Dissertation on

Measurement of Dimensions of object(s) in an Image using Image Processing and Machine Learning

Thesis submitted towards partial fulfillment of the requirements for the degree

Master of Technology in IT (Courseware Engineering)

Submitted by

Arijit Guria

EXAMINATION ROLL NO.: M4CWE24011

UNIVERSITY REGISTRATION NO.: 163780 of 2022-23

Under the guidance of

Mr. Joydeep Mukherjee

School of Education Technology

Jadavpur University

Course affiliated to

Faculty of Engineering and Technology

Jadavpur University

Kolkata-700032

INDIA

2024

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"**MEASUREMENT OF DIMENSIONS OF OBJECTS IN AN IMAGE USING IMAGE PROCESSING AND MACHINE LEARNING**"** is a bonafide work carried out by ARIJIT GURIA under our supervision and guidance for partial fulfillment of the requirements for the degree of Master of Technology in IT (Courseware Engineering) in School of Education Technology, during the academic session 2023-2024.

SUPERVISOR

School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR

School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN - FISLM

Jadavpur University,
Kolkata-700 032

M.Tech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination for evaluation of Thesis

_____

_____

_____

_____

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his **Master of Technology in IT (Courseware Engineering)** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME: ARIJIT GURIA

EXAMINATION ROLL NUMBER: M4CWE24011

UNIVERSITY REGISTRATION NO.: 163780 OF 2022-2023

THESIS TITLE: MEASUREMENT OF DIMENSIONS OF OBJECTS IN AN IMAGE USING IMAGE PROCESSING AND MACHINE LEARNING

SIGNATURE: DATE:

Acknowledgment

I feel extremely glad to be presenting this thesis at School of Education Technology, Jadavpur University, Kolkata, in the partial fulfillment of the requirements for the degree of Master of Technology in IT (Courseware Engineering).

I would like to express my sincere gratitude to my guide, Mr. Joydeep Mukherjee, for his vigilant supervision and constant encouragement. He spent her precious time reviewing the research work and provided many insightful comments and constructive criticism. Without his enthusiasm, advice and unique support this research work would never have become a reality.

I am also grateful to Prof. (Dr.) Matangini Chattopadhyay, Director of School of Education Technology, for her support, encouragement and timely advice. I am really indebted to Dr. Saswati Mukharjee for his continuous support during the entire course of the research.

Their advice and support were highly inspirational and motivating. I would also like to take this opportunity to thank all of my classmates from my Master of Technology in IT (Courseware Engineering) course who motivated me to complete my research work successfully. I do wish to thank all of our departmental support staff and all of those who were associated with this research contributed in some form or the others.

Lastly but not least, I would like to acknowledge and thank my mother for the strong, unconditional support, inspiration and encouragement that they have provided me, without which I would be unable to complete this research.

Arijit Guria

Examination Roll Number: M4CWE24011

University Registration No.: 163780 of 2022-2023

Thesis Title: Measurement of dimensions of objects in an image using image processing and machine learning

Date:

Contents

Title **Page**

List of Figures

List of Abbreviations

ML	Machine Learning
CV	Computer Vision
YOLO	You only live once
R-CNN	Region-based Convolution Neural Networks
SSD	Single Shot Detector
BS	Background Substraction

List of Tables

Executive Summery

In the traditional way, dimensions of object(s) from an image are measured using various image processing techniques and steps. This paper represents collaboration of Pre-trained ML model (reference model) and Modified OpenCV technique using image processing technique.

Measurement is not only crucial in science and the engineering but also in agriculture, drone industry, chemical industry, building, manufacturing industries, business and many more professions and pursuits technique that uses to measure an object's overall size. Contactless measurement is very crucial in space mission, maintaining safe distance in contaminant diseases etc.

This paper represents a comprehensive approach for contactless object dimension measurement using the OpenCV library, a widely adopted computer vision tool, and pre-trained ML model for better accuracy, better speedy measurement and lesser computer resources.

Subsequently, reference objects with known dimensions are used to calibrate the system, enabling the conversion of pixel measurements to real-world units. Calibration methods based on known reference objects to ensure accurate size estimation.

The program initially detects the object from a two-dimensional image using the ML model and then applies different mathematical operations to determine required object(s) borders and dimensions. If the detected object(s) is not known to the ML model then the proposed method uses image processing technique using OpenCV.

This research created an accurate, dependable, and cost-effective method for measuring item measures (width, and height). Experimental results are presented to validate the proposed approach's performance under various conditions, including object types, sizes, and environmental factors. The comparative analysis with existing methods demonstrates the superiority of the ML and OpenCV-based approach in terms of accuracy and computational efficiency.

Chapter 1

1. Introduction

1.1. Overview

Object detection is frequently utilized in industrial stages of product quality. An industrial quality control system can use the suggested system. Measurement of object size and approximate area are becoming more important in many applications, particularly mobile autonomous systems, space missions etc.

This paper comprises two primary components: object detection and dimension assessment.

For object(s) detection two techniques are used. One is pre-trained ML model and second image processing. The Canny edge detector is utilized to identify the object's edges. To accurately ascertain the object's edges, Gaussian blur and Otsu thresholding are applied. Dilation and Erosion, fundamental morphological operations, are employed to eliminate noise from the image and to bridge any gaps along the object's edges. The objective here is to determine the object's dimensions, specifically its height and width.

1.2. Problem Statement

In traditional way, for measurement of object dimension, the following limitations are found-

- Exact object not detected i.e. detector took surrounding area as an object
- For calculation of area of a hollow object, interior hollow area not subtracted during calculation of area.

1.3. Objectives

- To minimize effect of lighting error while capturing image using camera.
- To maintain consistency across asymmetric objects also.
- Improve accuracy, object detection rate through this model.

1.4. Assumptions and Scope

1.4.1. Assumption

- Image must be good lighting condition
- Image should be taken almost 90 degree with object position
- Image should contain reference object (whose dimension is already known)

1.4.2. Scope

Contactless height and width measurement is a versatile technology with applications across various fields. Here's a brief explanation for each area:

For Industry

In industrial settings, contactless measurement systems are used for:

- **Quality Control:** Ensuring products meet specified dimensions without physical contact, which is crucial for delicate or high-speed production lines.
- **Automation:** Integrating with robotic systems to measure and sort items quickly and accurately.
- **Safety:** Reducing the need for manual measurements, thereby minimizing human error and enhancing worker safety.

For Medical Science

In medical science, contactless measurement technologies are applied for:

- **Patient Monitoring:** Measuring body dimensions such as height and limb length without physical contact, which is particularly useful for patients with mobility issues.
- **Surgical Planning:** Providing precise measurements for pre-surgical planning and custom prosthetics.
- **Telemedicine:** Enabling remote assessments where physical measurements are not feasible.

For Space Missions

In space missions, contactless measurement is critical due to the unique environment:

- **Equipment Monitoring:** Measuring dimensions of equipment and components in microgravity without physical contact, which can be challenging with traditional methods.
- **Astronaut Health:** Monitoring astronauts' body dimensions to track health changes over time.
- **Robotic Exploration:** Assisting robotic systems in measuring and navigating through unknown terrains on other planets.

These applications highlight the importance and versatility of contactless measurement technologies in enhancing efficiency, safety, and precision across various fields.

2. Organization of Thesis

➢ Chapter 1: Introduction

This chapter provides a comprehensive overview of the thesis. It includes:

- ❖ **Introduction:** A brief introduction to the topic, setting the stage for the research.
- ❖ **Objective of Thesis:** Clearly defined goals and objectives of the research.
- ❖ **Assumptions and Scope:** The assumptions made during the research and the scope of the study.

➢ Chapter 2: Literature Survey .

This chapter reviews existing literature related to the research topic. It includes:

- ❖ **Survey of Existing Literature:** A detailed survey of previous work on dimension measurement and image processing.
- ❖ **Discussion of Previous Work:** Analysis of previous research using OpenCV and MediaPipe for similar applications, highlighting gaps and areas for improvement.

➢ Chapter 3:Motivation and Background Study:

Reasons for choosing the topic, the motivation behind the research, and a summary of the background study that led to the research question.

➢ Chapter 4: Proposed Approach

This chapter outlines the proposed methodology for the research. It includes:

- ❖ **ML Model and OpenCV Techniques:** Description of the machine learning model and image processing techniques used.
- ❖ **Methods for Measuring Dimensions:** Detailed explanation of the methods used for dimension measurement using image processing.
- ❖ **Implementation of OpenCV and MediaPipe:** Step-by-step process of how OpenCV and MediaPipe were implemented in the research.

➢ Chapter 5: Results

This chapter presents the results of the proposed method. It includes:

- ❖ **Output of the Proposed Method:** Detailed presentation of the results obtained from the implementation of the proposed approach, including any relevant data, graphs, and visualizations.

➢ Chapter 6: Conclusion and Future Work

This chapter summarizes the findings and suggests future research directions. It includes:

- ❖ **Conclusion:** Summary of the research findings and their implications.
- ❖ **Future Development:** Suggestions for future research and potential improvements to the proposed method.

➢ References

This chapter lists all the references and sources used throughout the thesis. It includes:

- ❖ **References and Website Content:** Comprehensive list of all the books, articles, websites, and other sources cited in the thesis.

➢ Appendix

This chapter includes supplementary material that supports the thesis. It includes:

- ❖ **Appendix of Codes:** All the code used in the research, providing detailed documentation and explanations for each part.

Chapter 2

2. Literature Survey

N. A. Othman, M. U. Salur, M. Karakose and I. Aydin [1] proposed a measurement system of real-time object detection and measurement of its size. The proposed methodology involves capture image, object identification (gray scaling, blurring, non-maximum suppression, dilation and erosion), object measurement, save to local storage and then cross-check with actual size and proposed size.

R. A. Joshi, S. N. Helambe, & R.R.Deshmukh [5] proposed a surface area measurement system using digital image processing. Their proposed system calculated area values obtained for various regular and irregular objects. Then proposed area was compared along with its actual manual area values obtained from mathematical formulas. It also clarifies that the distance factor does not affect the accuracy of result.

C. Lü, H. Ren, Y. Zhang and Y. Shen [6] proposed a leaf area measurement system based on image processing technique. The proposed technique involves geometric distortion correction, image segmentation, contour extraction, region filling and leaf area calculation. This method give output with absolute error about 2 to 3 cm^2

Erbing Yang, Meiqing Wang, Hang Cheng, Rong Liu, Fei Chen [7] aimed to Improve the Precision of 2-Dimensional Size Measurement of Objects through Image Processing. The key purpose was to simplify the measurement process and consider a measurement model suitable for most nonprofessional cameras such as mobile phones, without calculating the internal and external parameters of the camera. In this paper, the influence of the camera position was eliminated by setting four reference objects at four corners.

B. M U, H. Raghuram and Mohana [8] introduced a “Real Time Object Distance and Dimension Measurement using Deep Learning and OpenCV” model. Several functions in OpenCV can be used to determine an object's length, breadth, and volume. Drawing a box all the way around an object was the simplest technique to measure its dimensions. Next, utilize the associated OpenCV functions to determine the box's length, breadth, and height. These measurements can then be applied to other things. Therefore, computer vision (webcam device and code) was employed to measure the object dimensions in real time.

Limeng Pu†, Rui Tian, Hsiao-Chun Wu, and Kun Yan [9] presented an algorithm for accurately measuring the sizes of multiple objects in the scene using one or two photo shots.The effectiveness of this proposed scheme was demonstrated via various experiments. The average error percentage was below 3%, while that resulting from most commercial mobile applications is around 30%.

O. Kainz, F. Jakab, M. W. Horečný and D. Cymbalák [10] presented an improved computer vision system to measure size from a static 2d image. The principal goal was to propose a solution, which is to utilize computer vision algorithms in the estimation of size of the object from images. Premise to achieve this is to have two types of images – one with a scene without the object and the other one with the object present in the scene. Another premise was to have information on the camera height. For different heights and different distances of objects from the camera, it was found that the deviation of the measurement is smaller than 10%.

H. M. Tran, K. T. Pham, T. M. Vo, T. H. Le, T. T. M. Huynh and S. V. T. Dao [11] proposed to measure physical properties of irregular shape. This paper focuses on new techniques which exhibit simple installation to generate multiple functions for estimating the dimensions, volume, and mass with high accuracy. The validated results were highly competitive with accuracy of about 99% for the volume and mass in 300 testing samples.

A. HajiRassouliha, E. J. L. P. Tang, A. J. Taberner, M. P. Nash and P. M. F. Nielsen [12] introduced a method for three-Dimensional measurements using widely angled stereoscopic camera. The method presented here addresses some of the limitations of using block matching methods for dense reconstruction with widely angled stereoscopic cameras. The method can deal with some degree of non-flatness, because the transformed image is used only as an initial estimate and the location of matched points are precisely found using subpixel image registration. However, the performance may decrease substantially for highly curved objects.

S. Karunarathne and R. Ganganath [14] proposed a sovereign button detection and measure the alignment using image processing technique.The system comprised of real time button detection and alignment. Limitations of this method to check button alignment use in the apparel industry.

S. B. Patil, J. C. Shimpi, A. G. Tanawade, P. G. Chavan and V. S. Tandulkar [15] introduced autonomous object detection and counting using edge detection and image processing techniques.Canny edge detection techniques used here.The number of objects were calculated. The same concept is integrated into Android applications.

P. Mostarac, S. L. Drmić, J. Janković, Ž. Ilić, G. Šišul and A. Šala [16] aimed to calibrate and define a system for automated ship length measurement. This paper presented for reconstructing a 3D scene and measuring the length of a ship and detecting the ship's registration number using stereovision is presented.

H. Yu, W. Yan, J. Sun, H. Wang and L. Zhang [17] designed of intelligent measurement system of vehicle dimensions based on structured light imaging and machine vision.The experimental results showed that the

proportion of vehicle dimension error within 1% is more than 90%, and the error of repeated measurement for the same vehicle is less than 0.5%.

S. K. Bhattacharyya and S. Pal [18] designed a methodologies to check rice quality using image processing technique.Image processing technique has been successfully applied for non-contact measurement of grain dimension.

Chapter 3

3. Motivation and Background Study

Reasons for Choosing the Topic

The topic of contactless dimension measurement is chosen due to its significant potential across various fields such as industry, medical science, and space exploration. The need for precise, efficient, and non-invasive measurement techniques is growing, driven by advancements in technology and the demand for higher accuracy and safety standards.

Motivation Behind the Research

The motivation for this research is inspired by the recent success of ISRO's Chandrayaan-3 mission. Chandrayaan-3 demonstrated advanced technologies for safe and soft landing on the lunar surface, which required precise measurements and control . This mission highlights the importance of accurate, contactless measurement systems in challenging environments, such as space, where traditional methods are not feasible.

Background Study

These works employ methods such as machine learning-based object detection, calibration, stereo vision, and image processing based on OpenCV. They aim to accurately measure object dimensions and distances, offering solutions for applications ranging from surveillance to autonomous vehicles and medical studies. These approaches leverage features like object detection models (e.g., YOLO,Google Mediapipe and R-CNN), reference objects, geometric principles, and calibration techniques to achieve precise and efficient object size measurements.

Some even introduce real-time measurement capabilities using acceleration signals, 3D measurement using two camera and image processing, demonstrating their practical potential in diverse scenarios.

In addition to the mentioned works, the field of object size measurement using OpenCV has seen a growing importance on combining computer vision techniques with deep learning models to enhance accuracy and robustness.

Researchers are increasingly leveraging pre-trained models like YOLO, Google mediapipe , SSD, and Faster R-CNN for object detection,. which can be fine-tuned on custom datasets to improve performance.

Moreover, the integrating of stereo vision and camera calibration has become pivotal in achieving highly accurate measurements of object dimensions and distances, making these methods appropriate for applications such as robotics, industrial automation, and augmented reality. These advancements reflect the ongoing efforts to develop versatile and accurate solutions that cater to a wide range of real-world scenarios where object size measurement is crucial.

Chapter 4

4. Proposed Approach

4.1. Overall proposed method

Step 1: Captured image is taken

Step 2: Initiate ML model

Step 3: Detection of object using ML model

Step 4: If object is detected ,initiate mathematical operation block

Step 5: If object not detected by ML model then pass image for OpenCV based image processing techniques

Step 6:Drawing bounding box and calculation of dimensions using mathematical block

Step 7:Calibration of dimensions to get real-time output

Step 8: Output of image with dimensions on screen

Below figure represents above steps graphically to visualize more efiiciently-

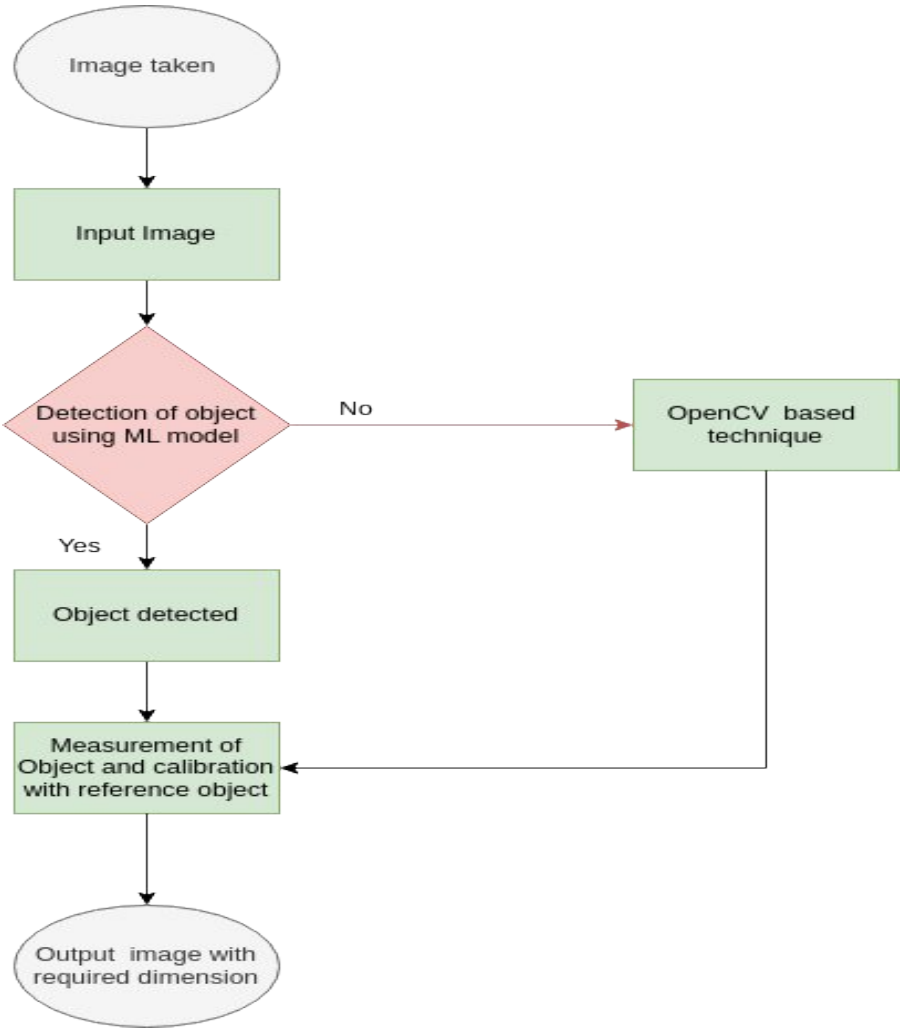

Figure 1:Flowchart of overall model (Self made using drawio software)

4.2. Approach based on OpenCV

After capturing image, the next steps are:

Step 1: Convert the image to grayscale

Step 2: Apply Gaussian blur to the grayscale frame

Step 3: Apply a otsu threshold to the blurred frame

Step 4: Background substraction of the Otsu threshold image

Step 5: Find contours using canny edge detection and dialation with erosion method on the substracted image

Step 6: Draw small crosshairs on the original frame

Step 7:Perform various calculations and plotting for each contour.

Below figure represents openCV based technique through a flowchart:-

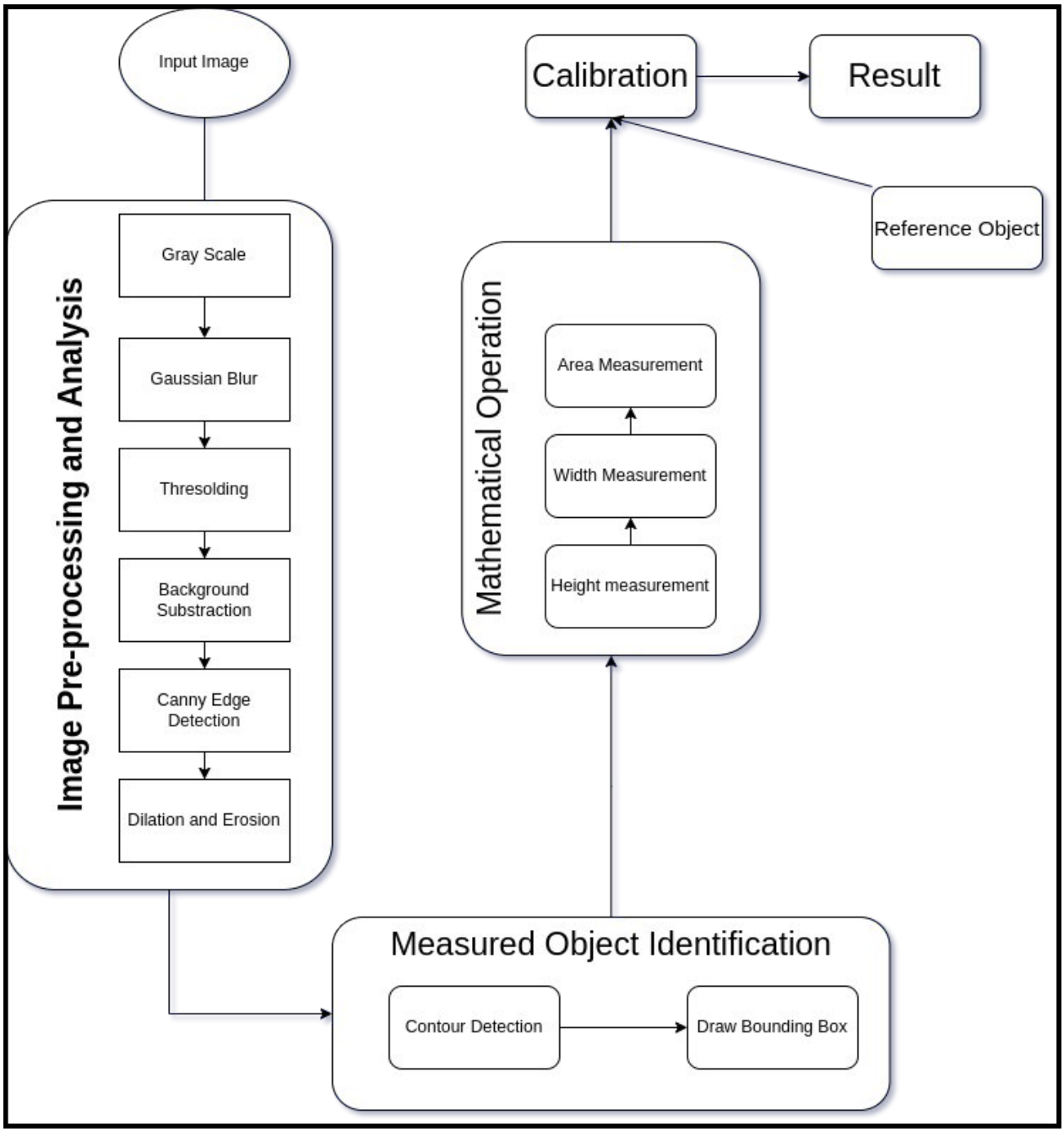

Figure 2:Flowchart of OpenCV based technique (self made using drawio software)

4.3. Using Machine Learning Model

The MediaPipe Object Detector task detects the presence and location of multiple classes of objects within images or videos. For

example, an object detector can locate dogs in an image. This task operates on image data with a machine learning (ML) model, accepting static data or a continuous video stream as input and outputting a list of detection results. Each detection result represents an object that appears within the image or video.

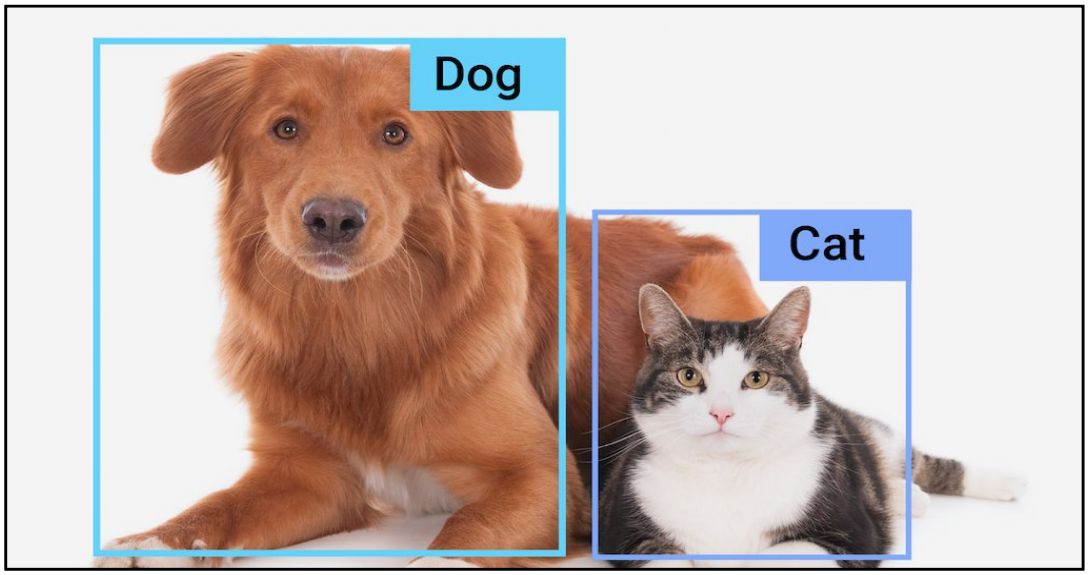

Figure 3:Detector locate a dog and a cat in an image
Source:https://developers.google.com/static/mediapipe/images/solutions/examples/object_detector.png

4.3.1. Input Image-

Following image is taken from local storage or cloud storage.

- Import the Image class from the Python Imaging Library (PIL).
- Open the image using Image.open("picture.jpg")

Figure 4:Image of a dog and a cat for determination of size
Source-https://pixabay.com/photos/pet-cute-animal-domestic-mammal-3157961/

4.3.2. After applying Mediapipe documentation model

The MediaPipe Object Detector task requires a trained model that is compatible with this task. For more information on available trained models for Object Detector, see the task overview Models section.Select and download a model, and then store it in a local

Directory:
model_path='/absolute/path/to/litemodel_efficientdet_lite0_detection_metadata_1.tflite'

Configuration options

This task has the following configuration options for Python applications:

Table 1:Mediapipe Documentation

Option Name	Description	Value Range	Default Value
running_mode	Sets the running mode for the task. There are three modes: IMAGE: The mode for single image inputs.	{IMAGE}	IMAGE
display_names	Sets the language of labels to use for display names provided in the metadata of the task's model, if available. Default is en for English.	Locale code	en
max_results	Sets the optional maximum number of top-scored detection results to return.	Any positive numbers	-1 (all results are returned)
score_threshold	Sets the prediction score threshold that overrides the one provided in the model metadata (if any). Results below this value are rejected.	Any float	Not set
category_allowlist	Sets the optional list of allowed category names. If non-empty, detection results whose category name is not in this set will be filtered out. Duplicate or unknown category names are ignored. This option is mutually exclusive with category_denylist and using both results in an error.	Any strings	Not set

category_denylist	Sets the optional list of category names that are not allowed. If non-empty, detection results whose category name is in this set will be filtered out. Duplicate or unknown category names are ignored. This option is mutually exclusive with category_allowlist and using both results in an error.	Any strings	Not set

4.3.3. Run the task

The Object Detector task will return the objects detected within the input image or frame.

Perform object detection on the provided single image.

```
detection_result = detector.detect(mp_image)
```

For a complete example of running an Object Detector on an image, see the code example for details.

Note the following:

- When running in the image , the Object Detector task will block the current thread until it finishes processing the input image or frame.

4.3.4. Handle and display results

Upon running inference, the Object Detector task returns an ObjectDetectionResult object which describes the objects that it has found in the input image.

The following shows an example of the output image from this task:

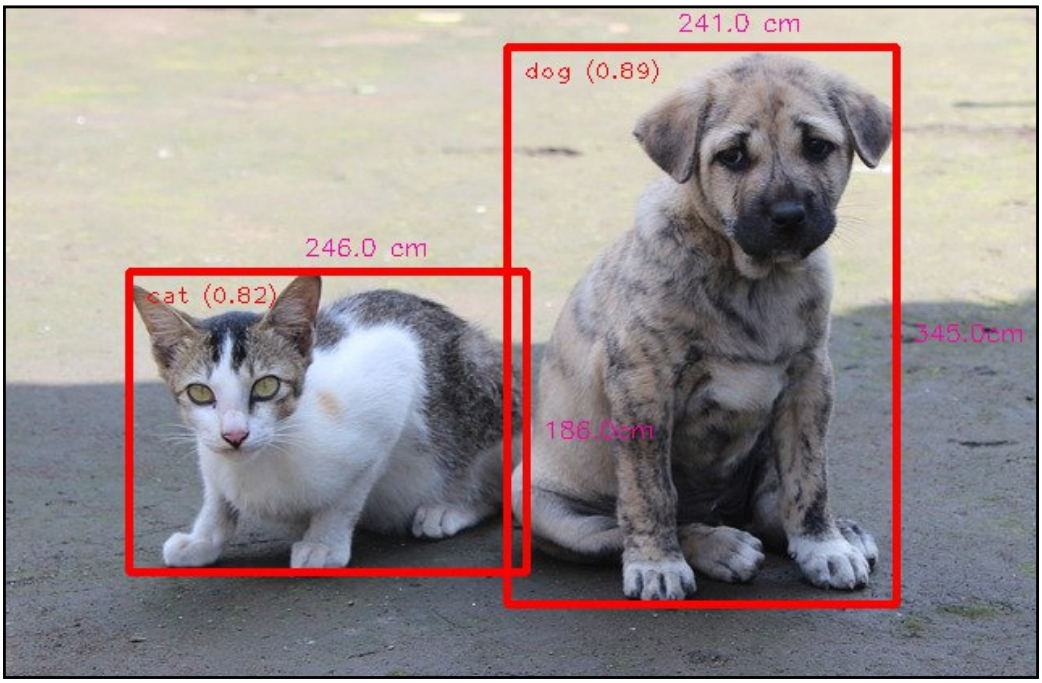

Figure 5:Detection of cat and dog with size(Uncalibrated)
Source- output generated from google colab

4.4. OpenCV based Image Processing Model

4.4.1. Input Image-

The initial task to perform is taking an image(loading targeted image whose dimension wanted to measure).

OpenCV offers support for the image formats Windows bitmap (bmp), portable image formats (pbm, pgm, ppm) and Sun raster (sr, ras).

- Import the Image class from the Python Imaging Library (PIL).
- Open the image using Image.open("picture.jpg")

Following figure(image) is capture using phone camera and imported to google colab-

Figure 6:Original image of a Highlighter
Source- output generated from google colab

4.4.2. Pre-processing

The pre-processing module is crucial for optimizing image quality and preparing it for accurate measurements. This stage encompasses several operations, including image resizing, noise reduction, and conversion to a specific color space. Resizing the image allows customization of the resolution or scale, ensuring compatibility with subsequent analysis. To eliminate unwanted noise or artifacts, techniques like blurring can be applied. Additionally, converting the color space facilitates subsequent calculations and analysis if required**.**

4.4.2.1. RGB to Gray

In digital photography, computer-generated imagery, and colorimetry, a **grayscale** image is one in which the value of each pixel is a single sample representing only an *amount* of light; that is, it carries only intensity information.

Grayscale images, a kind of black-and-white or gray monochrome, are composed exclusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest.

Transformations within RGB space like adding/removing the alpha channel, reversing the channel order, conversion from 16-bit RGB color , as well as conversion to/from grayscale using:

$$RGB\,[A]\ to\ Gray\colon Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad (1)$$

To split an RGB image into its individual color channels (Red, Green, and Blue), you can use OpenCV in Python. Here's how:

- Splitting Channels:
- Load your image using cv2.imread('image.jpg').
- Use cv2.split(image) to obtain separate numpy arrays for the blue, green, and red channels.
- Assign these arrays to variables (e.g., blue, green, red).
- Display each channel as a grayscale image

Following image visually represent above processes-

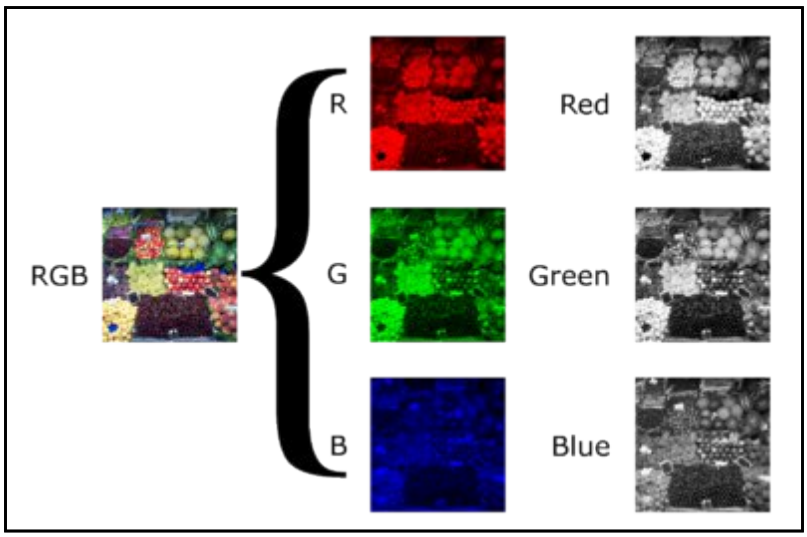

Figure 7:Composition of RGB from three gray scale images
Source-https://upload.wikimedia.org/wikipedia/commons/thumb/3/33/Beyoglu_4671_tricolor.png/400px-Beyoglu_4671_tricolor.png

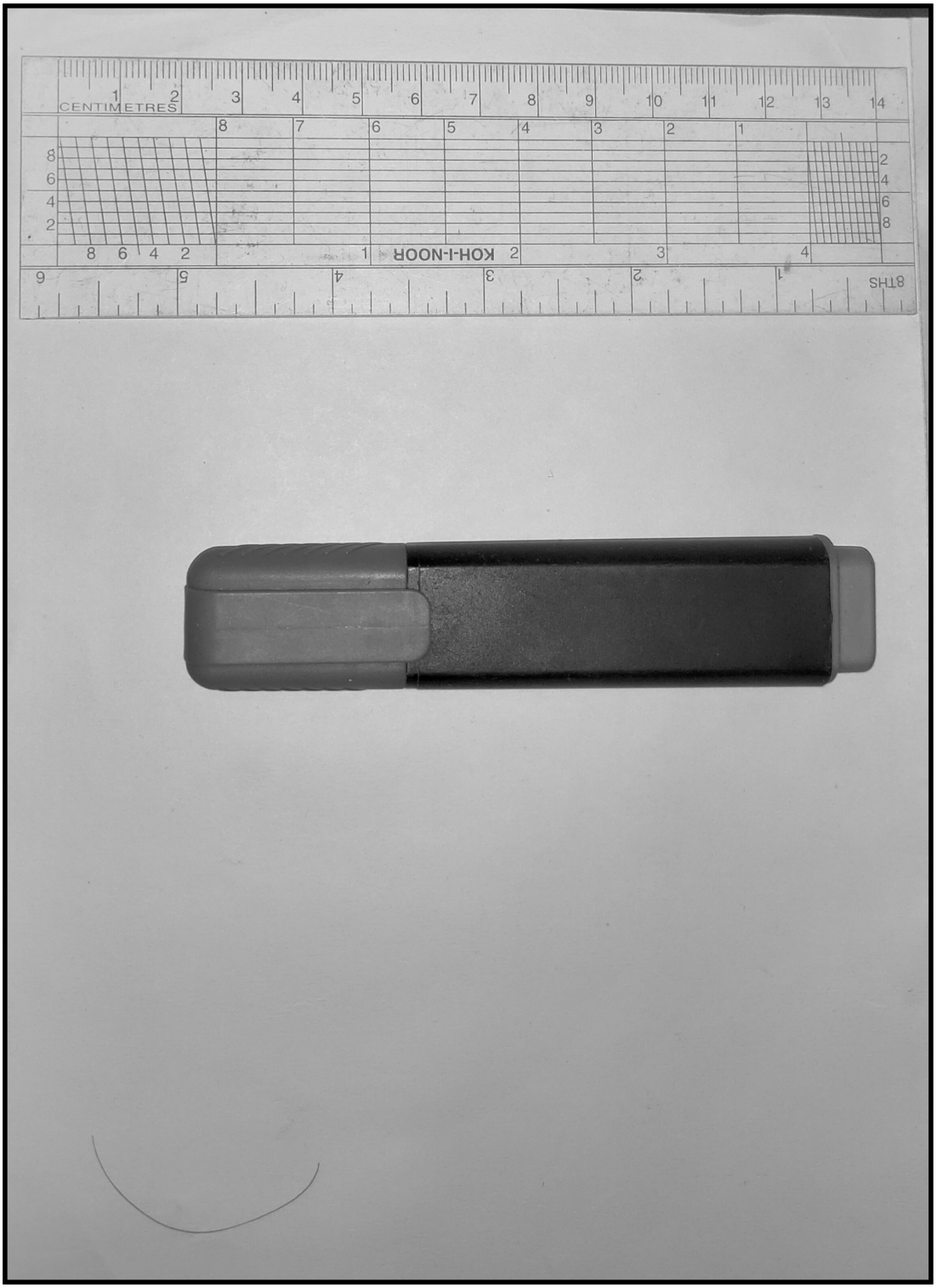

Figure 8:Gray Scale Image of a Highlighter and a scale as a reference object
Source- output generated from google colab

4.4.3. Gaussian Blur

It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination.

In two dimensions, it is the product of two such Gaussian functions, one in each dimension:

$$G(x,y) = \frac{1}{2\Pi\sigma^2} e^{-\frac{x^{2+}y^2}{2\sigma^2}} \qquad (2)$$

where

x is the distance from the origin in the horizontal axis; *y* is the distance from the origin in the vertical axis, *σ* is the standard deviation of the Gaussian distribution.

Common uses:

- Gaussian smoothing is commonly used with edge detection.
- Lower-end digital cameras, including many mobile phone cameras, commonly use gaussian blurring to obscure image noise caused by higher ISO light sensitivities.

* Gaussian blur is automatically applied as part of the image post-processing of the photo by the camera software, leading to an irreversible loss of detail.

To apply **Gaussian blur to a grayscale** image using OpenCV, following steps are appied in this thesis work-

- Load the Grayscale Image:
- First, load your grayscale image using cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE).
- Apply Gaussian Blur:
- Use the cv2.GaussianBlur() function to blur the image.
- The function takes three arguments:
    - The input grayscale image.
    - The kernel size (width and height). These values should be odd numbers (e.g., (5, 5)).
    - The standard deviation (sigmaX). If set to 0, it will be calculated automatically.

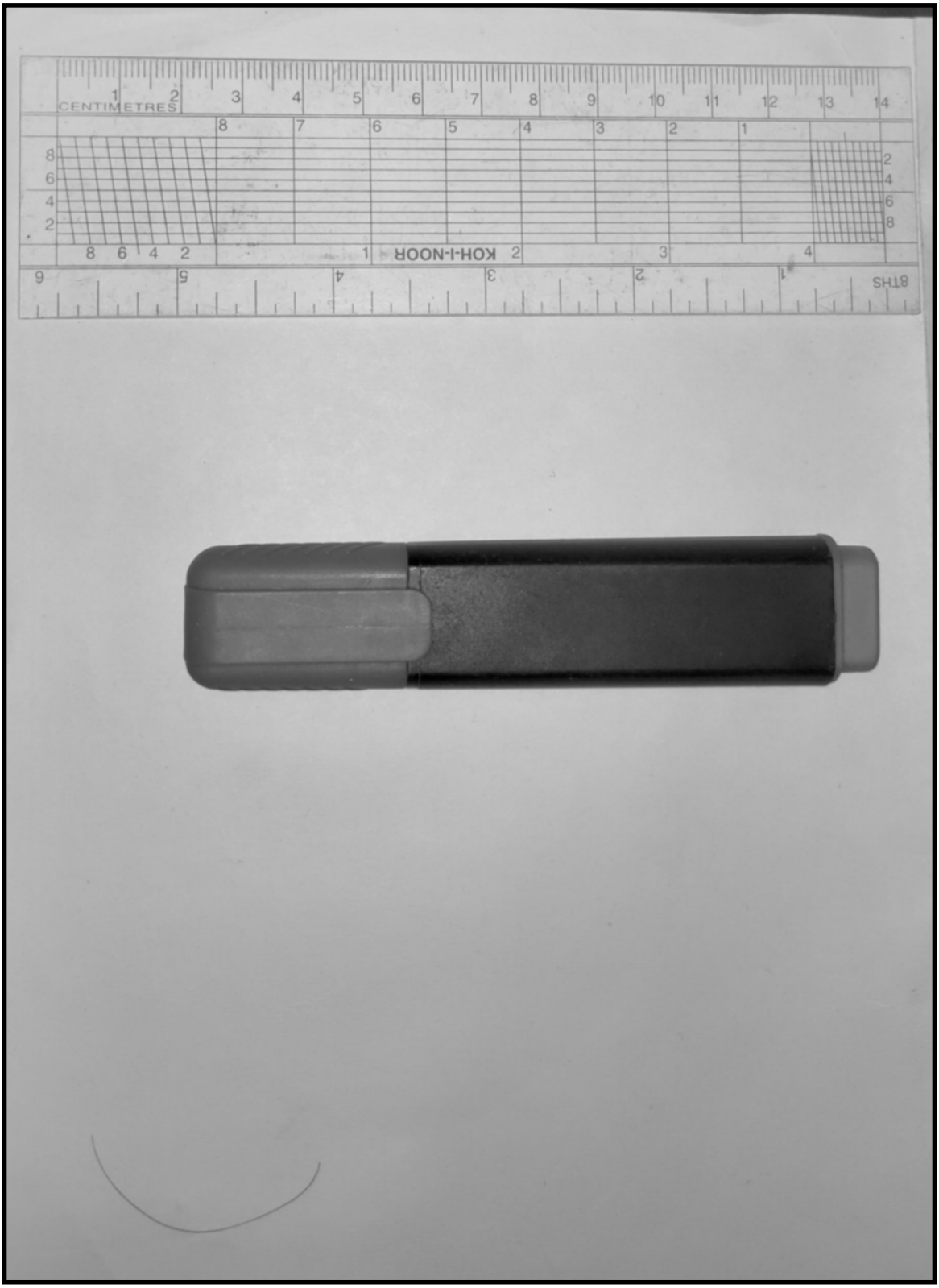

Figure 9:After applying Gaussian Blur
Source- output generated from google colab

4.4.4. Otsu Thresholding

Otsu's method, named after its creator **Nobuyuki Otsu**, is commonly used in computer vision and image processing to perform automatic image thresholding. The goal of thresholding is to segment an image into two classes: **foreground** and **background**. Here's how it works:

- **Image Thresholding**:
  Thresholding is the process of generating a **binary image** from a grayscale image by separating it into two regions based on a **threshold value**.
  Pixels with intensity values greater than the threshold are treated as **white (1)** in the output image, while others are considered **black (0)**.

- **Otsu's Method**:
  Otsu's method aims to find the **optimal threshold** that maximizes the **between-class variance**.
  The key idea is that well-thresholded classes of pixels must be distinct in terms of their intensity levels.
  By iterating through all possible threshold values, Otsu's method identifies the threshold that provides the best separation between foreground and background pixels.

- **Algorithm Steps**:
  Obtain the **image histogram**, which represents the distribution of pixel intensities.
  Compute the threshold value that minimizes the variance between the two classes.
  Segment the image into two separate classes based on this threshold.

- **Mathematical Formulation**:
  Let {0, 1, 2, …, L-1} denote the L distinct intensity levels of the pixels in an M x N image.
    - The normalized histogram components represent the probabilities of occurrence of each intensity level.

- The probability that a pixel belongs to class C1 (foreground) is given by:

$$P_1(K) = \sum_{i=0}^{k} p(i) \quad (3)$$

- Similarly, the probability of class C2 (background) is:

$$P_1(K) = \sum_{i=k+1}^{L-1} p(i) \quad (4)$$

- The mean intensity level of pixels in class C1 is:

$$\mu_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i.p(i) \quad (5)$$

➢ **Threshold Selection**:
Otsu's method selects the threshold value k that maximizes the **between-class variance**:

$$\sigma_B^2(k) = P_1(k).P_2(k).\left(\mu_1(k) - \mu_2(k)\right) \quad (6)$$

➢ **Output**:
The segmented image has only two classes of pixels: foreground (white) and background (black).

Application in Image Segmentation:-

Otsu's method is particularly useful for tasks like tumor detection in medical images or identifying natural disasters in satellite imagery. By automatically determining the threshold, it simplifies the process and ensures robust results.

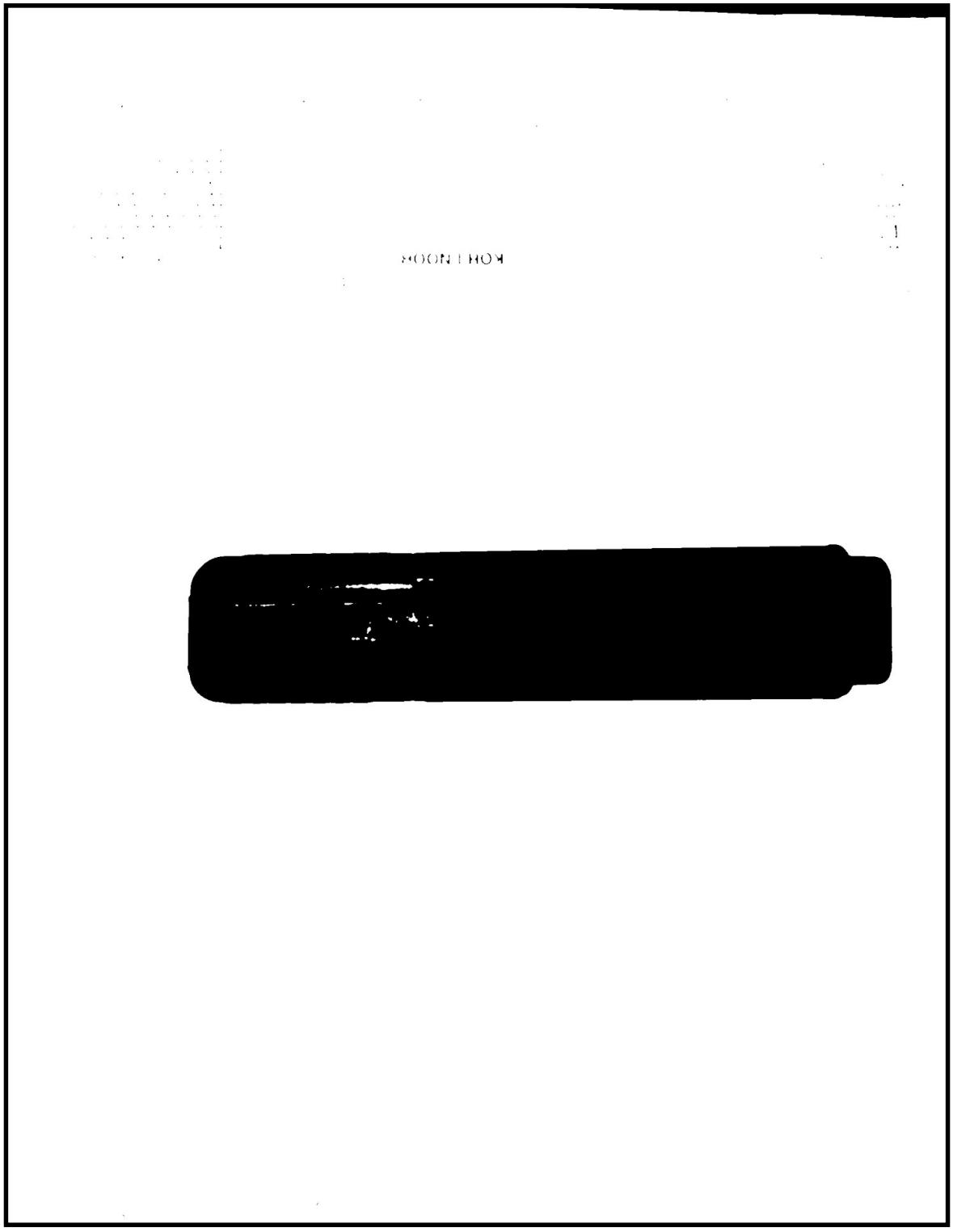

Figure 10:Otsu Thresholding
Source- output generated from google colab

4.4.5. Background Subtraction

Background subtraction is a fundamental technique used in computer vision and image processing to separate moving objects from a static background.

Background Subtraction Basics

- **Objective**: The goal of background subtraction is to generate a **foreground mask**, which is a binary image containing pixels belonging to moving objects in the scene.
- **Method**: It calculates the foreground mask by subtracting the current frame from a background model.
- **Background Model**: The background model represents the static part of the scene or anything that can be considered as background based on scene characteristics.
- **Steps**:
  - ❖ **Background Initialization**: Compute an initial model of the background.
  - ❖ **Background Update**: Continuously update the model to adapt to changes in the scene.

OpenCV Implementation:

Below is a Python example using OpenCV to perform background subtraction:

```
import cv2
bgsub = cv2.createBackgroundSubtractorMOG2()
_ = bgsub.apply(bg)
fgmask = bgsub.apply(th2)
plt.imshow(fgmask,cmap='gray'),plt.axis('off')

cv2_imshow(fgmask)
```

```
        # Press 'q' to exit
        if cv2.waitKey(30) & 0xFF == ord("q"):
            break

cap.release()
cv2.destroyAllWindows()
```

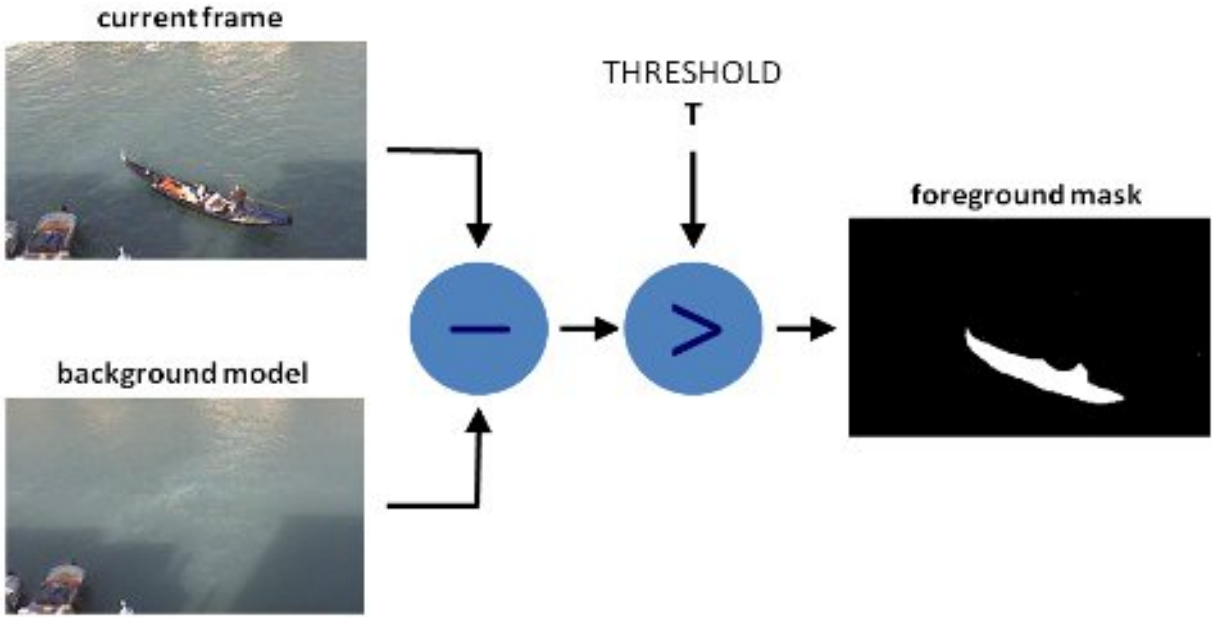

Figure 11:Technique of background substraction
Source-https://docs.opencv.org/3.4/Background_Subtraction_Tutorial_Scheme.png

Explanation:

- Create a BackgroundSubtractor object (in this case, BackgroundSubtractorMOG2).
- Capture two image one with object(s) and another without that object(s).
- Apply background subtraction to using fgbg.apply(image).
- Display the resulting foreground mask.

Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing

the pixels belonging to moving objects in the scene) by using static cameras.

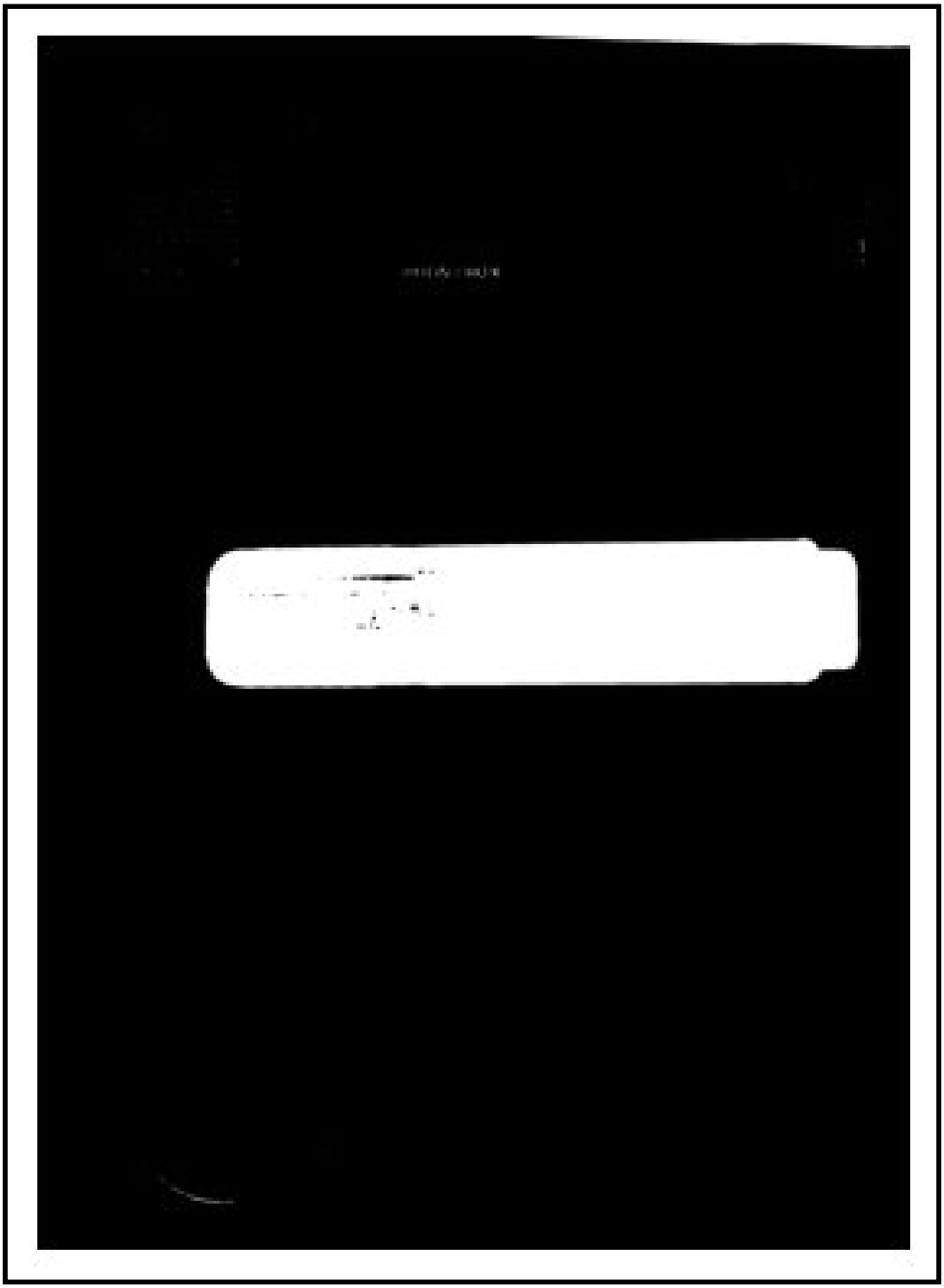

Figure 12: Image After Background Subtraction
Source- output generated from google colab

4.4.6. Edge detection

Edge detection plays a crucial role in various applications, including computer vision, object recognition, and feature extraction. Here are some key points:

- **. Motivation for Edge Detection**
  - **Edges** represent significant local changes in intensity within an image.
  - They typically occur at the boundaries between different regions or objects.
  - Humans rely on edges for object recognition, and line drawings (which emphasize edges) are almost as recognizable as the original images.
- **Types of Edges**
  - There are four possible sources of edges in an image:
    - **Surface normal discontinuity**: Occurs when the surface changes direction sharply.
    - **Depth discontinuity**: Arises when one surface is behind another.
    - **Surface color discontinuity**: Indicates a change in color within a single surface.
    - **Illumination discontinuity**: Associated with shadows or changes in lighting.
- **Edge Detection Operators**

  Edge detection operators help identify edges in an image. Following operator is used here-

  Canny Edge Detector

  - Gaussian-based operator.

- Computes the second derivative of the image using the Laplacian.
- Effective for abrupt transitions in gray levels.

❖ **Application and Importance**

➢ Edge detection allows us to:

- Extract important features (e.g., corners, lines, curves).
- Recognize objects.
- Recover image geometry and viewpoint.

The Canny edge detector is a multi-stage algorithm

➢ **Noise Reduction**

Since edge detection is susceptible to noise in the image, the first step is to remove the noise in the image with a Gaussian filter.

➢ **Finding Intensity Gradient of the Image**

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\ Gradient\,(G) = \sqrt{(G_x^2 + G_y^2)} \qquad (7)$$

$$Angle(\theta) = \tan^{-1}[\tfrac{Gy}{Gx}] \qquad (8)$$

➢ **Non-maximum Suppression**

After getting gradient magnitude and direction, a full scan of the image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, the pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:

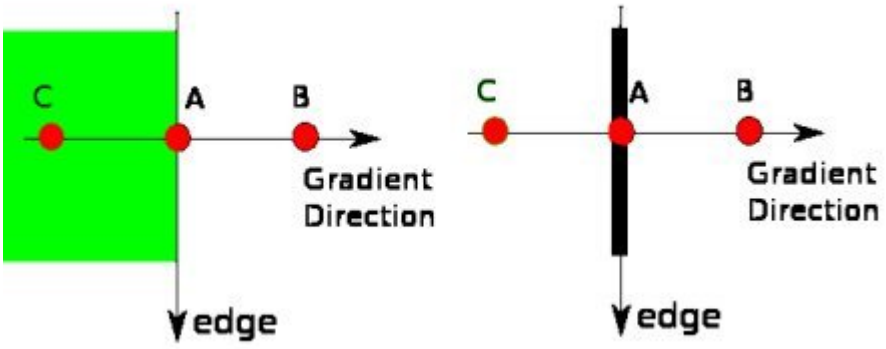

Figure 13:Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. Source:https://docs.opencv.org/4.x/nms.jpg

- **Hysteresis Thresholding**

This stage decides which all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:

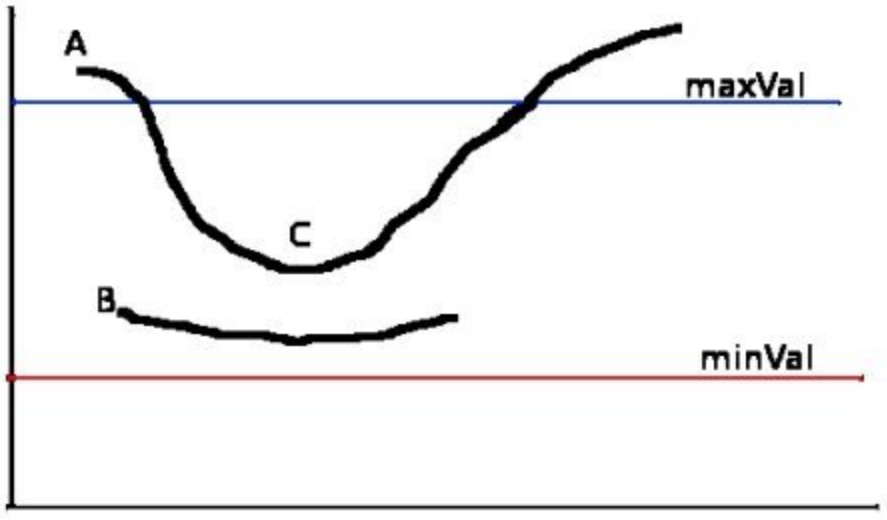

Figure 14:Image represents maximum and minimum value
Source-https://docs.opencv.org/4.x/hysteresis.jpg

Explanation:

The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered a valid edge and we get that full curve. But edge B, although it is above minVal and is in the same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

The edges of the image are highlighted by the combination of these steps. The binary image produced by this module shows the borders as white pixels and the remaining areas as black**.**

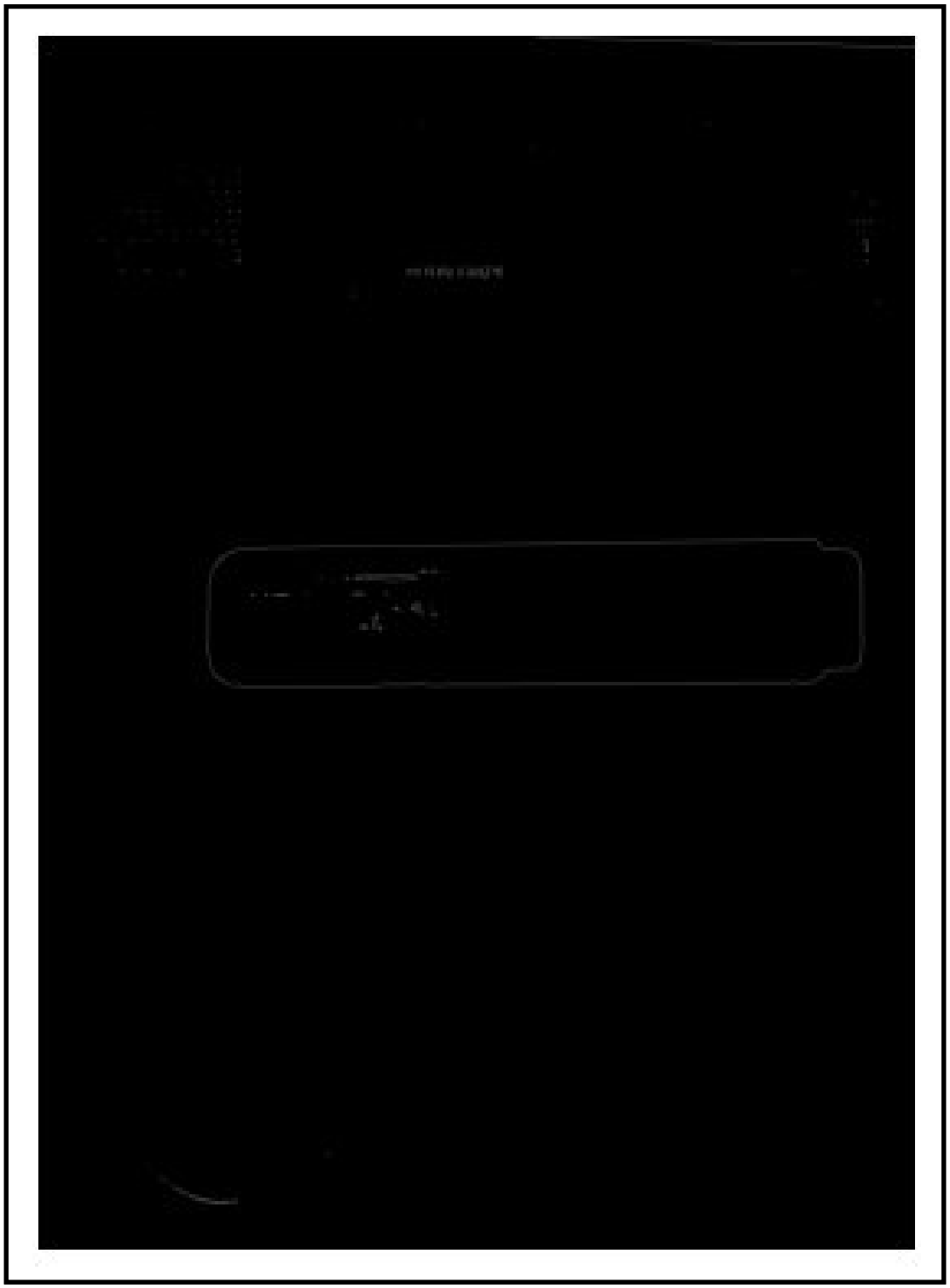

Figure 15:After applying Canny edge detection
Source- output generated from google colab

4.4.7. Drawing Contour

- **Contour drawing** is a fundamental technique where an artist creates a line drawing that outlines the edges and contours of a subject.
- In contour drawing, the artist focuses on the **outline** or **silhouette** of the subject, carefully observing its shape, proportions, and details.
- In the context of image processing, contour detection plays a crucial role in identifying and extracting object boundaries within an image.

Contour Detection in Image Processing

- **Contours**, represented as continuous lines or curves, define the complete boundary of an object.
- Here are the steps involved in contour detection using OpenCV:

Grayscale Conversion:
Convert the image to grayscale to simplify contour detection.

Binarization:
Create a binary image through thresholding or Canny edge detection.

Contour Detection:
Utilize cv2.findContours() to find contours in the binary image.

Draw Contours:
Visualize the detected contours on the original image.

Retrieval Modes and Approximation Methods

Retrieval Modes in OpenCV:

- RETR_LIST: Retrieve all contours without establishing a parent-child relationship.
- RETR_EXTERNAL: Return only the extreme outer contours, omitting child contours.
- RETR_CCOMP: Organize contours into a 2-level hierarchy, differentiating external and internal contours.
- RETR_TREE: Retrieve all contours and create a full family hierarchy list.

Approximation Methods:

- CHAIN_APPROX_NONE: Store all points along the line, providing accurate but potentially inefficient representation.
- CHAIN_APPROX_SIMPLE: Store only the end points of each line, offering a more efficient representation.

The contours are a useful tool for shape analysis and object detection and recognition.In OpenCV, finding contours is like finding white object from black background. So remember, the object to be found should be white and the background should be black.

4.4.8. Measurement of Dimension

The dimension computation module makes it possible to estimate the sizes of the items in the image using the extracted contours Area, perimeter, centroid, bounding box, and other attributes of contours can be computed using a variety of OpenCV methods. The dimensions, shapes, and orientations of the objects in the image can all be determined using these characteristics as useful metrics. For instance, the area of the contour can be used to estimate the size of the object

The results demonstrated that the system achieved a high level of measurement accuracy, with deviations between the system's measurements and ground truth values within an acceptable range. The accuracy varied depending on factors such as object complexity and image quality, but in general, the system consistently provided reliable size measurements.

The accuracy of a measuring system can be ex- pressed using the formula for relative error. Relative error is a way to quantify the discrepancy between a measured or calculated value and a true or reference value.Relative error is calculated as follows:

- Subtract the True Value from the Measured Value to find the difference between the two.
- Divide the difference by the True Value.
- Multiply the result by 100 % to express the relative error as a percentage.

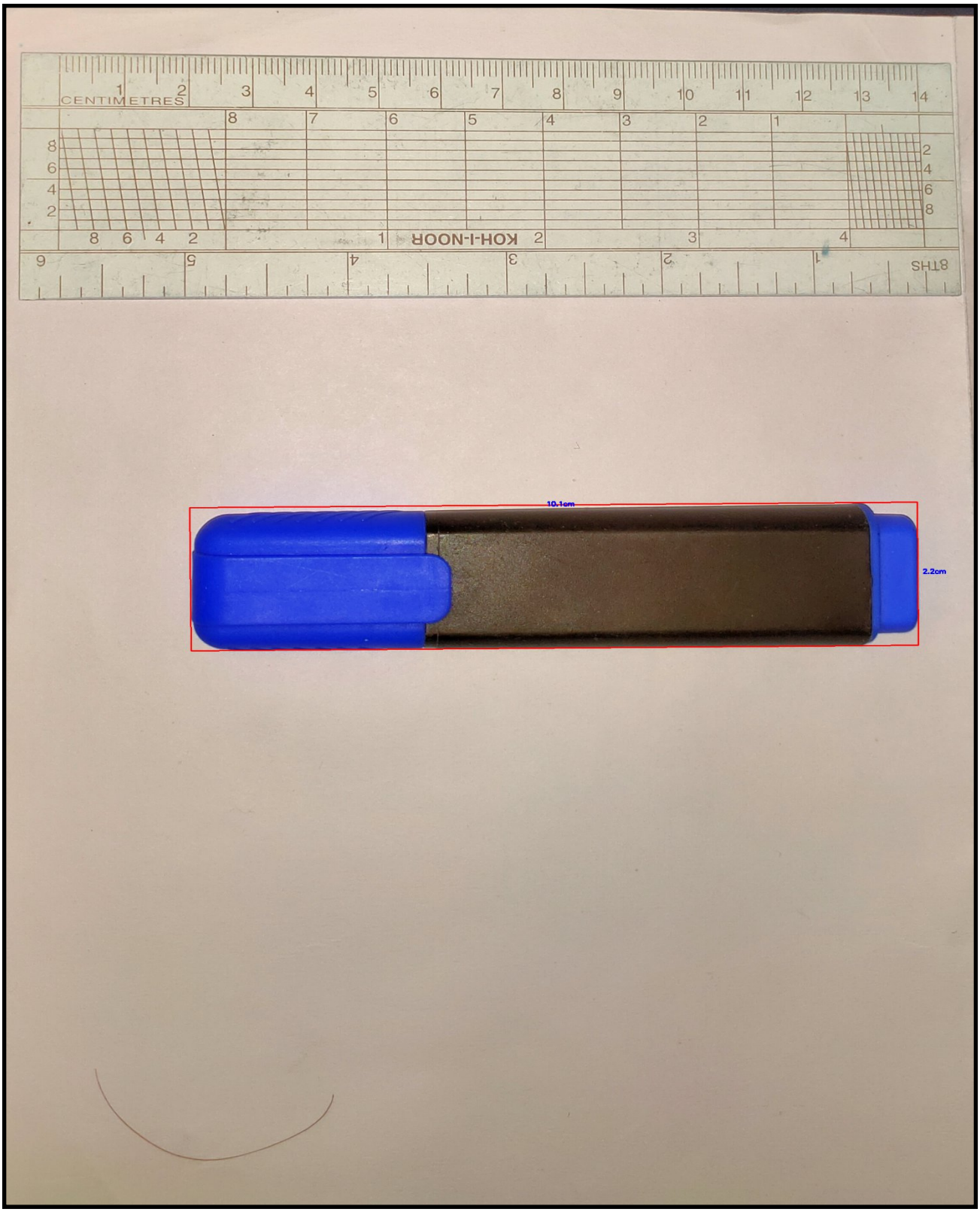

Figure 16:Bounding box around targeted object with height and width
Source- output generated from google colab

4.4.9. Calibration

In order to determine the size of an object in an image, we first need to perform a “calibration” using a reference object.

Calibration is necessary to get dimensions in terms of real word measurement units like cm,inch etc.

pixels_per_metric = object_width / know_width.....(10)

Chapter 5

5. Experimentation and Results

The accuracy of the object size measurements was assessed by comparing the system's measurements to ground truth values. Ground truth values were obtained through manual measurements of the objects using physical tools like rulers or calipers. The measurements were taken across a diverse set of objects with varying sizes, shapes, and complexities.

The accuracy of a measuring system can be expressed using the formula for relative error. Relative error is a way to quantify the discrepancy between a measured or calculated value and a true or reference value.Relative error is calculated as follows:

- Subtract the True Value from the Measured Value to find the difference between the two.
- Divide the difference by the True Value.
- Multiply the result by 100 % to express the relative error as a percentage.

Here are some experimental results shown in Table 2, which we have done practically using our own images .The accuracy of this model is around 97% for the images we have considered. The results might vary according to the input constraints

Table 2:Experimental Result

SL No.	Object	Original Height(cm)	Original Width(cm)	Height using Model(cm)	Width using Model(cm)	Relative Error in height(%)	Relative Error in Width(%)
1	Highlighter	11.3	2.4	11.1	2.4	1.76	0
2	Human	160		119.16		25.52	
3	Torch Light	19.5	4.7	17.9	4.8	8.21	2.12
4	pen	14.5	0.8	13.1	0.9	9.65	12.5

6. Conclusion and Future Scope

6.1. Conclusion

This approach eliminates the need for physical contact or manual measurements. Instead, it relies on computer vision techniques to assess an object's dimensions.

ML Model is trained using known objects (those used during model training). The ML model learns patterns and features from these objects.When presented with a new object in real-world images or videos, the model recognizes it and predicts its size based on the learned information.

OpenCV-based processing provides tools for image manipulation. Preprocessing steps enhance image quality and reduce noise. Objects are enclosed in bounding boxes. From these boxes, we extract length, width, and height.

Industries like manufacturing, logistics, healthcare, and retail benefit from accurate size estimation.The approach is efficient,faster (less time consuming) benefits from Canny edge detection, and holds practical applications in the real world.

6.2. Future Scopes

Advancements in object detection models (such as YOLO, Faster R-CNN, or EfficientDet) can significantly enhance accuracy and reliability. These models allow precise identification of objects within an image or video stream, aiding in subsequent size measurements. Applications range from inventory management to security surveillance.

By incorporating 3D vision techniques, we can move beyond 2D measurements (length and width) to estimate volume. Depth sensors (e.g., LiDAR, structured light) provide additional data for volumetric calculations. This is useful in scenarios like package dimensioning, volumetric analysis of containers, and architectural measurements.

Tracking algorithms play a crucial role in maintaining accurate measurements over time, especially in dynamic environments. Challenges include handling occlusions, object movement, and changing viewpoints. Reliable tracking ensures consistent size estimation in videos or real-time streams.

Edge devices (e.g., cameras, drones) can process measurements locally, minimizing communication delays. Real-time size estimation aids autonomous robots, collision avoidance, and safety protocols. Balancing accuracy with computational efficiency is essential.

In agriculture, crop size estimation, yield prediction, and precision farming benefit from the collaboration of geospatial mapping and remote sensing technology. Urban planning and environmental monitoring also rely on accurate size measurements.

Responsible deployment requires addressing privacy concerns. Ensuring anonymization and transparency about data collection and usage is crucial, especially in surveillance contexts.

References

[1] Othman, Nashwan & Salur, Mehmet & Karakose, Mehmet & Aydin, Ilhan. (2018). An Embedded Real-Time Object Detection and Measurement of its Size. 10.1109/IDAP.2018.8620812.

[2] Ankad, Sushma & Raj, B & Nasreen, Azra & Mathur, Surbhi & P., Ramakanth & Sreelakshmi, K. (2021). Object Size Measurement from CCTV footage using deep learning. 1-5. 10.1109/CSITSS54238.2021.9683671.

[3] K. Deshmukh, R. Kasture, S. Bhoite, S. L. Tade and S. A. Vaishnav, "Performance Analysis of Fruit Quality Detection Using Computer Vision and Object Detection," 2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA), Pune, India, 2023, pp. 1-8, doi: 10.1109/ICCUBEA58933.2023.10392090.

[4] K. Singh, V. K. N., A. Shinde and S. Y. Sawant, "Surface Area Calculation of Asymmetric/Axisymmetric Shapes Utilising Simple Image Processing and OpenCV," 2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA), Pune, India, 2023, pp. 1-8, doi: 10.1109/ICCUBEA58933.2023.10392058.

[5] R. A. Joshi, S. N. Helambe, & R.R.Deshmukh. (2020). Digital Image Processing based Surface Area Calculation. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 10(1), 186–190. [Online]. Available: https://www.ijitee.org/wp-content/uploads/papers/v10i1/A81501110120.pdf

[6] C. Lü, H. Ren, Y. Zhang and Y. Shen, "Leaf Area Measurement Based on Image Processing," 2010 International Conference on Measuring Technology and Mechatronics Automation, Changsha, China, 2010, pp. 580-582, doi: 10.1109/ICMTMA.2010.141.

[7] E. Yang, M. Wang, H. Cheng, R. Liu and F. Chen, "A Method to Improve the Precision of 2-Dimensioanl Size Measurement of Objects through Image Processing," 2022 21st International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Chizhou, China, 2022, pp. 197-200, doi: 10.1109/DCABES57229.2022.00010.

[8] B. M U, H. Raghuram and Mohana, "Real Time Object Distance and Dimension Measurement using Deep Learning and OpenCV," 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2023, pp. 929-932, doi: 10.1109/ICAIS56108.2023.10073888.

[9] Limeng Pu, Rui Tian, Hsiao-Chun Wu and Kun Yan, "Novel object-size measurement using the digital camera," 2016 IEEE Advanced Information

Management, Communicates, Electronic and Automation Control Conference (IMCEC), Xi'an, China, 2016, pp. 543-548, doi: 10.1109/IMCEC.2016.7867270.

[10] O. Kainz, F. Jakab, M. W. Horečný and D. Cymbalák, "Estimating the object size from static 2D image," 2015 International Conference and Workshop on Computing and Communication (IEMCON), Vancouver, BC, Canada, 2015, pp. 1-5, doi: 10.1109/IEMCON.2015.7344423.

[11] O. Kainz, F. Jakab, M. W. Horečný and D. Cymbalák, "Estimating the object size from static 2D image," 2015 International Conference and Workshop on Computing and Communication (IEMCON), Vancouver, BC, Canada, 2015, pp. 1-5, doi: 10.1109/IEMCON.2015.7344423.

[12] A. HajiRassouliha, E. J. L. P. Tang, A. J. Taberner, M. P. Nash and P. M. F. Nielsen, "A Method for Three-Dimensional Measurements Using Widely Angled Stereoscopic Cameras," 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Auckland, New Zealand, 2019, pp. 1-5, doi: 10.1109/I2MTC.2019.8827086.

[13] S. Karunarathne and R. Ganganath, "A Sovereign Button Detection and Measure the Alignment Using Image Processing," 2021 IEEE 16th International Conference on Industrial and Information Systems (ICIIS), Kandy, Sri Lanka, 2021, pp. 359-364, doi: 10.1109/ICIIS53135.2021.9660683.

[14] S. B. Patil, J. C. Shimpi, A. G. Tanawade, P. G. Chavan and V. S. Tandulkar, "Autonomous Object Detection and Counting using Edge Detection and Image Processing Algorithms," 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2023, pp. 1280-1286, doi: 10.1109/ICOEI56765.2023.10125716.

[15] P. Mostarac, S. L. Drmić, J. Janković, Ž. Ilić, G. Šišul and A. Šala, "Calibration and Definition of a System for Automated Ship Length Measurement," 2022 International Conference on Smart Systems and Technologies (SST), Osijek, Croatia, 2022, pp. 315-318, doi: 10.1109/SST55530.2022.9954740.

[16] H. Yu, W. Yan, J. Sun, H. Wang and L. Zhang, "Design of Intelligent Measurement System of Vehicle Dimensions Based on Structured Light Imaging and Machine Vision," 2019 6th International Conference on Systems and Informatics (ICSAI), Shanghai, China, 2019, pp. 1238-1243, doi: 10.1109/ICSAI48974.2019.9010242.

[17] S. K. Bhattacharyya and S. Pal, "Measurement of Parboiled and Non-parboiled Rice Grain Dimension during Hydro Thermal Treatment Using Image Processing," 2020 National Conference on Emerging Trends on Sustainable

Technology and Engineering Applications (NCETSTEA), Durgapur, India, 2020, pp. 1-5, doi: 10.1109/NCETSTEA48365.2020.9119920.

[18] N. Ladplee, A. Pimpin, W. Srituravanich and N. Damrongplasit, "Volumetric Measurement of Rectangular Parcel Box Using LiDAR Depth Camera for Dimensioning and 3D Bin Packing Applications," 2022 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Yeosu, Korea, Republic of, 2022, pp. 1-4, doi: 10.1109/ICCE-Asia57006.2022.9954650.

[19] X. Huang, Z. Liu and S. Zhou, "Automatic Measurement Method of Human Body Dimension Based on Frontal Human Image," 2022 Global Conference on Robotics, Artificial Intelligence and Information Technology (GCRAIT), Chicago, IL, USA, 2022, pp. 308-312, doi: 10.1109/GCRAIT55928.2022.00072.

[20] A. S. Parihar, M. Gupta, V. Sikka and G. Kaur, "Dimensional analysis of objects in a 2D image," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, India, 2017, pp. 1-7, doi: 10.1109/ICCCNT.2017.8203937.

[21] Mediapipe Solutions Guide,google ai edge,google AI for developers. Google. Acessed:Aug 24,2024. [Online.] Available: https://ai.google.dev/edge/mediapipe/solutions/guide

[22] OpenCV: Image processing in OpenCV. Acessed:Aug 24,2024.[Online.] Available:https://docs.opencv.org/4.x/d2/d96/tutorial_py_table_of_contents_imgproc.html .

Appendix

A: Python code for ML model

Object Detection with MediaPipe Tasks

- Let's start with installing MediaPipe.

```
!pip install -q mediapipe

!wget -q -O efficientdet.tflite -q https://storage.googleapis.com/mediapipe-models/object_detector/efficientdet_lite0/int8/1/efficientdet_lite0.tflite

import cv2
import numpy as np

MARGIN = 10  # pixels
ROW_SIZE = 10  # pixels
FONT_SIZE = 1
FONT_THICKNESS = 1
TEXT_COLOR = (255, 0, 0)  # red


def visualize(
        image,
        detection_result
) -> np.ndarray:
  Args:
        image: The input RGB image.
        detection_result: The list of all "Detection" entities to be visualize.
  Returns:
        Image with bounding boxes.
  """
  for detection in detection_result.detections:
        # Draw bounding_box
        bbox = detection.bounding_box
        start_point = bbox.origin_x, bbox.origin_y
        end_point = bbox.origin_x + bbox.width, bbox.origin_y + bbox.height
        cv2.rectangle(image, start_point, end_point, TEXT_COLOR, 3)
        print(bbox)
        # Draw label and score
        category = detection.categories[0]
        category_name = category.category_name
        probability = round(category.score, 2)
        result_text = category_name + ' (' + str(probability) + ')'
        text_location = (MARGIN + bbox.origin_x ,
                MARGIN + ROW_SIZE + bbox.origin_y )
        new_width = int(bbox.width / 2)
        new_height = int(bbox.height / 2)
        text_location_x = (bbox.origin_x  + new_width - 15, bbox.origin_y - 10)
        text_location_y = (bbox.origin_x  + bbox.width + 10, bbox.origin_y + new_height + 10 )
```

```
        cv2.putText(image, result_text, text_location, cv2.FONT_HERSHEY_PLAIN,
                    FONT_SIZE, TEXT_COLOR, FONT_THICKNESS)
        cv2.putText(image, "{:.1f} cm".format(bbox.width),text_location_x,
                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 167), 1)
        cv2.putText(image, "{:.1f}cm".format(bbox.height), text_location_y,
                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 167), 1)

  return image

 !wget -q -O image.jpg https://storage.googleapis.com/mediapipe-
tasks/object_detector/cat_and_dog.jpg

 IMAGE_FILE = 'image.jpg'

 import cv2
 from google.colab.patches import cv2_imshow

 img = cv2.imread(IMAGE_FILE)
 cv2_imshow(img)
```

Here are the steps to run object detection using MediaPipe.

- STEP 1: Import the necessary modules.

```
import numpy as np
import mediapipe as mp
from mediapipe.tasks import python
from mediapipe.tasks.python import vision
```

- STEP 2: Create an ObjectDetector object.

```
base_options = python.BaseOptions(model_asset_path='efficientdet.tflite')
 options = vision.ObjectDetectorOptions(base_options=base_options,
                                        score_threshold=0.5)
 detector = vision.ObjectDetector.create_from_options(options)
```

- STEP 3: Load the input image.

```
image = mp.Image.create_from_file(IMAGE_FILE)
```

- STEP 4: Detect objects in the input image.

```
detection_result = detector.detect(image)
```

- STEP 5: Process the detection result. In this case, visualize it.

```
image_copy = np.copy(image.numpy_view())
annotated_image = visualize(image_copy, detection_result)
rgb_annotated_image = cv2.cvtColor(annotated_image,
cv2.COLOR_BGR2RGB)
cv2_imshow(rgb_annotated_image)
```

B: Python code for Image processing based OpenCV model-

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import imutils
from google.colab.patches import cv2_imshow
from matplotlib import pyplot as plt
from google.colab import files
import os
from imutils import perspective
from imutils import contours
from scipy.spatial.distance import euclidean
from PIL import Image

def show_images(images):
    for i, img in enumerate(images):
        cv2_imshow(img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

- ```
  UPLOAD=files.upload()
  !wget "https://t4.ftcdn.net/jpg/02/11/17/49/360_F_211174954_zZonCPGppAr0MfRj52qhY5zENe3QbESQ.jpg"
  !wget "https://images.rawpixel.com/image_800/cHJpdmF0ZS9sci9pbWFnZXMvd2Vic2l0ZS8yMDIyLTA1L3BmLXMxMjQtYWstMjY4MV8yLmpwZw.jpg""
  ```

```
from google.colab import drive
drive.mount('/content/drive')
os.chdir('/content/drive/My Drive/Thesis/image')

uploaded_img=cv2.imread('Highlighter.jpg')
img = Image.open('phone.jpg')
bg=cv2.imread('phoneBG.jpg')
if uploaded_img is None:
        print('Image file is empty.')
else:
        org_img=cv2.cvtColor(uploaded_img, cv2.COLOR_BGR2RGB)
        gray_img=cv2.cvtColor(uploaded_img,cv2.COLOR_BGR2GRAY)

images=[uploaded_img,
        org_img,
        gray_img]
titles=['Uploaded Image','original Image','Gray Scale Image']

for i in range(3):
  plt.subplot(1,3,i+1),plt.imshow(images[i],cmap='gray')
```
```

```
  plt.title(titles[i])
  plt.xticks([]),plt.yticks([])
plt.show()

plt.imshow(org_img,cmap='gray'),plt.axis('off')

Gaussian = cv2.GaussianBlur(gray_img, (9, 9), 0)
plt.subplot(221),plt.imshow(Gaussian,cmap='gray')
hist = cv2.calcHist([Gaussian], [0], None, [256], [0, 256])
plt.subplot(223),plt.plot(hist)

cv2_imshow(Gaussian)
```

- global thresholding

```
ret,th1 = cv2.threshold(Gaussian,127,255,cv2.THRESH_BINARY)
th2 =
cv2.adaptiveThreshold(Gaussian,255,cv2.ADAPTIVE_THRESH_MEAN_C,
\
        cv2.THRESH_BINARY,11,2)
th3 =
cv2.adaptiveThreshold(Gaussian,255,cv2.ADAPTIVE_THRESH_GAUSSIA
N_C,\
        cv2.THRESH_BINARY,11,2)
```

- Otsu's thresholding after Gaussian filtering

```
blur = cv2.GaussianBlur(Gaussian,(5,5),0)
ret2,th2 =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
plt.subplot(221),plt.imshow(th2,cmap='gray')
hist = cv2.calcHist([th2], [0], None, [256], [0, 256])
plt.subplot(223),plt.plot(hist)

cv2_imshow(th2)

plt.imshow(bg, cmap='gray')

bgsub = cv2.createBackgroundSubtractorMOG2()
_ = bgsub.apply(bg)
fgmask = bgsub.apply(th2)
plt.imshow(fgmask,cmap='gray'),plt.axis('off')

cv2_imshow(fgmask)

inv_img=~th2
plt.subplot(1,2,1),plt.imshow(inv_img,cmap='gray')
hist = cv2.calcHist([inv_img], [0], None, [256], [0, 256])
plt.subplot(1,3,3),plt.plot(hist)

edges = cv2.Canny(inv_img,200,200)
plt.imshow(edges,cmap='gray'),plt.axis('off')

edged = cv2.dilate(edges, None, iterations=1)
```

```
edged = cv2.erode(edged, None, iterations=1)

cnts = cv2.findContours(edged,
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
biggest_contour=max(cnts,key=cv2.contourArea)
```

- Sort contours from left to right as leftmost contour is reference object

```
(cnts, _) = contours.sort_contours(cnts)
```

- Remove contours which are not large enough

```
cnts = [x for x in cnts if cv2.contourArea(x) > 1000]
```

- Reference object dimensions

```
box = cv2.minAreaRect(ref_object)
box = cv2.boxPoints(box)
box = np.array(box, dtype="int")
box = perspective.order_points(box)
(tl, tr, br, bl) = box
dist_in_pixel = euclidean(tl, tr)
dist_in_cm = 2
pixel_per_cm = dist_in_pixel/dist_in_cm

pixel_per_cm=237
max_wid = []
max_ht=[]
for cnt in cnts:
        box = cv2.minAreaRect(cnt)
        box = cv2.cv.BoxPoints(box) if imutils.is_cv2() else
cv2.boxPoints(box)
        box = np.array(box, dtype="int")
        box = perspective.order_points(box)
        (tl, tr, br, bl) = box
        cv2.drawContours(org_img, [box.astype("int")], -1, (0, 0, 255), 2)
        mid_pt_horizontal = (tl[0] + int(abs(tr[0] - tl[0])/2), tl[1] + int(abs(tr[1] -
tl[1])/2))
        mid_pt_verticle = (tr[0] + int(abs(tr[0] - br[0])/2), tr[1] + int(abs(tr[1] -
br[1])/2))
        wid = euclidean(tl, tr)/pixel_per_cm
        ht = euclidean(tr, br)/pixel_per_cm
        cv2.putText(org_img, "{:.1f}cm".format(wid),
(int(mid_pt_horizontal[0] -15), int(mid_pt_horizontal[1] -15)),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
        cv2.putText(org_img, "{:.1f}cm".format(ht), (int(mid_pt_verticle[0] +
10), int(mid_pt_verticle[1])),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

tl

show_images([org_img])

print('Width= '+ str(round(wid,3))+ ' cm'+ '\n' + 'height= '+ str(round(ht,3))+ '
cm')
```