

**DETECT FERTILIZED, NON-FERTILIZED EGGS AND CRACKED
EGGSHELL FROM CANDLED EGG IMAGES USING IMAGE
PROCESSING TECHNIQUES.**

A PROJECT REPORT SUBMITTED BY

W.A.A.N. WERAGODA

(S / 19 / 172)

in partial fulfillment of the requirement for the course of
CSC 3141 Image Processing Laboratory

to the
Department of Statistics and Computer Science

of the

FACULTY OF SCIENCE
UNIVERSITY OF PERADENIYA
SRI LANKA
2024

DECLARATION

I do hereby declare that the work reported in this project report was exclusively carried out by me under the supervision of Prof. Amalka Pinidiyaarachchi .It describes the results of my own independent work except where due reference has been made in the text. No part of this project report has been submitted earlier or concurrently for the same or any other degree.

Amali.

Date: 2024/09/16

Signature of the Candidate

Certified by:

1. Instructor: **Mr. Kansaji Kotuwage**

Date:

Signature:.....

2. Supervisor: **Prof. Amalka Pinidiyaarachchi**

Date:

Signature:.....

3. Head of the Department: **Dr. Sachith Abeysundara**

Date:

Signature:.....

ABSTRACT

The detection of fertilized and non-fertilized eggs, along with the identification of cracked eggshells, is very important in the industry for quality control and incubation success. This project aims to automate the process of detecting these characteristics using image processing techniques applied to candled egg images. The process involves several stages, including image acquisition, preprocessing through contrast enhancement, noise reduction, and binary thresholding using Otsu's method. Cracked eggshells are identified through edge detection and contour analysis, while fertilized eggs are detected by analyzing the mean pixel intensity within isolated egg regions. The system classifies eggs as fertilized or non-fertilized and highlights visible cracks, offering a solution for egg quality inspection.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to everyone who supported me throughout the course of this project.

TABLE OF CONTENTS

DECLARATION	2
ABSTRACT.....	1
INTRODUCTION	4
1.1. INTRODUCTION	4
1.2. PROBLEM STATEMENT	4
1.3. SOLUTION	4
RELATED WORK	5
METHODOLOGY	6
3.1. IMAGE PROCESSING PIPE LINE	6
3.2. USED FUNCTION AND THEIR APPLICATIONS	8
3.3. USED TOOLS TECHNOLOGIES AND LIBRARIES	9
3.4. PROCEDURE	10
3.5. PROCEDURE EXPLAINED	12
RESULTS AND DISCUSSION	20
4.1. MAJOR RESULTING STEPS	20
4.2. SOME TESTED IMAGE RESULTS	21
4.3. DISCUSSION	25
CONCLUSIONS.....	26
5.1. ADVANTAGES OF THE IMPLEMENTED SYSTEM	26
5.2. ISSUES	26
5.3. CONCLUSION	26
REFERENCES	27

LIST OF FIGURES

Figure No	Title	Page No
Figure 1	Method of image processing techniques for detecting fertility	6
Figure 2	Method of image processing techniques for crack detection in eggshells	6
Figure 3	Apparatus for image acquisition	10
Figure 4	Original Image	12
Figure 5	Isolated Egg Image	12
Figure 6	Enhanced Egg Image	13
Figure 7	Image after Noise Removal	13
Figure 8	Image after applying Otsu's Threshold	14
Figure 9	Non-fertilized Egg	15
Figure 10	Fertilized Egg	15
Figure 11	Red & Green Channel	16
Figure 12	Blurred Red Channel	16
Figure 13	Binary Image (Red)	16
Figure 14	Edge Detection	17
Figure 15	After Morphological Closing	17
Figure 16	After Bitwise Operation	17
Figure 17	Detected Crack on the Egg	18
Figure 18	outputs of some major resulting steps of fertility detector	20
Figure 19	outputs of some major resulting steps of crack detector	21

LIST OF TABLES

Table No	Title	Page No
Table 1	Results of Fertilized and Non-Fertilized Egg Detection	21
Table 2	Results of Crack Detection	23

CHAPTER 01

INTRODUCTION

1.1. INTRODUCTION

This project uses image processing techniques to provide an automated solution for analyzing candled egg images, ensuring better accuracy and efficiency in egg inspection. Currently, identifying fertilized and non-fertilized eggs, as well as detecting cracked eggshells, is done manually on small farms. The process is labor intensive and prone to human error. Therefore, automated egg quality control has become essential in the industry.

1.2. PROBLEM STATEMENT

The traditional methods of inspecting eggs for fertilization status and cracks are labor intensive, time consuming, and lead to errors when the yield is high. These challenges can result in inconsistent quality control, leading to the misclassification of eggs. To avoid these issues and improve productivity, there is a need for a more efficient and accurate inspection process.

1.3. SOLUTION

This project proposes an automated egg analysis system that utilizes image processing techniques to detect fertilized and non-fertilized eggs and identify cracks in eggshells. By using methods such as thresholding, edge detection, contrast enhancement and contour analysis, the system enhances accuracy and efficiency in egg quality inspection while minimizing human error.

CHAPTER 02

RELATED WORK

The detection of fertilized and non-fertilized eggs, along with the identification of cracked eggshells, has been a significant area of research in the agricultural industry. Several studies have explored different approaches and techniques to automate this process.

Previous Research

- **Image-based methods:** Researchers have employed various image processing techniques, such as thresholding, edge detection, and feature extraction, to analyze egg images and detect abnormalities.[Alvin S. Alon et. al., International Journal of Advanced Trends in Computer Science and Engineering, 8(6),November - December 2019, 2794 - 2799]
- **Deep learning models:** Recent advancements in deep learning have led to the application of convolutional neural networks (CNNs) for egg quality inspection. CNNs have shown promising results in accurately classifying eggs based on their visual characteristics. [Saifullah, Shoffan & Drezewski, Rafal & Yudhana, Anton & Pranolo, Andri & Kaswijanti, Wilis & Suryotomo, Andiko & Putra, Seno & Alin, Khaliduzzaman & Prabuwo, Anton Satria & Japkowicz, Nathalie. (2023). Nondestructive Chicken Egg Fertility Detection Using CNN- Transfer Learning Algorithms. Jurnal Ilmiah Teknik Elektro Komputer dan Informatika. 9. 854-871. 10.26555/jiteki.v9i3.26722.]

CHAPTER 03

METHODOLOGY

3.1. IMAGE PROCESSING PIPE LINE

Fertilized and non-fertilized egg detection

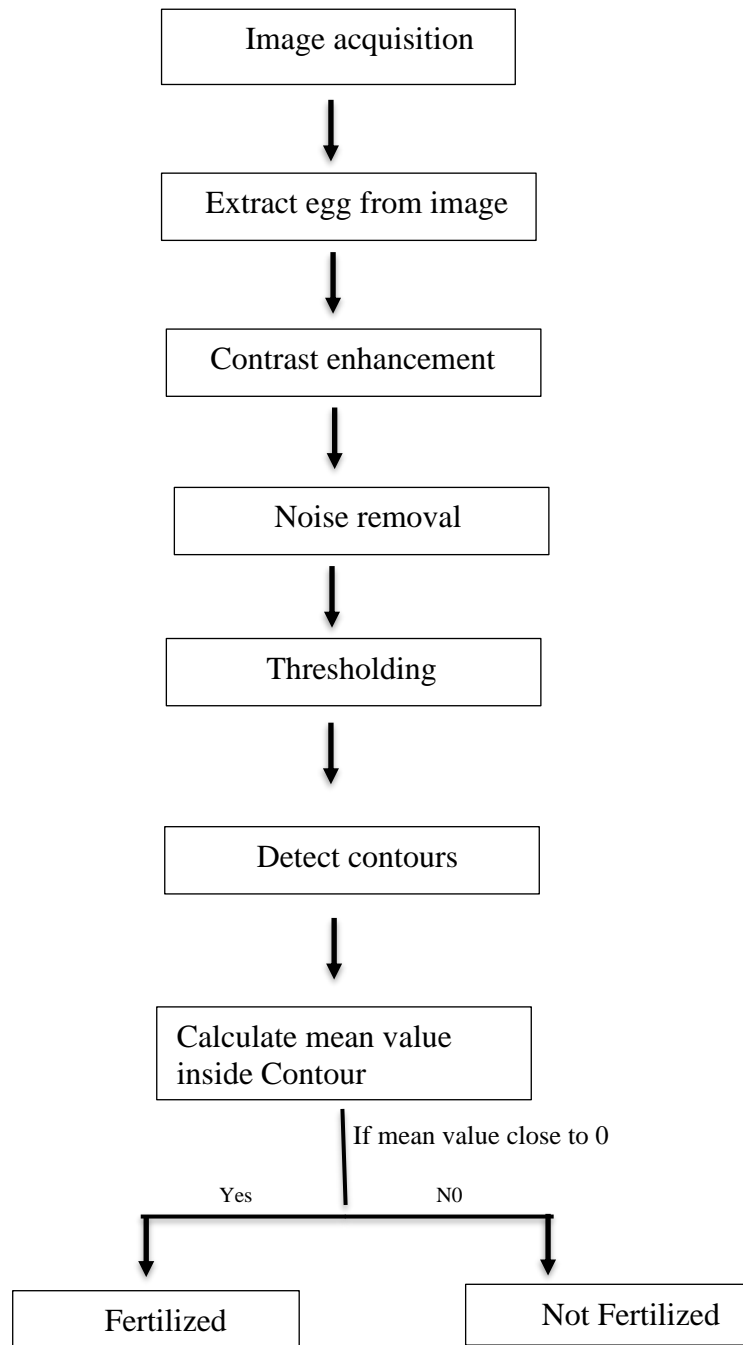


Figure 1. Method of image processing techniques for detecting fertility

Cracked egg shell detection

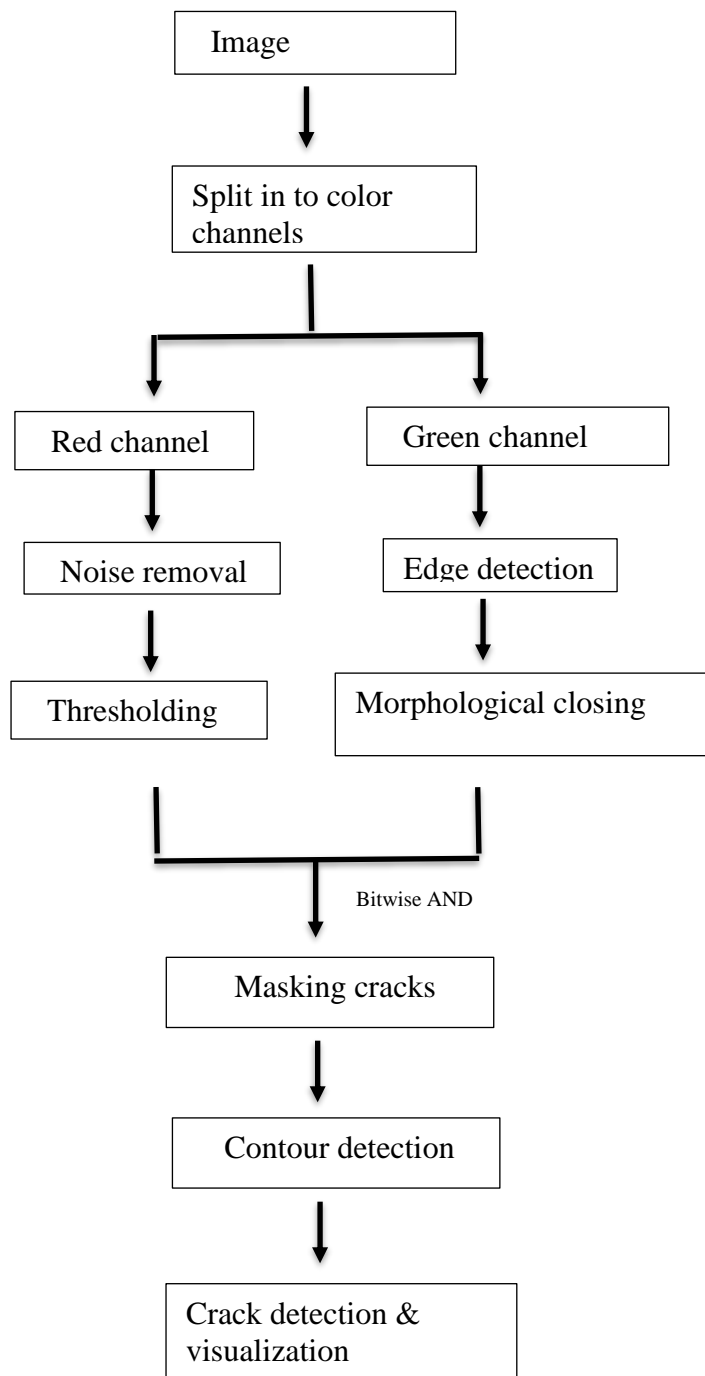


Figure 2. Method of image processing techniques for detect crack in eggshell

3.2. USED FUNCTION AND THEIR APPLICATIONS

cv2.imread()

Syntax: image = cv2.imread(filename, flags)

Application: Loads the image from the specified path.

cv2.cvtColor()

Syntax: converted_image = cv2.cvtColor(src, code)

Application: Converts images between different color spaces (BGR, grayscale, etc.).

cv2.split()

Syntax: b, g, r = cv2.split(image)

Application: Splits a multi-channel image (e.g., BGR or RGB) into its individual color channels

cv2.threshold()

Syntax: retval, dst = cv2.threshold(src, thresh, maxval, type)

Application: Converts a grayscale image to a binary image by thresholding pixel values.

cv2.Canny()

Syntax: edges = cv2.Canny(image, threshold1, threshold2)

Application: Detects edges in an image using the Canny edge detection algorithm. It finds areas of rapid intensity change, typically indicating boundaries or edges.

cv2.subtract()

Syntax: result = cv2.subtract(image1, image2)

Application: Subtracts one image from another or removes parts of an image. This can highlight differences between images or remove unwanted portions of an image.

cv2.bitwise_and()

Syntax: result = cv2.bitwise_and(image1, image2, mask=mask)

Application: Performs a bitwise AND operation on two images or an image and a mask, typically used to apply masks to images.

cv2.findContours()

Syntax: contours, hierarchy = cv2.findContours(image, mode, method)

Application: Finds contours (i.e., boundaries of objects) in a binary image.

cv2.drawContours()

Syntax: cv2.drawContours(image, contours, contourIdx, color, thickness)

Application: Draws detected contours on an image for visualization purposes.

cv2.boundingRect()

Syntax: x, y, w, h = cv2.boundingRect(array)

Application: Finds the smallest rectangle that can enclose a contour.

cv2.createCLAHE()

Syntax: output = cv2.medianBlur(src, ksize)

Application: Enhances the contrast of an image using adaptive histogram equalization, specifically useful for highlighting finer details.

cv2.medianBlur()

Syntax: `output = cv2.medianBlur(src, ksize)`

Application: Removes salt-and-pepper noise from an image while preserving edges, making it suitable for image preprocessing.

cv2.GaussianBlur()

Syntax: `output = cv2.GaussianBlur(src, ksize, sigmaX)`

Application: Smoothens an image by reducing noise and detail. It is often used as a preprocessing step to make the image easier to analyze.

cv2.mean()

Syntax: `mean_value = cv2.mean(src, mask=None)`

Application: Calculates the average color or intensity of a specified area in the image.

plt.imshow()

Syntax: `plt.imshow(image, cmap=None)`

Application: Displays an image in a new window or plot using Matplotlib.

plt.subplot()

Syntax: `plt.subplot(nrows, ncols, index)`

Application: Creates a grid of plots where each subplot shows different stages of the image processing pipeline or comparison between original and processed images.

plt.text()

Syntax: `plt.text(x, y, text, fontsize=12, ha='center', va='center')`

Application: Adds text to a plot at specified coordinates. It is useful for labeling, annotating, or adding information to visualizations.

os.listdir()

Syntax: `file_list = os.listdir(path)`

Application: Lists all files in a specified directory.

os.path.join()

Syntax: `full_path = os.path.join(path1, path2, ...)`

Application: Constructs a valid file path by combining directory and file names.

3.3. USED TOOLS TECHNOLOGIES AND LIBRARIES

- OpenCV (Used for image processing and computer vision tasks.)
- NumPy (Handles numerical operations and image data as arrays.)
- Matplotlib (Visualizes images and data.)
- Operating System (OS) Library (Manages file and directory interactions.)
- Python

3.4. PROCEDURE

Image Acquisition :

Used candled egg images. Egg candling is a non-destructive procedure that consists on applying light against an egg. Images were acquired using a mobile phone camera.

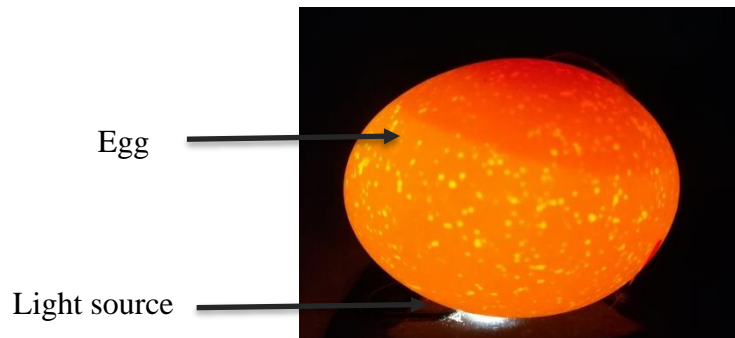


Figure 3. Apparatus for image acquisition

For Fertilized and Non-Fertilized egg detection

- This process is designed to identify the region in the image where the embryo is located and verify whether the egg is fertilized

Egg Extraction from Image:

The image is converted to grayscale, and the egg is isolated from the original image.

Contrast Enhancement:

The contrast of the isolated egg image is improved using CLAHE (Contrast Limited Adaptive Histogram Equalization).

Noise Removal:

A median filter with a (3x3) kernel and Gaussian filters are applied to reduce noise in the image.

Thresholding:

After noise removal, Otsu's method is used for binary thresholding.

Contour Detection:

Contours in the binary image are found, and a mask is created for the contour to isolate the region inside the contour.

Fertilization Status Check:

The grayscale image within the masked area is analyzed, and the mean pixel value for the region is computed. 0 (indicating a dark region) is suggestive of the presence of a fertilized egg.

For cracks detection

Color Channel Separation:

The image is split into its color channels: blue, green, and red.

Red Channel Processing:

Gaussian Blur is applied to the red channel to remove noise and smooth the image.

Binary Mask Creation:

A binary mask is created by thresholding the blurred red channel.

Edge Detection:

Canny edge detection is applied to the green channel to detect edges in the image.

Morphological Closing:

Morphological closing is applied to the detected edges using a (3x3) kernel to close small gaps.

Crack Masking:

A mask for cracks inside the egg is created using bitwise operations on the morphologically closed image.

Contour Detection:

Contours are found in the masked image where cracks are detected.

Contour Filtering:

Contours are filtered based on their area to identify significant cracks.

Crack Drawing:

The detected cracks are drawn on a copy of the original image.

Result Display:

The original image and the image with detected cracks are displayed. If no cracks are detected, a message is shown indicating that no cracks were found.

3.5. PROCEDURE EXPLAINED

For Fertilized and Non-Fertilized egg detection

Necessary libraries were imported

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
```

Egg sub image was created

```
def extract_egg_subimage(image_path):
    img = cv2.imread(image_path)

    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply threshold
    _, binary_mask = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY)

    # Find contours
    contours, _ = cv2.findContours(binary_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if contours:
        largest_contour = max(contours, key=cv2.contourArea)

        x, y, w, h = cv2.boundingRect(largest_contour)

        # Extract the sub-image (ROI) from the original color image
        egg_subimage = img[y:y + h, x:x + w]

        return egg_subimage
    else:
        print("No egg found in the image.")
        return None
```

original image



isolated egg



Figure 4. Original Image

Figure 5 . Isolated egg Image

The contrast of the isolated egg image is improved using CLAHE (Contrast Limited Adaptive Histogram Equalization).

```
def enhance_contrast(image):  
    # Apply CLAHE  
    image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)  
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))  
    enhanced_image = clahe.apply(image)  
    return enhanced_image
```

First image was converted to grayscale image and then used CLAHE. Lower clipLimit values reduce contrast, while higher values increase it. A smaller tile size increases local contrast adjustments, while a larger tile size reduces the effect.

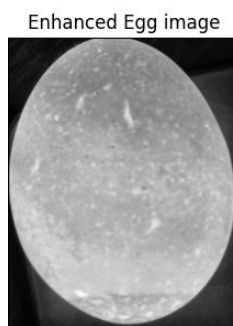


Figure 6. Enhanced egg Image

A median filter and Gaussian filters are applied to reduce noise in the image.

```
def remove_noise(image):  
    # Apply median filter to remove noise  
    denoised_image = cv2.medianBlur(image, ksize=3)  
  
    # Apply Gaussian smoothing for further noise reduction  
    smoothed_image = cv2.GaussianBlur(denoised_image, (5, 5), sigmaX=0)  
  
    return smoothed_image
```

used (3x3) kernel for meadinBlur function. The median blur is effective in removing salt and pepper noise by replacing each pixel's value with the median of the neighboring pixels. Applies Gaussian smoothing with a 5x5 kernel to the denoised image for further noise reduction.

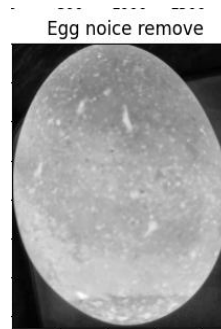


Figure 7. Image after Noise removal

Otsu's Thresholding was to convert the smoothed image into a binary image.

```
def otsu_threshold(smoothed_image):
    _, binary_image = cv2.threshold(smoothed_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    return binary_image
```

Otsu's method automatically finds the best threshold value for separating foreground from background. Converts pixel values; those below the threshold become black (0), and those above become white (255).

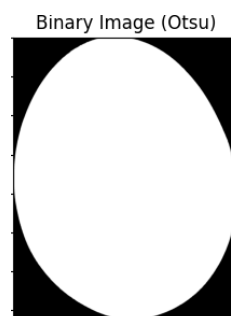


Figure 8. Image after applying Otsu's threshold

Fertilized and non-fertilized eggs were detected using a thresholded image based on the presence of the embryo. If the embryo exists, it shows a black spot inside the egg


```
def is_fertilized(thresh,image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if not contours:
        return False

    # Analyze contours
    for contour in contours:
        # Create a mask for the current contour
        mask = np.zeros_like(gray)
        cv2.drawContours(mask, [contour], -1, 255, thickness=cv2.FILLED)

        # Compute the mean color inside the contour area
        mean_value = cv2.mean(gray, mask=mask)

        # The mean value should be close to 0 for a fertilized egg (black area)
        if mean_value[0] < 40:
            return True

    return False
```

Firstly, blank mask (all pixels set to 0) was created with the same dimensions as the grayscale image of the input image. Then contour was drawn on the mask making the area inside the contour white (255) while keeping everything outside as black (0). After that the mean value of the pixel intensities was computed inside the contour area defined by the mask. In a grayscale image, lower values represent darker areas. This threshold is used to determine if the area inside the contour is dark enough to classify it as a fertilized egg. If the mean intensity is low (close to 0), it indicates that the contour area is mostly dark, which is expected for a fertilized egg.

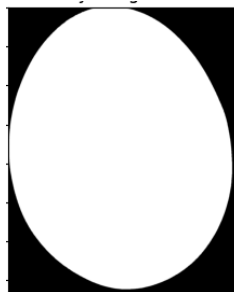


Figure 9. Not fertilized

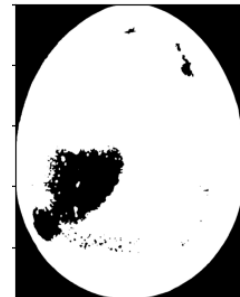


Figure 10. Fertilized

Input folder of images

```
image_folder = 'imageSet01'
image_files = [f for f in os.listdir(image_folder) if f.endswith((''.jpg', '.jpeg', '.png', '.webp'))]

image_paths = [os.path.join(image_folder, file) for file in image_files]

process_multiple_images(image_paths)
```

Plotting images

```
def process_multiple_images(image_paths):
    for idx, image_path in enumerate(image_paths):
        isolated_egg = extract_egg_subimage(image_path)
        if isolated_egg is None:
            continue

        enhanced_egg_subimage = enhance_contrast(isolated_egg)
        final_egg_subimage = remove_noise(enhanced_egg_subimage)
        binary_result = otsu_threshold(final_egg_subimage)

        is_fertilized_result = is_fertilized(binary_result, cv2.imread(image_path))
        fertilized_status = "Fertilized" if is_fertilized_result else "Not Fertilized"

        img = cv2.imread(image_path)

        # Plot the results
        plt.figure(figsize=(12, 6))

        plt.subplot(1, 3, 1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.title("Original Image")

        plt.subplot(1, 3, 2)
        plt.imshow(cv2.cvtColor(isolated_egg, cv2.COLOR_BGR2RGB))
        plt.title("Isolated Egg Subimage")

        plt.subplot(1, 3, 3)
        plt.imshow(binary_result, cmap='gray')
        plt.title(f"Thresholded Image\n({fertilized_status})")

        plt.suptitle(f"Egg Analysis {idx+1} - {fertilized_status}", fontsize=16)
        plt.show()
```

For cracks detection

Necessary libraries were imported

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
```

The image was split into three color channels, and then the red and green channels were used for further processing

```
# Split into color channels
b, g, r = cv2.split(image)

gray_red = r
gray_green = g
```

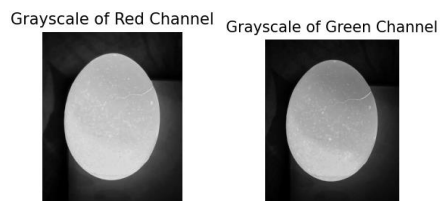


Figure 11. Red & Green channel

Noise was removed and applied binary thresholding to the red channel

```
# Apply Gaussian Blur to the red channel
blurred_red = cv2.GaussianBlur(gray_red, (11, 11), 0)

# Threshold the blurred red channel to create a binary mask for the egg
_, binary_image = cv2.threshold(blurred_red, 150, 255, cv2.THRESH_BINARY)
```

Gaussian blur was applied to the red channel with a kernel size of 11×11 to reduce noise and smooth the image. Then the blurred red channel image was converted to a binary image.

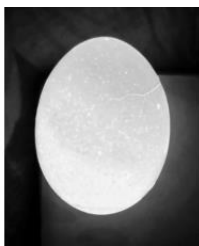


Figure 12. Blurred red channel

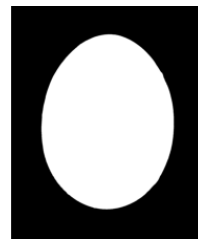


Figure 13. Binary image(red)

Canny edge detection and morphological closing was applied to the green channel

```
edges = cv2.Canny(gray_green, 50, 250)

# Apply morphological closing to the edges to close small gaps
kernel = np.ones((3, 3), np.uint8)
morph_closed = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel)
```

3×3 structuring element was used for morphological operations. The morphological closing operation was applied to the detected edges. Closing helps to close small gaps and holes in the edge contours by eroding the image and then dilating it. This makes the edges more continuous and solid.

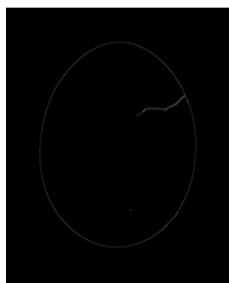


Figure 14. Edge detection



Figure 15. After applying morphological closing

Applied binary mask to edge image

```
masked_cracks_inside = cv2.bitwise_and(morph_closed, morph_closed, mask=binary_image)
```

Bitwise AND operation was applied between the result of morphological closing and itself, using binary image as a mask. This operation effectively isolates the cracks within the binary mask.



Figure 16. After applying bitwise and operation

contours were detected and Filtered them to keep only those whose area is greater than the minimal value. This removes small contours that are likely noise, retaining only significant cracks.

```
# Find contours of cracks inside the egg
contours_inside, _ = cv2.findContours(masked_cracks_inside, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Filter contours
min_area = 500
crack_contours_inside = [c for c in contours_inside if cv2.contourArea(c) > min_area]

crack_image_inside = image.copy()

# Draw the detected cracks
if crack_contours_inside:
    cv2.drawContours(crack_image_inside, crack_contours_inside, -1, (0, 0, 255), 3) # Draw inside cracks in red
    crack_detected = True
else:
    crack_detected = False
```

A copy of the original image was Created to draw contours on, preserving the original image.



Figure 17. Detected crack on the egg

Input image folder

```
input_folder = "imagest02"

image_files = [f for f in os.listdir(input_folder) if os.path.isfile(os.path.join(input_folder, f))]

for image_file in image_files:
    # Load the image
    image_path = os.path.join(input_folder, image_file)
    image = cv2.imread(image_path)

    if image is None:
        print(f"Error loading image {image_file}")
        continue

    # Split into color channels
    b, g, r = cv2.split(image)
```

Plotting images

```
if crack_detected:
    plt.subplot(1, 2, 2)
    plt.imshow(cv2.cvtColor(crack_image_inside, cv2.COLOR_BGR2RGB))
    plt.title("Detected Cracks In Egg")
    plt.axis('off')
else:
    plt.subplot(1, 2, 2)
    plt.text(0.5, 0.5, 'No Crack Detected In Egg', fontsize=12, ha='center', va='center')
    plt.title("Crack Detection Result")
    plt.axis('off')

plt.show()
```

CHAPTER 04

RESULTS AND DISCUSSION

4.1. MAJOR RESULTING STEPS

Fertilized and Non-Fertilized egg detection

- **Image Loading:** Load the image using `cv2.imread`.
- **Egg Extraction:** Convert to grayscale, threshold, and extract the egg subimage using contours.
- **Contrast Enhancement:** Enhance image contrast using CLAHE.
- **Noise Removal:** Apply median and Gaussian filters to reduce noise.
- **Thresholding:** Use Otsu's method to convert the image to a binary format.
- **Fertilization Detection:** Analyze contours and mean values to check for fertilization.
- **Batch Processing:** Iterate through multiple images, applying the above steps.
- **Result Visualization:** Display original, isolated, and thresholded images with fertilization status using `matplotlib`.

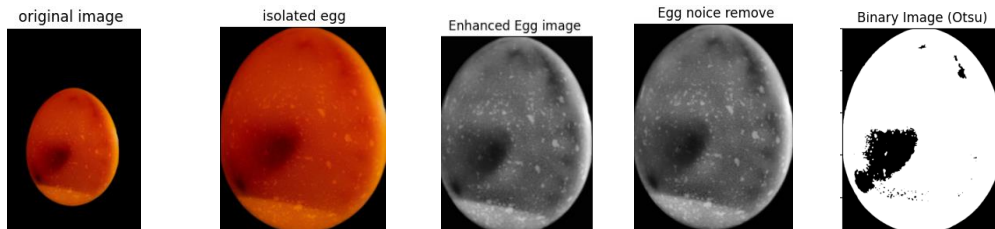


Figure 18. outputs of some major resulting steps of fertility detector

cracks detection

- **Image Loading:** Load images from a folder using `cv2.imread`.
- **Color Channel Splitting:** Split the image into blue, green, and red channels with `cv2.split`.
- **Red Channel Processing:** Apply Gaussian blur to the red channel and use thresholding to create a binary mask for the egg.
- **Edge Detection:** Detect edges in the green channel using the Canny edge detector.
- **Morphological Operations:** Apply morphological closing on the edges to remove small gaps.
- **Mask Application:** Use `cv2.bitwise_and` to isolate the cracks within the egg using the binary mask.

- **Contour Detection:** Find contours of the cracks inside the egg and filter them based on a minimum area.
- **Crack Drawing:** If cracks are detected, draw the contours on the image; otherwise, indicate that no crack was found.
- **Result Visualization:** Display the original image and crack detection results using matplotlib.

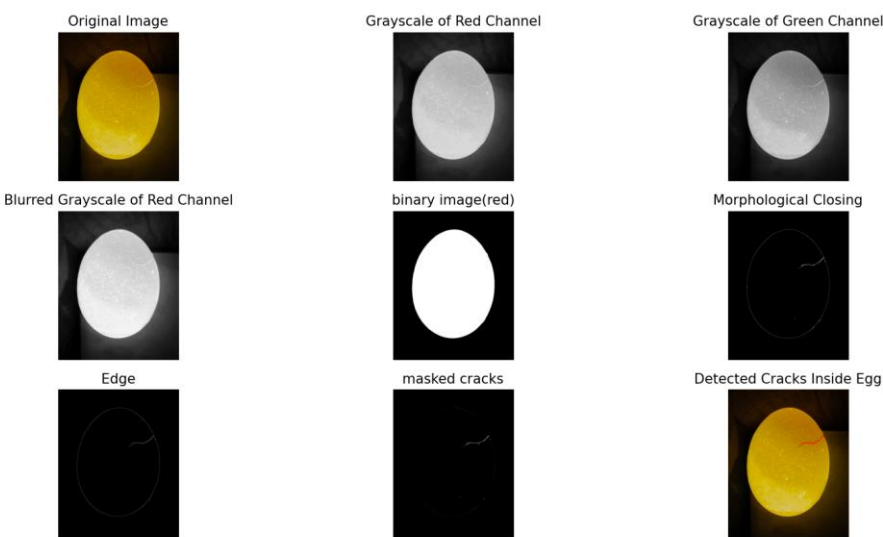
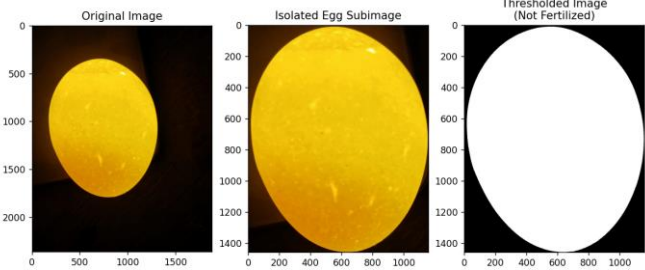
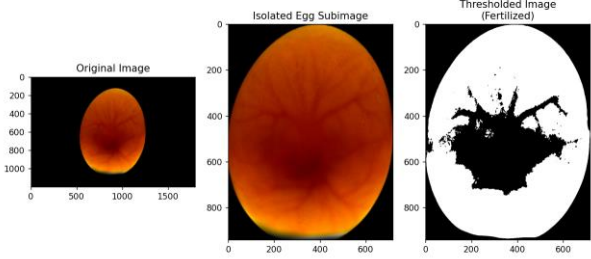
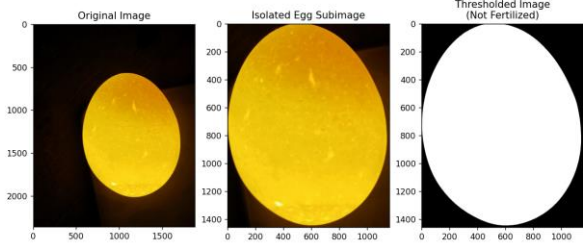
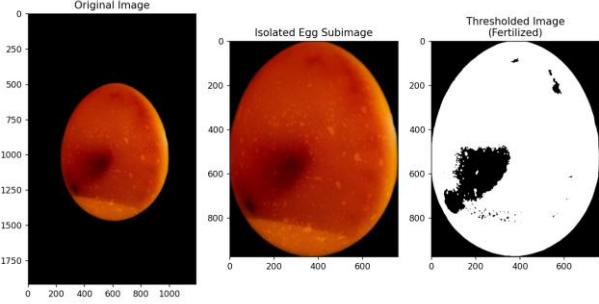


Figure 19. outputs of some major resulting steps of crack detector

4.2.SOME TESTED IMAGE RESULTS

Fertilized and Non-Fertilized egg detection

Output	Correct Result
	Not Fertilized

<p>Egg Analysis 2 - Not Fertilized</p> 	<p>Not Fertilized</p>
<p>Egg Analysis 3 - Fertilized</p> 	<p>Fertilized</p>
<p>Egg Analysis 4 - Not Fertilized</p> 	<p>Not Fertilized</p>
<p>Egg Analysis 5 - Fertilized</p> 	<p>Fertilized</p>

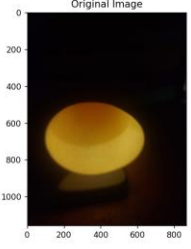
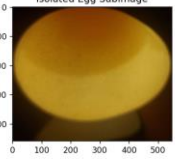
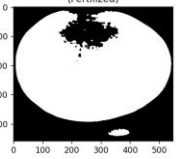
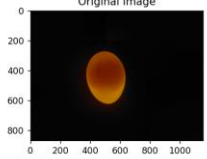
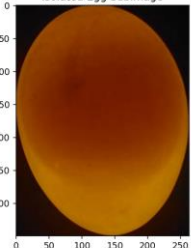
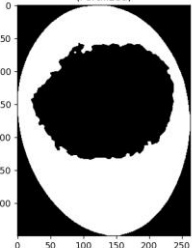


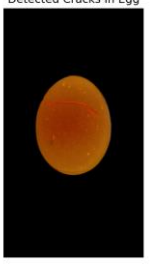
<p style="text-align: center;">Egg Analysis 6 - Fertilized</p> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;"> <p>Original Image</p>  </div> <div style="text-align: center;"> <p>Isolated Egg Subimage</p>  </div> <div style="text-align: center;"> <p>Thresholded Image (Fertilized)</p>  </div> </div>	Fertilized
<p style="text-align: center;">Egg Analysis 7 - Fertilized</p> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;"> <p>Original Image</p>  </div> <div style="text-align: center;"> <p>Isolated Egg Subimage</p>  </div> <div style="text-align: center;"> <p>Thresholded Image (Fertilized)</p>  </div> </div>	Fertilized

Table 01. Results of fertilized and non-fertilized egg detection

cracks detection

Output	Correct Result
<p style="text-align: center;">Crack Detection Result</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Original Image: c1.jpg</p>  </div> <div style="text-align: center;"> <p>No Crack Detected In Egg</p> </div> </div>	Not crack
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Original Image: c2.jpg</p>  </div> <div style="text-align: center;"> <p>Detected Cracks In Egg</p>  </div> </div>	crack






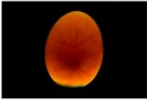
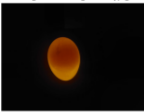
<p>Crack Detection Result</p> <p>Original Image: c3.jpg</p>  <p>No Crack Detected In Egg</p>	<p>Not crack</p>
<p>Original Image: c4.jpg</p>  <p>Detected Cracks In Egg</p> 	<p>crack</p>
<p>Original Image: c5.jpg</p>  <p>Detected Cracks In Egg</p> 	<p>crack</p>
<p>Crack Detection Result</p> <p>Original Image: f2.jpeg</p>  <p>No Crack Detected In Egg</p>	<p>Not crack</p>
<p>Crack Detection Result</p> <p>Original Image: f9.jpg</p>  <p>No Crack Detected In Egg</p>	<p>Not crack</p>

Table 01. Results of crack detection

4.3.DISCUSION

Based on the result detecting fertilized versus non-fertilized eggs and identifying cracks can be done effectively using image processing techniques but also highlight areas for improvement. For fertilized egg detection, while the overall methodology is effective, some misclassifications suggest that further refinement in thresholding ,contrast adjustment, accurate determination of the minimum mean value is needed. Similarly, in crack detection, although the process successfully identifies most cracks, there are instances of missed detections, indicating a need for more precise parameter tuning, adjusting the minimum area threshold for contour detection and noise reduction for accurately identifying cracks and avoiding missed detections. Continuous testing and refinement will be key to achieving optimal performance. However this project successfully demonstrates the application of image processing techniques to detect cracks and fertilization in eggs. It provides a foundation for automating egg quality inspection processes.Future developments could include integration with automated sorting systems and the application of machine learning techniques to further improve detection accuracy and adapt to new types of defects.

CHAPTER 05

CONCLUSIONS

5.1. ADVANTAGES OF THE IMPLEMENTED SYSTEM

- The system provides automated detection of fertilized and non-fertilized eggs and cracks, reducing the need for manual inspection and reduce errors.
- This system utilizes simple image processing techniques, making it both cost effective and feasible.
- The system can achieve high accuracy when the conditions are well controlled.
- Unlike machine learning-based systems, this system does not require extensive training data, making it easier to set up and use.
- The system is scalable and can be adapted to handle larger volumes of images.

5.2.ISSUES

- Performance may degrade with changes in lighting or image quality.
- High noise levels could affect detection accuracy.
- Requires extensive manual adjustment of parameters.
- Requires proper adjustment in the minimum area threshold for contour detection and determination of the minimum mean value.
- Dirt can interfere with accurate detection of fertilized and non-fertilized eggs.
- Finding optimal threshold values for binary masking and edge detection can be challenging and may require extensive testing.

5.3.CONCLUSION

In conclusion, the project effectively utilizes simple image processing techniques for detecting fertilized and non-fertilized eggs and identifying cracks, demonstrating cost effectiveness and efficiency. The system also enhances accuracy in egg quality inspection while reducing human error. Although there are areas for improvement, such as refining detection parameters. Despite issues, the system provides a solid foundation for automated egg analysis with potential for further development.

REFERENCES

1. [OpenCV: OpenCV modules](#)
2. [\(PDF\) Chicken Egg Detection Based-on Image Processing Concept: A Review \(researchgate.net\)](#)
3. [Sensors | Free Full-Text | Development of an Early Embryo Detection Methodology for Quail Eggs Using a Thermal Micro Camera and the YOLO Deep Learning Algorithm \(mdpi.com\)](#)
4. [Crack on Eggshell Detection System Based on Image Processing Technique | IEEE Conference Publication | IEEE Xplore](#)
5. [\(PDF\) A Machine Vision System for Detecting Fertile Eggs in the Incubation Industry \(researchgate.net\)](#)
6. <https://iopscience.iop.org/article/10.1088/1742-6596/2261/1/012021/pdf#:~:text=The%20detection%20of%20a%20crack,in%20closed%20environments%20is%20fixed.>