T.C

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DIGITAL IMAGE PROCESSING

TERM PROJECT REPORT

Name/Surname	Abdullah Doğan GÜVENER
Number	244201008002

TABLE OF CONTENTS

ABSTRACT

Title

X-ray Phantom Image Analyzer

The aim of this project is to analyze the phantom image of X-ray to evaluate the correct threshold to calibrate the X-ray devices by using digital image processing techniques.

Keywords: X-ray device, phantom image, DIP

1 INTRODUCTION

1.1.1. Subtitle

1.1 Aim of the project

This project was developed to address a system-level testing problem for X-ray devices. During the production of these X-ray devices, a comprehensive system test is performed at the end of the assembly process. As part of this test, a specific phantom image is placed in the X-ray field, and an image is captured by the device to assess its resolution. A threshold value is used to calibrate the device, and the objective of this process is to visually evaluate the captured image against the threshold value to ensure proper calibration of the X-ray device.

2
MATERIAL AND METHOD

2.1 Material

• Software **and Libraries**:

- Python programming language
- Libraries such as OpenCV, NumPy, Matplotlib, or any other relevant libraries are used.
- Development environment (e.g., Jupyter Notebook with Visual Studio Code)

• Hardware:

- Computer specifications:
- Processor: Intel i5-11300H
- RAM = 16GB DDR4
- GPU = NVIDIA GeForce RTX 3050)

• Dataset/**Images**:

- Images taken from ASELSAN Mobile X-Ray Device Shown in Figure 1.1

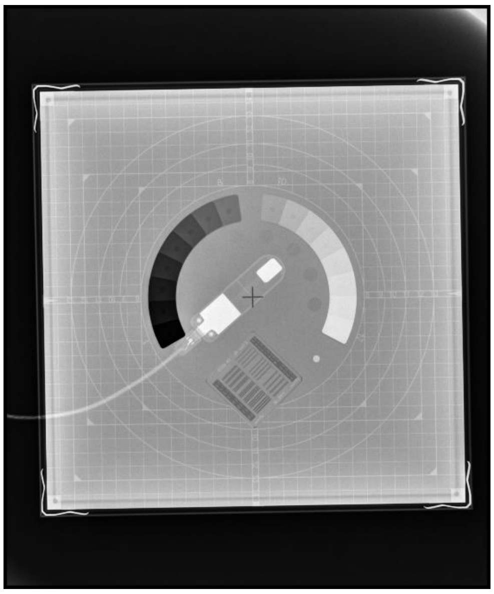

Figure 2.1 Example X-Ray Phantom Image Shot

2.2 Method

Phantom Image Analysis:

There are several test phantom images for x-ray devices as shown in Figure 1.2. X-ray device producers are free to choose one of those phantom test images.

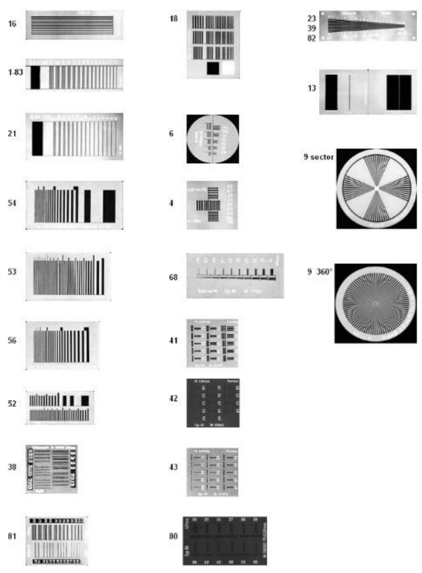

Figure 2.2 Phantom Types

In ASELSAN's Mobile X-Ray device a specific test phantom is used and its shown in Figure 1.3. This phantom contains lines of varying widths, and after capturing an image with the X-ray device, the highest LP/mm value at which all three lines are still clearly visible is considered the threshold value.

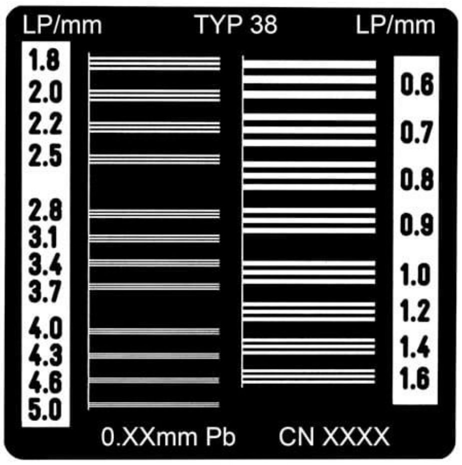

Figure 2.3 Original Phantom

Image Processing Workflow:

After taking shot with this phantom original image is like shown in Figure 1.4.

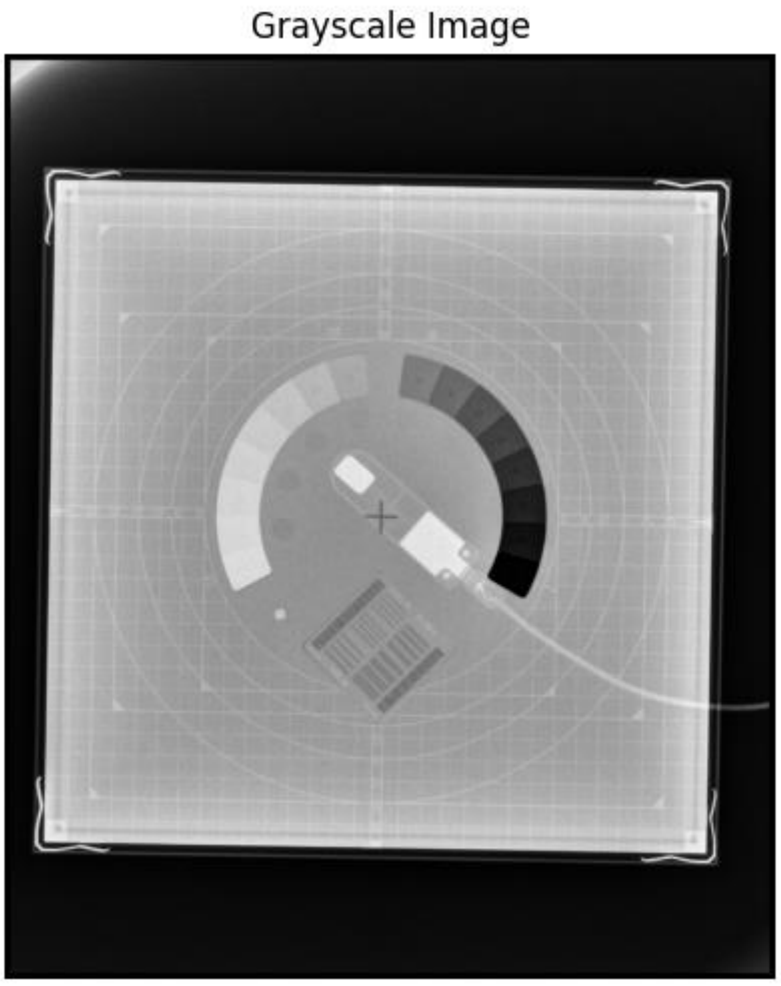

Figure 1.4 Original Phantom Image

As seen from Figure 1.4, the shot also includes unmeaningful areas and the point is to get close to the phantom shown in Figure 1.3. To do this first I evaluated a process to get the corners of the phantom. In every shot those corners are the same and because of that reason I have clipped equally the corners of the phantom and use them as comparator to identify new images corners and make them ready for further operations. The Kernels are shown in Figure 1.5

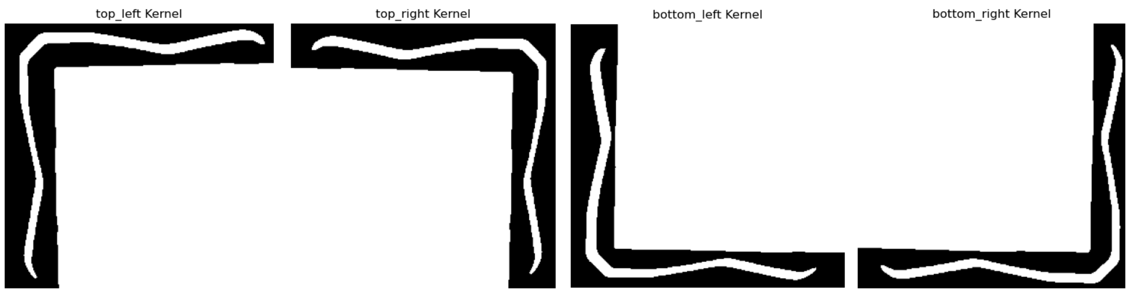

Figure 1.5 Corner Kernels

The similarity search operation in this project is a technique used to identify and locate specific regions in an input image that closely matches predefined template images (kernels). The process involves converting the input image to grayscale and applying a binary threshold to simplify its representation. Each kernel, representing a distinct corner type, is compared against overlapping regions of the input image using a sliding window approach. The similarity between the kernel and the region is calculated as the proportion of matching pixels. To optimize performance, the search is conducted in predefined steps rather than pixel-by-pixel, ensuring computational efficiency. The best match for each kernel is determined by the region with the highest similarity score exceeding a defined threshold. The detected positions are visually annotated on the output image with bounding rectangles and labeled points, aiding in the accurate localization of key features within the input image. This operation is crucial for precise pattern recognition and automated visual analysis in image processing tasks.

An example output of this algorithm is shown in Figure 1.6.

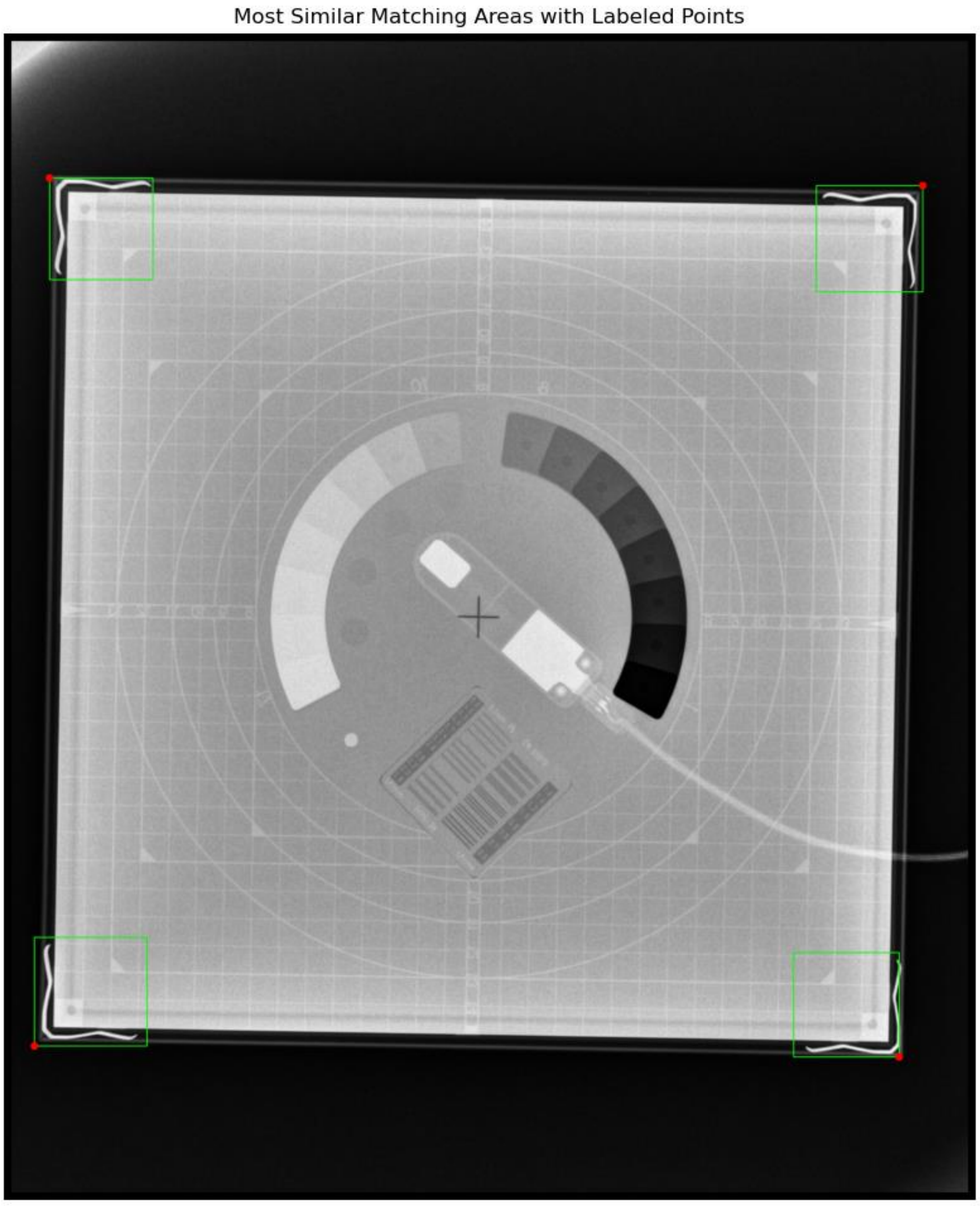

Figure 1.6 Phantom Corner Detection Algorithm Results

After detecting those 4 points the image is clipped from those points as shown in Figure 1.7.

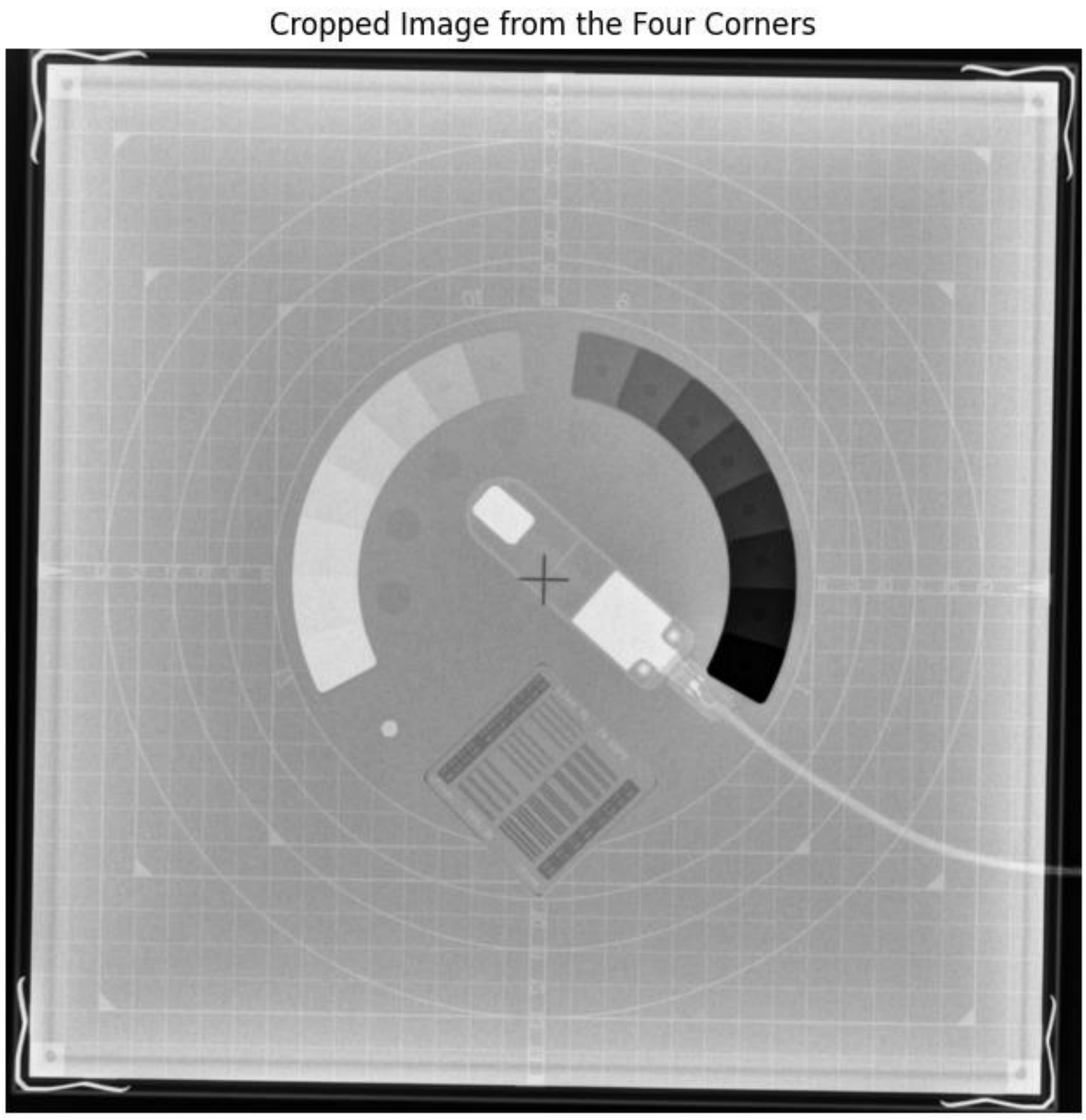

Figure 1.7 Cropped Image

But there is a problem with rotation. By using detected 4 points I can calculate the rotation and fix it.

- The rotation angle (θ) between two points (x1,y1) and (x2,y2) is calculated using the following formula:

$\theta = \operatorname{atan2}(y2 - y1, x2 - x1)$

- Pixel Mapping for Rotation

To rotate the image by an angle

θ around its center (center$x$,center$y$)(center x ,center y), the coordinates of a pixel (x,y)(x,y) in the rotated image are mapped to the corresponding coordinates (x',y')(x′ ,y′) in the original image using the following formulas:

x′=cos(θ)·(x−centerx)+sin(θ)·(y−centery)+centerx

y′=−sin(θ)·(x−centerx)+cos(θ)·(y−centery)+centery

- Bounding Box for Cropping

The bounding edges for cropping are determined using the corner points:

Left Edge: left_edge=min(xtop_left,xbottom_left)

Right Edge: right_edge=max(xtop_right,xbottom_right)

Top Edge: top_edge=min(ytop_left,ytop_right)

Bottom Edge: bottom_edge=max(ybottom_left,ybottom_right)

After implementing those processes to the images the result is shown in Figure 1.8.

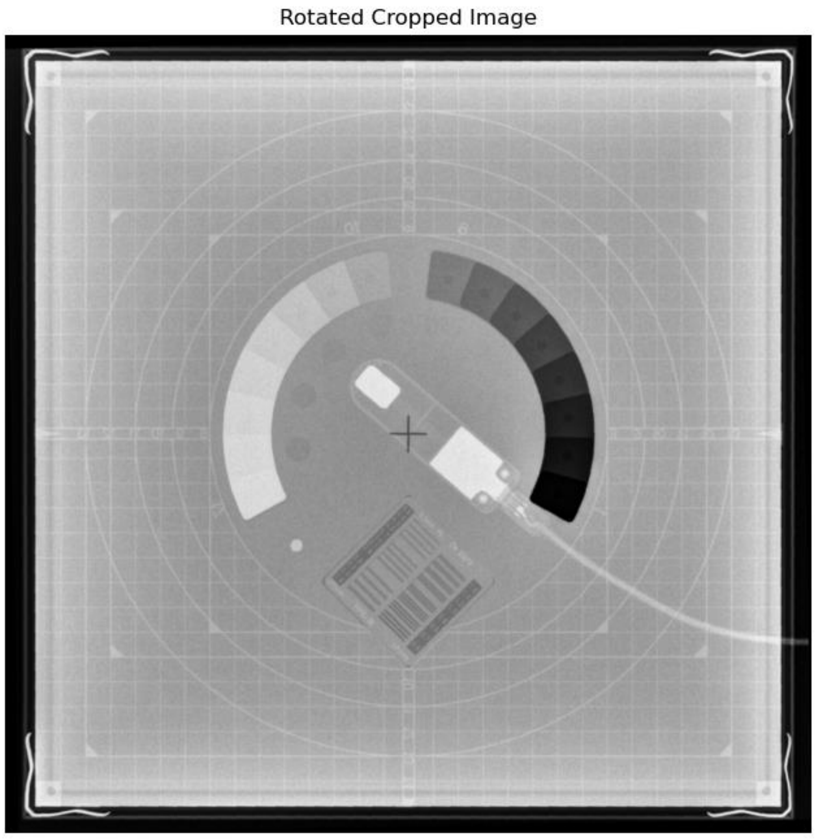

Figure 1.8 Rotated Cropped Image

After rotating the image now its time to detect the phantom image. To detect it I have used a kernel for it as shown in Figure 1.9.

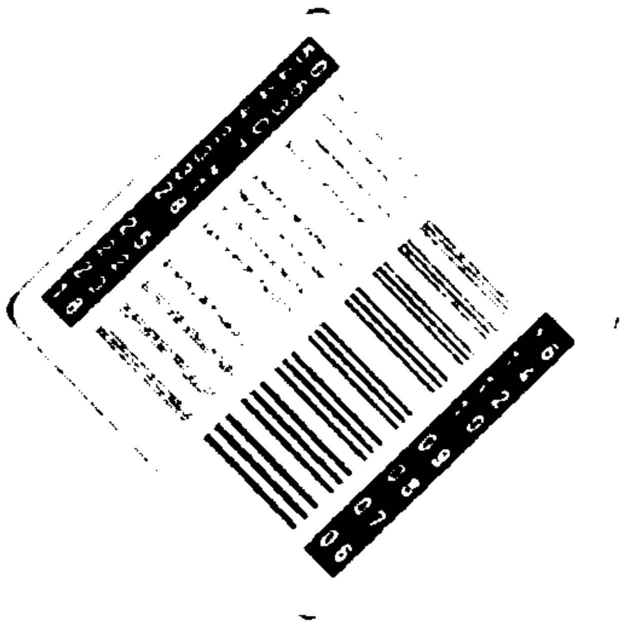

Figure 1.9 Phantom Kernel

I have applied same similariy check operation that I have done for corner detection. After detecting the image, the final cropped image is also rotated 45 deggre in anti clock wise direciton is shown in Figure 1.10.

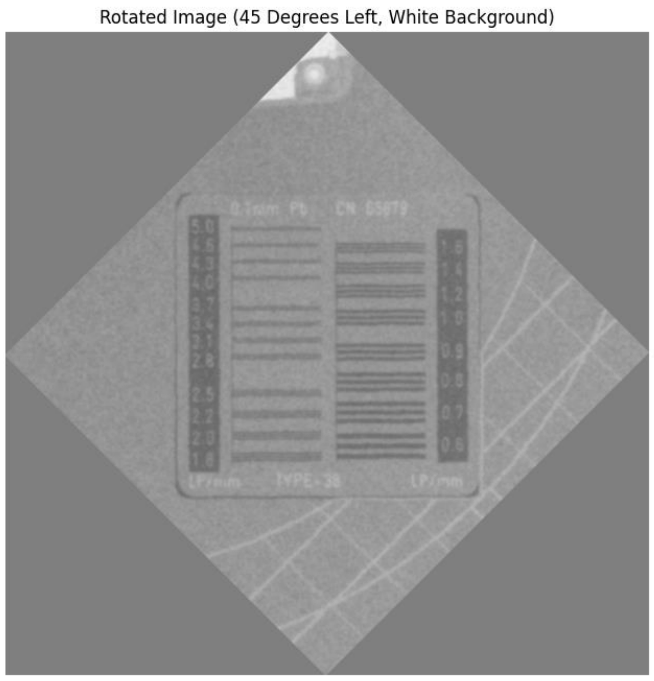

Figure 1.10 Cropped Phantom Image

After detecting the kernel, a specific code run for the user to clip the region of lines. The reason we did at that point is that image resolution is very high and not able to select the region. The region is specified by the user. The defined region is shown in Figure 1.11.

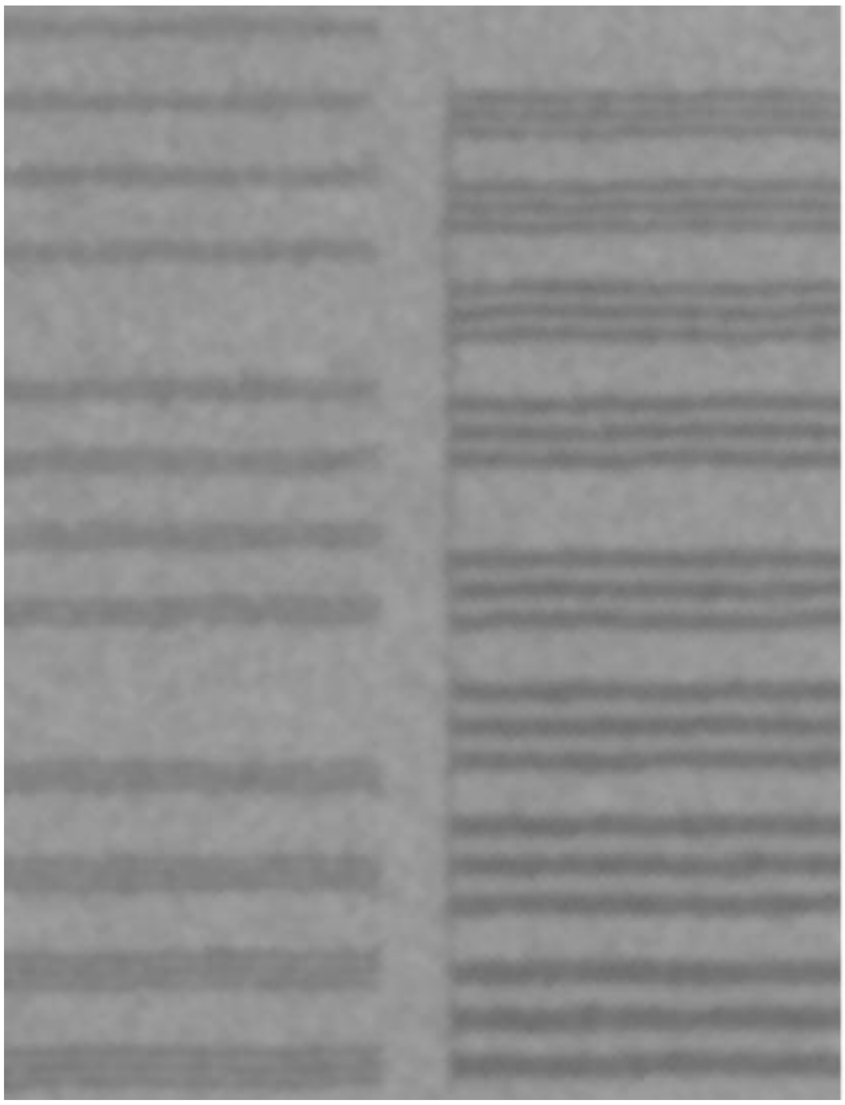

Figure 1.11 Defined Region for Analysis

After this process a binary thresholding is applied to the image and divided into two pieces to make analysis as shown in Figure 1.12.

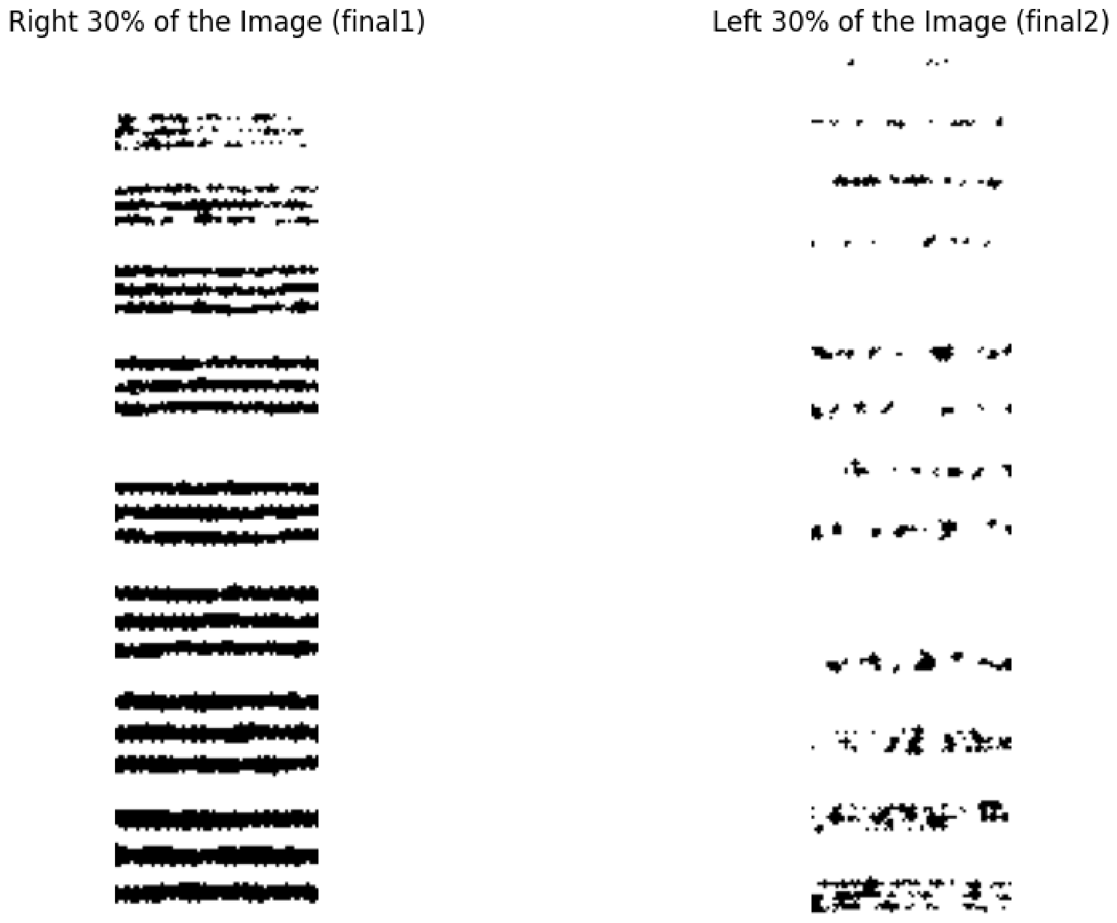
Right 30% of the Image (final1)
Left 30% of the Image (final2)

Figure 1.12 Images Ready for Analysis

After that point a specific analysis algorithm is applied. The analysis conducted demonstrates the effectiveness of identifying and quantifying discontinuities in horizontal line patterns within the binary image. The results depict the ratio of black pixels to total pixels for each line, visualized as a continuous graph. Significant regions of interest were highlighted where the black pixel ratio exceeded the predefined threshold of 0.4, grouping contiguous lines within a maximum distance of 3. These regions were further evaluated for their average black pixel ratio, which is annotated on the graph for clarity. Additionally, the transposed binary image offers an alternate perspective for visual inspection of line patterns, reinforcing the accuracy of the automated quality assessment. This method successfully identifies key patterns and provides a reliable metric for evaluating the quality of the horizontal line formations in Figure 1.13 and Figure 1.14.

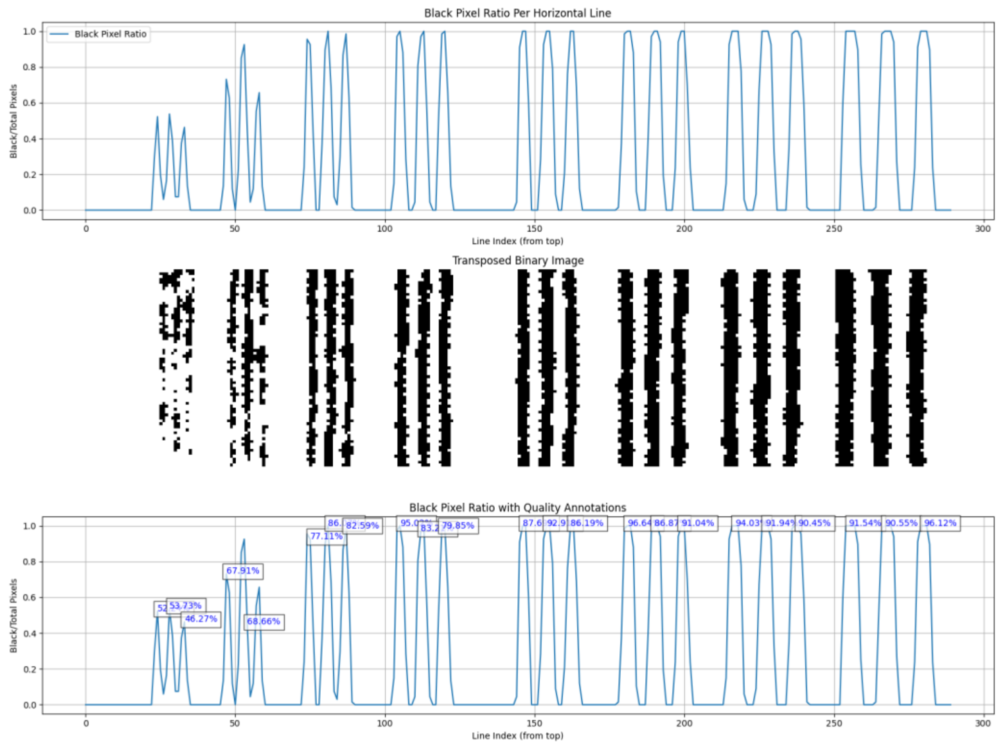

Figure 1.13 Analysis Results-1

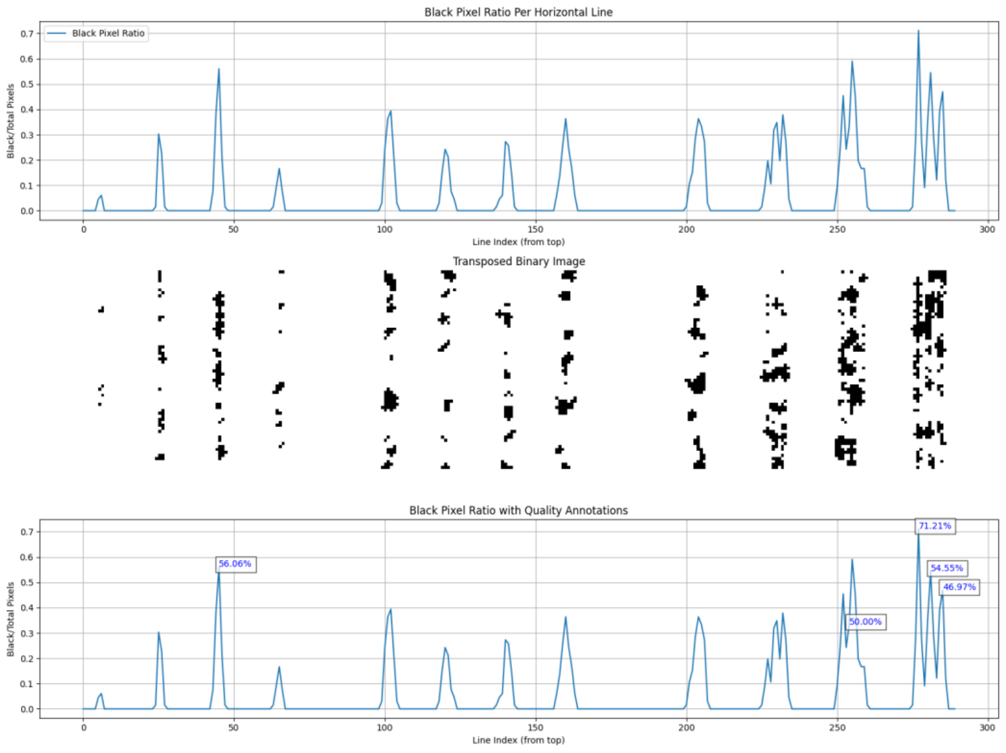

Figure 1.14 Analysis Results-2

An extra approach to this analysis opening is applied to the image to have better lines. The result for opening is shown in Figure 1.15. But this method is not worked well the one applied in Figure 1.13 and Figure 1.14.

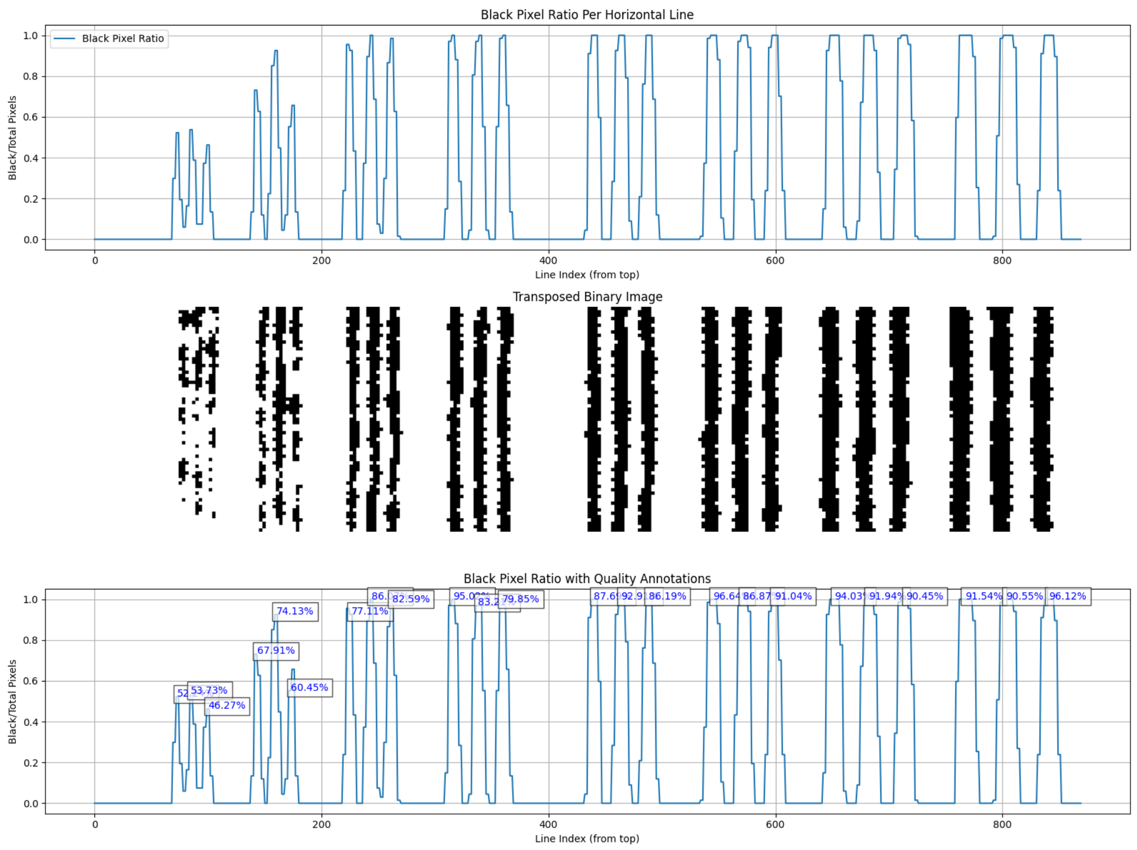

Figure 1.15 The Results After Opening

3
DISCUSSION

3.1 Discussion

This project aimed to analyze X-ray phantom images to assist in the calibration of X-ray devices. The method developed worked effectively in detecting discontinuities in the phantom's horizontal line patterns. By calculating the black pixel ratio for each line and grouping contiguous lines based on a threshold, we were able to identify important regions and evaluate their quality.

One of the main challenges was handling rotation in the captured images. Using the detected corner points, we calculated the rotation angle and successfully corrected it, ensuring the images were properly aligned for further analysis. Another challenge was testing alternative methods like morphological opening, but as shown in the results, this approach didn’t perform as well as the primary method.

While the analysis was successful overall, there are some areas for improvement. For example, the process currently requires user input to define regions for line detection, which could be automated in the future. Additionally, the method might be sensitive to noise or variations in the phantom image, which should be addressed in further studies.

This work provides a useful tool for evaluating phantom images and can be a reliable addition to X-ray device calibration processes. In the future, we could explore more advanced techniques, like machine learning, to improve automation and make the process even more robust.

REFERENCES

The Implementation of the project and codes are available at github.

[1] https://github.com/adguvener/x-ray-line-detection.git