Designing a Shape-Based Contour Detection System for the Identification of Nuts and Washers

Nathanael Adrian T. Cua
Department of Electronics and Computer Engineering
Gokongwei College of Engineering
De La Salle University Manila
Manila, Philippines
nathanael_cua@dlsu.edu.ph

Gospel Ernes G. Blances
Department of Electronics and Computer Engineering
Gokongwei College of Engineering
De La Salle University Manila
Manila, Philippines
gospel_blances@dlsu.edu.ph

Luis Miguel G. Perez IV
Department of Electronics and Computer Engineering
Gokongwei College of Engineering
De La Salle University Manila
Manila, Philippines
luis_pereziv@dlsu.edu.ph

Psalm Darius C. Regis
Department of Electronics and Computer Engineering
Gokongwei College of Engineering
De La Salle University Manila
Manila, Philippines
psalm_regis@dlsu.edu.ph

***Abstract*—This project aims to develop a computer vision system for the identification and counting of nuts and washers in industrial retail environments through shape-based contour detection techniques. The system can highlight objects in images and distinguish between nuts (polygonal in shape) and washers (circular). Furthermore, the classification accuracy is verified through ground truth comparison. The system ultimately aims to enhance hardware inventory management, improve manufacturing automation, and offer quality control in detecting defective parts.**

Keywords—computer vision, object classification, image preprocessing, edge detection, manufacturing

I. Introduction

The increasing demand for automation in industries such as manufacturing and retail has driven the need for advanced systems capable of efficiently identifying and classifying various objects. This project focuses on developing a computer vision system that utilizes shape-based contour detection to identify and count various types of nuts and washers based on their distinct geometric features. The primary objectives of the project are to highlight all nuts and washers in an image, identify these objects through contour analysis, and verify the accuracy of the algorithm's output. The system will distinguish between nuts, typically characterized by a polygonal shape with a hollow center, and washers, which are circular with a central hole. By utilizing various contour detection techniques, the project aims to improve the automated identification of these hardware components.

This approach provides an efficient method for sorting and identifying objects in real-time, which is crucial for applications such as hardware store inventory management and automation in manufacturing. In a manufacturing setting, the system could support robotic systems for object sorting and assembly, reducing the need for manual labor. Additionally, the system could be adapted to detect irregularities or defects in these components, helping to ensure quality control by preventing defective items from being used or shipped. The project also holds potential for consumer applications, such as in self-checkout systems, offering a more accurate and efficient method for counting small items in retail environments.

II. Design and Application

A. Objectives

This project focuses on developing a computer vision system to identify and count nuts and washers within a hardware store. Using shape-based contour detection, the objectives include:

1. Highlight all nuts and washers within an image using shape-based contour techniques.
2. Identify nuts and washers based on distinct features-nuts are hollow polygons while washers are circular with center holes.
3. verify the output through comparison with ground truth data to evaluate the classification accuracy

These objectives align with the need for efficient inventory management and automation in both industrial and retail environments.

B. System Architecture

The proposed system operates in five stages: image acquisition, preprocessing, contour extraction, shape analysis, and classification. Each stage is detailed below.

1. Image Acquisition

Images of nuts and washers are captured under controlled conditions using standard cameras. The dataset comprises diverse orientations, lighting, and object arrangements to ensure robustness in real-world scenarios.

2. Preprocessing

Preprocessing ensures image quality and reduces computational load.

- **Grayscale Conversion:** Input images are converted to grayscale using cv2.cvtColor().

- **Gaussian Blurring:** Noise is minimized using cv2.GaussianBlur() with a kernel size of 5×5.
- **Edge Detection:** The Canny algorithm (cv2.Canny()) is applied to detect object boundaries effectively.

3. Contour Extraction

Contours, representing object boundaries, are extracted using OpenCVs cv2.findContours(). Filtering criteria include area thresholding to remove large or excessive contours, and polygon approximation, to simplify the contours to reduce complexity.

4. Shape Analysis

Geometric properties are used to differentiate washers from nuts:

- Circularity (C)

$$C = \frac{4\pi \, x \, Area}{Perimeter^2}$$

By getting a circularity value close to 1, the object can be defined or approximated to be a circle.

- Aspect Ratio (AR)

The ratio of width to height of the bounding rectangle is computed to assess elongation.

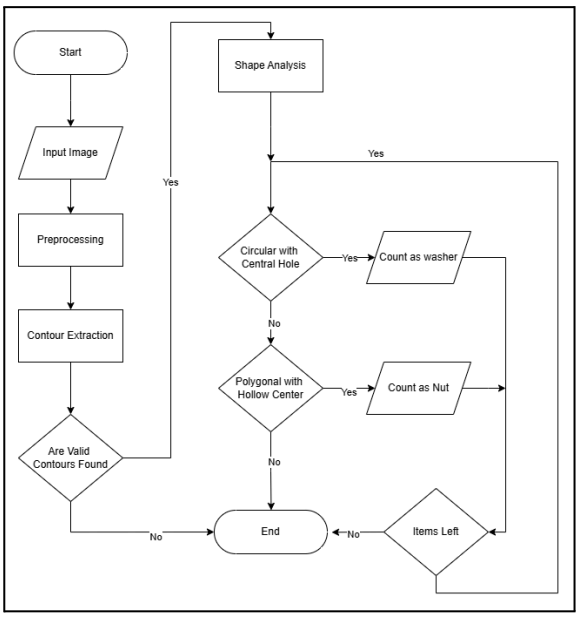
- Hole Detection

Internal contours are examined to confirm the presence of central holes, characteristic of both nuts and washers.

5. Classification

The system will classify objects based on the shape analysis: nuts with hollow polygons and washers with high circularity.

C. System Architecture



III. Review of Related Literature and Studies

Contour Detection and Object Recognition

The study by Charmia et al. (2010) introduces a method for improving contour detection by integrating geometric shape priors into region-based active contours. This approach addresses the limitations of traditional contour models, such as their vulnerability to noise, occlusion, and initialization errors. By using Fourier-based shape alignment, the method ensures that evolving contours are constrained to resemble a predefined reference shape, accounting for geometric transformations like scaling, rotation, and translation. This capability is particularly relevant for identifying objects with similar contours, such as nuts and washers, where distinguishing subtle differences is crucial. These features align well with the objectives of a shape-based contour detection system for nuts and washers. The study provides a theoretical foundation and practical insights that can inform the development of reliable detection mechanisms. This is further supported by a study by Xin-Yi et al. (2018) in a study entitled "An Overview of Contour Detection Approaches" – a paper that reviews contour detection methods. It classifies approaches into four major categories: pixel-based, edge-based, region-based, and deep convolutional neural network (DCNN)-based methods. The paper emphasizes that while significant progress has been made in contour detection, challenges such as handling noise, incomplete contours, and scalability remain. A study by Segaert, Nygård, and Wagemans (2009) investigated how motion influences object identification using kinetic contours derived from random dot displays. After conducting two experiments, both demonstrated that object identification improved with motion coherence and less compact objects. While the combination of speed and direction differences had a super-additive effect. This knowledge may be useful in manufacturing applications such as nuts on a conveyor belt.

Hardware Inventory Automation

Industrial automation has been pivotal in enhancing the efficiency and accuracy of inventory management systems. The use of automated sorting techniques, as discussed by Raffik et al. (2021), reduces human effort and time consumption in industrial applications. Automated systems utilize proximity sensors and IIoT web servers to monitor stock levels and demand, ensuring real-time updates and reducing errors in inventory tracking. The study by Raffik et al. (2021) highlights the use of low-cost automation (LCA) systems for sorting lightweight objects based on size variations. These systems employ photoelectric sensors and programmable logic controllers (PLCs) to achieve precise sorting with minimal errors, thereby enhancing overall production efficiency.

The use of automation systems for hardware inventory uses various hardware and software methods for different items. A study by Vellgas-Ch et al. (2024), explores how their platform, which uses convolutional neural networks with open source libraries led to a significant improvement in both counting time and accuracy. Their output reduced inventory time by 45% and improved stock efficiency by 50%. By cutting the time for inventory using computer vision, it shows that tedious and high error percentage jobs like counting can be replaced. Moreover, computer vision systems for counting overcome the limitations of traditional counting systems.

Advances in Image Preprocessing for Contour Detection

Researchers have made recent advancements in image preprocessing to significantly improve the accuracy and efficiency of contour detection. A study by Feng et al. (2021) provided a new detection algorithm that integrates fuzzy radial basis fusion with existing edge detection techniques. This study focuses on improving edge detection by addressing noise suppression and enhancing accuracy using fuzzy radial basis fusion. The results showed that the method enhances training speed by optimizing input data in the input layer and increasing data in the enhancement layer. The study also introduced an improved method called the radial basis width learning improvement method (RBF-BLS+).

A study by Zhou et al. (2020) utilized dynamic threshold analysis, contour extraction, and deep learning in developing a new iris recognition algorithm. The algorithm used the first two techniques to isolate the pupil edge and then proceeds with edge detection and gray-level calculations to precisely locate the iris. Finally, a deep learning network received features of normalized images for deep learning. This method enhanced the speed and accuracy of iris segmentation. The study was able to address the challenges posed by high equipment costs and algorithmic limitations and make advancements towards large-scale biometric applications.

IV. Significance of the Project

The system provides a more convenient way to perform inventory and hardware management. The system aims to identify and classify nuts and washers with efficiency and accuracy, which can reduce time-consuming manual labor and minimize human errors in identification and classification. The real-time processing capability ensures that inventory management is performed faster. The system also demonstrates the practical application of image processing and computer vision by applying edge detection and shape-based contour detection.

V. Methodology

Flowchart

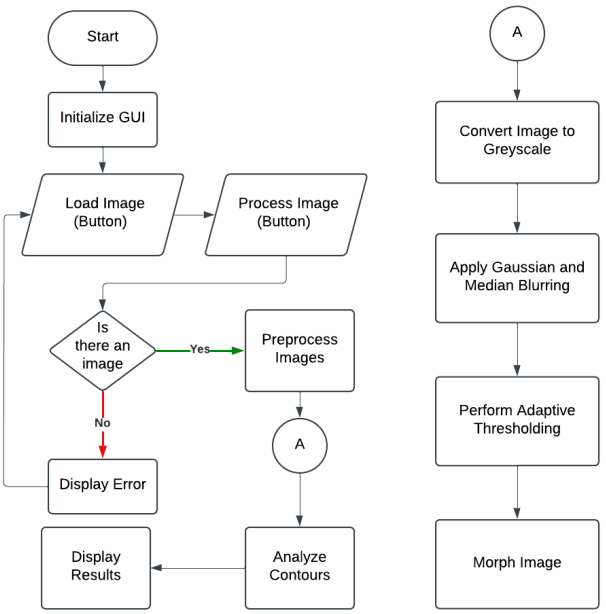


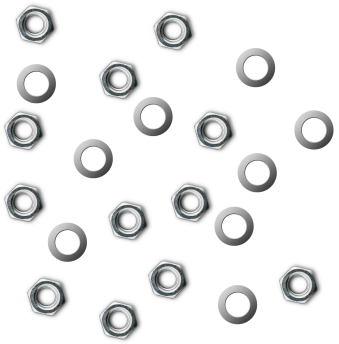
Figure 5.1 Flowchart of code.

The program is a Python-based application that combines computer vision techniques and a graphical user interface (GUI) to analyze images, detect nuts and washers, and display results. It is built using the tkinter library for the GUI, OpenCV for image processing, and PIL for efficient image manipulation. It begins with the initialization of a NutWasherDetector class, which sets up the GUI, including buttons for loading and processing images, an image display area, and a text box to show results. The interface is designed to be user-friendly and supports multiple image file formats like .png, .jpg, and .jpeg.

The load_image function allows users to select an image, which is then read using OpenCV and displayed in the GUI. Once an image is loaded, the process_image method analyzes it by invoking the detect_nuts_and_washers function. This function preprocesses the image (specifically handling the grayscale conversion, blurring, and thresholding), identifies contours, and evaluates shapes and features to differentiate nuts (hexagonal shapes with inner circles) and washers (circular shapes with inner holes).

The processed image is then displayed with highlighted contours, and the counts of detected items are presented in the results text box.

VI. Results and Discussion



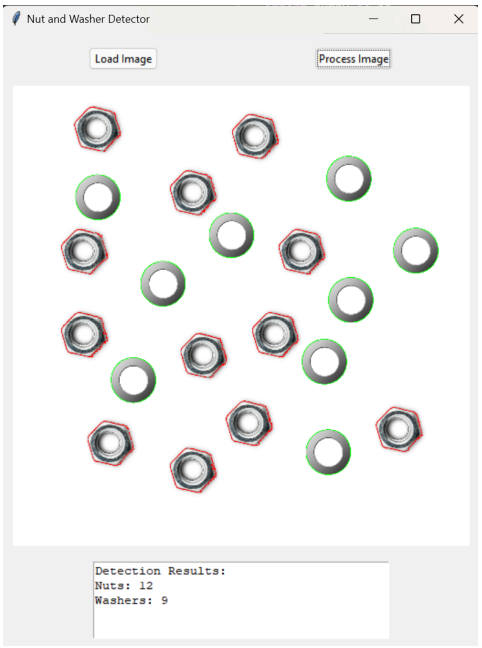
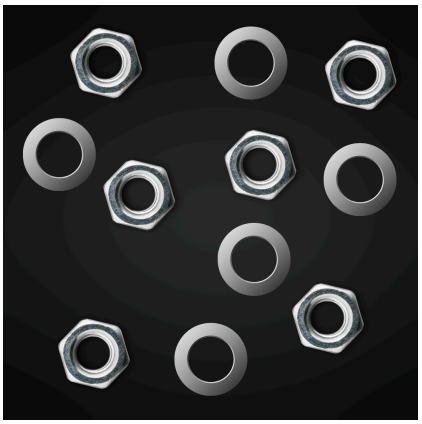


Figure 5.2 Inputted Image and Resulting Image (1)

The image above is an example of a photo the algorithm expects containing both nuts and washers from a top-down view.

The output of the image includes the inputted image with annotations on nuts (enclosed in red) and washers (enclosed in green). Moreover, the program displays the count of nuts and washers in the image. In the example above (Resulting Image (1)), it is able to detect twelve (12) nuts and nine (9) washers. This example shows that the nut and washer detection is capable in light backgrounds.



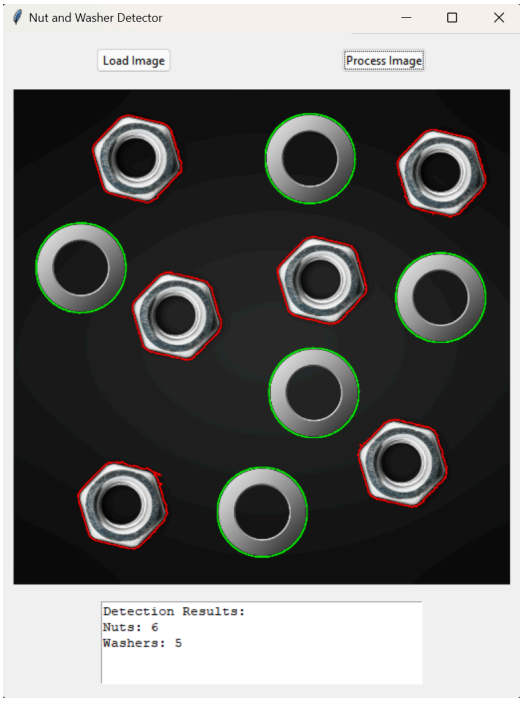


Figure 5.3 Inputted Image (2)

In the example above (Resulting Image (2)), it is able to detect six (6) nuts and five (5) washers. Notably this example showcases that the program also works in dark backgrounds.

VII. Recommendation

The performance of the system may be further enhanced by adding some more preprocessing techniques such as adaptive thresholding and morphological transformations in order to increase the contour clarity and remove noise. It should be tested on a wide range of environments that will face real-world applications including changing lighting and orientations. Incorporating metrics like ellipticity or solidity into advanced shape analysis would make the classification more accurate. The researchers suggest that further improvements can be made in exploring the use of machine learning or deep learning algorithms for automatic feature extraction and classification, thereby enhancing the system's adaptability and precision. Integration into a larger inventory management platform or robotic automation pipeline may also ensure scalability for applications in industrial settings.

VIII. Conclusion

The results show that the system is able to highlight nuts and washers in an image using shape-based contour detection. The system was able to identify the features of each nut and washer by the nut's hexagon-like shape and the washer's circular shape and to count the number of nuts and washers in an image accurately. It was also able to perform the analysis with varying sizes, showing that the system can identify the same object with different sizes. For the first image of small-scaled objects, the system was able to count 12 nuts and 9 washers. For the 2nd image of larger-scaled objects, the system was able to count 6 nuts and 5 washers. All of these results show that the system had great efficiency and accuracy in identification, classification, and counting.

The development of a shape-based contour detection system for identifying nuts and washers can serve as a foundational program that can be used for the automation of inventory management and quality control processes in industrial and retail settings. By utilizing advanced image preprocessing techniques and contour analysis, the system effectively distinguishes between nuts and washers based on their geometric features such as polygons and circularity. The scalability also allows the system to adjust to varying inventory management frameworks with different sizes of objects. This automatic analysis can help reduce the dependency on manual labor, minimize human error, and significantly improve operational efficiency.

Acknowledgment

We would like to express our gratitude to our CPECOG1 professor, Mr. Jay Robert del Rosario, for providing us with guidance, expertise, and insightful feedback throughout the lectures and the development of the project. He provided us with his knowledge of computer vision through lectures, which would support us in creating a project that utilizes contour-based edge detection.

We would also like to thank DLSU for providing the curriculum that taught us every topic and foundation important to this project. The curriculum gradually built up our knowledge for each activity up until this project. The project serves as a practice of all the topics from CPECOG1 and from all subjects before CPECOG1.

References

[1] Feng, L., Wang, J., and Ding, C. (December 21, 2021). Image edge detection algorithm based on fuzzy radial basis neural network. https://doi.org/10.1155/2021/4405657

[2] K. Segaert, G. E. Nygård, and J. Wagemans, "Identification of everyday objects on the basis of kinetic contours," Vision Research, vol. 49, no. 4, pp. 417–428, 2009, doi: 10.1016/j.visres.2008.11.012.

[3] Raffik, R., Rakesh, D., Venkatesh, M., & Samvasan, P. (2021). Supply chain control and inventory tracking system using industrial automation tools and IIoT. 2021 International Conference on Industrial Engineering and Operations Management (IEOM). https://doi.org/10.1109/IEOM.2021.9675774

[4] Villegas-Ch, W., Maldonado Navarro, A., & Sanchez-Viteri, S. (2024). Optimization of inventory management through computer vision and machine learning technologies. Intelligent Systems with Applications, 24, 200438. https://doi.org/10.1016/j.iswa.2024.200438

[5] X.-Y. Gong, H. Su, D. Xu, Z.-T. Zhang, F. Shen, and H.-B. Yang, "An Overview of Contour Detection Approaches," International Journal of Automation and Computing, vol. 15, no. 3, pp. 226-243, June 2018, doi: 10.1007/s11633-018-1117-z.

[6] Zhou, W., Ma, X., and Zhang, Y. (August 2020). Research on image preprocessing algorithm and deep learning of iris recognition. https://doi-org.dlsu.idm.oclc.org/10.1088/1742-6596/1621/1/012008

[7] Charmia, M. A., Mezghicha, M. A., M'Hiria, S., Derrode, S., & Ghorbela, F. (n.d.). Geometric shape prior to region-based active contours using Fourier-based shape alignment.

[8] Yang, D., Peng, B., Zaid Al-Huda, Malik, A., & Zhai, D. (2022). An overview of edge and object contour detection. Neurocomputing, 488, 470–493. https://doi.org/10.1016/j.neucom.2022.02.079

Appendix

```
import tkinter as tk
from tkinter import ttk, filedialog
import cv2
import numpy as np
from PIL import Image, ImageTk

class NutWasherDetector:
    def __init__(self, root):
        self.root = root
        self.root.title("Nut and Washer Detector")

        # Main frame
        self.main_frame = ttk.Frame(root, padding="10")
        self.main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N,
tk.S))

        # Buttons
        ttk.Button(self.main_frame, text="Load Image",
command=self.load_image).grid(row=0, column=0, pady=5)
        ttk.Button(self.main_frame, text="Process Image",
command=self.process_image).grid(row=0, column=1, pady=5)

        # Circularity slider
        self.circularity_label = ttk.Label(self.main_frame, text="Washer
Circularity Threshold:")
        self.circularity_label.grid(row=1, column=0, sticky=tk.W, pady=5)

        self.circularity_var = tk.DoubleVar(value=0.87)
        self.circularity_slider = ttk.Scale(
            self.main_frame,
            from_=0.5,
            to=1.0,
            orient=tk.HORIZONTAL,
            length=200,
            variable=self.circularity_var,
            command=self.update_circularity_label
        )
        self.circularity_slider.grid(row=1, column=1, sticky=tk.W, pady=5)

        self.circularity_value_label = ttk.Label(self.main_frame, text="0.87")
        self.circularity_value_label.grid(row=1, column=2, sticky=tk.W,
pady=5)

        # Image display
        self.image_label = ttk.Label(self.main_frame)
        self.image_label.grid(row=2, column=0, columnspan=3, pady=10)

        # Results display
        self.result_text = tk.Text(self.main_frame, height=5, width=40)
        self.result_text.grid(row=3, column=0, columnspan=3, pady=5)

        self.current_image = None
        self.processed_image = None

    def update_circularity_label(self, value):
        """Update the label to show current circularity threshold"""
        formatted_value = "{:.2f}".format(float(value))
        self.circularity_value_label.config(text=formatted_value)

    def load_image(self):
        file_path = filedialog.askopenfilename(
            filetypes=[("Image files", "*.png *.jpg *.jpeg *.bmp *.gif *.tiff")]
```

```
        )
        if file_path:
            self.current_image = cv2.imread(file_path)
            self.display_image(self.current_image)

    def detect_nuts_and_washers(self, image):

        circularity_threshold = self.circularity_var.get()


        # Convert to grayscale and preprocess
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        blurred = cv2.GaussianBlur(gray, (11, 11), 0)
        blurred = cv2.medianBlur(gray, 7)


        # Adaptive threshold
        thresh = cv2.adaptiveThreshold(
            blurred, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
            cv2.THRESH_BINARY_INV, 7, 1
        )

        # Morphological operations to reduce noise and shadows
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
        thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)

        # Find contours and hierarchy
        contours, hierarchy = cv2.findContours(
            thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE
        )

        nut_count = 0
        washer_count = 0
        processed_image = image.copy()

        for i, contour in enumerate(contours):
            area = cv2.contourArea(contour)

            # Filter by area
            if area < 10 or area > 100000:
                continue

            perimeter = cv2.arcLength(contour, True)
            approx = cv2.approxPolyDP(contour, 0.03 * perimeter, True)

            # Bounding box and aspect ratio
            x, y, w, h = cv2.boundingRect(contour)
            aspect_ratio = w / float(h)

            # Check for nuts (hexagonal shape with inner circle)
            if len(approx) == 6 and 0.8 <= aspect_ratio <= 1.2:  # Hexagonal
outer shape
                # Check for inner hole (child contour)
                has_inner_circle = False
                for j, h in enumerate(hierarchy[0]):
                    if h[3] == i:  # Child contour
                        inner_area = cv2.contourArea(contours[j])
                        inner_perimeter = cv2.arcLength(contours[j], True)
                        circularity = (
                            4 * np.pi * inner_area / (inner_perimeter ** 2)
                            if inner_perimeter > 0
                            else 0
                        )
                        if inner_area > 10 and circularity > 0.7:  # Confirm circular
hole
                            has_inner_circle = True
                            break

                if has_inner_circle:
                    cv2.drawContours(processed_image, [contour], -1, (0, 0, 255),
2)  # Red
                    nut_count += 1
                    continue

            # Check for washers (circular with inner hole)
            circularity = 4 * np.pi * area / (perimeter ** 2) if perimeter > 0 else
0
            if circularity > circularity_threshold and 0.9 <= aspect_ratio <= 1.1:
                # Verify inner hole (child contour)
                has_inner_hole = False
                for j, h in enumerate(hierarchy[0]):
                    if h[3] == i:  # Child contour
                        inner_area = cv2.contourArea(contours[j])
                        inner_perimeter = cv2.arcLength(contours[j], True)
                        inner_circularity = (
                            4 * np.pi * inner_area / (inner_perimeter ** 2)
                            if inner_perimeter > 0
                            else 0
                        )
                        if inner_area > 10 and inner_circularity > 0.8:  # Confirm
circular hole
                            has_inner_hole = True
                            break

                if has_inner_hole:
                    cv2.drawContours(processed_image, [contour], -1, (0, 255, 0),
2)  # Green
                    washer_count += 1

        return processed_image, nut_count, washer_count

    def process_image(self):
        if self.current_image is None:
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(tk.END, "Please load an image first!")
            return

        self.processed_image, nut_count, washer_count =
self.detect_nuts_and_washers(self.current_image)
        self.display_image(self.processed_image)

        self.result_text.delete(1.0, tk.END)
        self.result_text.insert(tk.END, f"Detection Results:\nNuts:
{nut_count}\nWashers: {washer_count}")

    def display_image(self, cv_image):
        if cv_image is None:
            return

        # Resize image to fit display
        max_size = 500
        height, width = cv_image.shape[:2]

        if height > max_size or width > max_size:
            scale = max_size / max(height, width)
            new_width = int(width * scale)
            new_height = int(height * scale)
            cv_image = cv2.resize(cv_image, (new_width, new_height))

        # Convert CV2 image to PhotoImage
        image = cv2.cvtColor(cv_image, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(image)
        photo = ImageTk.PhotoImage(image)

        # Update label
        self.image_label.configure(image=photo)
        self.image_label.image = photo

def main():
    root = tk.Tk()
    root.title("Nut and Washer Detector")
    app = NutWasherDetector(root)
    root.mainloop()

if __name__ == "__main__":
    main()
```