6
2.1 Image Processing .....	6
2.2 Artificial Intelligence in Medical Imaging.....	8
2.3 Fast Fourier Transform and Monte Carlo Simulation.....	11
2.4 Forward and Inverse Kinematic Models .....	12
Chapter 3 Methodology .....	13
3.1 Overview .....	13
3.2 Deep Learning .....	14
3.2.1 Convolutional Neural Network .....	14
3.2.2 Supervised Learning .....	22
3.2.3 Network Training .....	23
3.3 Image Processing .....	26
3.3.1 Image Generation .....	26
3.3.2 Projective Transformation.....	28
3.3.3 Point Feature Matching.....	31
3.4 Automatic Repositioning .....	32
3.4.1 Camera Calibration .....	33
3.4.2 Angle Extraction .....	37
Chapter 4 Results and Discussion .....	39
4.1 Object Detection .....	39

4.2 Image Registration .....	48
4.3 Experimental Validation and Verification .....	49
Chapter 5 Conclusion.....	55
5.1 Summary .....	55
5.2 Outcome .....	56
5.3 Implications.....	56
5.4 Future Work .....	57
References .....	58
Appendix A Object Detection for Orthopedic Surgery .....	63
Appendix B Image Generation using FFT and MCS.....	74
Appendix C Image Registration using Projective Transformations .....	75
Appendix D Tilt and Orbital Angles of Rotation.....	77

## **List of Tables**

Table 1. R-CNN Learning Progression without Image Generation.....	41
Table 2. Object Classification without Image Generation.....	42
Table 3. R-CNN Learning Progression with Image Generation.....	44
Table 4. Object Classification with Image Generation.....	45
Table 5. Experimental Results – Matrices .....	52
Table 6. Experimental Results – Angles.....	53
Table 7. Experimental Results – Positions.....	54



## List of Figures

Figure 1. Schematic of C-Arm Components and Their Movements [2].....	1
Figure 2. Angles of Rotation (a) Tilt – Front View (b) Orbit – Side View .....	3
Figure 3. The Scope of this Study (Orange Bounds) in Context to the Overall Concept ..	4
Figure 4. Flow Chart of the Overarching Approach .....	13
Figure 5. Convolutional Neural Network [28].....	14
Figure 6. Features Used for Vertical Edge Detection (a) Background to Bone (b) Bone to Background .....	15
Figure 7. Test Image Portraying X-ray of Legs .....	16
Figure 8. Filtering Process .....	16
Figure 9. Filtering Outcome.....	17
Figure 10. Convolutional Map .....	18
Figure 11. Max Pooling Layer Filtration of Rectified Convolutional Map .....	19
Figure 12. Deep Stacking.....	20
Figure 13. Fully-Connected Layer .....	21
Figure 14. Pre-Labeled Input Images.....	23
Figure 15. Error Function.....	24
Figure 16. Gradient Descent on a Multi-Variable Error Function [33] .....	25
Figure 17. Image Generation using FFT and MCS.....	27
Figure 18. Projective Transformation [36] .....	30
Figure 19. Overlap of Fixed Image (Green) and Moving Image (Violet) .....	31
Figure 20. Multiple Control Point Pairs using SURF Point Matching .....	32
Figure 21. Overlap of Fixed Image (Green) and Registered Moving Image (Violet) .....	32
Figure 22. Identical Focal Point at Differing Orientations [40].....	33
Figure 23. C-Arm Mini with Mounted Camera .....	34
Figure 24. Calibration Images.....	35
Figure 25. Corner Extraction of Checkerboard.....	35
Figure 26. Extrinsic Parameters during Calibration Process (all units in cm).....	36
Figure 27. R-CNN Architecture.....	40
Figure 28. Image Generation Sample .....	43
Figure 29. Object Detection of (a) Chest, (b) Leg, and (c) Neck.....	47

Figure 30. Control Point Pairs (11) using Point Feature Matching .....	48
Figure 31. Registration Attempt including Point Feature Matching.....	49
Figure 32. Experimental Setup.....	50
Figure 33. Comparison of Experimental Pre- and Post-Procedural Images of a Neck Phantom .....	51
Figure 34. Detection of the Neck Phantom.....	52
Figure 35. Repositioning Validation through Inspection of Image Realignment .....	54

## List of Equations

Equation (1).....	25
Equation (2).....	25
Equation (3).....	29
Equation (4).....	29
Equation (5).....	29
Equation (6).....	29
Equation (7).....	30
Equation (8).....	30
Equation (9).....	30
Equation (10).....	30
Equation (11).....	37
Equation (12).....	37
Equation (13).....	37
Equation (14).....	37
Equation (15).....	38
Equation (16).....	38
Equation (17).....	38
Equation (18).....	38
Equation (19).....	38

## Abbreviations

**C-Arm** – mobile C-Arm medical x-ray imaging systems  
**CAM** – C-Arm Mini  
**CNN** – convolutional neural network  
**FFT** – fast Fourier transform  
**IR** – infra-red  
**MCS** – Monte Carlo simulation  
**R-CNN** – regions with convolutional neural network features  
**ReLU** – rectified linear units  
**SURF** – speeded up robust features

## Nomenclature

$B$  – biases  
 $E$  – error function  
 $\Phi$  – features  
 $f_x$  – focal length in x-direction  
 $f_y$  – focal length in y-direction  
 $\gamma$  – step size  
 $[H]$  – homography  
 $[K]$  – camera calibration matrix  
 $v$  – y-coordinate of control points in moving image  
 $m$  – number of control point pairs  
 $n$  – number of steps taken towards local minimum  
 $N$  – normalization factor  
 $p$  – point on error function  
 $p_x$  – principal point x-coordinate  
 $p_y$  – principal point y-coordinate  
 $[R]$  – rotation matrix  
 $r_1$  – first column of rotation matrix  
 $r_2$  – second column of rotation matrix  
 $r_3$  – third column of rotation matrix  
 $s$  – skew coefficient  
 $\theta_o$  – orbital rotation angle  
 $\theta_t$  – tilt rotation angle  
 $[T]$  – projective transformation matrix  
 $u$  – x-coordinate of control points in moving image  
 $\Omega$  – weights  
 $x$  – x-coordinate of control points in fixed image  
 $y$  – y-coordinate of control points in fixed image

## **Abstract**

### **ACUTE ANGLE REPOSITIONING IN MOBILE C-ARM USING IMAGE PROCESSING AND DEEP LEARNING**

Armin Yazdanshenas

Thesis Chair: Chung Hyun Goh, Ph.D.

The University of Texas at Tyler

June 2019

During surgery, medical practitioners rely on the mobile C-Arm medical x-ray system (C-Arm) and its fluoroscopic functions to not only perform the surgery but also validate the outcome. Currently, technicians reposition the C-Arm arbitrarily through estimation and guesswork. In cases when the positioning and repositioning of the C-Arm are critical for surgical assessment, uncertainties in the angular position of the C-Arm components hinder surgical performance. This thesis proposes an integrated approach to automatically reposition C-Arms during critically acute movements in orthopedic surgery. Robot vision and control with deep learning are used to determine the necessary angles of rotation for desired C-Arm repositioning. Novelty lies within the integration of the methods and their application to the C-Arm for the purpose of accurate repositioning. More specifically, a convolutional neural network is trained to detect and classify internal bodily structures. Image generation using the fast Fourier transform and Monte Carlo simulation is included to improve the robustness of the training progression of the neural network. With the incorporation of image generation, classification of the chest, neck, and leg were improved by 13.75%, 10.00%, and 11.67%, respectively. Image processing and feature matching techniques are used to couple the output of the neural network with control points. Matching control points between a reference x-ray image and a test x-ray image allows for the determination of the projective transformation relating the images. From the projective transformation matrix, the tilt and orbital angles of rotation of the C-Arm are calculated. This information is then sent to the kinematic model of the C-Arm so that it may reposition itself back to the desired reference position based on the input angles.

Overall, the approaching method links several different procedures into one large system that governs C-Arm movements. The mode of the overarching methodology innovatively applies image processing with convolutional neural networks to extract rotation matrices that may be used for assessing the orientation of the C-Arm with respect to the frame of the reference image. The originality of the proposition is presented in a way such that methods are not limited to the application of C-Arm repositioning. In fact, any robotic system associated with automatic repositioning of its linkages to a desired orientation is compatible to the modes outlined in this thesis. For validation, an experiment is run on a mobile C-Arm Mini prototype to justify the cogency of the integrated approach. When rotating the tilt and orbital linkages of the C-Arm Mini prototype by  $5^\circ$  and  $8^\circ$ , respectively, from the origin, the proposed method repositions the linkages by  $5.4^\circ$  and  $8.3^\circ$ , resulting in an overshoot of nearly  $0.5^\circ$ . Key results indicate that the proposed method is successful in repositioning mobile C-Arms to a desired position within 8.9% error for the tilt and 3.5% error for the orbit. Comparisons between the spatial coordinates before and after repositioning proves a positional error of only 0.08 inches along the y-coordinate of the global coordinate system; no errors existed for the x- and z-coordinates of the positions. As a result, the guesswork entailed in fine C-Arm repositioning is replaced by a better, more refined method. Ultimately, confidence in C-Arm positioning and repositioning is reinforced, and surgical performance with the C-Arm is improved. This includes minimizing operation time, reducing operation costs, eliminating unwanted exposure to radiation from multiple x-ray images, curtailing the risk of infection for the patient, and reducing the uncertainties entailed in the repositioning of mobile C-Arms.

# Chapter 1

## Introduction

### 1.1 Background/Motivation

The mobile C-Arm medical x-ray imaging system (C-Arm) is a radiographic machine that captures fluoroscopic x-ray images of patients during surgery. The name originates from the device's physical appearance in which a large “C” shaped linkage connects the x-ray source generator to an image intensifier or flat-panel detector [1]. During surgery, operating subjects rest on a flatbed positioned between the ends of the C-Arm, making it possible for surgeons to view medical x-ray images of patients in real time. C-Arms found in operating rooms in hospitals across the United States and parts of Europe are most commonly equipped with five degrees of freedom.

The two translational degrees of freedom account for the up-down and in-out movements; the remaining three degrees of freedom constitute the rotational movements, yaw, roll, and pitch, of the C-Arm. In context to C-Arm imaging systems, the yaw, roll, and pitch are colloquially referred to as the wig-wag, tilt, and orbit, respectively, as shown in Fig 1. Because of its remarkable mobility, the C-Arm has become a major medical device and a key contributor to successful operations.

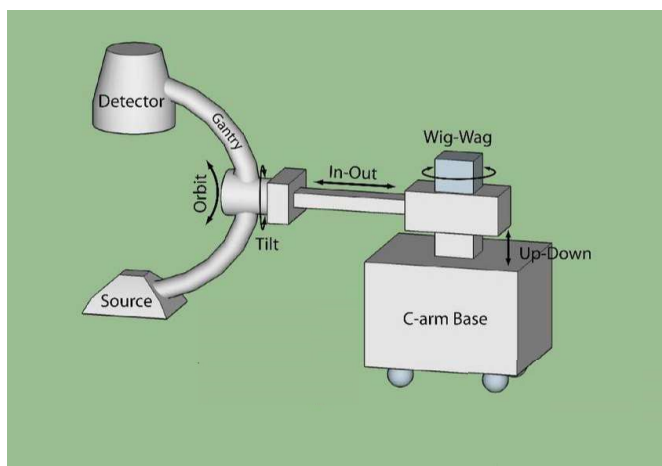


Figure 1. Schematic of C-Arm Components and Their Movements [2]

However, C-Arms still lack capability in terms of fine auto-repositioning. Interestingly enough, the mobile C-Arm itself has no issues in executing a movement with

exactness. Rather, it is the movement approximations made by the C-Arm technician that prevent accurate repositioning. Especially in surgeries requiring subtle movements, the lack of precision in current C-Arm repositioning methods often leaves surgeons exasperated. Attempts have been made to succor C-Arm technicians, as discussed in upcoming sections, but no foolproof method has yet been proposed.

## **1.2 Problem Statement**

With as much success as the C-Arm has had, the system, however, is in no shape perfect. Plenty of room for improvements exists. For instance, C-Arms are still maneuvered manually during surgery and require a skilled x-ray technician to operate. The manual maneuvering of the C-Arm is sufficient most of the time; however, complications with positioning accuracies arise during sensitive procedures. Certain surgical procedures require large equipment and the interaction of multiple operators. In those surgeries where large equipment is used, the C-Arm is rolled off to the side to allow for more room. The problem emerges when the C-Arm must be repositioned back to its original position with respect to the patient. Placing the C-Arm back to its exact position is vital for before-and-after x-ray image comparisons, validation of the surgical procedures, and diagnosis of remaining medical conditions. This repositioning of the C-Arm oftentimes becomes difficult, inaccurate, or a repetitive process. At times, multiple x-rays are taken to validate the repositioning of the C-Arm, exposing patients, surgeons, and x-ray technicians to unnecessary radiation [3]. Furthermore, for every minute that passes because of the repositioning the C-Arm, the cost of surgery will increase. The cost includes the payment of the x-ray technician and rent of the operation room and equipment. In addition to the cost, the longer a patient is exposed to the environment through surgery, the risk of infection increases for that person, meaning that if the repositioning time of the C-Arm was reduced, the risk of infection is also reduced in correlation.

## **1.3 Scope/Framework**

The objective of this thesis is to develop a method for automatic and autonomous repositioning of acute movements in current C-Arms. More specifically, robot vision and deep learning algorithms will be applied to the x-ray images and coupled to a kinematic



model of the C-Arm system for relatively small albeit precise movements. With a physical prototype, referred to as the C-Arm Mini (CAM), a complete kinematic model for the system was derived based on the actual mechanics, boundaries, and dimensions of the model. To reposition the C-Arm back to a previous location, Vicon motion capture technology was used. A local coordinate system was assigned to the C-Arm, and a global coordinate system was assigned to the operation flatbed, allowing the orientation of the C-Arm to be referenced to the flatbed. This will take care of the up-down, in-out translations and the wig-wag, tilt and orbital rotations as part of the main repositioning process (macro-repositioning). This thesis will focus on improving the final rotation movements, the tilt and orbital rotations  $\theta_t$  and  $\theta_o$  as shown in Fig. 2, to result in a much more accurate repositioning process (micro-repositioning). Focus is placed on what the desired input angles should be, how to detect them, and how to match them to reference points/angles through image processing and deep learning.

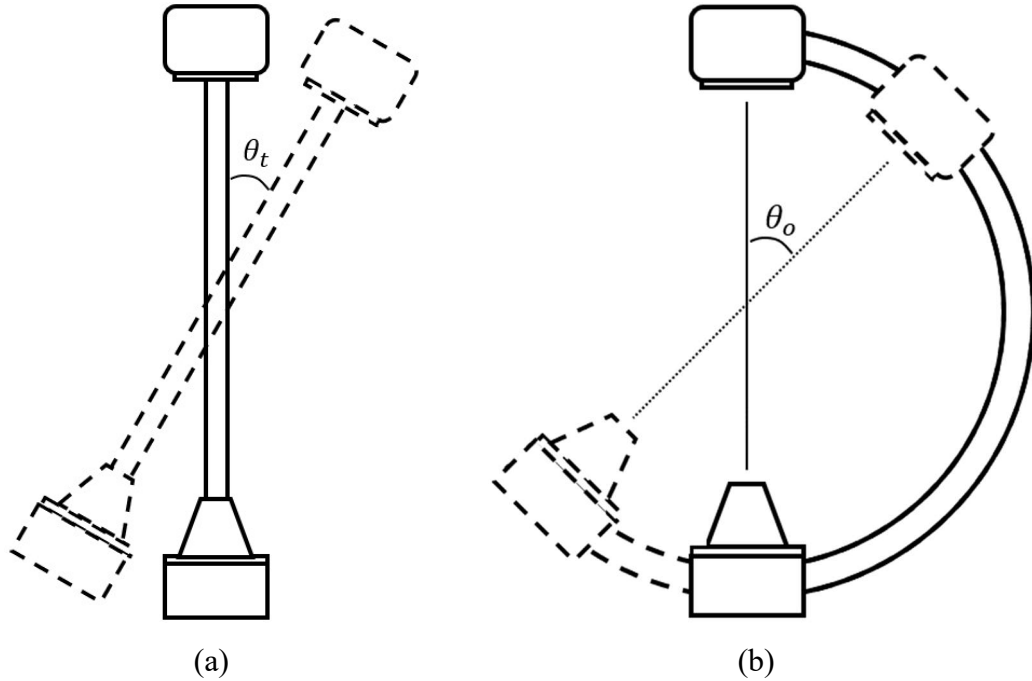


Figure 2. Angles of Rotation (a) Tilt – Front View (b) Orbit – Side View

Figure 3 details the scope of this thesis with respect to the overall concept. The region highlighted by the orange bounds entails the micro-repositioning approach used to acutely move the C-Arm's tilt and orbital rotations for accurate repositioning. The region

encompassed by the green bounds represents the initial and quicker macro-repositioning movements. The processes shaded by the blue bounds is the portion of the overall concept that is shared between the macro- and micro-repositioning approaches. The shared processes include the kinematic model and physical repositioning of the C-Arm. The macro-repositioning approach uses markers and reflected infra-red (IR) signals with a Vicron Motion Capture system to reposition a virtual C-Arm model in synchronization with the physical C-Arm. This research is focused solely on the micro-repositioning approach in which difference between x-ray image orientations are used to make fine rotations in the tilt and orbit linkages. Again, the general process flow of the micro-repositioning used in this thesis is demonstrated within the orange bounds of Fig. 3.

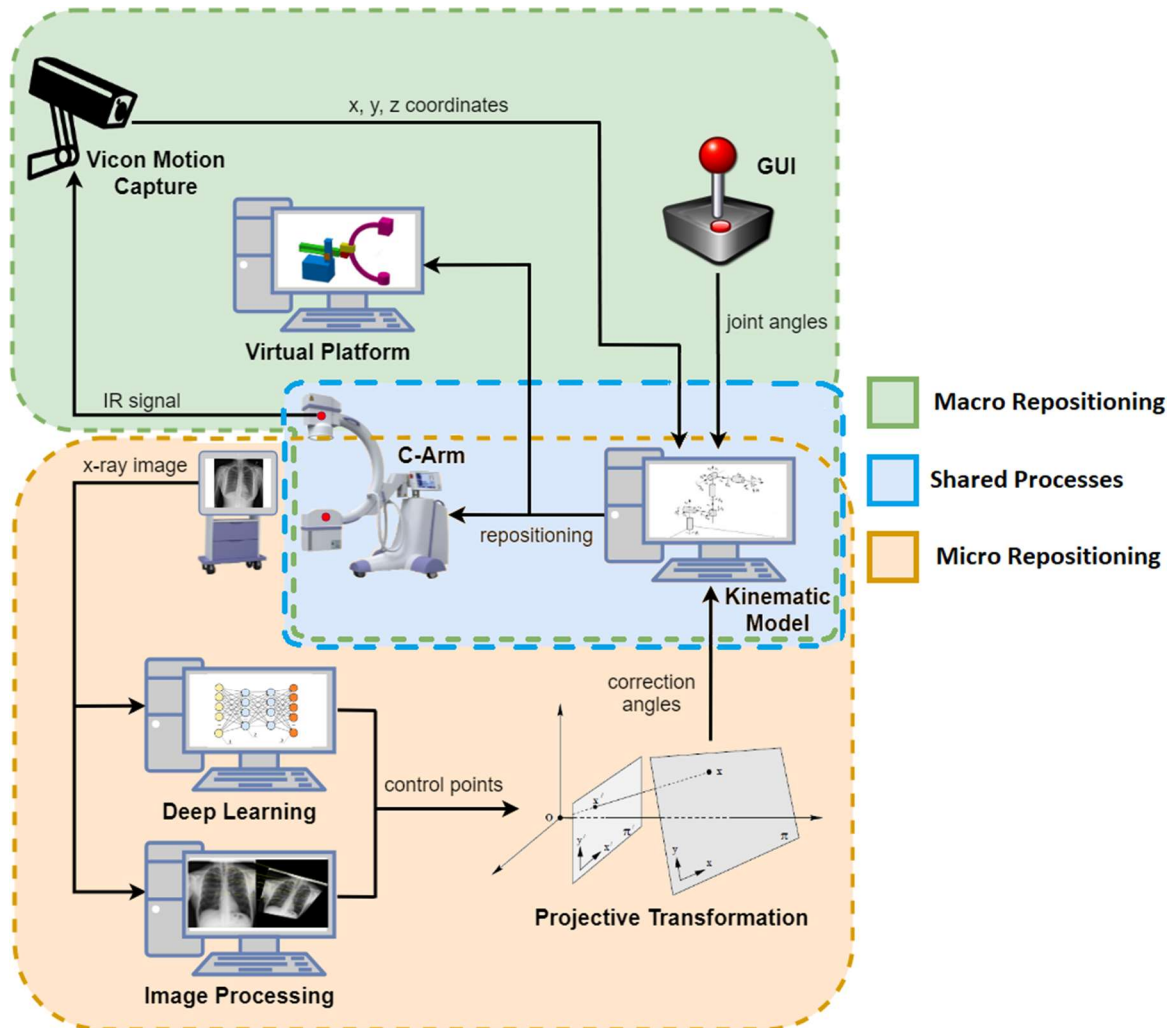


Figure 3. The Scope of this Study (Orange Bounds) in Context to the Overall Concept

The outcome of this work will improve the overall workings of C-Arms in the operation room; this includes minimizing time spent on maneuvering the C-Arm, eliminating the need to take multiple x-ray images to check for correct alignment, reducing the risk of infection, improving the accuracy of C-Arm repositioning, and decreasing the overall cost of operation.

#### **1.4 Outline of Thesis**

The thesis is structured as follows: In Chapter One, an introduction to the study was provided. In the introduction, the background, motivation, problem statement, and scope of the study were given. Chapter Two describes the literature review; current research on the C-Arm repositioning is examined with respect to image processing and artificial intelligence methods. The procedures and methodology of the study are detailed in Chapter Three. More specifically, the applications and methods of a convolutional neural network (CNN), supervised learning, network training, image generation, projective transformations, and point feature matching are all explained. The importance of camera calibration and the extraction method of the tilt and orbital rotation angles is also provided in Chapter Three. Key results are tabulated and discussed in Chapter Four. Chapter Four also discusses the results of an experimental case study using the CAM. To conclude the thesis, Chapter Five delivers a brief summary, discusses the implications of the study, and assigns future work and other possible implementations.

## **Chapter 2**

### **Literature Review**

#### **2.1 Image Processing**

Augmented reality is a major research topic in image-guided C-Arm surgery that focuses heavily on image processing. Computer vision and image processing techniques are used extensively with intraoperative x-ray images to visualize bone configurations for orthopedic, trauma interventional, and other such surgeries [4]. In these systems, the x-ray image source and video camera are paralleled. The unison of both image acquisition system synchronizes the footage that they collect in the time domain. Using homography and camera calibration processes, the positional overlay of both frames can be corrected in the space domain [5]. In other words, this method ensures that both the video images and x-ray images are in near-perfect alignment. The workings of the augmented reality method can be translated over and applied to the C-Arm for the purpose of automatic repositioning.

Image processing techniques such as image registration is also common in imaged-guided C-Arm systems. Similar to the methods used in augmented reality-based platforms, x-ray images are realigned based on pre-operative fluoroscopic calibrations and pose transformations [6]. Reference pins of known orientation are used to define the pose of the surgical object, typically a bone fragment. The change in pin orientation can then be used to calculate the equivalent transformation of the x-ray image orientation. Uploading this information into software for 3D projection contour modeling allows the 3D model to be registered (realigned) with the 2D x-ray image. The registered image allows the surgeon to see more detail and, thus, improving surgical outcome. However, this method requires external parameters such as the pin and knowledge of its orientation for every registration attempt. Registering images without the need of an external reference pin would make registration processes less demanding and improve the workflow.

Researches in academia have attempted to address a few of the persisting issues that exist with repositioning the C-Arm during operations. The current literature proposes a few developing methods of C-Arm repositioning that focus on automatic movement. Imaging methods have been developed for areas of interest that lie within the border of the radiographs. In one of these methods, an image can be re-centered by choosing a new location to be the image center; with the user's input, the edge-lying area of interest is

moved to the center of the image by automatic repositioning of the C-Arm [7]. This method of automated positioning and repositioning has been made possible through image processing; however, this proposed repositioning method is not fully automatic as it requires user input for determination of the desired region. In other words, as it currently stands, the C-Arm is capable of moving on its own, but it still needs an external input from a technician to approximate or pinpoint the desired center point.

Developments have been made to where the C-Arm may be maneuvered more accurately using three-dimensional coordinate systems. An initial reference element is used in the coordinate system to pinpoint the current location of the C-Arm. The orientation deviation between the reference element and the location point is measured using markers and computerization. Using an intraoperative approach, a motorized exact positioning unit is then capable of positioning the C-Arm based on directional data provided by the system, after which the surgeon assesses the results of the C-Arm repositioning [8]. Although such an approach fares well in determining the current position of the C-Arm and how it should move to reposition to a desired point, the repositioning itself relies on a global positioning system that makes acute movements and small deviations difficult to process. Because of this, the validity of the reposition of the C-Arm is largely based on the satisfaction of the surgeon.

The current literature has also ventured into the possibility of incorporating image processing applications to assist in patient repositioning. The method involves taking a reference x-ray image of the patient body part of surgical interest, mapping the x-ray image to reconstruct a radiograph from three-dimensional scanning data, superimposing the generated image with a differently orientated x-ray image, detecting the positional error using specific landmarks compatible to both images, and relocating the patient based off the detected positional error [9]. With this process, the C-Arm's rotational and translational components are assumed to be locked and unchanging, and only the patient and flatbed are the variable components. This can be problematic in situations where the C-Arm's orbit and tilt rotations must be moved to unhinge the C-Arm from the patient and flatbed. Unless an automated flatbed capable of tilting the patient at various angles is used, no positioning of the patient or flatbed can be made to exactly reposition the patient back to the reference orientation. Moreover, the positioning method presented by the current

literature still relies on manual maneuvering of the moving object, which is the patient and flatbed in this case.

Considerable efforts have been made to improve the repositioning capabilities of the C-Arm by optimizing the use of the computed tomography and C-Arm imaging with external tracking systems. To alleviate some of the drawbacks faced by these systems, a variety of Camera-Augmented Mobile C-Arm solutions have been proposed to calibrate the C-Arm's projection geometry online by attaching a camera and image intensifier and detector to the mobile C-Arm. The integrated Camera-Augmented Mobile C-Arm system makes it possible to enable intraoperative navigation without an external tracking system; however, additional procedures are required for the setup of the Camera-Augmented Mobile C-Arm as well as its associated calibrations, which translates over to a need for additional time and professional skills [10] [11].

Distortions are also present in both optical video camera and x-ray imaging in some of the current developments. These systems require calibration at every orientation for more exact distortion rectifications. Distortions requiring corrections not as urgently depend on models and precomputed look-up tables [10]. In addition, these systems lead to inaccuracy due to a slightly reduced distance between the radiation source and image intensifier, creating more problems that require further handling by well-trained operators for additional calibration processes. Although these methods improve on the C-Arm repositioning accuracy, they are ultimately tradeoffs for additional operation time and more knowledgeable and experienced technicians. To overcome these issues, this thesis uses the calibration parameters of the C-Arm x-ray image capturing system to account for unwarranted distortions. Additionally, the proposed solution needs to be executed only once per C-Arm.

## **2.2 Artificial Intelligence in Medical Imaging**

Recent advancements in artificial intelligence have also made their way into the realm of operating rooms, specifically with x-ray imaging. The current literature presents methods of detecting a variety of diseases through artificial intelligence radiographic models. Tuberculosis, pneumonia, cancer, and other medical conditions can be detected and classified with deep learning neural networks as accurately as expert radiologists [12].

Deep learning radiographic models are also in the works of determining the likelihood of cancer development in the body and predicting the probability of its spread. Such models make use of databases composed of thousands of x-ray images. However, these studies have focused solely on the chest for potential diseases; the potential to apply these models for the purpose of C-Arm movement has not yet been investigated. This may be due to some of the limitations that come with deep learning. For instance, open access to large databases of real x-ray images is difficult to come by. Although there are some open sources, a vast majority of x-ray images are locked away in hospitals due to privacy rights and other legal jurisdiction. Not only this but access to multiple areas of the body, not just the chest, is needed to create a well-rounded deep learning algorithm. Developing further into these deep learning models, specifically detection and classification, can be beneficial for determining points of interest within an x-ray image.

Researchers, however, have been able to use deep learning for the detection of body parts during orthopedic procedures. In spinal surgery, detection of lumbar vertebrae in C-Arm x-ray images have made it possible for surgeons to find and detail the area of surgical interest more accurately [13]. Even in abnormal cases where pathological conditions and uncertain, multi-angle image views are involved, the lumbar vertebrae detection of the deep learning algorithm highly improves surgical confidence. To automatically detect the lumbar vertebrae, a CNN was modeled to indicate the location of the region of surgical interest within a C-Arm x-ray image. The model utilizes a two-stage approach: pre-processing of the training data and fusion of features to combine vertebrae shape and texture information for detection improvements. In this case, detection was the only goal, and the possibility of incorporating the work for C-Arm repositioning was not considered.

Deep learning models with CNN have also been used to address challenges faced in C-Arm surgeries involving pedicle screw implants [14]. The neural network categorizes regions within x-ray images into the screw head, screw shaft, and background. Once the pedicle screws are detected, geometric relationships were used to map the transformation needed to position the screw. A biplanar setup correlates the object of interest and acquired x-ray images to the preoperative and global coordinate frames. With this relationship, the major axes of the pose estimated screws are compared against each other. The deviation between the axes are calculated to provide an error estimation between their respective

orientations. The locations of the screw tips are also compared to estimate their difference. Overall, the CNN detected and categorized the pedicle screw shaft and screw head with 84.5% and 67.1% accuracy, respectively. The deviation errors for the tunnel axis angle and screw tip locations ranged from  $1^{\circ}$  to  $3^{\circ}$  and  $1mm$  to  $2mm$ , respectively [14]. The methods and procedures conducted by this research can be represented as a leitmotif to the work presented in this thesis, making it an appealing source for the automatic repositioning of the C-Arm.

The use of CNNs for the purpose of image registration in x-rays becomes obvious when the workings of CNNs are studied. By using pixel spatial correlations, CNNs correlate each pixel to its neighboring pixels [15]. The closer two pixels are on the image plane, the more highly correlated will those pixels be. As more correlations are formed, the number of neurons and parameters relating the correlations are increased. Depending on the image size, the number of neurons and parameters quickly multiply over the training progression. With hidden layers added in between, exponential overflow of neuron connections occurs, making the CNN impossible large to assess. To overcome this issue, a single parameter may be assigned to a multitude of neurons as through grouping. As a result, every successive group of neurons will reduce the number of parameters by a set factor and, thus, preventing unreasonable deepness in the network layers.

Improved variations of CNNs have also been developed by researchers working on deep learning networks. When labeled training data is scarce, a Regions with CNN features (R-CNN) can be administered to maintain quality results even with minimal training data. R-CNNs capitalize on supervised pre-training and domain-specific fine-tuning to gauge the training process towards a desired end result [16]. Instead of computing CNN features on an entire x-ray image, region proposals are extracted at the initial stages of the learning process to filter out the background and other likely sources of noise. The R-CNN then performs convolution and gradient descent analysis on the proposed regions only. Ultimately, R-CNNs facilitate network training of relatively small training data sets.

Artificially intelligence has also proven itself to be useful in the augmented reality platform. With the use of a Kinect sensor, machine learning is used to improve the augmented reality scene in such a way that surgeons are able to better perceive depth and overall scene understanding [17]. The Kinect sensor provides the machine learning model



information such as the color space, depth, and x-ray image data for training. With the help of image processing and computer vision, the information provided to the machine learning model can be used to produce background modeling, object identification, and determination of anatomy in x-ray images. The results were coupled into the augmented reality system to render a more realistic virtual model. Experimental trials were also performed to collect qualitative results from expert surgeons and 4<sup>th</sup> year medical students. In the end, the machine learning-based augmented reality system presented promising results for mitigating confusion and challenges in surgery simulations.

### **2.3 Fast Fourier Transform and Monte Carlo Simulation**

The training of CNNs is an intensive process requiring extensive amounts of training data. In context to this thesis, a plethora of x-ray images are needed to adequately train a CNN for object detection of orthopedic structures. Due to the limitations of this research, x-ray images were generated from pre-existing x-ray images and incorporated into the training data set of the CNN. Using the fast Fourier transform (FFT) coupled with the Monte Carlo simulation (MCS), image generation is made possible. Therefore, a thorough review of FFT with MCS is proposed as a preliminary study.

With exponentially-growing technological advancements, images are becoming larger and more refined as image data increases in capacity. With larger image data files, image processing and image handling is, in turn, becoming more and more complex and exhaustive. This not only impedes on the viability of image processing, but it also increases processing time. Additionally, the transfer of image data from computer sources to server system destinations via electronic modes of transference is becoming growingly expensive [18]. Fortunately, the development of FFT has made it possible to convert data functions into integrals of wave functions. Represented as sine and cosine composites, the FFT allows for easier manipulation and transfer of structured data, including digitized x-ray images. Once processed in the frequency domain, the inverse FFT may be applied to revert the data back to its original form without losing any unwarranted information [19]. Currently, FFT has been used in image processing for a variety of goals: contrast enhancement, edge detection, sharpening, linear and non-linear filtering, zooming, noise removal, etc. [20].

In order to create unique images, MCS was implemented for randomization of the image data. Researches who have already made endeavors with MCS have used its methods to address quantitative problems. For instance, MCS has been used to generate random objects and process for the observation of system behaviors in economics, to generate repeated samples for the estimation of numerical quantities in science applications, and to optimize complicated engineering problem solutions through randomized algorithms [21].

## **2.4 Forward and Inverse Kinematic Models**

Multiple kinematic representations of several different C-Arm models have been presented in the current literature. Some of the most common methods of establishing a kinematic model involve the use of a geometric approach to define an exact mathematical configuration. In such methods, the C-Arm is modeled as a robotic arm, and an analytical solution to the inverse kinematics is developed [22]. The forward kinematic model can also be developed using the Denavit-Hartenberg parametric scheme [23]. Inverse kinematic models have been established through closed-form solutions to place the C-Arm fluoroscope at a desired position [24] [25].

Radiation exposure is another motive for developing kinematic models for C-Arms. As discussed before, unwarranted radiation is a major problem in extensive and challenging interventional procedures. Both patient and staff are forced to endure the damaging effects of prolonged exposure to radiation. Because of this, pose optimization methods have been introduced to improve the efficiency of C-Arm movement during trying operations [26]. General solutions for fluoroscopic C-Arms have also been formulated through joint parameterization of robotized C-Arms [25].

Researchers at the University of Texas at Tyler have already designed, built, and manufactured a mobile prototype of an autonomous CAM capable of robotic movements in five degrees of freedom. With their partnership, the CAM will be used for experimental purposes of the methods proposed in this thesis. A full kinematic model of the CAM was also developed to establish a reliable function relating the input parameters, such as positions or angles, to the inverse kinematic model. The actual movements of the C-Arm will be performed using the forward kinematic model.

## Chapter 3

### Methodology

#### 3.1 Overview

Before diving into the details of the methods and procedures, a swift rundown of the methodology is issued to provide a general understanding of the organization and order of the approach as a whole. A comprehensive overview of the overall workings and procedures of the methodology is described in Fig. 4.

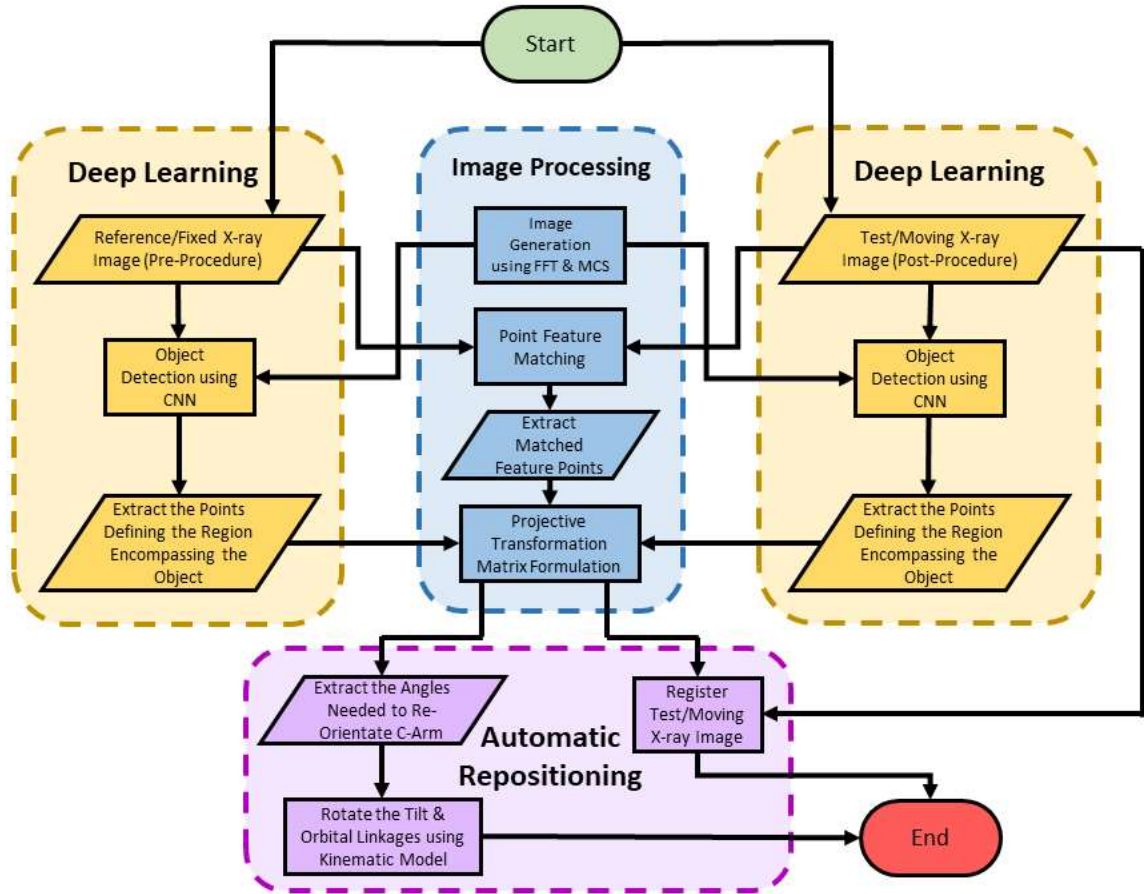


Figure 4. Flow Chart of the Overarching Approach

Following the schematic in Fig. 4, a reference and test x-ray image are both sent to a deep learning neural network for object (bone) detection where the points defining the region of interest of the detected object are found. It should be noted that the neural

network uses generated images, using fast Fourier transform (FFT) and Monte Carlo simulation (MCS), in its training database to improve detection and classification accuracies. The reference and test x-ray images are also imported to a feature matching algorithm where similar points between the two x-ray images are linked. All of the extracted points from the reference image are correlated to all of the extracted points from the test image to create a projective transformation matrix. The resulting transformation matrix can then be used to register, or realign, the skewed test image. The transformation matrix can also be used to extract the angles necessary for rotating the tilt and orbital rotations of the C-Arm such that it automatically repositions itself to the desired orientation.

### 3.2 Deep Learning

Deep learning applications with MATLAB were used to create a training network capable of object detection [27]. A CNN is one of the most popular algorithms for deep learning with images and videos due to its high efficiency and accuracy in featuring objects. Because of this, a CNN was used in performing feature detection of various x-ray images depicting bodily structures such as the neck, chest, and leg. An overview of the fundamental components driving a standard CNN is summarized in Fig. 5. As shown in the figure, CNN's are composed of several stages, or layers, that power the detection and classification processes. In the following subsections, details of each of the layers are discussed in application to object detection in digital x-ray images.

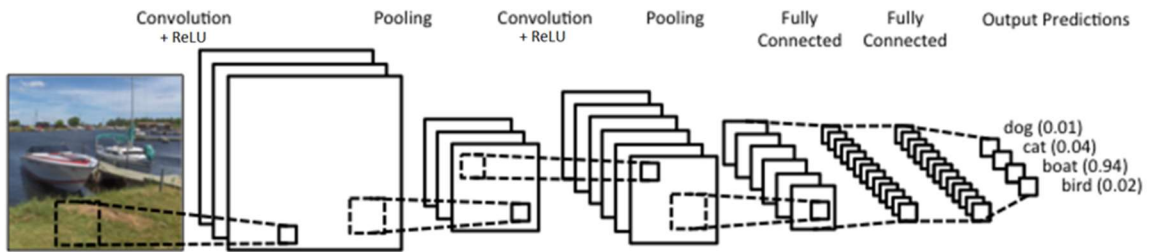


Figure 5. Convolutional Neural Network [28]

#### 3.2.1 Convolutional Neural Network

CNNs work by matching smaller parts of images, called features, to one another. Using image features rather than entire images is more beneficial for a number of reasons.

For one, computers are literal in their image matching capabilities, which means that even the slightest difference between two images will be seen as a mismatch, regardless of how similar the images may be. For instance, two identical images cannot be matched if one of them is altered in the slightest. Since the likelihood that the entirety of an image's pixel intensity matrix matches exactly with that of another image is extremely low, object detection becomes practically impossible. However, matching image features is far more likely to result in a match since only small but relevant components are considered. This makes the use of features a much more reliable method for detecting objects within images.

Examples of these features are mini-images that can vary in size anywhere from  $3 \times 3$  pixels up to any odd-numbered square image. These features are typically basic geometric shapes such as edges, lines, squares, rectangles, circles, or any other simple yet commonly found shape; Fig. 6 is an example of two different features. For simplicity, black, white and gray pixels were used to represent the pixel intensity of the feature.

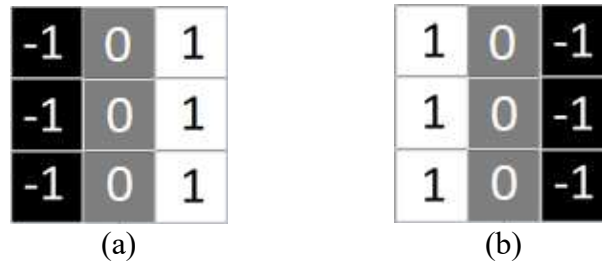


Figure 6. Features Used for Vertical Edge Detection (a) Background to Bone (b) Bone to Background

The features in Fig. 6 can be used to determine whether an image portrays an edge of a leg bone. This can be done through image filtering where the features are passed over a test image to estimate the likelihood that the image does in fact depict a leg bone. Unlike strict matching where the end result is either a perfect match or no match at all, filtering allows the CNN to calculate the probability of a match. Filtering works by lining up the feature and the test image, multiplying each image pixel by the corresponding feature pixel, adding up them all up, and dividing by the total number of pixels in the feature. To demonstrate this process, the feature in Fig. 6 (a) will be used to filter the test image in Fig. 7.



Figure 7. Test Image Portraying X-ray of Legs

For this example, the center of the feature is placed over the third column and the third row of the test image. The  $3 \times 3$  area of the test image that is overlapped with the feature image will be referred to as the image patch, shown as the green, yellow, and red boxes in Fig. 8. The corresponding pixel values of the feature are then multiplied by the pixel values of the image patch. The top left pixel value of the feature is multiplied by the top left pixel value of the image patch. Then, the top center pixel value of the feature is multiplied by the top center pixel value of the image patch. The same is done to the remaining pixels of the feature and image patch; step by step and pixel by pixel, they are all multiplied by each other.

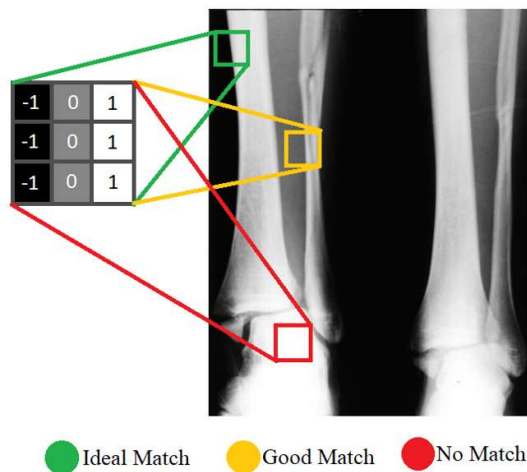


Figure 8. Filtering Process

The results of each of the products are then summed up and divided by the total number of pixels in the feature. In other words, the average of the results is taken so that a single value may be used to represent the overall match between the feature and the image patch. Since the image patch bordered in green is nearly identical to the feature, it gets a high match score where 1 is an exact match and  $-1$  is no match. The yellow has a slightly lower match score, and the red lower still. This average value will then be placed in a new blank image. The location of the average value in the blank image must be the same as the location of the center pixel of the image patch with respect to the whole test image. Inserting it into the newly constructed image results in Fig. 9 where the boxes represent the size of the image patch and feature. In order to make this insertion, the feature image size should have a pixel center. For this reason, the size of the feature should be an odd-numbered square image. Although it is possible to follow through with the filtering process using even-number sized features, the center point of these features is unclear, and placement of the filtered results becomes indistinct. With odd-numbered square features, however, all of the confusion is cleared by a definite center point.

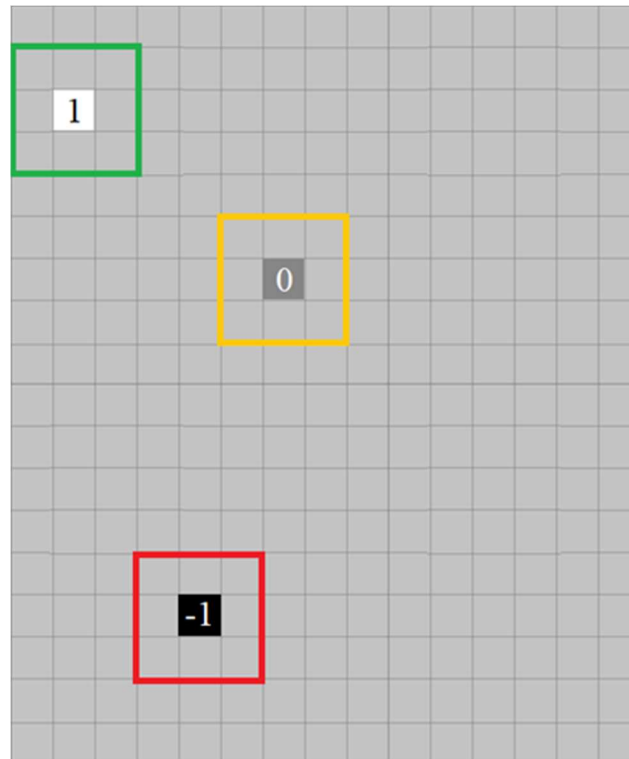


Figure 9. Filtering Outcome

The feature is continuously moved to other locations and overlapped with new areas on the test image, creating newer image patches to be matched with the feature. The filtering process is repeated with the new image patches, and new values are inserted into the blank image. By moving the filter around to different places in the test image, different values are found that indicate how well a feature is expressed at a certain position [29] [30]. After covering the entirety of the test image and all possible positions with the feature, Fig 9. will transform into a map of where the feature occurs most predominantly. Applying the filtering processes over the whole image results in the repeated application of the feature over and over again. This is referred to as convolution. Figure 10 illustrates the convolutional map of the feature from Fig. 6 (a).

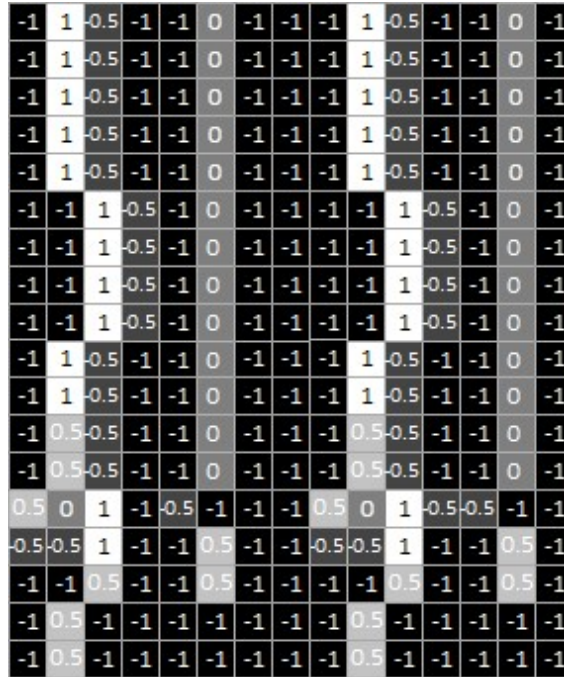


Figure 10. Convolutional Map

In convolution, one image becomes a stack of filtered images (convolutional maps), and there are as many filtered images as there are filters (features). After convolution, the output goes through a normalization process. In CNNs, normalization is performed by a rectified linear units (ReLU) layer. This type of layer performs a simple threshold operation, where any input value less than zero will be set to zero [27]. Such an operation preserves the mathematical development and prevents disruptions from perpetuating the



network. Compared to other activation functions, the ReLU activation function performs substantially better for deep learning applications despite of its simplicity and lack of linearity and differentiability [31]. All convolutional maps are passed through the ReLU filter.

The third layer is called the max pooling layer, and it shrinks the output image stack of the ReLU layer. A square window, typically sized at  $2 \times 2$  or  $3 \times 3$ , is defined and paced over the rectified convolutional map in strides. The stride determines the number of pixels that the window steps down and to the right as it paces through the filtered image. With each step, the maximum value within the window is tracked and used to create a smaller image. Figure 11 demonstrates the max pooling process where the green box is the window location before the first stride and the red box is the window location after the first stride.

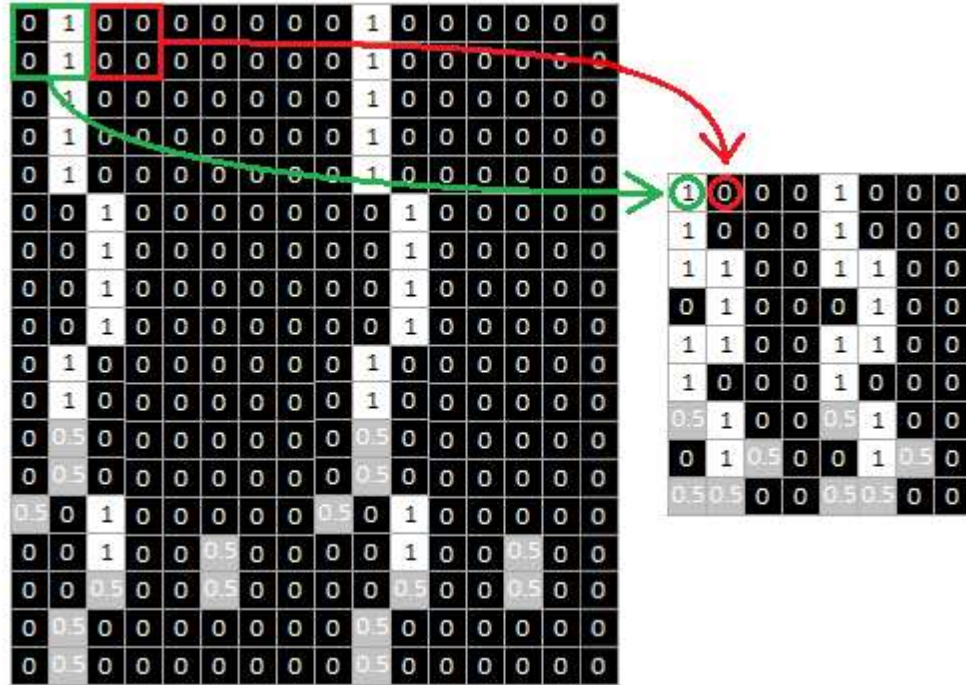


Figure 11. Max Pooling Layer Filtration of Rectified Convolutional Map

From the Fig. 11, it can be seen that max pooling generates a smaller version of the input image. The detected features, such as the white diagonal line, are preserved whilst minimizing the total number of pixels used. In other words, this layer reduces the size of the input image without losing any of the essential information. By choosing the greatest

value within the window, areas on the original image with the greatest coincidence of matching the feature are kept paramount, and since it does not matter where in the window the maximum value occurs, sensitivity to position is lessened. This means that a particular feature will still be detected regardless if it is translated slightly to one side or if it is partially rotated. Efficiency is also improved especially when considering large images and even larger data sets, as is common for neural networks, that would otherwise require undesirable processing time. All rectified convolutional maps are passed through the max pooling layer.

The initial layers of the CNN can be represented by the dimensions, color channels, and content of the original images; this is equivalent to the initial input of the entire network. The middle layers of the CNN can be summarized by a combination of the convolutional, ReLU, and max pooling layers. More specifically, three consecutive sets of convolutional, ReLU, and max pooling layers make up the middle layers. As shown in Fig. 12, the initial layers are passed through a convolutional layer whose output then becomes the input of a ReLU layer that sends its output to a max pooling layer. The output of the first max pooling layer then becomes the input of second convolutional layer. The data continues through all three sets until the output of the third and final max pooling is connected to the final layers.

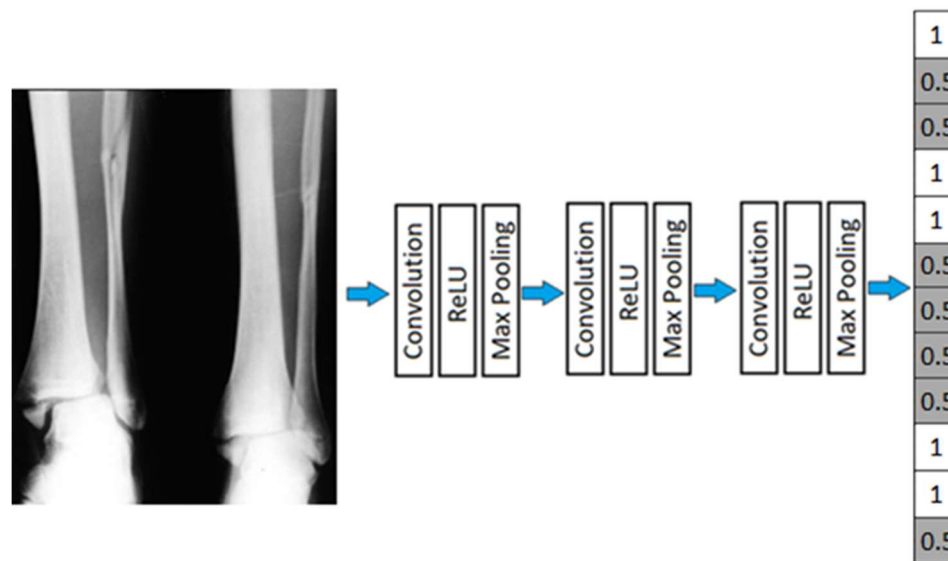


Figure 12. Deep Stacking

It should be noted that any combination and any number of layers can be used to create a CNN; however, most standard networks require at least a convolutional, ReLU, and max pooling layer. The output of the middle layers will be small matrices, and the number of matrices will depend on the number of features used for filtering. For visualization, they can be rearranged into a single column that lists the feature values. In the final layers, sets of fully-connected layers take the feature value lists and convert them into lists of indicators. The strength of these indicators depends on how strongly a value predicts the features, and they are ultimately used to determine what category is being depicted by an image. Figuratively speaking, a vote is casted by each of the values in the list, and the category with the highest overall vote is said to be what the image most likely portrays. Figure 13 is an example of how the fully-connected layers work, where the orange nodes symbolize the neurons within the hidden layers of the CNN.

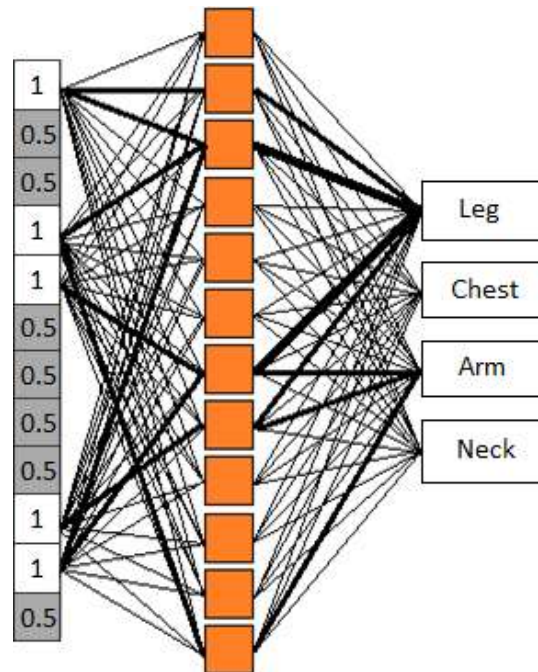


Figure 13. Fully-Connected Layer

Generally, multiple sets of fully-connected layers are incorporated. A fully-connected layer multiplies the input by a matrix and then adds a bias vector [27]. The multiplication matrix determines the weight of each vote, and it is symbolized by the thickness of the lines, or the strength of the connections, between the neurons. The bias

vector is responsible for determining how far off the predictions are from the input values, and it indicates whether a neuron tends to be active or inactive. A high bias correlates to faster learning, but as a result, they have lower predictivity on complex problems. On the other hand, a low bias uses more assumptions to increase flexibility and to improve predictive performance as a trade-off for a slower and more complex system. Together, the weights and biases function so that not all votes are casted equally; some are more partial to vote one way while others tend to vote another way. The actual values that are used for the weights, biases, and what the features should be are all governed by a process called gradient decent which will be covered in more detail in the section *3.2.3 Network Training*.

### *3.2.2 Supervised Learning*

Supervised learning is a form of deep learning that uses assisted classification to develop sensible conclusions. Unlike unsupervised learning which finds hidden structure within unlabeled data, supervised learning uses a pre-labeled training data set. The training data is equipped with both the input and the desired output of each classification. Each input image is paired with a constituting label, which is also referred to as the supervisory signal, so that the learning algorithm can formulate an inferred function. The inferred function demonstrates the intended relationship between the input and desired output by the means of function approximation through a set of fundamental assumptions. These assumptions state that the inferred function is compliant and consistent with the data, allowing for generalizations to be made. In other words, deep learning algorithms induct generalizations based on details and specifics that are provided to it. Also known as inductive bias, it is through the learning process of these generalizations that the algorithm is capable of making valid predictions from completely foreign inputs.

Figure 14 reveals x-ray input images and their pre-labeled output that is used as part of the training data set. The entirety of the x-ray image is used as the input, but only the user-defined region (yellow) is analyzed for features to be used for filtering. Anything outside of the bounded yellow box is ignored while all of the information within it is associated to the label. It is here where the CNN processes are conducted to construct an inferred function that links the data image input to the label output.

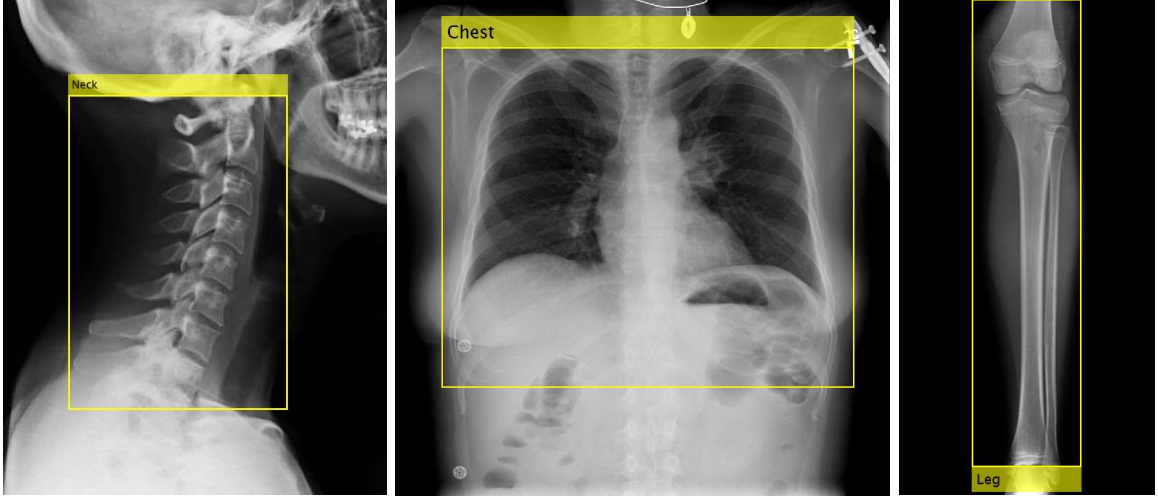


Figure 14. Pre-Labeled Input Images

As mentioned before, the yellow boxes encompassing the area of interest are predetermined by the user, so for each image, the size and location of the box as well as the correct label must be defined and placed into a matrix for proper grouping and indexing. Due to the limited resources and lack of open-source x-ray images, only three categories were investigated: neck, chest, and leg. These three categories were chosen strictly because of scarce availability of other x-rayed body parts. Since considerably large data sets are needed to develop a deep learning system with adequate decision-making capabilities and high accuracies and since these three categories were most abundant of the freely available x-ray images, this research will focus on the neck, chest, and leg regions of patients. Although this work is limited to these areas, the methods used in this research can be applied to any x-ray image just as effectively as long as large batches of x-ray images of the body part are included in the training data set.

### 3.2.3 Network Training

Before the CNN can be trained, several training options must be set. Firstly, the mini-batch size must be defined. The mini-batch size is the number of training images that will be used for each iteration. One iteration is equivalent to one pass through a mini-batch sized number of images. The number of iterations that it takes to cover the entire training data set depends on the mini-batch size and the total number of samples used in the training data set. Once all images in a training data set have been passed through, an epoch is said

to have been completed. The maximum number of epochs can vary based on user input. For example, if a training data set composed of 1,000 training images is passed through using a mini-batch size of 100, it will take 10 iterations to complete a single epoch. The more epochs the training sessions goes through, the more thorough will the network's training on the data set be.

In order for the remaining training options (initial learn rate, learn rate drop factor, and learn rate drop period) to make sense, the workings of the training process must be explained. Initially, the network starts with random features, weights, and biases. Using backpropagation, the network computes the error at the output, and since the desired output is provided by the image labels, the error is defined as the difference between the desired output and the predicted output. Because the initial features, weights, and biases are random, the first few iterations will most likely result in large errors. However, if the features, weights, biases are adjusted, the error can be lessened. To adjust the features, weights, and biases in such a way that the error is minimized, a process called gradient descent is used. As more and more iterations are performed, an error function based on the adjustments of the features, weights, and biases is formed. This error function  $E$  is a representation of the relationship between the output error and the features  $\Phi$ , weights  $\Omega$ , and biases  $B$ . Figure 15 is an example of a simple error function.

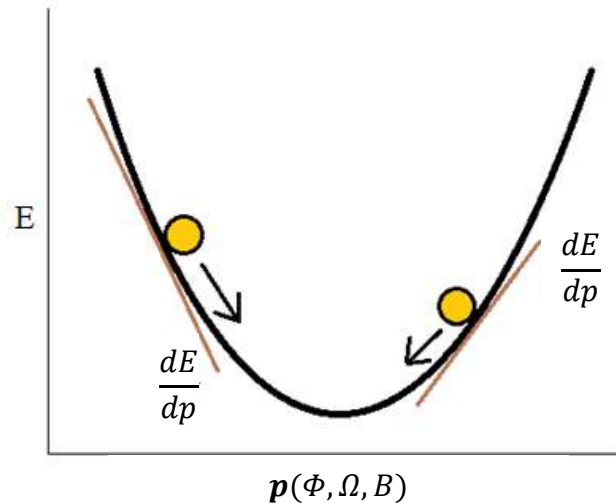


Figure 15. Error Function

The gradient of the multi-variable error function at any given point can be calculated using Eq. (1) to determine the direction of the closest local minimum

$$\mathbf{p}_{n+1} = \mathbf{p}_n - \gamma_n \nabla E(\mathbf{p}_n), \quad n \geq 0 \quad (1)$$

where  $\mathbf{p}_n$  is a point on the error function,  $n$  is the number of steps taken towards the local minimum, and  $\nabla E$  represents the gradient of the function  $E$ . The step size,  $\gamma_n$ , determines the size of the step to be taken to reach the next point [32]. The step size can change at every iteration, and it is defined by Eq. (2).

$$\gamma_n = \frac{(\mathbf{p}_n - \mathbf{p}_{n-1})^T [\nabla E(\mathbf{p}_n) - \nabla E(\mathbf{p}_{n-1})]}{\|\nabla E(\mathbf{p}_n) - \nabla E(\mathbf{p}_{n-1})\|^2} \quad (2)$$

With each gradient, the features, weights, and biases are adjusted slightly so as to bring the error closer towards the local minimum, and as the error approaches convergence, the adjustments of the features, weights, and biases are made smaller and smaller to avoid excessive overshoot. Once the slope of the tangent line is zero, the lowest possible output error has been reached, and the network will cease to adjust the features, weights, and biases for the neural connections relating to that output error. Since a multi-variable error function is considered for training virtually all neural networks, Fig. 16 is a more accurate depiction of the gradient decent process.

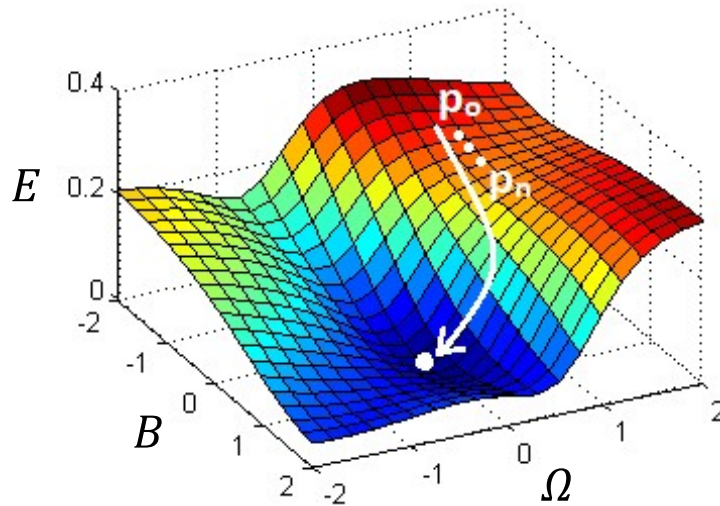


Figure 16. Gradient Descent on a Multi-Variable Error Function [33]



The initial learn rate training option parameter tells the network how drastically the changes to the features, weights, and biases should be to move the error down towards the closest local minimum. If the initial learn rate is too low, the training process will take a long time. If it is too high, the training process is likely to get stuck at a suboptimal result [27]. The learn rate drop factor is a multiplicative factor that is applied to the learning rate every time a certain number of epochs has passed. The learn rate drop period is the parameter that determines the number of epochs that should be passed before the learn rate drop factor is applied.

### **3.3 Image Processing**

The purpose of the image processing portion of this research is to improve the accuracy of the CNN by generating more images to be used in the training data set and to utilize geometric transformations to transform the post-procedural x-ray image to the orientation of the pre-procedural reference x-ray image. The resulting transformation matrix can then be used to dictate the tilt and orbital angles required to move the C-Arm so that all proceeding x-ray images are near-perfectly aligned with the fixed image.

#### *3.3.1 Image Generation*

Unfortunately, freely available open-source x-ray images of bodily structures are difficult to find. This poses a problem during the training portion of the neural network. To ensure that the CNN can reliably and accurately detect a specific object in the x-ray image, thousands of x-ray images are required in its training database. In other words, the larger the training dataset, the better the network will perform. Because of this, the CNN was trained only for chest, neck, and leg detection, but still, availability of x-ray images remains scarce. To make up for the lack of data, the FFT was used with MCS to generate additional data sets. The FFT converts the input (parent) image from the spatial domain to the frequency domain, making the data easier to manipulate in the MCS. The MCS creates a filtering ring that is used to filter out certain frequencies in the FFT of the parent image. The radius and thickness of the ring are altered randomly within a given range. Through iterative loops, any number of x-ray images can be generated from a single x-ray image. Since the radius and thickness of the filters are randomized within a threshold, each



generated x-ray image is entirely unique. Figure 17 shows how the FFT can be used with MCS to generate new images.

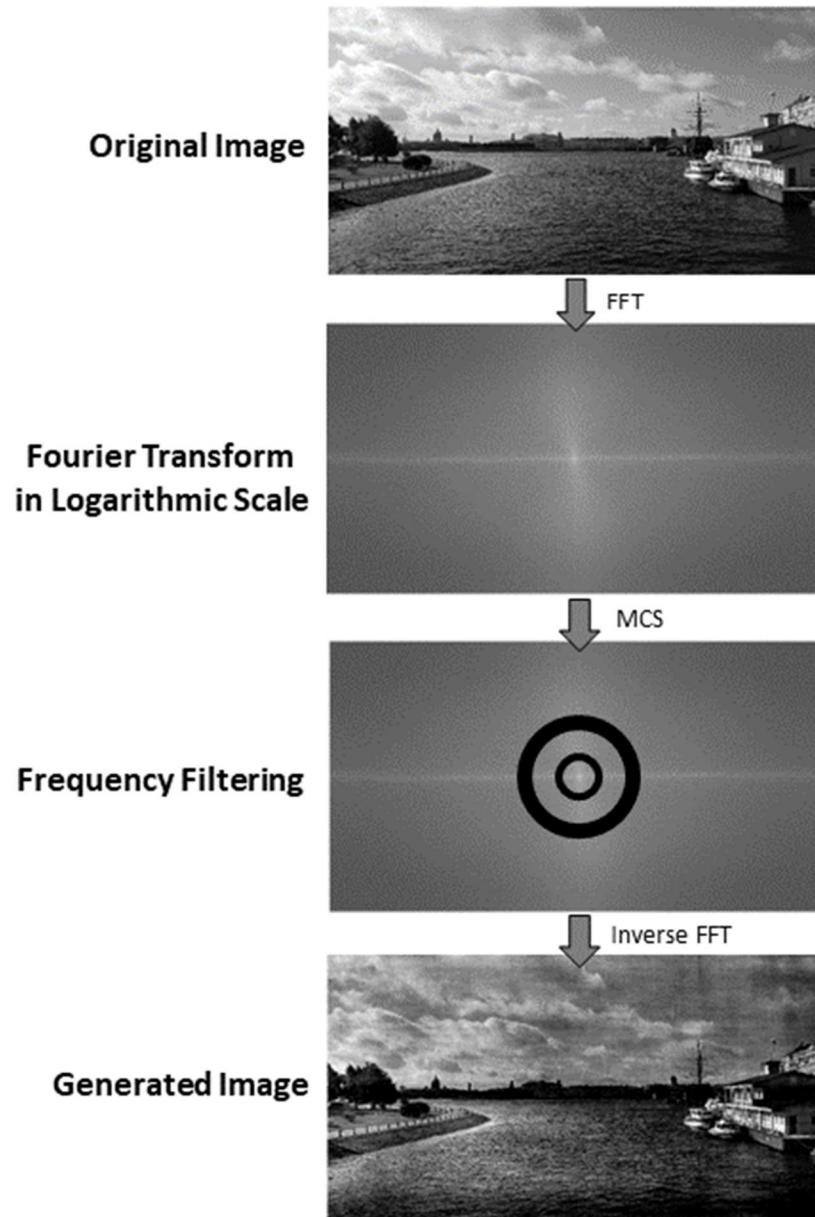


Figure 17. Image Generation using FFT and MCS

The generated image in Fig. 17 is much darker in the dark regions of the original and much brighter in the bright regions of the original. Not only this, but the generated image is also more saturated in some areas and dimmer in other areas. For the neural network, this means that the details of the generated image are different from the original,

but the information and the label classification are inherent to both. Applying this process to all of training images can staggeringly increase the size of the training data set.

### 3.3.2 Projective Transformation

After the CNN is fully trained and ready to be applied to a test image for body part detection, image processing is needed to realign the test image with the reference image. The reference image, also called the fixed image, is the original x-ray image that is taken prior to having the C-Arm relocated out of the way so that the surgeons may perform the operation. The x-ray image taken after the operation is referred to as the moving image, and it will closely resemble the fixed image save for minor translation, rotation, stretching, and tilting.

To register the moving image to the desired orientation of the fixed image, controls points from the fixed image must be paired up and matched with controls points of the moving image. When a control point in the fixed image is matched with a control point in the moving image, the location of both control points is said to be the same in a global system. Since both control points indicate the same location on both the fixed and moving images, more and more pairs of control points can be used to create a projective transformation matrix that defines the rotations, shifting, tilting, etc., relating the two images. To formulate a workable projective transformation matrix, at least four pairs of control points are required, and the more control point pairs are used, the more accurate the matching and realignment capabilities of the transformation matrix will be.

To determine suitable control point pairs between the fixed and moving image, the CNN will be run on both images to detect the bounds of the body part under investigation. The four points of the bounding box in the fixed image can then be paired and matched with the four points of the bounding box in the moving image. These points are depicted in Fig. 18 where the control point pairs are  $(A, F)$   $(B, G)$   $(C, H)$  and  $(D, E)$ . For a projective transformation, Eq. (3) and (4) are used where  $u$  and  $v$  are the x- and y-coordinates of a control point in the moving image,  $x$  and  $y$  are the x- and y-coordinates of a control point in the fixed image, and  $T_{1...8}$  are the unknown values of the projective transformation matrix ( $T_9 = 1$  for 2D transformations) [34] [35].

$$u = \frac{(T_1x + T_2y + T_3)}{(T_7x + T_8y + T_9)} \quad (3)$$

$$v = \frac{(T_4x + T_5y + T_6)}{(T_7x + T_8y + T_9)} \quad (4)$$

Both equations are multiplied by the denominator, and since  $T_9 = 1$ , they can be rewritten as Eq. (5) and (6).

$$u = [x \quad y \quad 1 \quad 0 \quad 0 \quad 0 \quad -ux \quad -uy] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} \quad (5)$$

$$v = [0 \quad 0 \quad 0 \quad x \quad y \quad 1 \quad -vx \quad -vy] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} \quad (6)$$

Equations (5) and (6) can be combined to form a linear system consisting of  $m$  number of equations. The number of equations in Eq. (7) should be equal to the number of paired control points.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_m \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -u_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -u_4x_4 & -u_4y_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_m & y_m & 1 & 0 & 0 & 0 & -u_mx_m & -u_my_m \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1x_1 & -v_1y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -v_2x_2 & -v_2y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -v_3x_3 & -v_3y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -v_4x_4 & -v_4y_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & x_m & y_m & 1 & -v_mx_m & -v_my_m \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} \quad (7)$$

Rewriting Eq. (7) in more general terms results in Eq. (8).

$$[U] = [X][T] \quad (8)$$

Solving for the  $[T]$  gives Eq. (9) which can be used to find the column vector that defines any translation, rotation, stretching, or tilting between the images.

$$[T] = [X]^{-1}[U] \quad (9)$$

This column vector is then reshaped into a  $3 \times 3$  matrix by transposing the first three, middle three, and last three (includes  $T_9$ ) rows into columns as shown in Eq. (10)

$$[T] \rightarrow \begin{bmatrix} T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 \\ T_7 & T_8 & T_9 \end{bmatrix} \quad (10)$$

Applying the inverse of the transformation matrix to the moving image, as shown in Fig. 18, will result in a registered, or realigned, moving image.

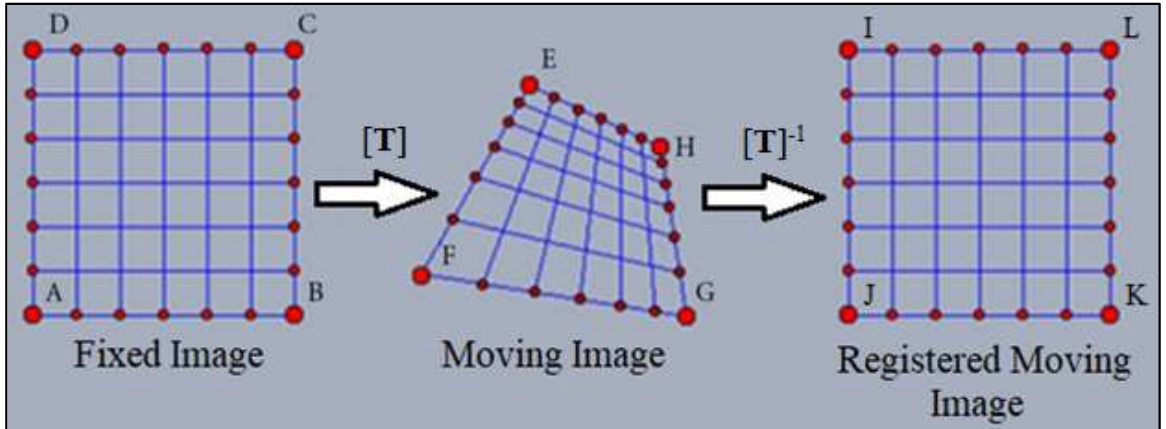


Figure 18. Projective Transformation [36]

### 3.3.3 Point Feature Matching

Figure 19 is an example of the differences between the fixed image and the moving image. The fixed image is shown in green and the moving image as violet. From the figure, it can be seen that the moving image is taken at a different camera height and camera angle when compared to the fixed image (8 DOF); this difference is a representation of the expected differences between the pre- and post-procedure x-ray images that the surgeon will encounter. A closer look at Fig. 18 reveals that a uniform white color is present at points where the moving image overlaps with the fixed image, meaning that if the moving image were to be altered so that it perfectly matches the orientation of the fixed image, the colors of the two images would overlap exactly to portray the original reference image in black and white.

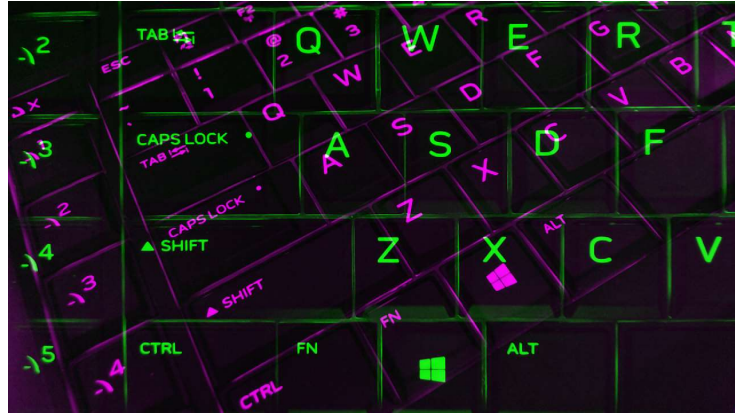


Figure 19. Overlap of Fixed Image (Green) and Moving Image (Violet)

Because the CNN is only capable of outputting four potential control point pairs, the accuracy in alignment of the projective transformation matrix tends to be lower than desired. To overcome this problem, feature points from both the fixed and moving image will be extracted and compared against each other. Using speeded up robust features (SURF), a descriptor is used to describe an image feature by using the pixel intensity distributions local to a point [37] [38]. Descriptors from different images can be compared to find matching pairs. Those feature points that have the strongest feature similarities are paralleled as control point pairs while those that lack or have minimal correlation with other points are discarded. Figure 20 is a demonstration of an additional 29 control point pairs that were formed through point feature matching.

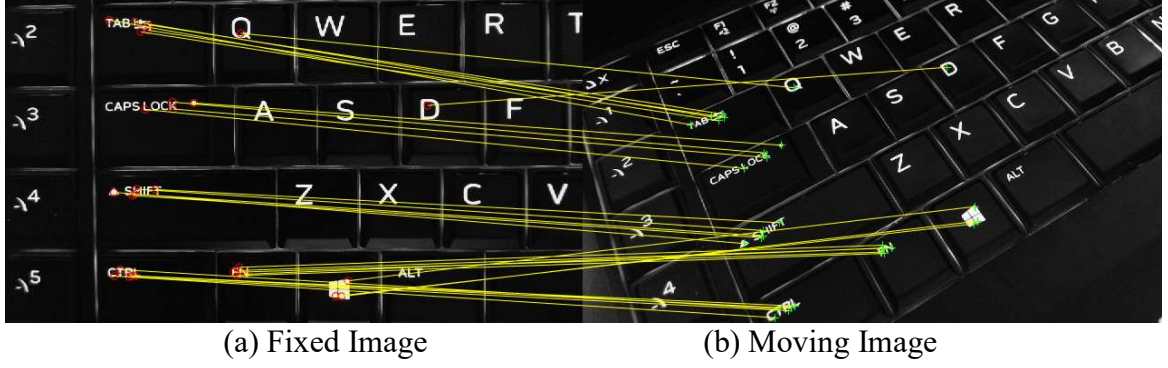


Figure 20. Multiple Control Point Pairs using SURF Point Matching

The total number of control points from Fig. 20 plus the additional control points from the CNN dictate the number of rows  $n$  to be used in Eq. (7). From this, the projective transformation matrix needed to register the images can be calculated. Figure 21 shows the miniscule error, high accuracy, and overall correctness of the realignment aptitude of the projective transformation matrix.



Figure 21. Overlap of Fixed Image (Green) and Registered Moving Image (Violet)

### 3.4 Automatic Repositioning

To automatically reposition the C-Arm, an automated C-Arm system is needed. In collaboration with the research team at the University of Tyler at Texas, the mobile CAM prototype and its kinematic model was used. During the automatic repositioning process, the kinematic model receives data from the projective transformation matrix as part of the input. The projective transformation matrix holds the necessary information required to

move the tilt and orbital rotations such that the C-Arm can be repositioned to the desired orientation. The implementation of the projective transformation matrix to the kinematic model is vital for properly orientating the C-Arm. This is because the tilt and orbital rotations allow the C-Arm to have the same focal point at an infinite number of different orientations. Figure 22 illustrates how the focal point at one orientation can occupy the same exact position in space when in another orientation. Not only is this true for the orbital rotation but also for the tilt rotation [39]. Together, the tilt and orbit make it extremely difficult for an algorithm to decide which linkages to activate and move, especially when an infinite number of possibilities exist. The realignment properties of the projective transformation matrix, however, nullifies the confusion caused by the limitless focal point positions.

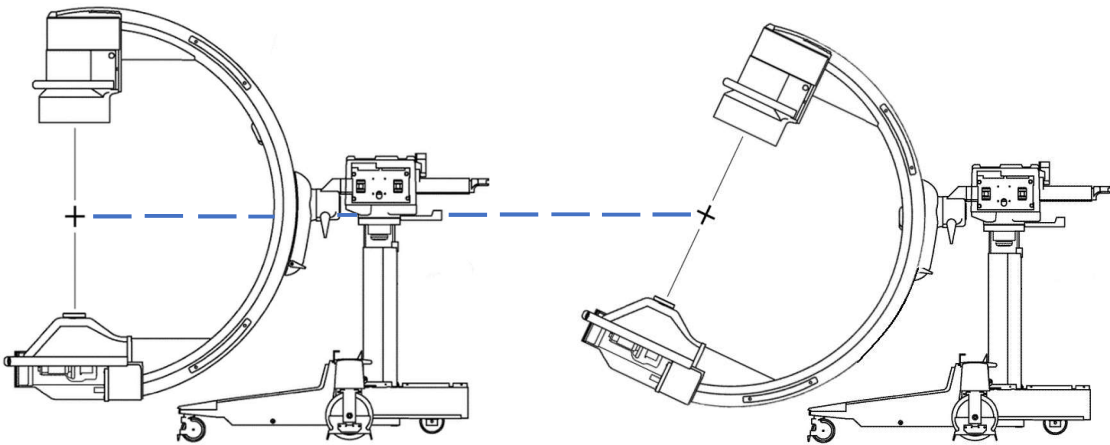


Figure 22. Identical Focal Point at Differing Orientations [40]

### 3.4.1 Camera Calibration

In order to orientate the C-Arm such that the post-procedure x-ray image is registered with the pre-procedure x-ray image, only one possibility exists, and it is stored in the projective transformation matrix. To retrieve the tilt angle of rotation and the orbital angle of rotation from the projective transformation matrix, the calibration matrix of the image intensifier electron lens system is needed. Since the electron lenses have intrinsic properties that cause distortions and aberrations between the world frame and image frame (much like how optical lenses of a camera inadvertently alter the output image slightly), the calibration parameters are needed to account for the errors introduced by the electron



lenses [41] [42]. Because the calibration matrix contains only intrinsic parameters, the calibration process must only be executed and stored once. Once found, the calibration matrix can be used for any and all future operations of that specific C-Arm. Since the CAM is only a prototype and that does not use an actual x-ray image intensifier and flat-panel, a camera is used instead as shown in Fig. 23. Although optical cameras and x-ray imaging systems use different methods of image acquisition, the principles of image distortion remains the same for both cases. Therefore, replacing the x-ray imaging system with a camera mounted to one end of the C-orbital arm should result in a fundamentally equivalent image capturing C-Arm system.

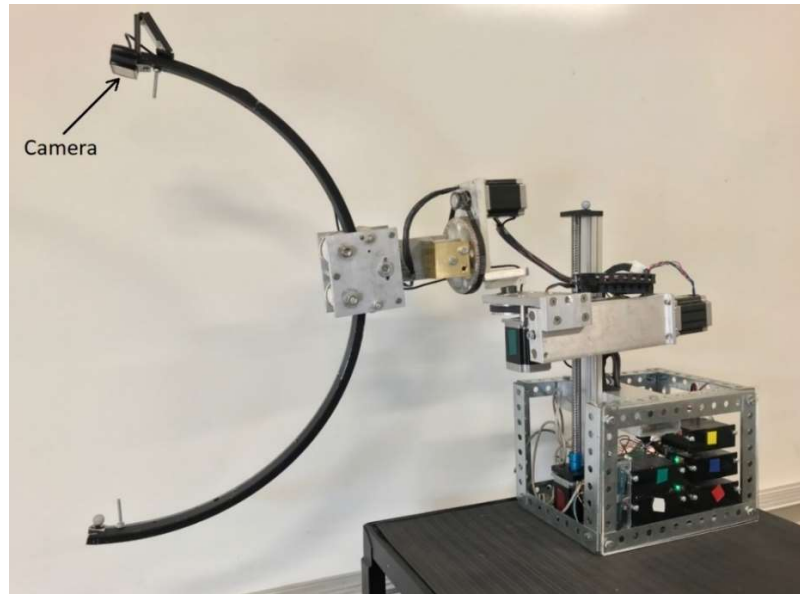


Figure 23. C-Arm Mini with Mounted Camera

With the help of the camera calibration toolbox for MATLAB, the camera was parameterized for calibration before being actually mounted to the CAM [43]. The calibration process involved the use of 10 image of a black and white checkerboard; Fig. 24 shows four examples. The squares are  $30 \times 30 \text{ cm}^2$  in size and distributed evenly across the page. Keeping the camera grounded and in a fixed orientation, images of the checkerboard were taken at different orientations. The greater the variety of orientations used, the more robust will the final calibration parameters be.



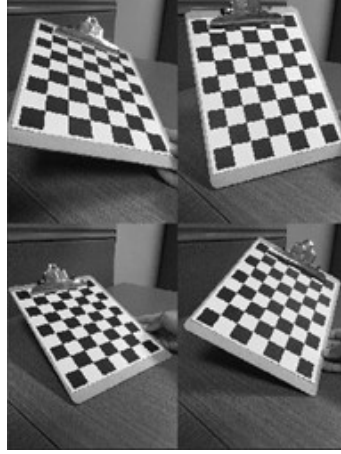


Figure 24. Calibration Images

In each calibration image, a  $5 \times 7$  rectangle encompassing 35 squares was defined. Since the size of each square is known, a grid can be formed so that all vertices of the squares may be defined by a point on the image. Figure 25 shows the grid formation process on one of the calibration images. The top left corner of the grid is used as the origin, and the remaining three corners are used to define the grid boundaries. Straight lines are used to connect the four corners and, thus, creating a rectangle in the world frame. In the image frame, however, the lines are not parallel; in fact, they form vanishing points. Using basic geometry, the angle of intersection of the gridlines can be calculated and used to determine the location of the vanishing points as well as the shape of the rectangular grid in the image frame [44].

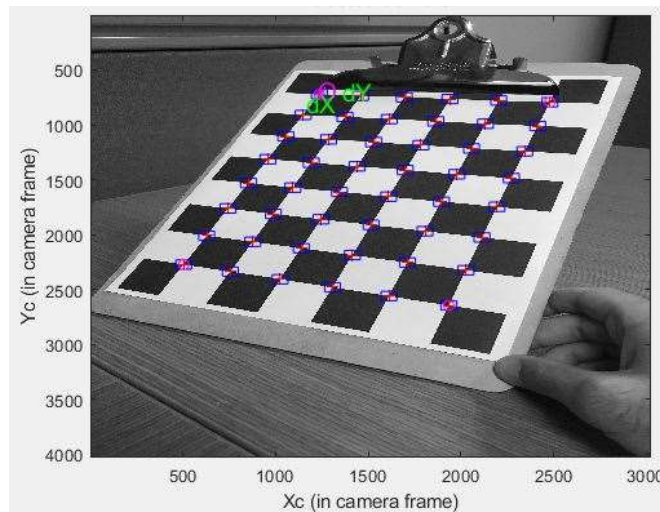
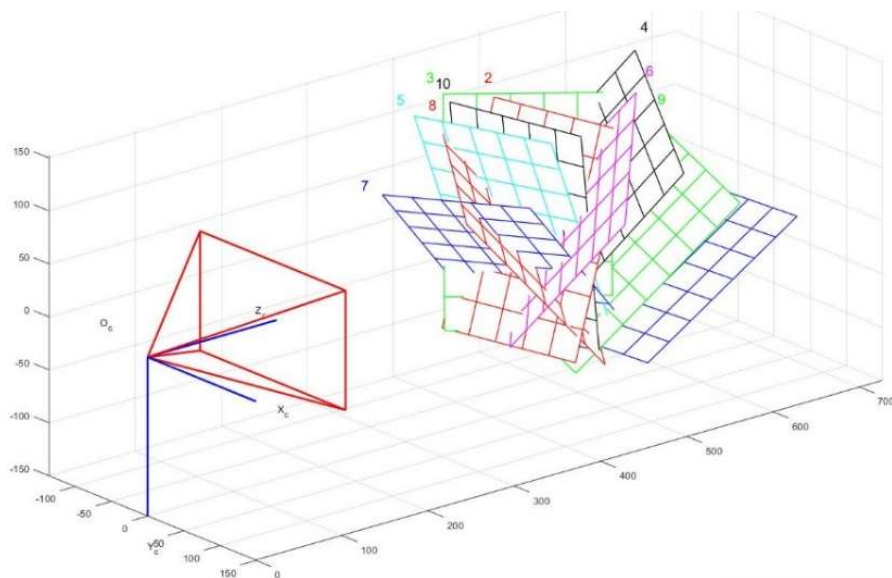
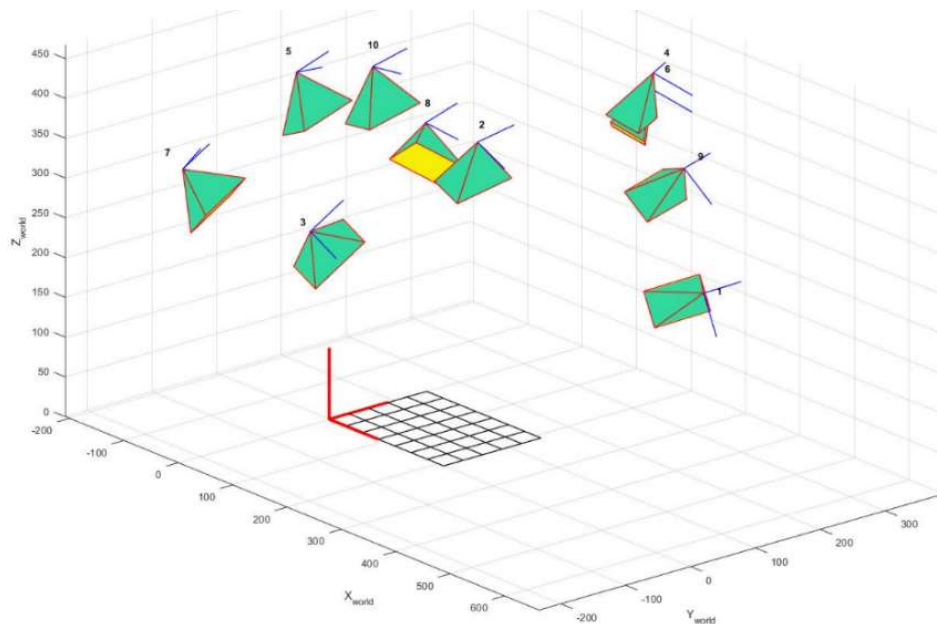


Figure 25. Corner Extraction of Checkerboard

As this process is performed over the entire calibration image data set, the intrinsic and extrinsic calibration parameters are computed by minimizing the reprojection error over all of the calibration parameters. Non-linear optimization is also done iteratively through gradient descent and by explicit computation of the Jacobian matrix [43] [45]. Figure 26 illustrates the extrinsic parameters that were used during the calibration process.



(a) Camera Centered



(b) World Centered

Figure 26. Extrinsic Parameters during Calibration Process (all units in cm)

Intrinsic calibration parameters that cause deviations between the world frame and image frame include the focal length in the x-direction,  $f_x$ , and y-direction,  $f_y$ , the principal point x-coordinate,  $p_x$ , and y-coordinate,  $p_y$ , and the skew coefficient,  $s$  [46]. Together, these parameters are responsible for the irregularities seen so often in images. Incorporating them into a single camera calibration matrix,  $[K]$ , results in Eq. (11).

$$[K] = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad (11)$$

### 3.4.2 Angle Extraction

To ensure that the projective transformation matrix gives a true and accurate mapping between the fixed and moving images, these distortion parameters must be accounted for [47]. As shown in Eq. (12), the projective transformation matrix can be multiplied by the inverse of the calibration matrix to result in a corrected projective transformation matrix (homography),  $[H]$ , that maps the world frame to the image frame [48] [49].

$$[H] = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} = [K]^{-1}[T] \quad (12)$$

The normalization factor,  $N$ , can be calculated using Eq. (13).

$$N = \sqrt{H_{11}^2 + H_{21}^2 + H_{31}^2} \quad (13)$$

The rotation column vector  $\mathbf{r}_1$  can then be found using the normalization factor and the first column of the corrected projective transformation matrix in Eq. (14).

$$\mathbf{r}_1 = \frac{1}{N} \begin{bmatrix} H_{11} \\ H_{21} \\ H_{31} \end{bmatrix} \quad (14)$$

Likewise, rotation column vector  $\mathbf{r}_2$  is found using Eq. (15).

$$\mathbf{r}_2 = \frac{1}{N} \begin{bmatrix} H_{12} \\ H_{22} \\ H_{23} \end{bmatrix} \quad (15)$$

Equation (16) finds rotation column vector  $\mathbf{r}_3$  by cross-multiplying  $\mathbf{r}_1$  and  $\mathbf{r}_2$ .

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \quad (16)$$

The results of Eq. (14-16) are then used to create rotation matrix  $[\mathbf{R}]$  as shown in Eq. (17) [50] [51].

$$[\mathbf{R}] = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3] \quad (17)$$

Finally, Eq. (18) and (19) can be used to solve for the tilt angle of rotation,  $\theta_t$ , and the orbital angle of rotation,  $\theta_o$ .

$$\theta_t = \tan^{-1} \left( \frac{R_{13}}{R_{33}} \right) \quad (18)$$

$$\theta_o = \sin^{-1}(R_{23}) \quad (19)$$

## **Chapter 4**

### **Results and Discussion**

#### **4.1 Object Detection**

As discussed in the previous sections, this study is scarce in labeled training data. However, thanks to the R-CNN method, the performance of a traditional CNN can be drastically improved, regardless of insufficient training data. R-CNN's exploit supervised pre-training and domain-specific fine-tuning to greatly improve performance [16]. Simply put, a pre-trained CNN capable of detecting basic objects with about 75% accuracy is fine-tuned and tailored towards the interests of this study. The pre-trained CNN can distinguish lines, edges, surfaces, and regions but lacks the proper training for complete and accurate classification. A neural network at such a stage is generalized and can function as a foundation for more specific object detection through fine-tuning. The fine-tuning is conducted on the pre-trained CNN by simply introducing it to what little training data exists for bone structure detection in x-ray images. In this manner, a fully trained CNN is developed for detection of bone in orthopedic procedures without the need of extravagantly large training data sets. The complete architecture of the neural network is portrayed in Fig. 27. The x-ray images are read in by the image input layer. The middle layers then pass the image data through three stages of convolution, ReLU, and max pooling. Substantial downsizing of the image data will have occurred at this point. In the final layers, two fully-connected layers, one ReLU layer, and a softmax layer are used to form neurological connections in the hidden layers. The softmax function was adopted because of its ideal multi-classification in the logistic regression model and for the different layers of neural networks [52]. The detection and classification results are then displayed by the classification output layer.

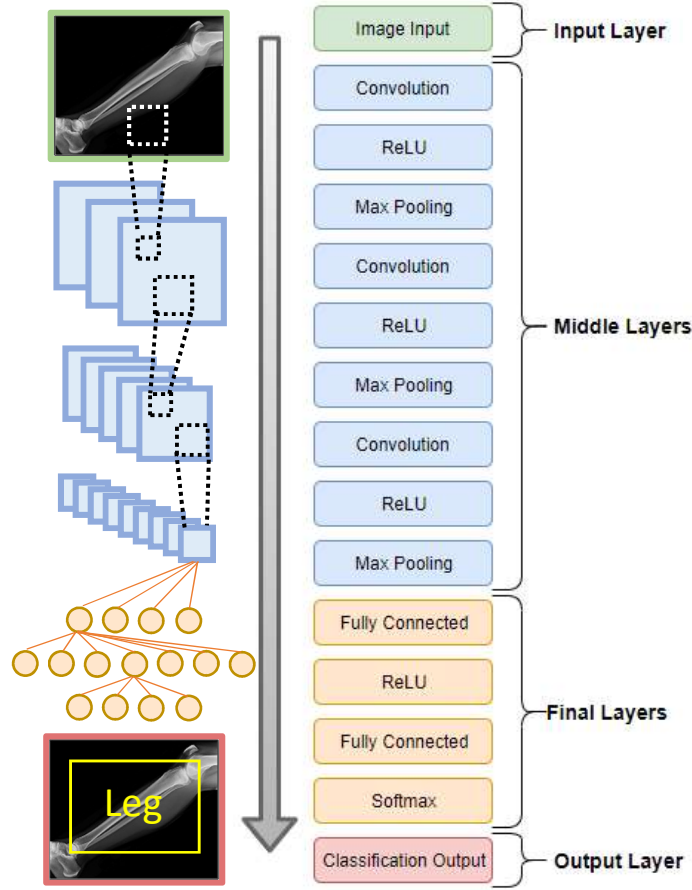


Figure 27. R-CNN Architecture

The pre-trained CNN used 10,000 images sized at  $32 \times 32$  pixels that were labeled and placed into ten different categories such as dogs, cats, ships, trucks, etc. [23]. To fine-tune the CNN specifically for the detection of bone structures, 260 x-ray images depicting the chest, neck, and legs were labeled and comprised as the training data set. This is the entirety of the training data set without the implementation of image generation. Out of the 260 x-ray images, 143 were of the category “chest”, 58 were “neck”, 45 were “leg”, and 14 were combinations of the three categories (more than one body part was depicted in the x-ray image e.g. chest and neck). The size and number of color channels of the x-ray images were used as the image input layer parameters. For the convolutional layers, a total of 32 filters were used except for the third and final convolutional layer which used 64 filters. Each of the filters was sized to be a  $5 \times 5$  matrix, and a padding space of 2 pixels was used to prevent the filter from overlapping past the image borders. This will cause a 2-pixel wide frame to be excluded from the convolutional process.

Because areas of interest in x-ray images rarely lie along image borders, the exclusion of the frame will seldomly impact the outcome. Even if the area of interest did fall somewhere on an image border, the small width of 2 pixels will have an insignificant effect on the accuracy. For the max pooling layer parameters, a  $2 \times 2$  sized window at a stride of 2 pixels was used. After passing through the fully-connected layers, the final output is a  $4 \times 64$  weight function and a bias vector of 4 units. Table 1 details the learning process of the R-CNN as it progresses through 10 epochs without the incorporation of any generated images in the training data set.

Table 1. R-CNN Learning Progression without Image Generation

<i>Epoch</i>	<i>Time Elapsed (hh:mm:ss)</i>	<i>Mini-Batch Accuracy</i>	<i>Mini-Batch Loss</i>	<i>Base Learning Rate</i>
<b>1</b>	00:00:00	14.84%	1.4338	0.0010
<b>1</b>	00:00:22	92.18%	0.2992	0.0010
<b>2</b>	00:00:46	92.97%	0.2359	0.0010
<b>3</b>	00:01:09	96.09%	0.1335	0.0010
<b>4</b>	00:01:32	99.22%	0.0273	0.0010
<b>5</b>	00:01:57	98.44%	0.0678	0.0010
<b>6</b>	00:02:22	99.22%	0.0610	0.0001
<b>7</b>	00:02:48	99.22%	0.0278	0.0001
<b>8</b>	00:03:13	100.00%	0.0156	0.0001
<b>9</b>	00:03:39	100.00%	0.0096	0.0001
<b>10</b>	00:04:05	99.22%	0.0202	0.0001
<b>10</b>	00:04:21	100.00%	0.0091	0.0001

The test data set consisted of 200 images. From these, 80 depicted the chest, 60 depicted the leg, and 60 depicted the neck. The test images were not included in the training data set, making it the CNN’s first exposure to those x-ray images. To determine the accuracy of the R-CNN, the network was run on all images of the test data set. Overall, the R-CNN correctly classified 85% of the test images with their true category. Even though a R-CNN was used, the small size of the training data set prevented the neural network from properly categorizing a tenth of the test images. Incorporating a larger variety of images in the training data set will improve these results. A summary of the classification test results is listed in the confusion matrix shown in Table 2.

Table 2. Object Classification without Image Generation

<i><b>Ground Truth</b></i>	<i><b>Prediction</b></i>		
	Chest	Neck	Leg
Chest	86.75%	8.33%	11.67%
Neck	6.25%	85.00%	8.33%
Leg	7.50%	6.67%	80.00%

After recording the results shown in Table 2, the R-CNN was reset with the same parameters but with the inclusion of 350 additional x-ray images into the training data set courtesy of the FFT and MCS image generation process. Out of the 350 added x-ray images, 50 were of the category “chest”, 150 were “neck”, and 150 were “leg”. Substantially more neck and leg x-ray images were generated to compensate for the lack of images in those categories. Figure 28 reveals a small sample of the 350 x-ray images that were generated using FFT and MCS. Although the differences between the x-ray images are difficult to see, all of them are distinguishable. The pixels that make up each of these images are distributed differently along the rows and columns of the images, and the pixel intensity at a certain point on a generated image is likely to be different than that of another generated image. Considering that these slight differences exist over the entirety of the x-ray images, all of the generated images are unique in their own. In other words, a simple computer cannot make correlations between any two generated images at face-



value. However, a more sophisticated algorithm is still capable of detecting features and general trends depicted in the images. Thus, the R-CNN is still capable of making out what each x-ray image means to portray.



Figure 28. Image Generation Sample

With a total of 610 x-ray images in the training data set, the R-CNN was re-trained from scratch. Table 3 lists the training progression of the R-CNN with the updates training data set size.

Table 3. R-CNN Learning Progression with Image Generation

<i>Epoch</i>	<i>Time Elapsed (hh:mm:ss)</i>	<i>Mini-Batch Accuracy</i>	<i>Mini-Batch Loss</i>	<i>Base Learning Rate</i>
<b>1</b>	00:00:00	43.75%	1.3065	0.0010
<b>1</b>	00:00:20	91.41%	0.2271	0.0010
<b>1</b>	00:00:41	99.22%	0.0368	0.0010
<b>2</b>	00:01:02	98.44%	0.0586	0.0010
<b>2</b>	00:01:22	98.44%	0.0558	0.0010
<b>3</b>	00:01:44	97.66%	0.0439	0.0010
<b>3</b>	00:02:05	100.00%	0.0077	0.0010
<b>3</b>	00:02:26	99.22%	0.0198	0.0010
<b>4</b>	00:02:47	100.00%	0.0119	0.0010
<b>4</b>	00:03:08	96.88%	0.0992	0.0010
<b>5</b>	00:03:30	98.44%	0.0568	0.0010
<b>5</b>	00:03:51	100.00%	0.0098	0.0010
<b>6</b>	00:04:13	99.22%	0.0439	0.0001
<b>6</b>	00:04:35	99.22%	0.0182	0.0001
<b>6</b>	00:05:01	99.22%	0.0084	0.0001
<b>7</b>	00:05:23	100.00%	0.0080	0.0001
<b>7</b>	00:05:45	99.22%	0.0164	0.0001
<b>8</b>	00:06:05	100.00%	0.0014	0.0001

<b>8</b>	00:06:25	100.00%	0.0015	0.0001
<b>9</b>	00:06:46	100.00%	0.0108	0.0001
<b>9</b>	00:07:07	99.22%	0.0123	0.0001
<b>9</b>	00:07:29	100.00%	0.0018	0.0001
<b>10</b>	00:07:49	99.22%	0.0210	0.0001
<b>10</b>	00:08:11	100.00%	0.0059	0.0001
<b>10</b>	00:08:23	100.00%	0.0020	0.0001

After training with the newly generated x-ray images, the R-CNN was applied to the same 200 test x-ray images. Table 4 accentuates the improvements that the generated x-ray images have made in predicting the correct category. Overall, the implementation of image generation using FFT MCS improved the detection accuracy in all three categories, indicating that the added images improved the quality of the learning process.

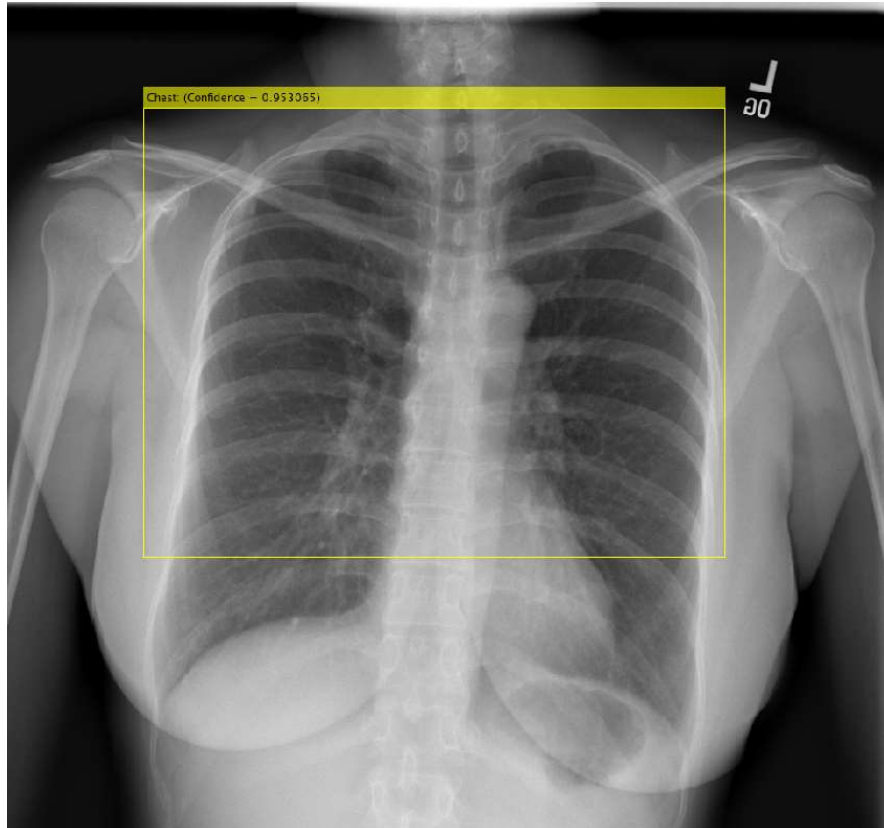
Table 4. Object Classification with Image Generation

<b><i>Ground Truth</i></b>	<b><i>Prediction</i></b>		
	Chest	Neck	Leg
Chest	100%	1.67%	3.33%
Neck	0%	95.00%	6.67%
Leg	0%	3.33%	91.67%
<b><i>Improvement</i></b>	+13.75%	+10.00%	+11.67%

Using the same 200 test images, the prediction of the chest, neck, and leg improved by about 13.75%, 10.00%, and 11.67%, respectively. Due to the close similarity between the chest x-ray images, the newly improved R-CNN was able to detect all of the chest x-rays without any issues. The high accuracy of the chest detection can also be attributed to the larger chest x-ray samples in the training data set. The neck and leg prediction

categories also improved thanks to the revamped R-CNN; however, a few mis-categorizations still remained. Although they have been greatly diminished, these persisting errors indicate that the R-CNN struggles to decipher neck and leg x-ray images. This is attributed to the fact that a much greater variety of neck and leg x-rays were used. For example, uncertainty was introduced to the learning process when both the sagittal and coronal views of the neck were used. Some of the leg x-ray images included the knee whereas others included parts of the foot. These inconsistencies in the training data set makes it harder for the R-CNN to extract desired information.

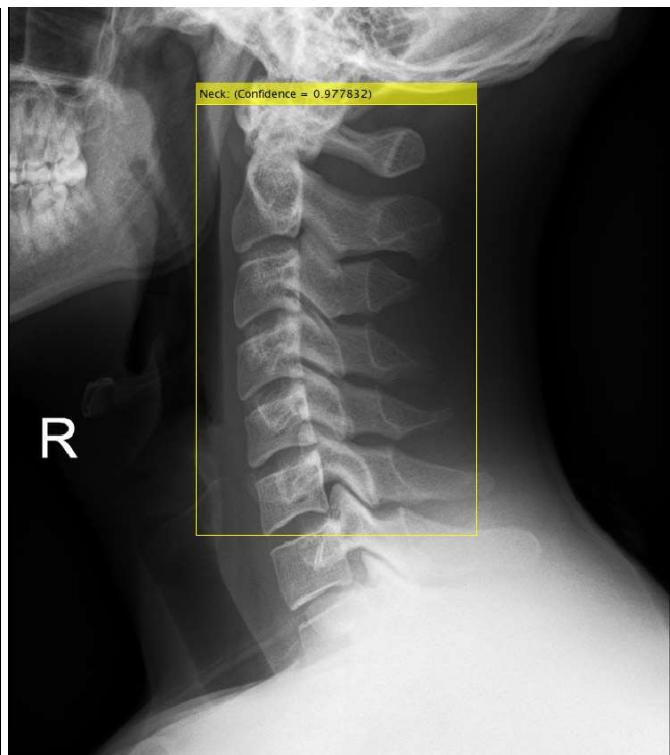
Although the accuracy of the deep learning neural network was improved, the 350 additional training images increased the overall processing time of the learning progression. By comparing the elapsed time between the training progression in Table 1 and Table 3, it can be seen that the more accurate neural network requires nearly twice the processing time. As is expected with most computationally extensive processes, an increase in a system's accuracy and reliability takes away from its speediness. However, once fully trained, the new and improved R-CNN is capable of detecting and classifying objects within an x-ray image as quickly and as swiftly as the initial R-CNN. Since the training of a neural network is only ever required once for the application of the C-Arm, the increase in processing time as a trade-off for higher detection accuracy is considered favorable. Figure 29 illustrates some of the successful detections made by the R-CNN after having incorporated the generated images using FFT and MCS.



(a)



(b)



(c)

Figure 29. Object Detection of (a) Chest, (b) Leg, and (c) Neck

In Fig. 29 (a), a chest x-ray image is shown. The newly trained CNN correctly determined that the bodily structure of interest depicted in the image was a chest. The chest itself is located and encompassed by the yellow bounding box to indicate its location and to differentiate it from the rest of the image. The confidence with which the neural network made the detection of the chest in Fig. 29 (a) was calculated by the algorithm to be about 95.3%. In Fig. 29 (b), the x-ray image of a leg was detected with a confidence of 99.5%. In Fig. 29 (c), the neck was detected and classified with 97.8% confidence.

## 4.2 Image Registration

In Fig. 30, the moving image is rotated clockwise on the image plane and tilted such that two vanishing points are present. A total of 15 control point pairs was used in formulating the projective transformation matrix. Out of the 15 points, 4 were drawn from the trained R-CNN, and the remaining 11 points were established using point feature matching.

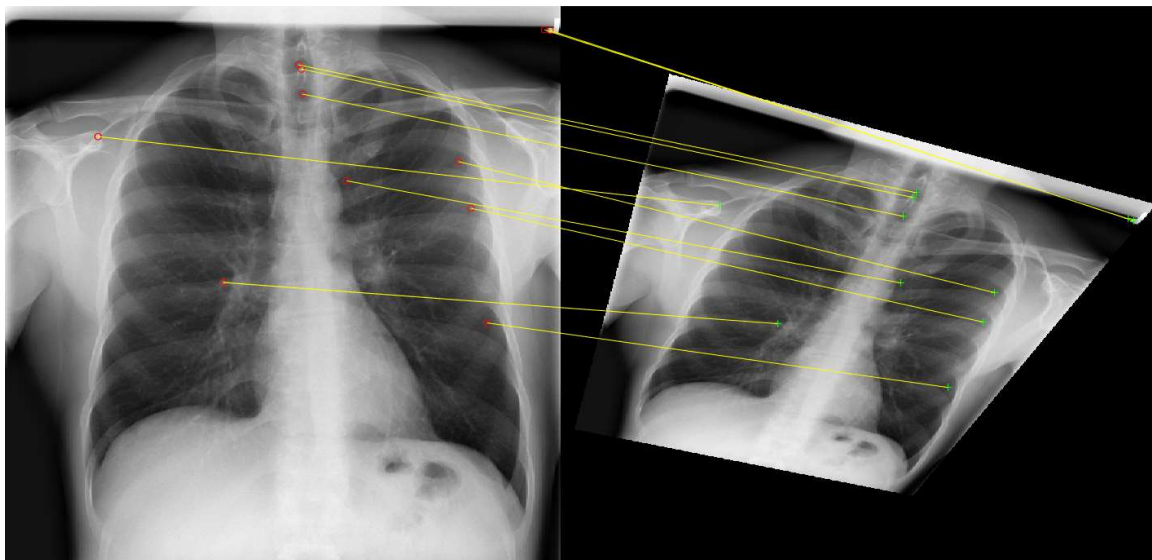


Figure 30. Control Point Pairs (11) using Point Feature Matching

Figure 31 compares the fixed and moving images of two x-ray images before and after having applied the projective transformation matrix. Applying the inverse of the projective transformation matrix resulted in Fig. 31 (b). From the comparison made between the fixed image and the registered (re-orientated) moving image, it can be seen

that the projective transformation matrix was able to align the moving image with the fixed image nearly perfectly, as made evident by the overwhelmingly white colors and lack of both green and violet. Nevertheless, some slight differences between the two exist. The error can be traced back to the accuracy of the R-CNN and the bounding boxes that it forms. Some error also exists in the fact that not all control point pairs from the point feature matching method are exact matches.

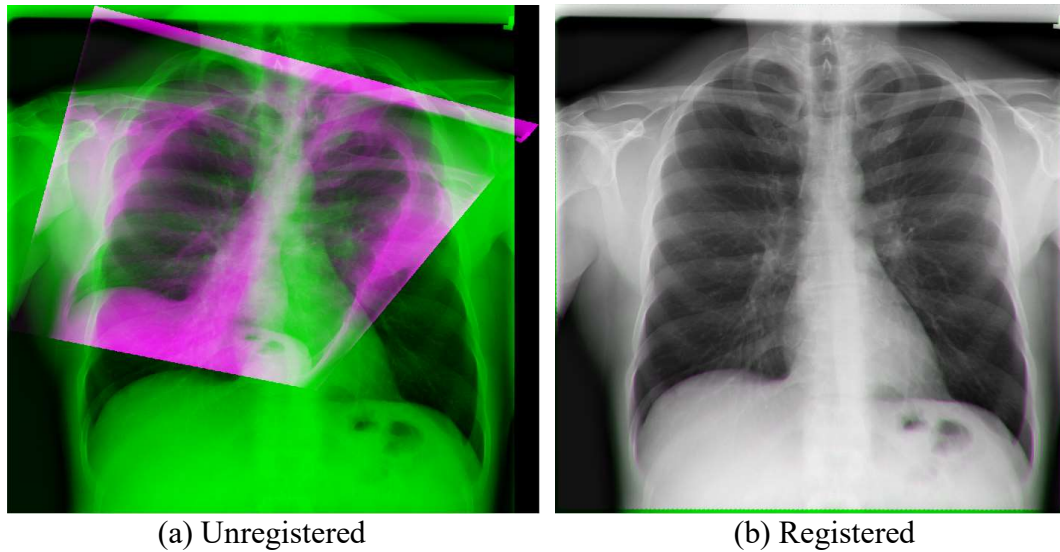


Figure 31. Registration Attempt including Point Feature Matching

### 4.3 Experimental Validation and Verification

As part of the validation and verification process for the automatic repositioning of the C-Arm, the CAM was fully equipped and placed in a laboratory environment for experimentation and testing. Since the CAM is only a prototype model, the x-ray image capturing system is represented using a camera. By replacing the image intensifier and flat-panel detector with a single camera mounted to one end of the C-orbital, an equivalent image capturing system is developed. Images from the camera may then be treated as x-ray images that the CAM could potentially generate.

To perform the experiment, a scenario was prepared to mimic actual C-Arm operation during an orthopedic procedure in which the lower cervical vertebrae in the neck is fractured. In the setup, the CAM and its mounted camera were placed by a flatbed on which a neck phantom lies. A phantom is simply a pictorial representation of an orthopedic

structure for simulation of a clinically realistic scenario. In this case, an x-ray image of a neck is the phantom. The CAM was positioned so that a clear image of the neck can be taken; this image will be the pre-procedural reference/fixed image. The experimental setup is shown in Fig. 32.

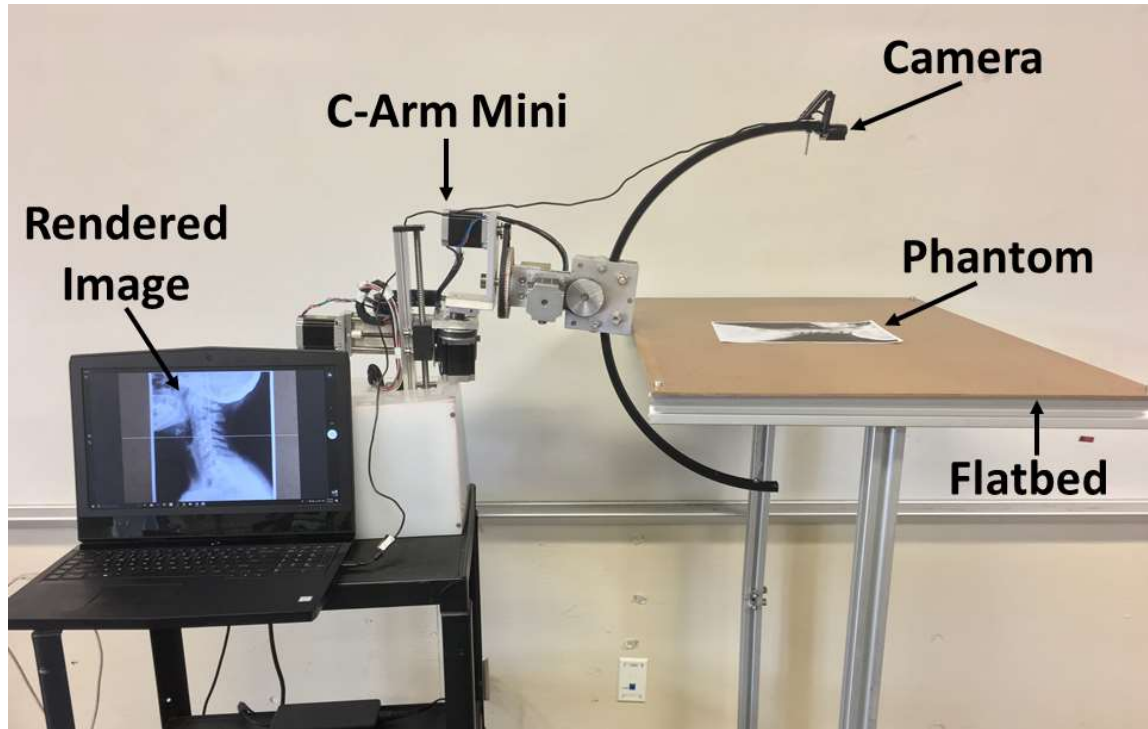


Figure 32. Experimental Setup

The CAM's tilt and orbital angular positions were then changed arbitrarily (but within reasonable amount) by  $5^\circ$  and  $8^\circ$ , respectively, to introduce controlled variation to the post-procedure test/moving image. A post-procedural image was taken and then compared to the pre-procedural image, as exemplified in Fig. 33. At this stage, the procedures and steps outlined in Fig. 4 were initiated.





(a) Before Moving the Tilt and Orbital Linkages



(b) After Moving the Tilt and Orbital Linkages by  $5^\circ$  and  $8^\circ$ , respectively.

Figure 33. Comparison of Experimental Pre- and Post-Procedural Images of a Neck Phantom

Figure 34 is the output of the R-CNN using the image from Fig. 33 (a). The neck phantom was successfully detected with 82.14% confidence. The yellow heading detailing the classification and confidence results was enlarged for better visualization. The bounding box outlining the region of detection fully encompassed the neck phantom. This ensures that the control points are drawn within close proximity to the neck phantom.

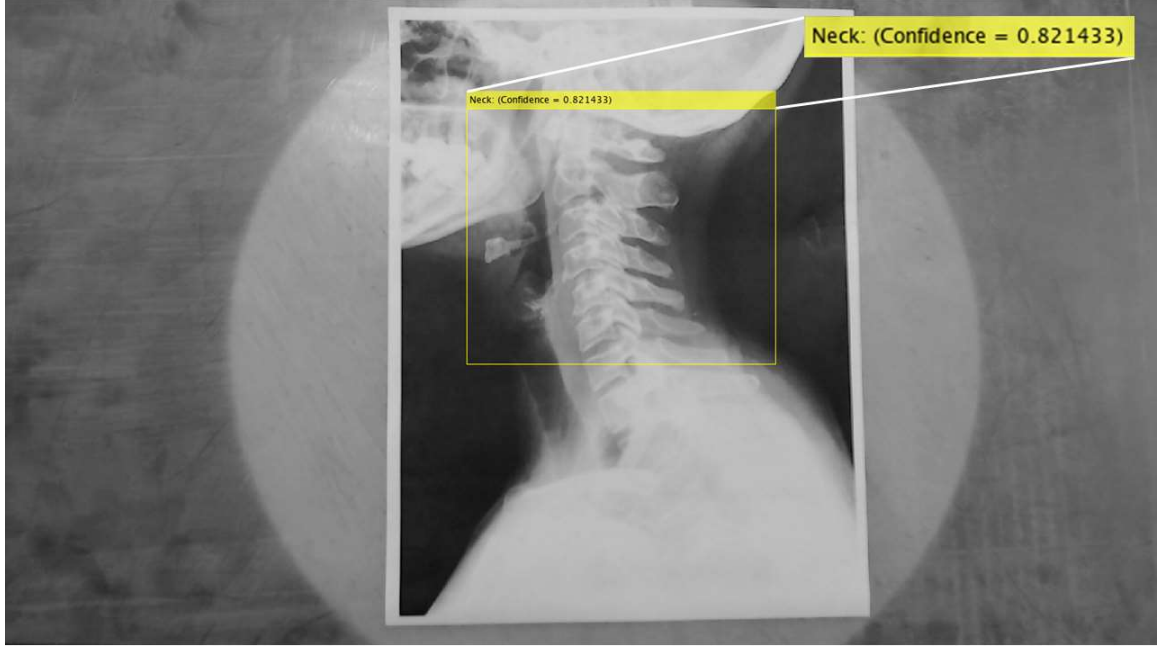


Figure 34. Detection of the Neck Phantom

After extracting the projective transformation matrix from the deep learning network and point feature matching system, the camera calibration matrix of the mounted camera was found using MATLAB's camera calibration toolbox [43]. With the camera calibration matrix and the projective transformation matrix, Eq. (11-19) were used to find the rotation matrix and the tilt and orbital angles of rotation. Tables 5-7 list the results of the first part of the experiment using the images in Fig. 33. In Table 5, the resulting projective transformation, camera calibration parameters, and rotation matrix are shown.

Table 5. Experimental Results – Matrices

<i>Matrix</i>	<i>Symbol</i>	<i>Values</i>
Projective Transformation	$[T]$	$\begin{bmatrix} 1.238 & 0.065 & 7.96 \times 10^{-5} \\ 0.099 & 1.230 & 1.166 \times 10^{-4} \\ -200.8 & -175.2 & 1 \end{bmatrix}$
Camera Calibration	$[K]$	$\begin{bmatrix} 1431 & -3.287 & 1010 \\ 0 & 1434 & 515.8 \\ 0 & 0 & 1 \end{bmatrix}$
Rotation	$[R]$	$\begin{bmatrix} 0.989 & 0.036 & 0.147 \\ -0.019 & 0.987 & 0.058 \\ -0.148 & -0.064 & 0.977 \end{bmatrix}$

In Table 6, the angles of rotation needed for automatic repositioning of the CAM are investigated. Since the tilt and orbital linkages were rotated by  $5^\circ$  and  $8^\circ$ , the theoretical angles needed to properly reposition the CAM back to its original orientation are  $-5^\circ$  and  $-8^\circ$ . The actual values that the CAM rotated its tilt and orbital linkages during the automatic repositioning process are listed in Table 6 as well. The difference between the actual values and the expected values are also shown. When compared to the initial angles, both the tilt and orbital angles of rotation were off from the expected value within less than half of a degree.

Table 6. Experimental Results – Angles

<i>Angle of Rotation</i>	<i>Symbol</i>	<i>Actual</i>	<i>Theoretical</i>	<i>% Difference</i>
Tilt	$\theta_t$	$-5.4465^\circ$	$-5.00^\circ$	8.936%
Orbital	$\theta_o$	$-8.2802^\circ$	$-8.00^\circ$	3.503%

The spatial position of the CAM was also considered during the automatic repositioning process of the experiment. The forward kinematic model was used to triangulate the instantaneous position of the camera. The focal point of the C-Orbital does not change during the movement of the tilt and orbital rotations, making it unsuitable for validation through spatial coordinates. For this reason, the camera position was deemed more appropriate for the purpose of this experiment. Table 7 details the position of the camera in a global coordinate system. The CAM starts at the origin and rotates  $5^\circ$  and  $8^\circ$  in tilt and orbital movement to the offset position. The coordinate point of the offset position is recorded. Using the actual angles from Table 6, the CAM is automatically repositioned back to the origin. The results indicate that the angles sent to the tilt and orbital linkages repositioned the CAM back to the origin with acute precision and high accuracy. To be more specific, the  $x$ - and  $z$ - coordinates fell exactly back on the origin (within tolerance), and the  $y$ -coordinate was off just short of a tenth of an inch.

Table 7. Experimental Results – Positions

<i><b>Position</b></i>	<i><b>Spatial Coordinates</b></i>		
	<i><b>x</b></i>	<i><b>y</b></i>	<i><b>z</b></i>
Starting (origin)	0.00 in	0.00 in	0.00 in
Offset	1.029 in	1.64 in	−0.16 in
Corrected (back to origin)	0.00 in	−0.08 in	0.00 in

For the second part of the experimental validation and verification process, a third and final validation image of the neck phantom was taken after the CAM had repositioned itself back to the desired position and orientation. Figure 35 is an overlap of Fig. 33 (a) with the final validation image of the neck phantom after having repositioned the C-Arm back to the desired orientation. Visual inspection of this image accentuates the success of the proposed automatic repositioning method. Save for a few minor distinctions, the final validation image is virtually indistinguishable from the starting position depicted in Fig. 33 (a). It is important to note that, although the two images look as if they are the same image, they are both in fact completely separate and distinct images taken at different instances during the repositioning validation process; all similarities are simply a result of the accurate repositioning of the C-Arm.

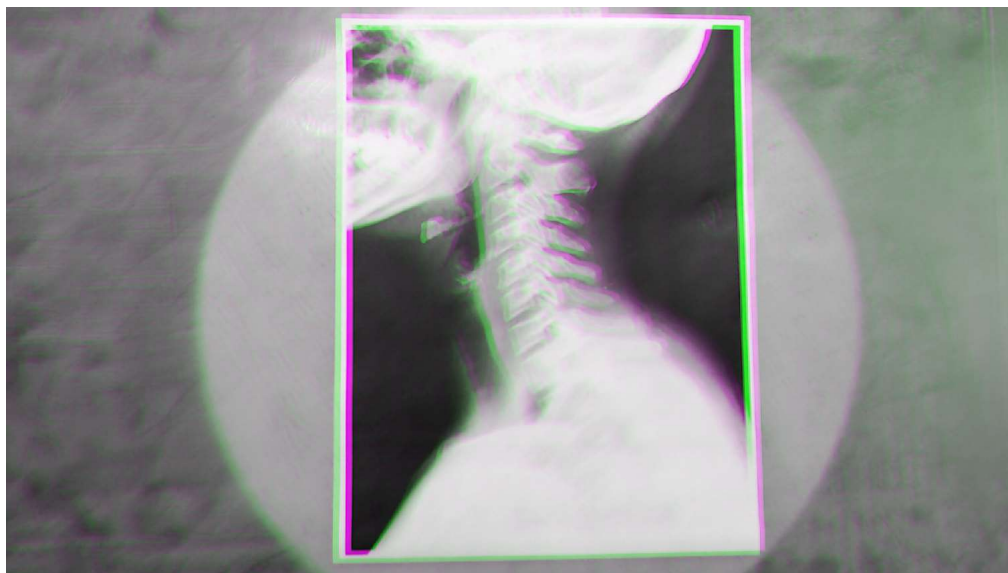


Figure 35. Repositioning Validation through Inspection of Image Realignment

## Chapter 5

### Conclusion

#### 5.1 Summary

In this thesis, a vigorous method for automatic repositioning of the C-Arm was established using deep learning and image processing. Background on the C-Arm was given, and a thorough investigation of the C-Arm in the current literature was provided. The literature survey explored the latest advancements of C-Arm repositioning amelioration with respect to artificial intelligence and robot vision. More specifically, the integration of CNNs and image processing with C-Arms were reviewed. The existent collaboration with other researchers and their development of a C-Arm Mini prototype for experimental purposes was also noted.

Detailed explanation behind the theoretical workings of the proposed automatic repositioning method was instated. A deep learning CNN was trained to detect bone structures in x-ray images during orthopedic surgery. Due to limited resources, a pre-trained R-CNN was fine-tuned for the detection and classification of three distinct body parts: chest, neck, and leg. Not only did the R-CNN improve the processing time of the training functions, but it also allowed the use of much smaller training data sets by reducing the application of feature filters only to specifically proposed regions. Image generation with FFT and MCS was another key component attributed to the success of this research. By filtering and processing training images for generation, the overall dependability of the neural network was drastically improved. The points defining the region of detection were then coupled with feature matching points to create a projective transformation matrix. Because of their contribution to the distortion of x-ray images, the importance of electron lens, or optic lens, camera calibration parameters were introduced. Accounting for the distortions caused by these lens parameters, the rotation matrix and, thus, the tilt and orbital angles of rotation of the C-Arm could be calculated from the corrected homography. These angles can be used to automatically reposition a mobile C-Arm to the desired orientation when fluoroscopic image realignment is required by the surgeon. Key results indicate that the proposed method makes for satisfactory detection of orthopedic structures and proper registration of fluoroscopic images.

An experiment was conducted with the C-Arm Mini prototype to validate the results. A mock surgery was constructed to observe the real-life application of the proposed methods in practice. The projective transformation matrix, tilt and orbital angles of rotation, and spatial positions were recorded during the automatic repositioning process. Based on the miniscule angle difference error and spatial deviation error, the mockup was deemed a success. A final visual comparison between the pre- and registered post-procedural x-ray image of the neck phantom further illustrated the validity of the reposition.

## **5.2 Outcome**

Ultimately, an integrated approach was developed to effectively better the fine auto movements during the repositioning of the C-Arm using deep learning CNNs and image processing. The proposed method also rectifies the kinematic-based issue of having an infinite number of possible C-Arm orientations for a given focal point down to a single possible orientation through projective transformations and angle transformations within reference x-ray images. Neither professional experience nor detailed calculations are required in the repositioning process.

## **5.3 Implications**

The C-Arm repositioning method outlined in this study can be used to reposition the C-Arm back to its original position with respect to the patient in an accurate and efficient fashion. This allows surgeons to assess and validate surgical procedures in real time with little to no realignment uncertainties. Additionally, the repetitive process of having to take and retake multiple x-ray images for comparison purposes is eliminated. As a result of this, the patient, surgeon, and surgical staff are no longer in jeopardy of unwarranted radiation exposure. Furthermore, the operation time for surgical procedures requiring multiple x-ray images and constant C-Arm movements would be greatly reduced. Since time consuming surgeries tend to be more expensive, the faster operation time consequently reduces the overall cost of operation. Reliable automatic repositioning of the C-Arm system also obviates the need for a C-Arm technician. Although the C-Arm will not be completely self-operating and will always require supervision to some degree, a

highly skilled technician is no longer required. As a final implication, the reduced surgery time minimizes the patient's exposure to the environment, lowering the risk of infection.

## **5.4 Future Work**

A few aspects of this study could be improved as part of the future work. For one, the size of the training and testing data sets of the deep learning network can be increased. This will not only improve the detection and classification of the CNN, but it will also improve the accuracy of the region of detection and bounding box dimensions. As a result, the overall workings and precision of the automatic repositioning will be enhanced. Expanding the number of classification categories to include other body parts would also improve the robustness of the deep learning network. Additional orthopedic structures may include the spine, the hip, or any other area of the body requiring extensive surgical care and C-Arm precision.

Another interesting facet to consider for improving the repositioning accuracy of the C-Arm entails the use of control feature points only within the region of detection of an x-ray image. Although this would reduce the total number of control points used, the quality of the points would be increased since points not of particular interest (i.e. points lying outside of the surgical area) would not be included. Doing so may improve the realignment process and worsen image registration or vice versa. Further testing is needed to assess these assumptions.

To further increase the credibility of this study, an experiment can be conducted in which an actual x-ray image intensifier and flat-panel detector are used instead of a camera. Rather than calibrating the optic lens camera on the C-Arm Mini prototype, the calibration process will be performed on the x-ray capturing system and its electron lenses of an industrial mobile C-Arm. The resulting calibration matrix can then be used to correct the projective transformation matrix between two actual x-ray images. The outcome of this experiment would validate the automatic repositioning of real C-Arms currently used in the medical industry.

## References

- [1] M. Herzmann, "What is a Mobile C-Arm," Ziehm Imaging, 2006. [Accessed Dec. 23, 2018].
- [2] H. Esfandiari, "Photogrammetric Advances to C-Arm Use in Surgery," Master of Science in Geomatics Engineering, University of Calgary, Calgary, Alberta, 2014. [Accessed Dec. 23, 2018].
- [3] Y. Lee, H. Lee, J. Cho and H. Kim, "Analysis of Radiation Risk to Patients from Intraoperative use of the Mobile X-Ray System (C-Arm)," *Journal of Research in Medical Science*, vol. 20, pp. 7-12, 2015.
- [4] M. Unberath, J. Fotouhi, J. Hajek, A. Maier, G. Osgood, R. Taylor, M. Armand and N. Navab, "Augmented Reality-Based Feedback for Technician-in-the-Loop C-Arm Repositioning," *IEEE, Helthcase Technology Letters*, vol. 5, no. 5, p. 143, 2018.
- [5] X. Chen, H. Naik, L. Wang, N. Navab and P. Fallavollita, "Video-Guided Calibration of an Augmented Reality Mobile C-Arm," *International Journal of Computer Assisted Radiology and Surgery*, vol. 9, no. 6, pp. 987-96, 2014.
- [6] G. Dagnino, I. Georgilas, S. Morad, P. Gibbons, P. Tarassoli, R. Atkins and S. Dogramadzi, "Intra-Operative Fiducial-Based CT/Fluoroscope Image Registration Framework for Image-Guided Robot-Assisted Joint Fracture Surgery," *International Journal of Comptuer Assisted Radiology and Surgery*, vol. 12, no. 8, pp. 1383-97, 2017.
- [7] N. Binder, L. Matthaus, R. Burgkart and A. Schweikard, "A Robotic C-Arm Fluoroscope," *International Journal of Medical Robotics and Computer Assisted Surgery*, pp. 108-116, 2005.
- [8] N. Suhm, "Method of Automatic Guiding a C-Arm X-ray Device". United States Patent 6,491,429 B1, 10 December 2002.
- [9] S. Froehlich, C. Schlossbauer and A. Blumhofer, "Exact Patient Positioning by Comparing Reconstructed X-ray Images and Linac X-ray Images". United States Patent 6,516,046 B1, 4 February 2003.
- [10] N. Navab, S. Heining and J. Traub, "Camera Augmented Mobile C-Arm (CAMC): Calibration, Accuracy Study, and Clinical Applications," *IEEE Transactions on Medical Imaging*, vol. 29, no. 7, pp. 1412-23, 2010.



- [11] T. Klein, S. Benhimane, J. Traub, S. Heining, E. Euler and N. Navab, "Interactive Guidance System for C-arm Repositioning Without Radiation," *Bildverarbeitung für die Medizin*, pp. 21-25, 2007.
- [12] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri and R. Summers, "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases," *IEEE Computer Vision and Pattern Recognition*, 2017.
- [13] Y. Li, W. Liang, Y. Zhang, H. An and J. Tan, "Automatic Lumbar Vertebrae Detection Based on Feature Fusion Deep Learning for Partial Occluded C-Arm X-Ray Images," *IEEE Conference for Engineering in Medicine and Biology Society*, 2016.
- [14] H. Esfandiari, R. Newell, C. Anglin, J. Street and A. Hodgson, "A Deep Learning Framework for Segmentation and Pose Estimation of Pedicle Screw Implants Based on C-Arm Fluoroscopy," *International Journal of Computer Assisted Radiology and Surgery*, vol. 13, no. 8, pp. 1269-82, 2018.
- [15] A. Gad, "Convolutional Neural Networks," in *Practical Computer Vision Applications using Deep Learning with CNNs*, Menoufia, Egypt, Apress, pp. 183-227, 2018.
- [16] R. Girshick, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *Computer Vision and Pattern Recognition*, Berkeley, California, 2014.
- [17] O. Pauly, B. Diotte, P. Fallavollita, S. Weidert, E. Euler and N. Navab, "Machine Learning-Based Augmented Reality for Improved Surgical Scene Understanding," *Elsevier, Computerized Medical Imaging and Graphics*, vol. 41, pp. 55-60, 2014.
- [18] A. Woodard, G. Lynch and D. Berenson, "Server Based Extraction, Transfer, Storage, and Processing of Remote Settings, Files and Data". United States Patent 7032011 B2, 18 April 2006.
- [19] I. Ismaili, S. Khowaja and W. Soomro, "Image Compression, Comparison Between Discrete Cosine Transform and Fast Fourier Transform and the Problems Associated with DCT," in *Internatinoal Conference on Image Processing, Computer Vision, and Pattern Recognition*, 2013.
- [20] S. Arunachalam, S. Khairnar and B. Desale, "Implementation of Fast Fourier Transform and Vedic Algorithm for Image Enhancement using MATLAB," *Applied Mathematical Sciences*, vol. 9, no. 45, pp. 2221-32, 2015.

- [21] D. Kroese and R. Rubinstein, "Monte Carlo Methods," *Computational Statistics*, vol. 4, no. 1, pp. 44-58, 2011.
- [22] J. Iqbal, R. u. Islam and H. Khan, "Modeling and Analysis of a 6 DOF Robotic Arm Manipulator," *Canadian Journal on Electrical and Electronics Engineering*, vol. 3, no. 6, 2012.
- [23] F. Steinparz, "Co-ordinate Transformation and Robot Control with Denavit-Hartenberg Matrices," *Elsevier, Journal of Microcomputer Applications*, vol. 8, no. 4, pp. 303-16, 1985.
- [24] L. Wang, "Closed-Form Inverse Kinematics for Interventional C-Arm X-ray Imaging with Six Degrees of Freedom: Modeling and Application," *IEEE Transactions on Medical Imaging*, vol. 31, no. 5, pp. 1086-99, 2012.
- [25] L. Matthaus, "Closed-Form Inverse Kinematic Solution for Fluoroscopic C-Arms," *Advanced Robotics*, vol. 21, no. 8, pp. 869-86, 2012.
- [26] N. Rodas, J. Bert, D. Visvikis and M. d. Mathelin, "Pose Optimization of a C-Arm Imaging Device to Reduce Intraoperative Radiation Exposure of Staff and Patient During Interventional Procedures," *IEEE International Confererence on Robotics and Automation*, 2017.
- [27] "Object Detection Using Deep Learning," 10 August 2018. [Online]. Available: [https://www.mathworks.com/help/vision/examples/object-detection-using-deep-learning.html?s\\_tid=mwa\\_osa\\_a](https://www.mathworks.com/help/vision/examples/object-detection-using-deep-learning.html?s_tid=mwa_osa_a). [Accessed 2 February 2019].
- [28] R. Gandhi, "Build Your Own Convolutional Neural Network in 5 mins," 18 May 2018. [Online]. Available: <https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f>. [Accessed 21 March 2019].
- [29] J. Wu, "Introduction to Convolutional Neural Networks," *National Key Lab for Novel Software Technology*, 2017.
- [30] "Introduction to Convolutional Neural Networks," Stanford University, Stanford, California, 2018.
- [31] X. Glorot, A. Bordes and Y. Bengio, "Deep Sparse Rectifier Neural Networks," University of Montreal, Montreal, Quebec, 2011.
- [32] J. Barzilai and J. Borwein, "Two-Point Step Size Gradient Methods," *IMA Journal of Numerical Analysis*, pp. 141-8, 1998.

- [33] H. Avjyan, "Stochastic Gradient Descent with Code and Implementation," 22 October 2018. [Online]. Available: <https://medium.com/@hakobavjyan/stochastic-gradient-descent-sgd-10ce70fea389>. [Accessed 19 January 2019].
- [34] A. Goshtasby, "Piecewise linear mapping functions for image registration," *Pattern Recognition*, vol. 19, pp. 459-66, 1986.
- [35] A. Goshtasby, "Image registration by local approximation methods," *Image and Vision Computing*, vol. 6, pp. 255-61, 1998.
- [36] K. Schmid, D. Marx and A. Samal, "Tridimensional Regression for Comparing and Mapping 3D Anatomical Structures," *Anatomy Research International*, vol. 2012, 2011.
- [37] H. Bay and A. Ess, "Speeded-Up Robust Features (SURF)," ETH Zurich, Katholieke Universiteit Leuven, 2008.
- [38] H. Bay and A. Ess, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346-59, 2008.
- [39] H. Livyatan, Z. Yaniv and L. Joskowicz, "Robust Automatic C-Arm Calibration for Fluoroscopy-Based Navigation: A Practical Approach," School of Computer Science and Engineering, The Hebrew University of Jerusalem, 2002.
- [40] D. Weijiang, "Lift Apparatus for Supporting C-Arm or U-Arm and Medical X-ray Machine Having the Same". United States Patent 20110243309, 10 October 2011.
- [41] P. Hawkes, "The Correction of Electron Lens Aberrations," *Elsevier, Ultramicroscopy*, 2015.
- [42] H. Liu, H. J and L. Fajardo, "Lens Distortion in Optically Coupled Digital X-ray Imaging," *American Association of Physicists in Medicine, Medical Physics*, vol. 27, no. 5, 2000.
- [43] J. Bouguet, "Camera Calibration Toolbox for Matlab," 14 October 2015. [Online]. Available: [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/). [Accessed 28 March 2019].
- [44] B. Caprile and V. Torre, "Using Vanishing Points for Camera Calibration," *International Journal of Computer Vision*, vol. 4, no. 2, pp. 127-39, 1990.
- [45] J. Weng, P. Cohen and M. Herniou, "Camera Calibration with Distortion Models and Accuracy Evaluation," *IEEE, Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 965-80, 1992.

- [46] M. Pollefeys, R. Koch and L. V. Gool, "Self-Calibration and Metric Reconstruction In spite of Varying and Unknown Intrinsic Camera Parameters," *International Journal of Computer Vision*, vol. 32, no. 1, pp. 7-25, 1998.
- [47] E. Dubrofsky, "Homography Estimation," Master's thesis, Department of Computer Science, The University of British Columbia Vancouver, 2009.
- [48] Z. Zhang, "A Flexible New Technique for Camera Calibration," Microsoft Research Technical Report, 1998.
- [49] Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations," in *International Conference of Computer Vision*, 1999.
- [50] R. Hartley, "Self-Calibration from Multiple Views with a Rotating Camera," in *European Conference of Computer Vision*, 1994.
- [51] M. Brown and D. Lowe, "Recognizing Panorama," in *International Conference of Computer Vision*, 2003.
- [52] S. Lim and D. Lee, "Stable Improved Softmax using Constant Normalisation," *Electronics Letters*, vol. 53, no. 23, pp. 1504-6, 2017.

## Appendix A

### Object Detection for Orthopedic Surgery

```
cifar10Data = 'D:\Machine Learning';
[trainingImages,trainingLabels,testImages,testLabels] =
helperCIFAR10Data.load(cifar10Data);

numImageCategories = 10;

% Create the image input layer for 32x32x3 CIFAR-10 images
[height, width, numChannels, ~] = size(trainingImages);

imageSize = [height width numChannels];
inputLayer = imageInputLayer(imageSize);

% Convolutional layer parameters
filterSize = [5 5];
numFilters = 32;

middleLayers = [
    convolution2dLayer(filterSize, numFilters, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(2, 'Stride',2)

    convolution2dLayer(filterSize, numFilters, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(2, 'Stride',2)

    convolution2dLayer(filterSize, 2 * numFilters, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(2, 'Stride',2)
];

finalLayers = [
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(numImageCategories)
    softmaxLayer
    classificationLayer
];

layers = [
    inputLayer
    middleLayers
    finalLayers];
layers(2).Weights = 0.0001 * randn([filterSize numChannels
numFilters]);

% Set the network training options
opts = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'InitialLearnRate', 0.001, ...
```

```

        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 8, ...
        'L2Regularization', 0.004, ...
        'MaxEpochs', 40, ...
        'MiniBatchSize', 128, ...
        'Verbose', true);

doTraining = false;

if doTraining
    % Train a network.
    cifar10Net = trainNetwork(trainingImages, trainingLabels, layers,
    opts);
else
    % Load pre-trained detector for the example.
    load('rcnnStopSigns.mat','cifar10Net');
end

% Extract the first convolutional layer weights
w = cifar10Net.Layers(2).Weights;

% rescale the weights to the range [0, 1] for better visualization
w = rescale(w);
figure;
montage(w);

% Run the network on the test set.
YTest = classify(cifar10Net, testImages);

% Calculate the accuracy.
accuracy = sum(YTest == testLabels)/numel(testLabels)

folderTrainDetector = 'D:\Machine Learning\Combination\TrainDetector';

pngFilesTrainDetector = dir(fullfile(folderTrainDetector, '*.png'));
jpgFilesTrainDetector = dir(fullfile(folderTrainDetector, '*.jpg'));
filesTrainDetector = [pngFilesTrainDetector; jpgFilesTrainDetector];

imageFilename = {};

for slice = 1 : length(filesTrainDetector)
    filenameTrainDetector = fullfile(folderTrainDetector,
    filesTrainDetector(slice).name);
    imageFilename = [imageFilename; filenameTrainDetector];
end

% Define the bounding boxes of each category found in the training
images (for Supervised Learning)
Chest =
{[200,200,700,600];[200,200,700,600];[200,200,700,600];[200,200,700,600
];[200,200,700,600];[200,200,700,600];[200,200,700,600];[200,200,700,60
0];[200,200,700,600];[200,200,700,600];...

[200,200,700,600];[200,200,700,600];[200,200,700,600];[200,200,700,600]

```

; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]; [200, 200, 700, 600]  
]; [200, 200, 700, 600]; [200, 200, 700, 600]; ...

[118, 84, 74, 80]; [112, 72, 76, 92]; [132, 82, 68, 82]; [330, 450, 302, 454]; [168, 132  
, 120, 134]; [130, 74, 78, 92]; [432, 126, 98, 118]; [108, 112, 100, 104]; []; [94, 48, 3  
8, 34]; ...

[]; []; [684, 684, 564, 394]; [152, 226, 220, 214]; [130, 140, 128, 116]; [94, 142, 138  
, 164]; [64, 70, 42, 48]; []; [30, 10, 204, 190]; [72, 34, 232, 200]; ...

[34, 10, 222, 220]; []; []; []; []; [100, 100, 850, 700]; [50, 50, 924, 924]; [2, 2, 1022  
, 1022]; []; []; ...

[]; []; []; [200, 200, 700, 600]; [200, 200, 700, 600]; []; []; []; []; [200, 200, 700, 6  
00]; ...

[]; []; [200, 200, 700, 600]; []; [200, 200, 700, 600]; []; []; []; []; []; ...  
[200, 200, 700, 600]; [200, 200, 700, 600]; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...





```

[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...

Leg = {[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...

[114,226,78,174];[100,224,96,194];[110,214,110,194];[];[];[122,220,60,1
90];[];[92,306,132,254];[];[92,110,40,84];...

[124,44,204,274];[74,78,182,286];[];[126,590,262,426];[124,358,136,86];
[];[150,58,120,62];[];[];[];...

[];[10,12,122,204];[28,90,104,102];[26,14,156,216];[124,2,140,604];[];[
];[];[140,6,272,788];[60,30,1050,1252];...
[];[];[];[];[];[];[118,10,152,616];[518,32,282,1110];[];[];...

[334,70,314,828];[];[];[];[];[];[270,10,380,1280];[];[4,4,844,1020];[4,
4,330,596];...
[];[];[];[194,6,150,570];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...

[140,6,272,788];[140,6,272,788];[140,6,272,788];[140,6,272,788];[140,6,
272,788];[140,6,272,788];[140,6,272,788];[140,6,272,788];[140,6,272,788
];[140,6,272,788];...
[];[];[];[];[];[];[];[];[];[];...
[];[];[];[];[];[];[];[];[];[];...

[670,220,330,1000];[20,5,250,580];[450,10,370,1000];[40,10,160,480];[62
0,20,470,1260];[50,10,70,220];[530,520,140,450];[20,360,1960,2060];[130
,350,100,260];[460,200,230,590];...

[670,220,330,1000];[670,220,330,1000];[670,220,330,1000];[670,220,330,1
000];[670,220,330,1000];[670,220,330,1000];[670,220,330,1000];[670,220,
330,1000];[670,220,330,1000];[670,220,330,1000];...

[670,220,330,1000];[670,220,330,1000];[670,220,330,1000];[670,220,330,1
000];[670,220,330,1000];[670,220,330,1000];[670,220,330,1000];[670,220,
330,1000];[670,220,330,1000];[670,220,330,1000];...

```



20]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; ...

[50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; ...

[50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; [50, 10, 70, 220]; ...

[[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...

Neck = {[]; []; []; []; []; []; []; []; []; []; []; ...

[[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...

[132, 64, 40, 26]; [134, 54, 30, 16]; [154, 66, 20, 14]; [342, 280, 166, 168]; [224, 90, 36, 36]; [158, 60, 22, 12]; [480, 84, 32, 40]; [136, 70, 46, 26]; [460, 430, 170, 197]; [104, 32, 16, 10]; ...

[]; []; [888, 526, 140, 134]; [218, 168, 70, 60]; [176, 100, 36, 28]; [142, 104, 40, 36]; [46, 76, 16, 22]; [54, 152, 70, 74]; []; []; ...  
[]; []; []; []; []; []; []; []; []; []; ...

[84, 100, 296, 450]; [76, 110, 244, 350]; [226, 222, 176, 258]; []; []; [320, 186, 850, 1144]; []; []; [24, 76, 304, 422]; []; ...

[]; [168, 130, 254, 360]; []; [150, 44, 178, 236]; []; [74, 324, 438, 552]; []; [644, 464, 1346, 1774]; []; []; ...

[]; []; [72, 94, 252, 396]; []; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; ...

[256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; ...

[256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; ...

[256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; [256, 110, 424, 696]; ...



```

; [226, 222, 176, 258]; [226, 222, 176, 258]; [226, 222, 176, 258]; [226, 222, 176, 258]
]; [226, 222, 176, 258]; [226, 222, 176, 258]; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...
[]; []; []; []; []; []; []; []; []; []; ...

[644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 13
46, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [6
44, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; ...

[644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 13
46, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [6
44, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; ...

[644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 13
46, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [6
44, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; ...

[644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 13
46, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [6
44, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; ...

[644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 13
46, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; [6
44, 464, 1346, 1774]; [644, 464, 1346, 1774]; [644, 464, 1346, 1774]; ...

T = table(imageFilename, Chest, Leg, Neck)
A = imread(T.imageFilename{1});
A =
insertObjectAnnotation(A, 'Rectangle', T.Chest{1}, 'Chest', 'LineWidth', 2, '
FontSize', 18);
figure
imshow(A)

doTraining = true;
if doTraining
    options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 5, ...
        'MaxEpochs', 10, ...
        'Verbose', true, ...
        'VerboseFrequency', 50);
    rcnn = trainRCNNObjectDetector(T, cifar10Net, options, ...
        'NegativeOverlapRange', [0 0.3], 'PositiveOverlapRange', [0.5
1])
else

```

```

    load('rcnnStopSigns.mat','rcnn')
end

%%
% Test R-CNN Bone Detector on a Single Test Image

testImage = imread('Leg (01).jpg');
testImage = rgb2gray(testImage);
[bbox,score,label] = detect(rcnn,testImage,'MiniBatchSize',128,
'SelectStrongest', true);

[score, idx] = max(score);
bbox = bbox(idx, :);
annotation = sprintf('%s: (Confidence = %f)', label(idx), score);
outputImage = insertObjectAnnotation(testImage, 'rectangle', bbox,
annotation);
figure, imshow(outputImage)

%%
% Test R-CNN Bone Detector on entire Test Data Set
folderTestDetector = 'D:\Machine Learning\Combination\Test';

pngFilesTestDetector = dir(fullfile(folderTestDetector, '*.png'));
jpgFilesTestDetector = dir(fullfile(folderTestDetector, '*.jpg'));
filesTestDetector = [pngFilesTestDetector; jpgFilesTestDetector];

for zef = 1:80
    TESTLabels(zef,:) = {'Chest'};
end
for zef = 81:140
    TESTLabels(zef,:) = {'Leg'};
end
for zef = 141:200
    TESTLabels(zef,:) = {'Neck'};
end

Zbboxes = zeros(length(filesTestDetector), 4);
Zscores = zeros(length(filesTestDetector), 1);
Zlabels = categorical(length(filesTestDetector),1);

for shizzz = 1:length(filesTestDetector)
    TESTImage = imread(filesTestDetector(shizzz).name);
    [Zbbox,Zscore, Zlabel] = detect(rcnn, TESTImage, 'MiniBatchSize',
128, 'SelectStrongest', true);

    [Zscore, Zidx] = max(Zscore);
    Zbbox = Zbbox(Zidx, :);
    Zlabel = Zlabel(Zidx, :);
    Zannotation = sprintf('%s: (Confidence = %f)', Zlabel, Zscore);
    ZoutputImage = insertObjectAnnotation(TESTImage, 'rectangle',
Zbbox, Zannotation);
    figure, imshow(ZoutputImage)

    Zscores(shizzz) = Zscore;

```

```

        Zlabels(shizzz,1) = Zlabel;

end

%%
% Calculate the accuracy.
Zaccuracy2 = sum(Zlabels == TESTLabels)/numel(TESTLabels);
average = nanmean(Zscores)
averageChest = nanmean(Zscores(1:80))
averageLeg = nanmean(Zscores(81:140))
averageNeck = nanmean(Zscores(141:200))

```

## Appendix B

### Image Generation using FFT and MCS

```
rng(0, 'twister');
b = 3.5;
a = 1.0;
r = (b-a).*rand(1000,1) + a;

foldername = 'D:\Machine Learning\Combination\TrainDetector';

for k = 1:200 % number of images to be generated
    randx = r(randperm(1000, 1));
    randy = r(randperm(1000, 1));

    I = imread('Parent_Image.jpg'); % original image
    I = rgb2gray(I);
    F = fft2(double(I));
    F = fftshift(F);

    [dimy, dimx] = size(F);
    [columnsInImage, rowsInImage] = meshgrid(1:dimx, 1:dimy);
    Cx = dimx/2;
    Cy = dimy/2;
    Ro = 5; % outter radius of filter ring
    Ri = 2; % inner radius of filter ring
    array2D = (rowsInImage - Cy).^2 + (columnsInImage - Cx).^2;
    ringPixels = array2D >= (Ri*randy).^2 & array2D <= (Ro*randx).^2;
    ring = imcomplement(ringPixels);

    T = zeros(dimy, dimx);
    for h = 1:dimx
        for e = 1:dimy
            T(e,h) = F(e,h)*ring(e,h);
        end
    end

    T2 = ifftshift(T);
    T2 = ifft2(T2);
    T2 = uint8(real(T2));
    figure, imshow([F,T]);
    figure, imshow([I,T2]);

    filename = ['Generated Image # ' num2str(k) '.jpg'];
    fullFileName = fullfile(foldername, filename);
    imwrite(T2, fullFileName);
end
```



## Appendix C

### Image Registration using Projective Transformations

```
fixedImage = imread('pre-procedure.jpg');
fixedImage = rgb2gray(fixedImage);
figure;
imshow(fixedImage);
title('Fixed Image');

movingImage = imread('post-procedure.jpg');
movingImage = rgb2gray(movingImage);

figure
imshowpair(fixedImage, movingImage, 'Scaling', 'joint');

fixedPoints = detectSURFFeatures(fixedImage);
movingPoints = detectSURFFeatures(movingImage);

[fixedFeatures, fixedPoints] = extractFeatures(fixedImage,
fixedPoints);
[movingFeatures, movingPoints] = extractFeatures(movingImage,
movingPoints);

pairs = matchFeatures(fixedFeatures, movingFeatures);

matchedFixedPoints = fixedPoints(pairs(:, 1), :);
matchedMovingPoints = movingPoints(pairs(:, 2), :);
figure;
showMatchedFeatures(fixedImage, movingImage, matchedFixedPoints, ...
    matchedMovingPoints, 'montage');
title('Matched Points (Including Outliers)');

[~, inlierFixedPoints, inlierMovingPoints] = ...
    estimateGeometricTransform(matchedFixedPoints, matchedMovingPoints,
'projective');

figure;
showMatchedFeatures(fixedImage, movingImage, inlierFixedPoints, ...
    inlierMovingPoints, 'montage');
title('Matched Points (Inliers Only)');

fixedPoints = zeros(inlierFixedPoints.Count, 2);
movingPoints = zeros(inlierMovingPoints.Count, 2);

for p = 1:inlierFixedPoints.Count
    fixedPoints(p, :) = inlierFixedPoints.Location(p, :);
    movingPoints(p, :) = inlierMovingPoints.Location(p, :);
end

tform = fitgeotrans(movingPoints, fixedPoints, 'projective');
```

```
movingReg = imwarp(movingImage, tform, 'OutputView',  
imref2d(size(fixedImage)));  
figure  
imshow(movingReg);  
  
figure  
imshowpair(fixedImage, movingReg)  
title('Image Registration Results')
```

## Appendix D

### Tilt and Orbital Angles of Rotation

```
% Extract Tilt and Orbital rotation angles
T = tform.T;
T = transpose(T);

% my camera's calibration matrix parameters
KK = [1.430712510883822e+03 -3.287386352501660
      1.009561839215500e+03; ...
      0 1.434233457820518e+03 5.157919135274227e+02; ...
      0 0 1];

% homography matrix
H = inv(KK)*T;

n = norm(H(:,1));
t = H(:,3)/n;
r1 = H(:,1)/n;
r2 = H(:,2)/n;
r3 = cross(r1,r2);

% rotation matrix
RM = [r1 r2 r3];

tilt = atan(r3(1)/r3(3))*(180/pi);
orbit = asin(r3(2))*(180/pi);
```