A

PROJECT REPORT

ON

Visual Inspection of object using OpenCV and Image processing

SUBMITTED TO

SHIVAJI UNIVERSITY, KOLHAPUR

IN THE PARTIAL FULFILLMENT OF REQUIREMENT FOR THE AWARD OF DEGREE BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

MR.	Arihant Shital Dugge	18UCS019
MR.	Sahil Sahadev Chahande	19UCS019
MR.	Vishal Vijaykumar Babar	19UCS007
MR.	Sarvadnya Krishnat Kamble	19UCS054

UNDER THE GUIDANCE OF

Prof. V. V. Kheradkar

DKTE

Promoting Excellence in Teaching, Learning & Research

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DKTE SOCIETY'S TEXTILE AND ENGINEERING INSTITUTE, ICHALKARANJI

2022-23

D.K.T.E.SOCIETY'S

TEXTILE AND ENGINEERING INSTITUTE, ICHALKARANJI
(AN AUTONOUMOUS INSTITUTE)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DKTE
Promoting Excellence in
Teaching, Learning & Research

CERTIFICATE

This is to certify that, project work entitled

Inspection of object using OpenCV and Image processing

is a bonafide record of project work carried out in this college by

MR.	**Arihant Shital Dugge**	**18UCS019**
MR.	**Sahil Sahadev Chahande**	**19UCS019**
MR.	**Vishal Vijaykumar Babar**	**19UCS007**
MR.	**Sarvadnya Krishnat Kamble**	**19UCS054**

is in the partial fulfillment of award of degree Bachelor in Engineering in Computer Science & Engineering prescribed by Shivaji University, Kolhapur for the academic year 2022-23

PROF. V. V. KHERADKAR
(PROJECT GUIDE)

PROF.(DR.) D.V.KODAVADE
(HOD CSE DEPT.)

PROF.(DR.) L.S.ADHMUTHE
(DIRECTOR)

EXAMINER: ______________

____

DECLARATION

We hereby declare that, the project work report entitled “Inspection of object using OpenCV ad Image processing” which is being submitted to D.K.T.E. Society’s Textile and Engineering Institute Ichalkaranji, affiliated to Shivaji University,Kolhapur is in partial fulfillment of degree B.E.(CSE). It is a bonafide report of the work carried out by us. The material contained in this report has not been submitted to any university or institution for the award of any degree. Further, we declare that we have not violated any of the provisions under Copyright and Piracy / Cyber / IPR Act amended from time to time.

MR.	Arihant Shital Dugge	18UCS019
MR.	Sahil Sahadev Chahnde	19CUS019
MR.	Vishal Vijaykumar Babar	19UCS007
MR.	Sarvadnya Krishnat Kamble	19UCS054

ACKNOWLEDGEMENT

With great pleasure we wish to express our deep sense of gratitude to Prof.V.V.Kheadkar for his valuable guidance, support and encouragement in completion of this project report.
Also, we would like to take opportunity to thank our head of department Dr. D.V.Kodavade for his co-operation in preparing this project report.
We feel gratified to record our cordial thanks to other staff members of Computer Science and Engineering Department for their support, help and assistance which they extended as and when required.

Thank you,

MR.	Arihant Shital Dugge	18UCS019
MR.	Sahil Sahadev Chahande	19UCS019
MR.	Vishal Vijaykumar Babar	19UCS007
MR.	Savadnya Krishnat Kamble	19UCS054

ABSTRACT

Visual inspection is a crucial process in many industries, including manufacturing, healthcare, and agriculture, to ensure the quality and efficiency of products. With the advancements in computer vision and image processing techniques, automated visual inspection of objects has become increasingly popular. OpenCV, an open-source computer vision library, provides various image processing tools that enable efficient and accurate visual inspection of objects.

In this paper, we propose a method for visual inspection of objects using OpenCV. The proposed method involves capturing images of the object using a camera or we can add images from local storage. The images are then processed using OpenCV techniques such as thresholding, filtering, and contour detection. The processed output is analyzed to detect any defects or irregularities in the object.

The first step in the proposed method is to capture images of the object. The images can be captured using various imaging devices, such as cameras or scanners, depending on the application. Once the images are captured, they are processed using OpenCV techniques. Thresholding is used to separate the object from the background, and filtering is used to remove noise and improve the image quality. Contour detection is then applied to detect the boundaries of the object.

Once the object boundaries are detected, the output is analyzed to detect any defects or irregularities in the object. The defects can be detected by comparing the detected contours to a reference contour or template. Any deviations from the reference contour can indicate defects or irregularities in the object.

The proposed method is highly effective and efficient in detecting defects and irregularities in objects. It can be implemented in various industries, including manufacturing, healthcare, and agriculture, to improve quality control and productivity. Furthermore, it can be easily integrated into existing systems and automated processes.

INDEX

1.Introduction :

The ability to derive useful measurements from images has transformed several industries in the current digital era, from computer vision and robots to manufacturing and construction. In many applications, such as quality control, item recognition, and geographical analysis, precise measurements of things are essential. It is at this point that OpenCV (Open Source Computer Vision Library) shows its true potential as a potent tool, offering a solid framework for picture processing and analysis.

The purpose of this project is to investigate OpenCV's capabilities and show how it has the capacity to measure objects accurately and consistently. We can accurately extract dimensions, distances, and other important information from images or video streams by using computer vision techniques. This improves the effectiveness of many industrial processes while also creating new opportunities for automation, augmented reality, and other things.

Initially at industrial level the Inspection was done manually which was more time consuming. Labours cost was increased and after particular time that is due to change in temperature there is slight error in instrument or human error. To avoid this, we introduce Visual Inspection Using Python.

A crucial role in many industries, such as manufacturing, robotics, and security, is visual inspection of things. Object inspection by hand can be labor-intensive, time-consuming, and prone to mistakes. Consequently, there is a demand for automated visual inspection systems that can properly and quickly recognise and classify items and their properties.

A well-liked library for computer vision and image processing applications is called OpenCV (Open Source Computer Vision). C++, Python, and Java are just a few of the programming languages supported by this open-source platform. OpenCV is a useful tool for visual inspection applications since it offers a variety of image processing, feature extraction, and machine learning functionalities.

Most important & time reducing work for Quality Department that is for checking dimension of Object. Which is Visual Inspection using python (OpenCV). We can find the dimensions using base marker called “Aruco” {whose perimeter is defined in code}. When Object is placed along with “Aruco” it calculates dimension of Object (Length and Breath) using Ratio and Proportion.

Assessing the accuracy and precision of the measuring techniques used in this project is a crucial component. To evaluate the system's dependability, we shall compare the measured values to actual data or established standards.

The main approach of project is to demonstrate the strength and adaptability of computer vision technology by delving into the field of object measuring using OpenCV. Through this project, we hope to develop image processing and offer insightful information to researchers, engineers, and enthusiasts that are interested in utilising OpenCV in their own applications.

1.1 Problem Definition :

The issue that the visual object inspection using OpenCV attempts to solve is the requirement for a quick and accurate object inspection in order to guarantee the effectiveness and quality of the final result. It is challenging to meet the expectations of contemporary industries using traditional methods of visual inspection since they take a long time and are prone to mistakes. In order to increase inspection speed and accuracy, the suggested method automates the visual inspection procedure using OpenCV, a well-known and effective computer vision library.

The key problem is to provide a method for flaw detection and image processing that is successful and efficient and can be used across a variety of industries and object kinds. The technique also needs to be strong enough to handle variations in background, lighting, and other environmental elements that can degrade image quality. Additionally, the system must be adaptive to various inspection conditions and requirements. The ultimate objective is to offer a dependable and affordable visual inspection solution that can raise product quality and productivity while lowering manual labour and errors

1.2 Aim and Objectives of project :

` using image processing techniques, the goal of visual object inspection using OpenCV is to provide an automated and effective method to find flaws or anomalies in an object. The suggested method attempts to decrease manual inspection's labour requirements and associated expenses while increasing inspection speed and accuracy..

The main objective of visual inspection using OpenCV is to develop an effective and robust system that can detect defects or irregularities in an object using image processing techniques. The system should be capable of processing images captured by cameras or scanners and detecting any deviations from the expected shape, size, or color of the object. The system should be able to detect various types of defects, such as scratches, cracks, dents, and deformations.

Additionally, the suggested solution intends to be flexible enough to accommodate varied inspection scenarios and requirements, making it appropriate for application across a range of sectors, including manufacturing, healthcare, and agriculture. The system must be capable of handling changes in background, illumination, and other environmental elements that can impair image quality.

In general, the goal of visual object inspection using OpenCV is to offer a quick, accurate, and affordable solution for visual inspection that can raise productivity and boost quality while lowering manual labour and errors.

1.3 Scope and Limitation of Project :

The scope of visual inspection of object using OpenCV is vast, as it can be applied in various industries and applications where quality control and defect detection are essential. Some of the areas where visual inspection using OpenCV can be applied include:

1. Manufacturing: Visual inspection can be used in the manufacturing industry to detect defects in products during the production process, ensuring that they meet the required standards.

2. Healthcare: Visual inspection can be used in healthcare applications, such as pathology and radiology, to detect abnormalities in medical images.

3. Agriculture: Visual inspection can be used in agriculture to detect defects in crops or fruits, ensuring that only high-quality products are sold.

4. Robotics: Visual inspection can be used in robotics to enable robots to detect and classify objects based on their visual appearance.

However, there are also limitations to the visual inspection of object using OpenCV. Some of these limitations include:

1. Complexity of objects: The visual inspection system may struggle to detect defects or irregularities in complex objects or objects with varying shapes and sizes.

2. Lighting conditions: The system's performance may be affected by changes in lighting conditions, which can cause shadows, reflections, or glare that may interfere with the detection process.

3. Limited accuracy: Although visual inspection systems can achieve high levels of accuracy, they may still miss defects that are too small or too subtle to be detected.

Overall, while visual inspection using OpenCV has great potential to improve product quality and efficiency, it is important to consider its limitations when implementing it in specific applications.

1.4 Timeline of the project :

Month	**Work Done**
July	Went through research papers from IEEE and shortlisted some of them for reference.
August	Worked on key points of synopsis and presentation like introduction, aim, goal and scope.
September	Worked to create some diagrams such as use case diagrams, Activity diagram, Data flow diagram, Sequence diagram.
October	Worked on methodologies required.
November	Completed the SRS and Design document report also covered the synopsis report.
December	Observation and Analysis of object detection techniques.
January	Worked on all Dependencies required for detection and predicting measurements.
February	Learned basics technologies required for project.
March	Worked on all environmental conditions for object detection.
April	Completed Object Detection and finding measurements in terms of pixels.
May	Prepared final Soft Copy for presentation and work in progress for creating website.

1.5 Project Management Plan :

1.Project Scope:

The scope of the project is to develop a visual inspection system using OpenCV to detect defects and anomalies in manufactured products. The system will use computer vision techniques to analyze images of the products and identify any deviations from the expected quality standards.

2.Project Objectives:

1.To develop a robust and reliable visual inspection system using OpenCV
2.To achieve high accuracy and precision in defect detection
3.To minimize the false-positive and false-negative rates
4.To optimize the processing time of the system

3.Project Team:

1.Project Manager: responsible for overall planning, coordination, and monitoring of the project.
2.Software Developer: responsible for designing and implementing the visual inspection system using OpenCV.
3.Image Processing Expert: responsible for optimizing the image processing algorithms and ensuring the accuracy of defect detection.
4.Quality Assurance Specialist: responsible for testing and verifying the system's performance against the quality standards.
5.Technical Writer: responsible for documenting the project requirements, design, and testing results.

4. Project Deliverables:

1.Detailed project requirements and design documents.
2.Source code and executable files of the visual inspection system.
3.Test reports and quality assurance documents.

5. Project Risks:

1.Technical difficulties in implementing the image processing algorithms.
2.Inadequate hardware resources for processing large volumes of images.
3.Insufficient user feedback and testing in real-world manufacturing environments.
4.Changes in project scope or requirements due to customer needs or market conditions.

7. Project Budget:

1.Personnel costs for the project team
2.Hardware and software resources for development and testing
3.Contingency funds for unexpected events or changes in project scope

8.Project Communication:

The project team will communicate regularly through team meetings, progress reports, and email updates. The project manager will also maintain regular communication with the project sponsor and other stakeholders to ensure their satisfaction with the project outcomes.

1.6 Project Cost :

Hardware	**Cost**
Computer System	40000
Camera (RaspberryPie or any other) / Mobile Phone (Android)	7000-8000
Python IDE to run machine learning module	Open Source
Libraries required	Open Source
Aruco Marker	Open Source

a. <u>Project cost</u>

Lines of Code (LOC):285
Effort = a * (LOC) ^ b
Time = c * (Effort) ^ d
Persons Required = Effort / Time

For COCOMO model parameters:
a = 2.4 (constant for organic projects)
b = 1.05 (exponent derived from historical data)
c = 2.5 (constant for organic projects)
d = 0.38 (exponent derived from historical data)

Calculate Effort:
Effort = 2.4 * (285) ^ 1.05
Effort = 2.4 * 378.082
Effort = 907.3991 Person-Hours (approximately)

Calculate Time:
Time = 2.5 * (907.3991) ^ 0.38
Time = 2.5 * 28.7737
Time = 71.9343 Hours (approximately)

Calculate Persons Required:
Persons Required = Effort / Time
Persons Required = 907.3991 / 71.9343
Persons Required = 12.6142 (approximately=12)

Therefore, based on the given 295 lines of code, the estimated cost using the COCOMO model is:
Effort: 907.3991 Person-Hours
Time: 71.9343 Hours
Persons Required: 12.6142 (approximately=12)

2. Background study and literature overview

2.1 Literature Overview :

1. In 2017, Bhumika Gupta et al. proposed improving the efficiency and accuracy of object detection using a color-based approach and implementing OpenCV library in Python with the help of NumPy.

1. In 2021, Shreyas N Srivatsa et al. proposed object detection using deep learning with OpenCV and Python using the You Only Look Once (YOLO) approach.

2. In 2017, Mukesh Tiwari and Dr. Rakesh Singhai proposed modifying the approach towards object tracking and detection using video sequences through different phases and identified gaps to improve object tracking over video frames.

3. In 2018, Karanbir Chahal and Kuntal Dey proposed a survey of modern object detection literature using deep learning and explored techniques to make networks portable through the use of lighter convolutional bases.

4. In 2019, R. Sujeetha and Vaibhav Mishra proposed a review of detection and tracking of objects using TensorFlow and faced problems in implementing real-time tracking over dynamic computation efficient performance.

5. In 2014, R. Girshick et al. proposed a simple and scalable detection algorithm that improves mean average precision (MAP) by combining high-capacity convolutional neural networks (CNNs) to bottom-up region proposals to localize and segment objects.

6. In 2016, K. He et al. proposed deep residual learning for image recognition, which naturally integrates low/medium/high level features and classifies an end-to-end multilayer fashion and levels.

7. In 2018, Pengchong Jin et al. proposed a Pooling Pyramid Network for Object Detection, which is effective in reducing a model size without reducing the quality.

8. In 2016, Shanghai Liao et al. proposed a fast and accurate unconstrained face detector by addressing challenges in unconstrained face detection and proposing a new image feature, the Normalized Pixel Difference (NDP).

2.2 Critical appraisal of other people's work :

1. "Object Detection and Recognition using OpenCV" by R. P. Jain and P. K. Chaudhari: This paper presents a method for object detection and recognition using OpenCV. The authors have used the Viola-Jones algorithm for face detection and SURF (Speeded-Up Robust Features) for object recognition. However, the paper lacks details about the performance of the proposed method in terms of accuracy and computational efficiency. Moreover, the experimental results are not presented with sufficient detail, making it difficult to evaluate the effectiveness of the proposed approach**.**

2. "Visual Inspection System for Defect Detection in Electronic Components using OpenCV" by V. S. S. Kumar, S. S. Anand, and S. Ramakrishnan: This paper proposes a visual inspection system for detecting defects in electronic components using OpenCV. The authors have used various image processing techniques such as thresholding, erosion, dilation, and contour detection to detect defects. The paper provides detailed experimental results and compares the proposed approach with other existing methods. However, the authors have not provided details about the dataset used for experimentation, which makes it difficult to evaluate the generality of the proposed approach.

3. "Automatic defect detection and classification of welding seams in steel plates using OpenCV" by T. Wu, Y. Jiang, and C. Zhang: This paper proposes an automatic defect detection and classification system for welding seams in steel plates using OpenCV. The authors have used various image processing techniques such as Canny edge detection, morphological operations, and contour detection for defect detection and classification. The paper provides detailed experimental results and compares the proposed approach with other existing methods. However, the authors have not provided details about the dataset used for experimentation, which makes it difficult to evaluate the generality of the proposed approach.

2.3 Investigation of current project and related work :

In the proposed system a USB Camera or an image from local storage is used to detect the object the environment which is and the image processing technique is implemented in the proposed system.
It followed various phases of image processing such as:
1.Image acquisition by camera.
2.Identifying contours.
3.Detecting Aruco
4.Image preprocessing
5.Analysis and result
Here we observed that the accuracy of output was improved ,the set up was easy as compared to old methods and most important the system was user friendly.

3.Requirement analysis :

3.1 Requirement Gathering :

The requirement gathering for the visual inspection of objects using OpenCV involves identifying and understanding the needs of the users and stakeholders of the system. This includes determining the specific objects to be inspected, the features and characteristics to be examined, the type of defects or anomalies to be detected, and the environmental conditions in which the inspection will take place.

1. Analyze existing systems: Existing systems that are similar to the proposed visual inspection system can be analyzed to identify their strengths and weaknesses. This information can be used to improve the proposed system.
2. Identify objects to be inspected: The objects to be inspected should be identified based on the application of the system. For example, if the system is being developed for defect detection in manufacturing, the objects could be machine parts or finished products.
3. Determine features to be inspected: The features and characteristics of the objects that need to be inspected should be identified. This includes determining the type of defects or anomalies to be detected.
4. Consider environmental conditions: The environmental conditions in which the inspection will take place should be considered. For example, lighting conditions, temperature, and humidity can affect the performance of the system.
5. Define performance metrics: Performance metrics should be defined to evaluate the effectiveness of the system. This includes metrics such as accuracy, precision, and recall.

3.2 Requirement Specification :

1.Functional Requirements:

1. The system should be able to take input from a camera or image files for inspection.
2. The system should be able to detect objects and their boundaries in the input image.
3. The system should be able to classify the object based on its features and identify the defects present.
4. The system should be able to provide an accurate measurement of the dimensions of the object.
5. The system should be able to store the inspection results and provide reports.

2.Non-Functional Requirements:

1. The system should have a user-friendly interface for ease of use and configuration.
2. The system should be able to handle high-speed processing for real-time inspection applications.
3. The system should have high accuracy and reliability for accurate defect detection.
4. The system should be easily scalable for integration into existing manufacturing or quality control environments.
5. The system should have good performance and stability.

3.Assumptions:

1. The system will be developed using OpenCV library and Python programming language.
2. The system will be developed for a Windows operating system.
3.The system will be integrated with a standard camera or webcam for image capture.
4.The system will be designed to inspect objects with a relatively uniform surface.
5.The system will require a calibration step to ensure accuracy and reliability.

Constraints:

1.The system must be developed within a budget of 40000 INR.
2.The system must be completed within a time frame of six months.
3.The system must be able to run on a standard desktop computer with a minimum of 8GB RAM and an Intel Core i5 processor.
4.The system must be able to operate in a manufacturing or quality control environment.

3.3 Use Case Diagram :

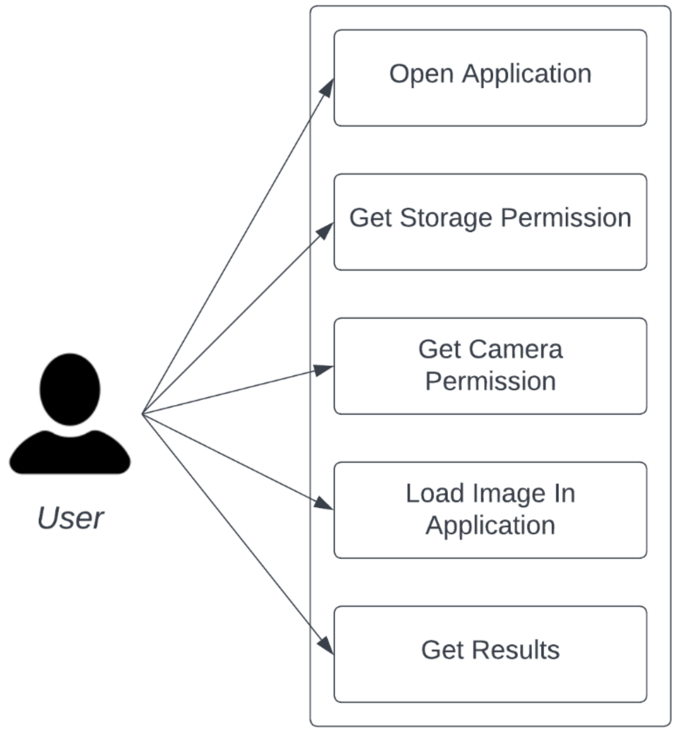


Fig 3.3.1

Actors:The person using the system to conduct activities like object measurement.
Image Source: Denotes the input source from which images or frames are acquired, such as a camera or previously recorded video.

Use Cases:The process of capturing or retrieving an image or video frame from the image source is started by the user.
Preprocess Image: The user asks the system to perform preprocessing methods to improve the acquired image, such as scaling, denoising, and colour correction.
Identify and find things of interest inside the preprocessed image using OpenCV's computer vision techniques. Detect things: The user initiates the object detection process.
Calculate Measurements: After the user starts the measurement computation process, the system uses calibration parameters, known objects, or reference scales to calculate the size or dimensions of the detected objects.
Display Measurements: The system generates reports and shows the user the measured object dimensions by superimposing them on the image or video frame.

4. System Design :

4.1Architectural Design Diagram :

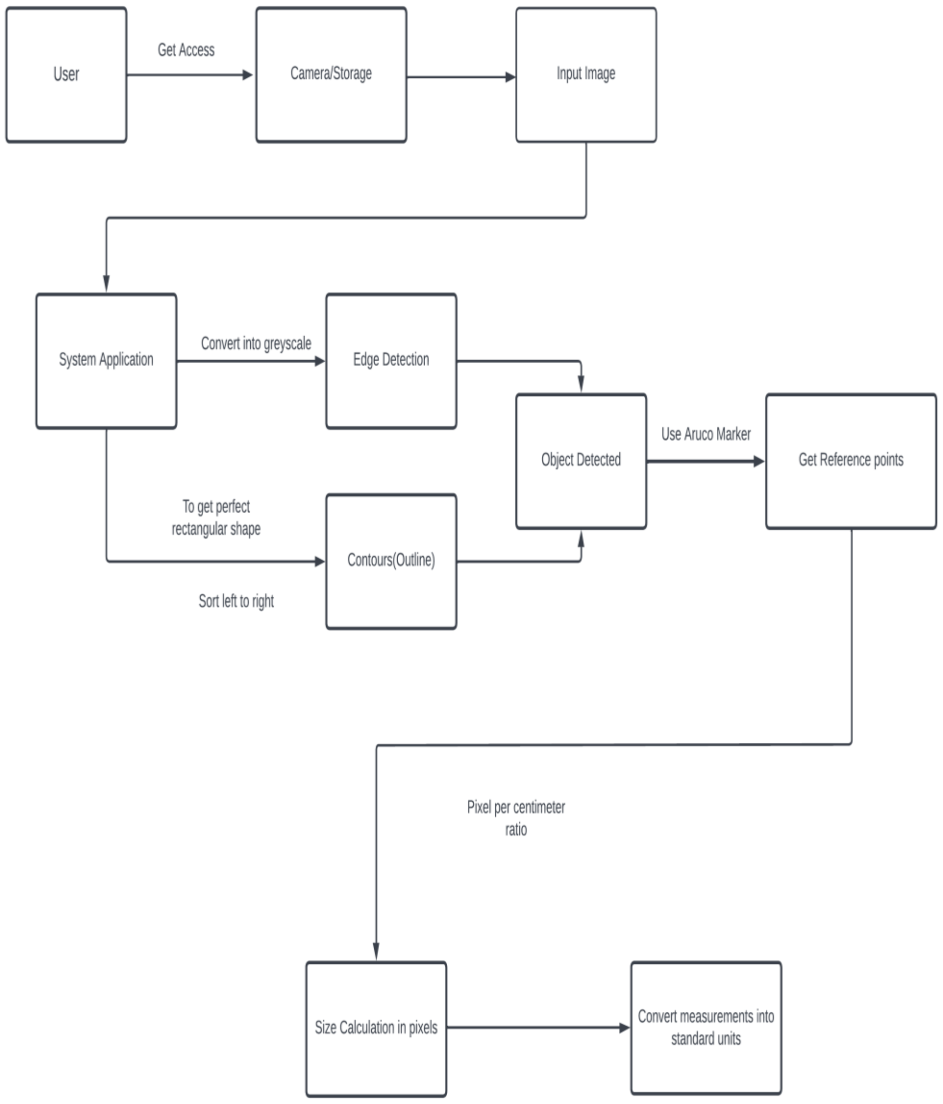
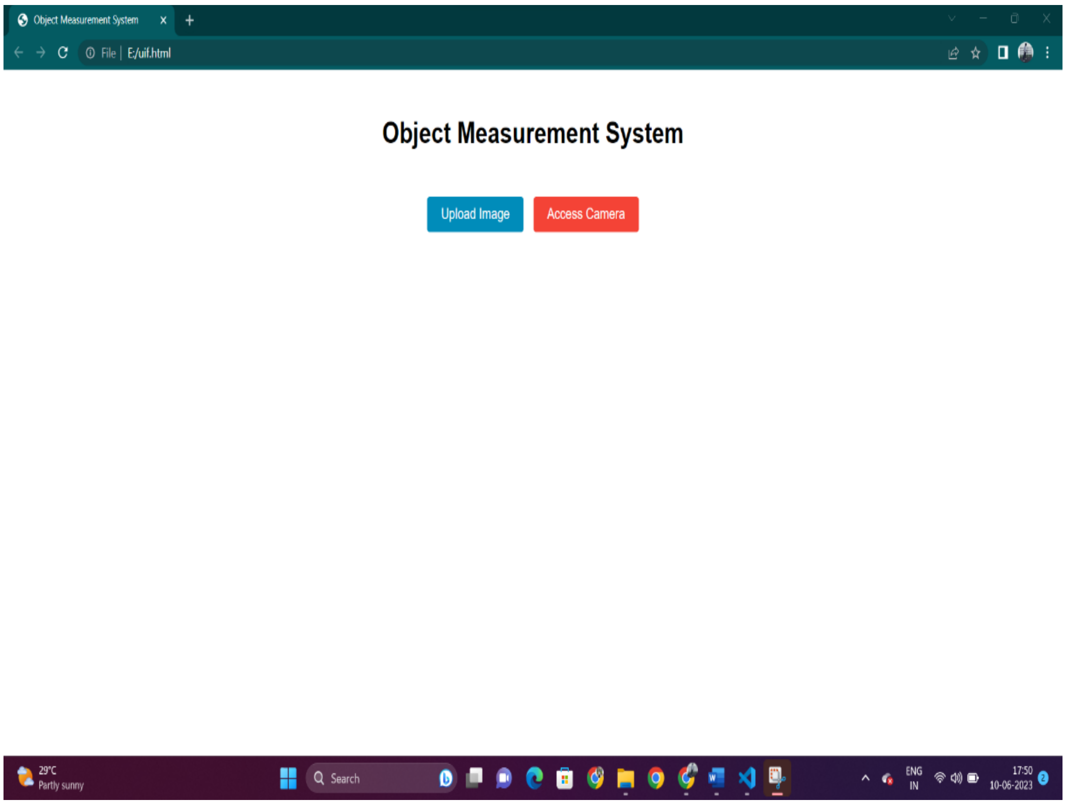


Fig 4.1.1

The user interacts with the system, providing input and receiving output. The Image Acquisition module captures or retrieves images or video frames from an input source. The Preprocessing module enhances the quality of the acquired frames by performing operations like resizing, denoising, and color correction.

The Object Detection module uses OpenCV's computer vision algorithms to detect objects in the preprocessed images, generating bounding boxes or contours around them. The Measurement Calculation module processes the detected objects and computes their measurements, incorporating reference objects or calibration parameters. The Calibration module adjusts the system to account for optical distortions or variations in the imaging setup. The User Interface module offers an interactive interface for users, presenting processed images with object measurements and enabling user input for object selection or parameter adjustment. The Output Visualization module creates visual representations of the measured object dimensions, such as annotated images or videos, reports, or real-time overlays on a live video feed.

4.2 User Interface Design :



4.3Algorithmic description of each module :

1.Initialize the system: The camera or video source should be set up for picture capture. Load the required libraries, such as OpenCV.

2.Aquire an image or video frame: Obtain or capture a frame of a picture from the input source.

3.Preprocess the image: Resize the picture to make it more practical.Utilise denoising methods to lessen noise.If necessary, correct any colour distortions or imbalances.

4.Detet Objects: Use an object detection technique to find items in the preprocessed image, such as YOLO, SSD, or Haar cascades. Draw the contours or bounding boxes around the identified objects.

5.Calculate measurements: Identify the calibration reference scale or known dimension, if available.Identify the object's pertinent characteristics or dimensions, such as its length, width, area, or other dimensions.In the absence of a reference scale, determine the object's size by comparing it to a known object or by using calibration parameters.

6.Display output and measurements: Measurements can be seen by superimposing them over an image or video frame.Present the measures in a way that is easy for the user to understand, for as using reports, annotated photos, or real-time overlays over a live video feed.

7.Terminate The system : End the program and ask for new input image

4.4System Modeling :

4.4.1Dataflow Diagram :

DFD Level 0

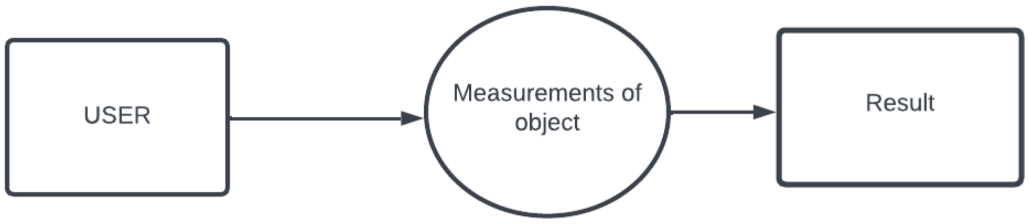


Fig 4.4.1.1

DFD Level 1

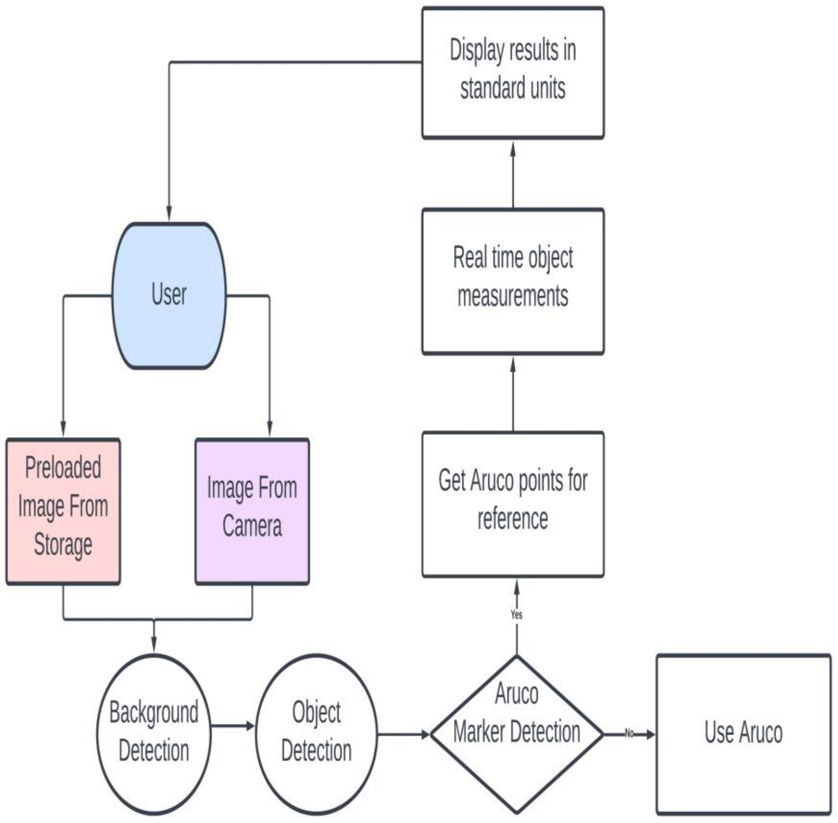

Display results in standard units
Real time object measurements
Get Aruco points for reference
User
Preloaded Image From Storage
Image From Camera
Background Detection
Object Detection
Aruco Marker Detection
Yes
No
Use Aruco

Fig 4.4.1.2

Entering Data:

Images or video frames from an input source, such as a camera or previously recorded video, make up the input data.

Process of acquiring an image:

After receiving the input data, the picture acquisition procedure takes or recovers the images or video frames from the input source.

Process of preprocessing:

Preprocessing improves the quality of the data by performing operations like scaling, denoising, and colour correction on the acquired pictures or video frames as input.

Process for detecting objects

The preprocessed data is provided to the Object Detection process, which then employs OpenCV's computer vision algorithms to find objects within the frames or images. It creates contours or bounding boxes around the observed items.

Measurement Calculation Process: The measurement calculation procedure calculates the measurements of the items that were discovered during the Object Detection phase. The dimensions of the items may be determined using reference objects, calibration parameters, or other methods.

User Interface Process: Users can interact with the system using an interactive interface that is provided by the user interface process. Users can choose an object to pick or change parameters as it presents the processed photos or frames with object measurements.

Output Data: The visual representations of the measured object dimensions that are shown to the user through the User Interface procedure make up the output data.

4.4.2 Sequence Diagram :

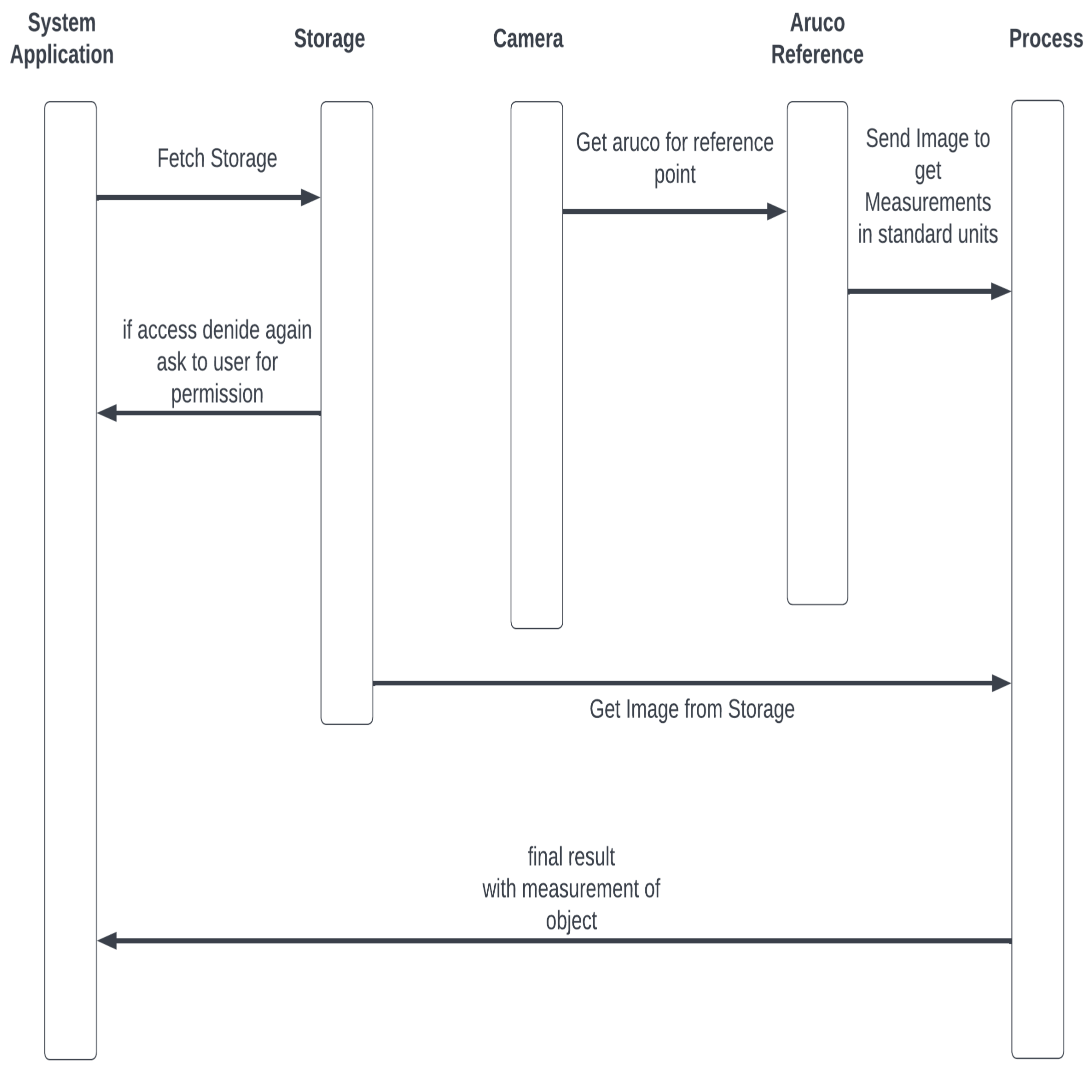

System Application
Storage
Camera
Aruco Reference
Process
Fetch Storage
Get aruco for reference point
Send Image to get Measurements in standard units
if access denide again ask to user for permission
Get Image from Storage
final result with measurement of object

Fig 4.4.2

1.To begin object measurement, the user makes a request.
2.After receiving the request, the user interface component presents them with the proper interface.
3.In order to collect or obtain images or video frames from the specified source, the User Interface component requests it from the Image Acquisition component.
4.The user interface component receives the images or video frames that were captured or retrieved by the image acquisition component.
5.The preprocessing component receives the collected data from the user interface component.
6.On the obtained data, the preprocessing component applies actions like scaling, denoising, and colour correction.
7.The Object Detection component receives the preprocessed data from the Preprocessing component.

8.The component for object detection uses computer vision techniques from OpenCV to identify items in the preprocessed data and create bounding boxes or contours around them.
9.The User Interface component receives the discovered objects from the Object Detection component.
10.The measurement calculation component receives the objects that have been detected from the user interface component.
11.Utilising calibration parameters or reference objects, the measurement calculation component analyses the detected items and determines their measurements.
12.The component for measurement calculation transmits the determined measures back to the component for the user interface.
13.The user is shown the processed images or frames with object measurements through the user interface component.
14.Through the User Interface component, the Output Visualisation component shows the user the visual representations.
15.The user can ask for further procedures or ask to stop the measurement process, continuing the interaction sequence.

4.4.3Activity Diagram :

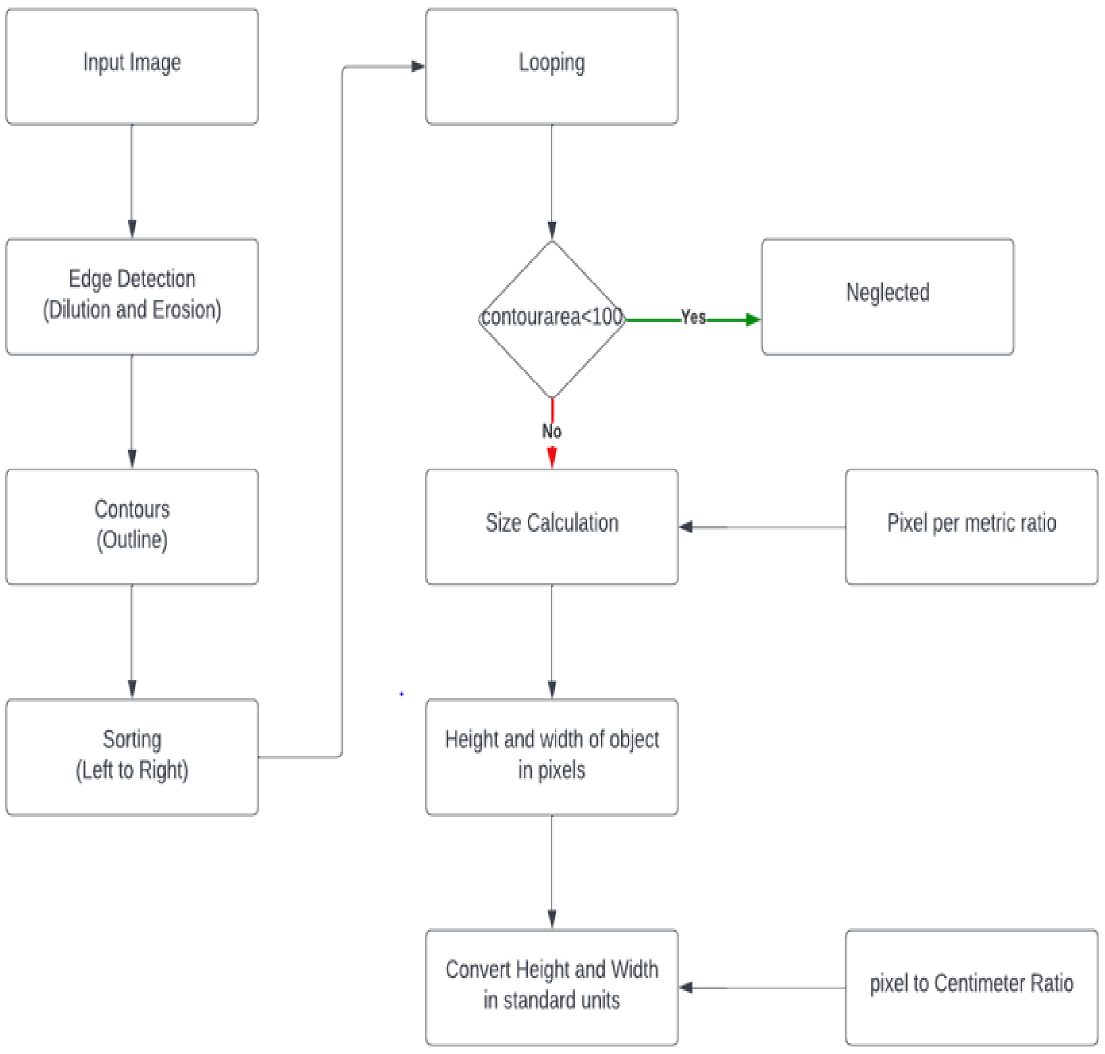

Input Image
Edge Detection (Dilution and Erosion)
Contours (Outline)
Sorting (Left to Right)
Looping
contourarea<100
Yes
Neglected
No
Size Calculation
Pixel per metric ratio
Height and width of object in pixels
Convert Height and Width in standard units
pixel to Centimeter Ratio

Fig 4.4.3

Start:
The first activity, which symbolises the commencement of the item measuring procedure, is where the activity diagram starts.

Image/Video Frame Capture:
The system extracts pictures or video frames from the input source and captures them. Interacting with the Image Acquisition component is required for this activity.

Processing Image:
Preprocessing techniques on the captured pictures or video frames include scaling, denoising,

and colour correction. The Preprocessing component carries out these actions.

Find Objects:

Computer vision techniques from OpenCV are used to process the preprocessed images in order to find things within them. This task is carried out by the Object Detection component, which also creates bounding boxes or contours around the items it finds.

Measurements are computed:

The Measurement Calculation component computes the measurements of the identified items before passing them forward. Utilising calibration parameters or reference items may be part of this activity.

Display Dimensions:

The user is shown by the system the measured object dimensions. This task entails creating visual representations, such as reports, reports with annotations, annotated photos or videos, or real-time overlays over a live video feed. This process is made easier by the Output Visualisation component.

4.4.4.Component Diagram **:**

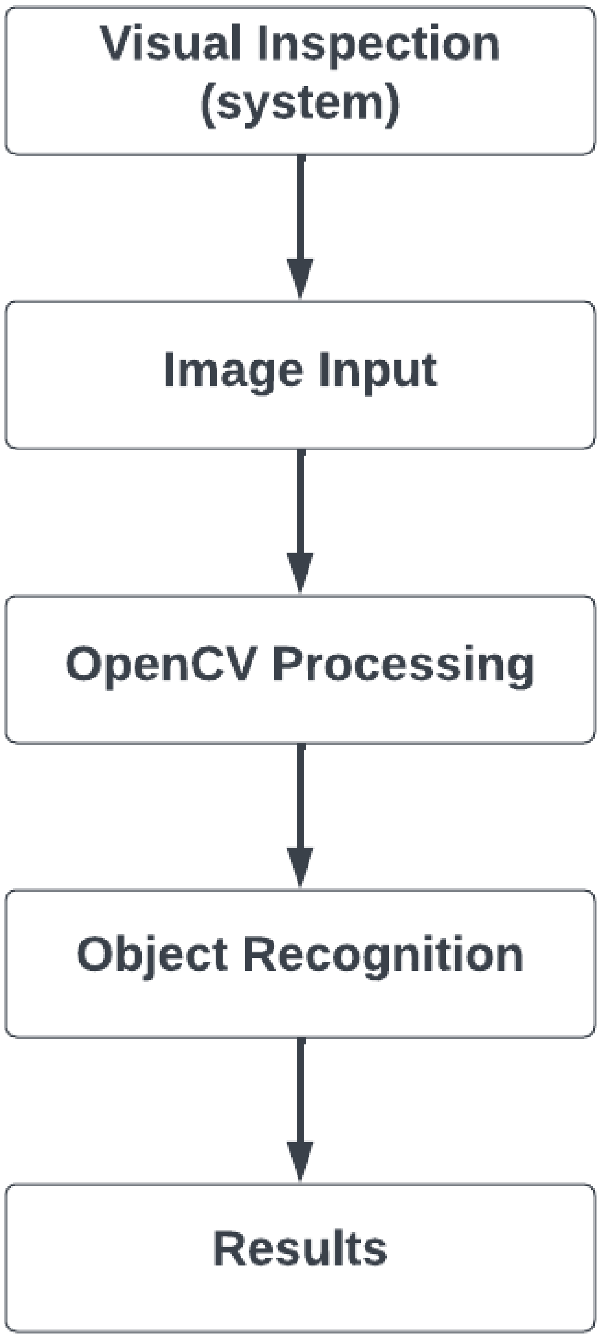


Fig 4.4.4

5.Implementation :

5.1 Environmental Setting for Running the Project :

Hardware: In order to process and analyse images, enough hardware must be employed. It should be equipped with a quick processor, sufficient memory, and a separate graphics card for picture processing. To capture the image in all of its detail, the camera being used should have a good resolution.

Software: In order to function, OpenCV needs a particular setup. It is advised to utilise the most recent OpenCV version along with a suitable IDE like PyCharm, Visual Studio, or Eclipse. It's crucial to confirm that the system has the correct version of Python or C++ installed.

Operating system: OpenCV is compatible with a variety of OSs, including Windows, Linux, and macOS. The operating system of choice should work with the applications used to create and maintain the system.

Lighting: To guarantee that the camera captures the image in detail, the object being examined should be well-lit. Accurate defect detection will benefit from a well-lit setting.

Temperature and Humidity: To avoid causing any harm to the hardware components, the environment's temperature and humidity should be under control. High humidity or temperature might harm the camera, limiting its capacity to take quality pictures.

Power supply: To prevent data loss or hardware damage, the power supply should be reliable and uninterruptible..

5.2 Detailed Description of Methods :

Image Thresholding : Thresholding is a simple image segmentation technique used to separate objects from the background in an image..

Image filtering : The process of applying several filters to a picture in order to minimise noise or highlight specific features, such as edge detection, sharpening, and blur, is known as image filtering. The filter2D() function in OpenCV can be used to apply a filter to an image.

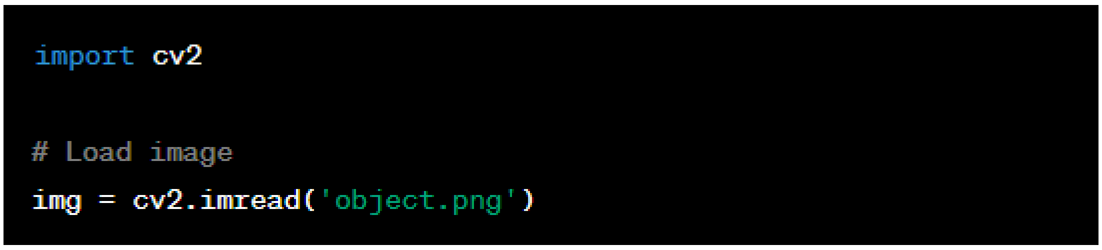
Contour detection: A curve linking all of an object's continuous points in an image serves as a representation of that object's border. Contour detection is the technique of locating and extracting an object's contours from an image. For this, OpenCV's findContours() technique can be utilised.

Object detection: It is the act of identifying the sort or class of an object, while object detection is the process of determining whether a specific object is present in a photograph. OpenCV provides a variety of methods for object detection and recognition, including the Haar cascade, the HOG (Histogram of Oriented Gradients), and Deep Learning-based algorithms like YOLO (You Only Look Once). Identifying if a certain object is present in a photo is the process of object detection, whereas identifying the kind or class of an object is the process of object recognition. OpenCV provides a variety of methods for object detection and recognition, including the Haar cascade, the HOG (Histogram of Oriented Gradients), and Deep Learning-based algorithms like YOLO (You Only Look Once).

5.3 Implementation Details :

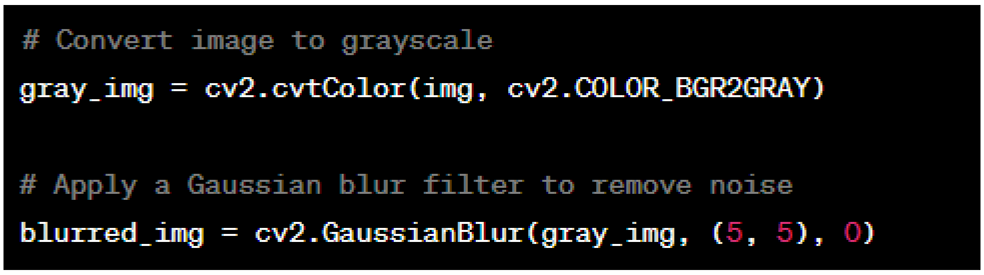
Upload a picture:

The image of the object you wish to investigate must first be loaded. To load the image, use OpenCV's imread() method.



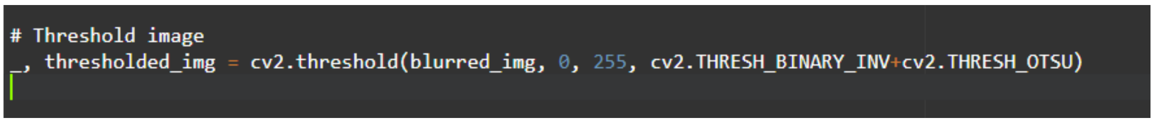
```
import cv2

# Load image
img = cv2.imread('object.png')
```

Preprocessing: You might need to preprocess the image before using any visual examination procedures. The image may need to be resized, made grayscale, or noise-reduction filters used.



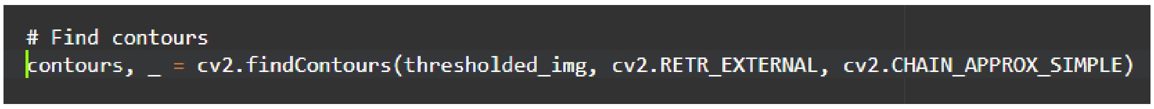
```
# Convert image to grayscale
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply a Gaussian blur filter to remove noise
blurred_img = cv2.GaussianBlur(gray_img, (5, 5), 0)
```

Thresholding: After the image has undergone preprocessing, thresholding can be used to separate the item from the background. We'll use a basic binary threshold in this example.



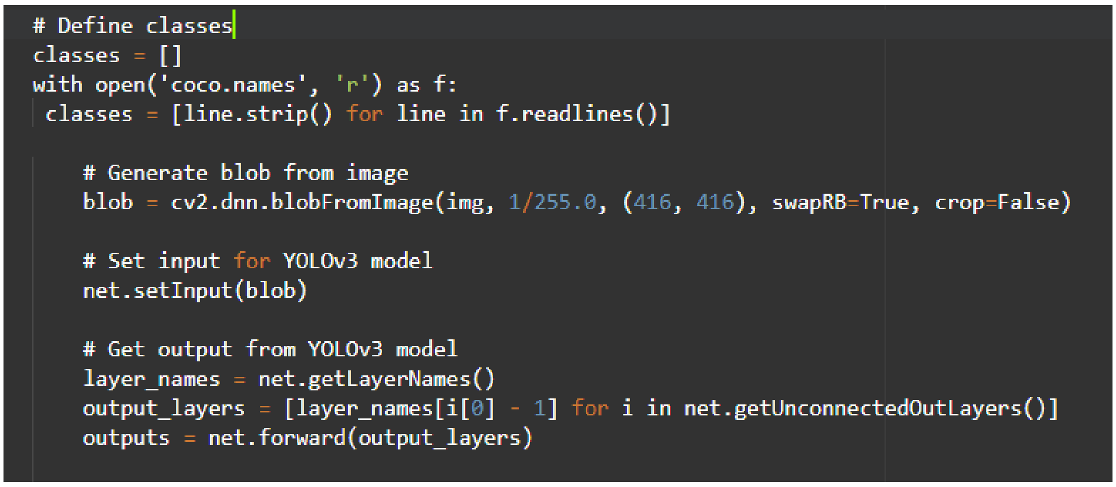
```
# Threshold image
_, thresholded_img = cv2.threshold(blurred_img, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

Contour Detection : Following that, we may utilise contour detection to locate the object's contour in the thresholded image.



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

Object Detection and Recognition : Last but not least, we may identify and categorise the object in the image using object detection and recognition algorithms. We'll apply the YOLOv3 deep learning model in this case.



```
# Define classes
classes = []
with open('coco.names', 'r') as f:
 classes = [line.strip() for line in f.readlines()]

    # Generate blob from image
    blob = cv2.dnn.blobFromImage(img, 1/255.0, (416, 416), swapRB=True, crop=False)

    # Set input for YOLOv3 model
    net.setInput(blob)

    # Get output from YOLOv3 model
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
    outputs = net.forward(output_layers)
```

6.Integration and Testing :

Integration: Integrate the essential code into your project by adding it to your current codebase or by developing a new module. Make sure to import all required OpenCV libraries, functions, and dependencies.

Testing: You may test the visual inspection code to verify if it produces the desired results by using a sample image of the object you'd like to inspect. To observe how different variables effect the result, you can experiment with changing the input parameters.To ensure that the code works properly in a range of settings, it is also crucial to test the code using a number of images and objects. Utilise test cases to automate testing and ensure that the code is functioning as intended..

Optimization : If the code is operating too slowly or generating unreliable results, it can be improved by adjusting the parameters or utilising more sophisticated methods. To increase the accuracy of the outcomes, you may, for instance, experiment with various thresholding techniques or employ a more sophisticated object detection model.

Overall, integrating and testing the visual inspection code entails experimenting with various settings and methods until you get the desired outcomes. A strong and trustworthy visual inspection system that can be applied in a number of applications can be built with enough testing and optimisation.

6.1 Description of the Integration Modules :

1.Image Acquisition module : This module is in charge of taking pictures or pulling video frames from an input source, either a camera or a previously recorded video. It might have features like starting up the camera, adjusting the resolution and frame rate, and giving users access to specific frames for additional processing.

2.Preprocessing module : The preprocessing module first improves the quality of the obtained images and gets them ready for further analysis. To assure uniformity and increase the accuracy of following processing processes, it may involve operations like resizing, cropping, denoising, and colour correction.

3.Object detection module : The object detection module locates and identifies things of interest in the captured images by using computer vision algorithms and techniques from OpenCV.

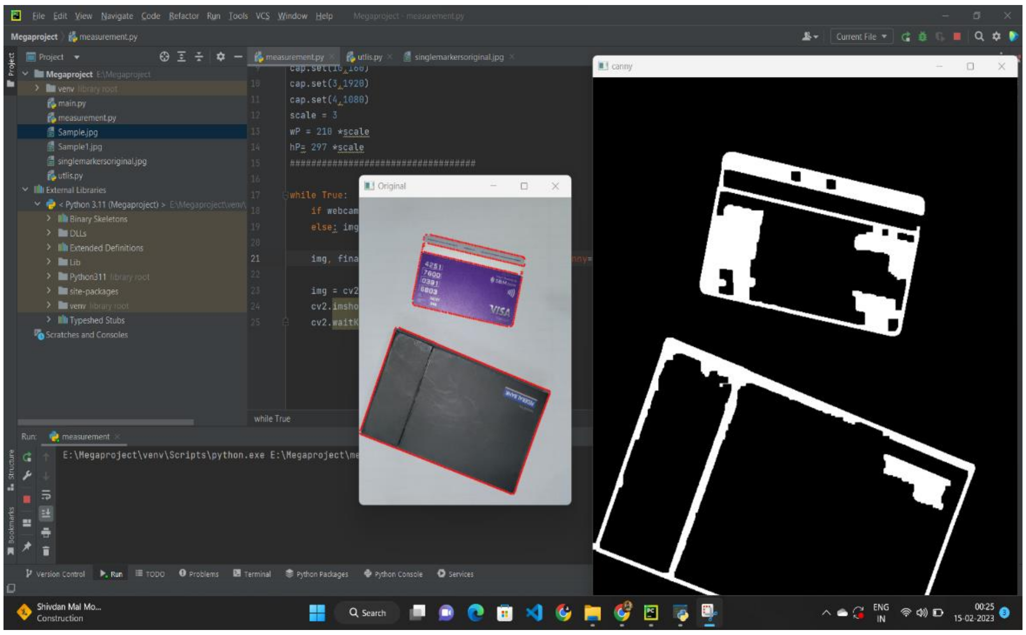
4.Measurement Calculation module : When an object is discovered, the measurement calculation module uses the detected regions to calculate the object's size or dimensions. This module can make use of a variety of methods, including employing well-known reference objects in the scene, comparing the detected objects' sizes to a reference scale, and using calibration parameters. It is possible to retrieve the measures in terms of length, width, area, or other pertinent dimensions.

5.User Interface module : Users can interact with the system using an interactive interface provided by the user interface module. It might have features like showing processed photos with object measurements superimposed, letting users manually select objects for measurement, giving users the chance to change settings and parameters, and presenting the final data in an approachable way.

6.Output Visualization module : The output visualisation module delivers the measured object dimensions in a comprehensible manner through visual representation. It can produce measurements-accompanied photos or videos of the things.

6.2Testing :

1.Unit Testing : Testing individual system parts such as object recognition, marker detection, and picture preprocessing is known as unit testing. Examples of photos or films that cover various scenarios and edge cases might be used for unit testing.



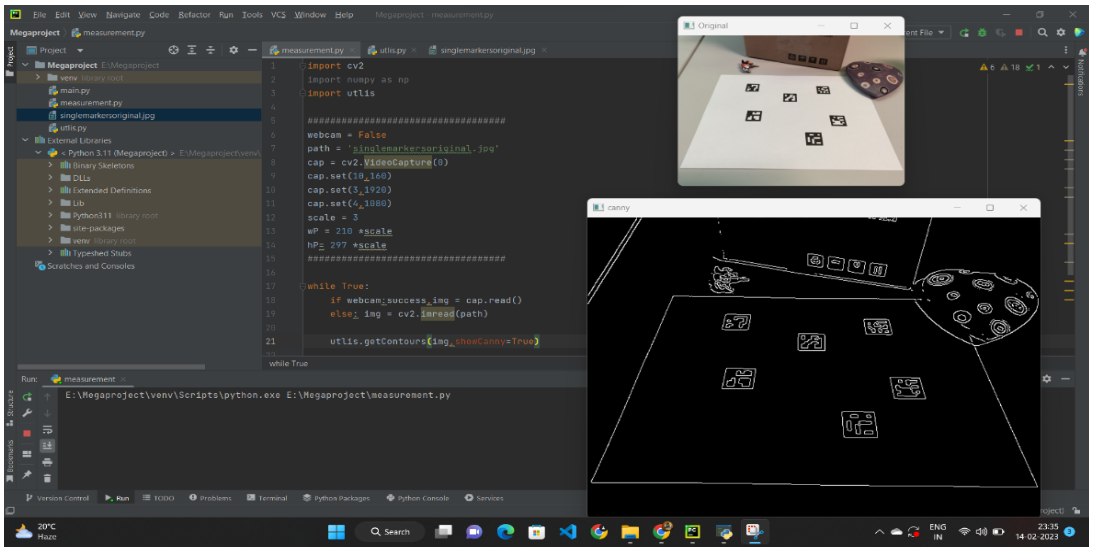
```
cap.set(10,160)
cap.set(3,1920)
cap.set(4,1080)
scale = 3
wP = 210 *scale
hP= 297 *scale
###################################

while True:
    if webcam
    else: img

    img, fina

    img = cv2
    cv2.imsho
    cv2.waitK
```

```
E:\Megaproject\venv\Scripts\python.exe E:\Megaproject\me
```

Fig 6.2.1

2.Integration Testing :

Testing the interactions between the various system components is known as integration testing. This comprises verifying the precision of camera calibration, the durability of the marker detection system, and the precision of the object recognition algorithm for visual inspection using ArUco markers. Actual photographs or videos of various lighting situations, camera angles, and object types can be used for integration testing.



```
import cv2
import numpy as np
import utlis

####################################
webcam = False
path = 'singlemarkersoriginal.jpg'
cap = cv2.VideoCapture(0)
cap.set(10,160)
cap.set(3,1920)
cap.set(4,1080)
scale = 3
wP = 210 *scale
hP= 297 *scale
####################################

while True:
    if webcam:success,img = cap.read()
    else: img = cv2.imread(path)

    utlis.getContours(img,showCanny=True)
```

```
E:\Megaproject\venv\Scripts\python.exe E:\Megaproject\measurement.py
```

Fig 6.2.2

3.Performance Testing :

First of all we predicted the center point of image which can lead us to find out the contours as well as the measurements of object.

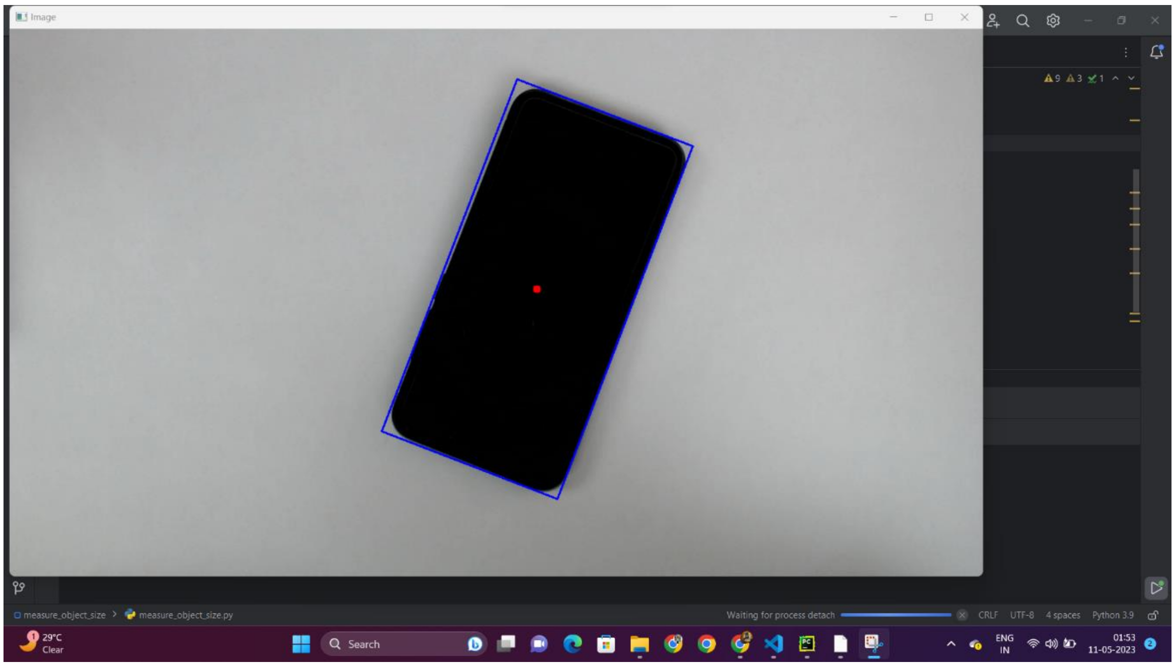
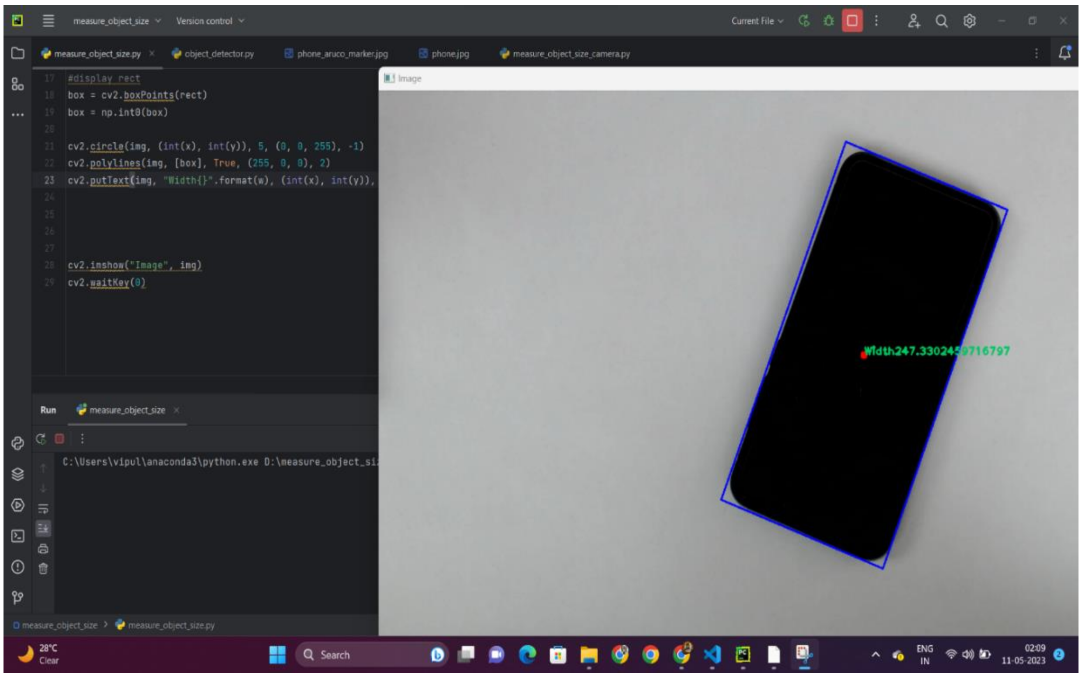


Fig 6.2.3



```
#display rect
box = cv2.boxPoints(rect)
box = np.int0(box)

cv2.circle(img, (int(x), int(y)), 5, (0, 0, 255), -1)
cv2.polylines(img, [box], True, (255, 0, 0), 2)
cv2.putText(img, "Width{}".format(w), (int(x), int(y)),



cv2.imshow("Image", img)
cv2.waitKey(0)
```

```
C:\Users\vipul\anaconda3\python.exe D:\measure_object_si
```

Fig 6.2.4

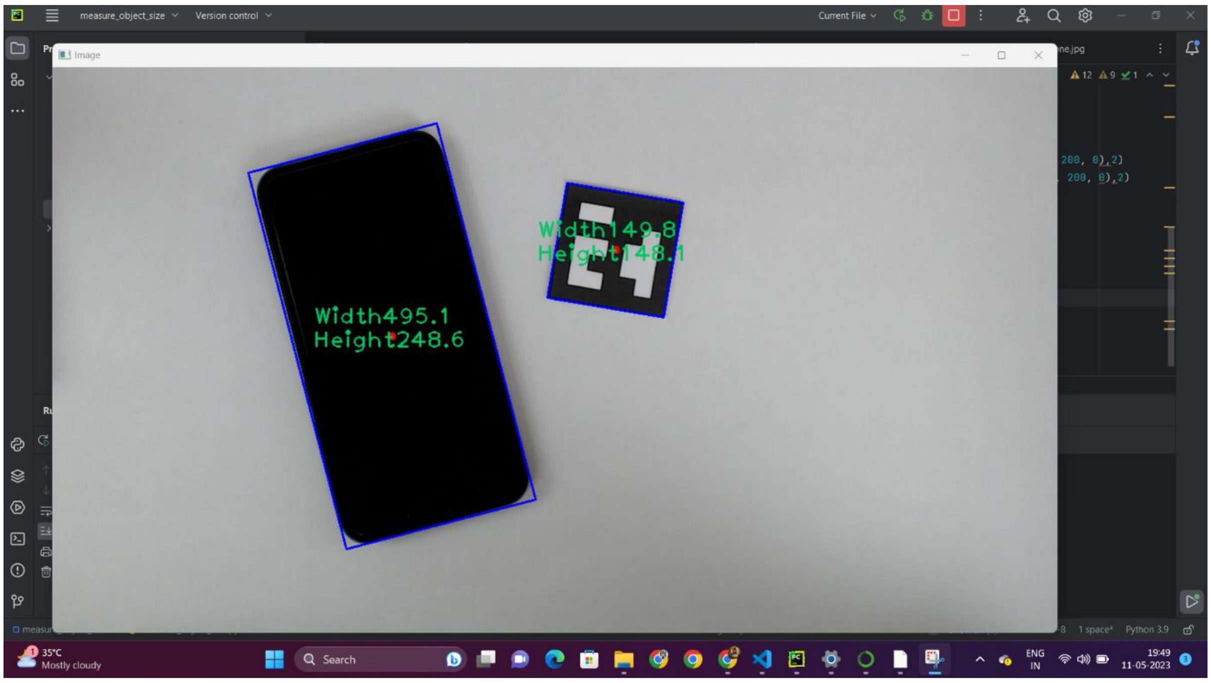
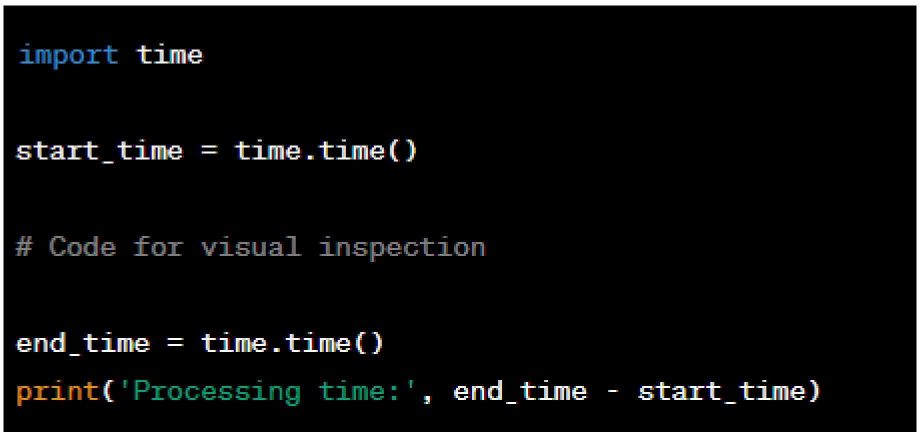


Fig 6.2.5

7.Performance Analysis :

Performance analysis of visual object inspection using OpenCV can be done by measuring several metrics, such as processing time, accuracy, and memory utilisation.

Processing Time: Processing time is the time it takes for a visual inspection code to study an image and deliver the intended outcome. The processing time of the code can be determined using Python's time package.



```
import time

start_time = time.time()

# Code for visual inspection

end_time = time.time()
print('Processing time:', end_time - start_time)
```

Accuracy : Accuracy is defined as the ability of the visual inspection code to correctly identify and classify objects in a picture. To assess correctness, the results of the visual inspection code can be put up against the actual data. For this, metrics like recall, accuracy, and F1-score can be applied.

Memory consumption: Memory usage is the amount of memory that the code uses for visual inspection during the examination. Memory utilisation can be calculated using the psutil library in Python.

8.Future Scope :

Visual object inspection using OpenCV is a current research and development topic with a number of potential applications.

Deep Learning: The bulk of OpenCV-based visual inspection techniques now in use rely on traditional computer vision algorithms. However, as deep learning techniques gain popularity, they may be used to develop increasingly complex object recognition and classification models.

Inspection in real-time : Real-time visual inspection presents a number of issues, one of which is conducting analysis. The evolution of hardware and software has made it possible to use OpenCV to build real-time inspection systems.

Automation : The process of automating the visual inspection is another area of growth. Systems that can detect flaws and recommend fixes or replacements or that can automatically alter the parameters based on the input image may need to be created in order to accomplish this.

Industrial Application : Visual inspection has a variety of industrial applications, such as quality control, defect detection, and product assembly. Specialised systems with features like 3D object recognition, high-speed analysis, and multi-camera systems could be developed and used in a variety of applications.

Robotics integration: Using OpenCV, visual inspection may be integrated into automated assembly and inspection applications.

9.Applications :

Quality Control : Visual inspection with OpenCV is used in quality control to find product flaws such as dents, scratches, and cracks. This can be accomplished by examining product photos and contrasting them with a reference image, or by utilising machine learning algorithms to find flaws in the products.

Automated Assembly: Automated assembly is possible by combining OpenCV-based visual inspection with robotics. Before assembly, component location and alignment must be verified using photographs in order to do this.

Object Tracking : Visual inspection with OpenCV can be used for object tracking, which is useful in robotics and surveillance systems, among other applications. This involves locating and tracking moving objects in a video feed and can be used for security or self-driving vehicle applications.

Medical Imaging: Using OpenCV, applications for visual inspection include X-ray analysis and cancer detection. This entails examining images for anomalies and can be utilised for diagnosis and treatment planning.

Agriculture : Visual inspection using OpenCV can be used for tasks like fruit grading and keeping track of the health of plants. Machine learning algorithms are used to examine crop photography in order to identify disease indicators or assess the quality of the food.

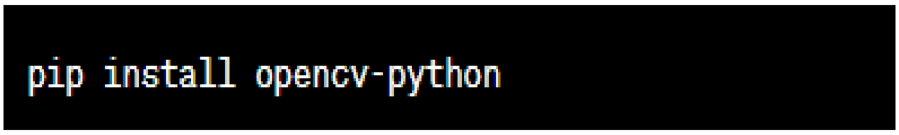
Robotics: Visual inspection using OpenCV can be used for autonomous navigation, object localization, and 3D mapping. Machine learning techniques are applied to photo analysis to identify objects and their position in space.

Overall, OpenCV-based visual inspection is a topic of continuing research and development and has many applications in a range of industries. Thanks to advancements in hardware and software, it is possible to design visual inspection systems that are more advanced, more accurate, and analyse data more quickly.

10.Installation Guide and User Manual :

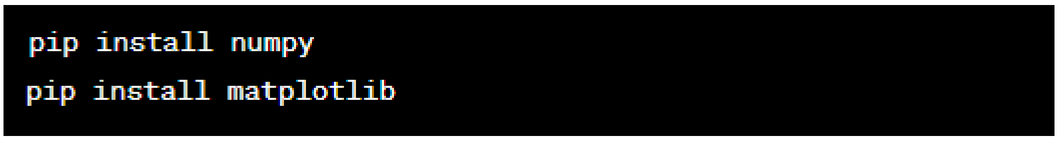
Install Python : Python needs to be installed on your machine before you can use OpenCV to perform visual inspection code. Python can be downloaded and installed from the official website.

Install OpenCV : Pip, a Python package installer, can be used to set up OpenCV. To install OpenCV, launch the command prompt and type the following command:



```
pip install opencv-python
```

Install additional prerequisites (Dependencies) : Numpy and Matplotlib may be additional dependencies needed by the visual inspection code. Use pip to install these dependencies.



```
pip install numpy
pip install matplotlib
```

Get the code here : Save the visual inspection code to your PC after downloading it from the source.

User Manual :

Open the code: Open the visual inspection code in a Python IDE or text editor.

Import the necessary libraries : Import the necessary libraries such as OpenCV, numpy, and matplotlib at the beginning of the code.

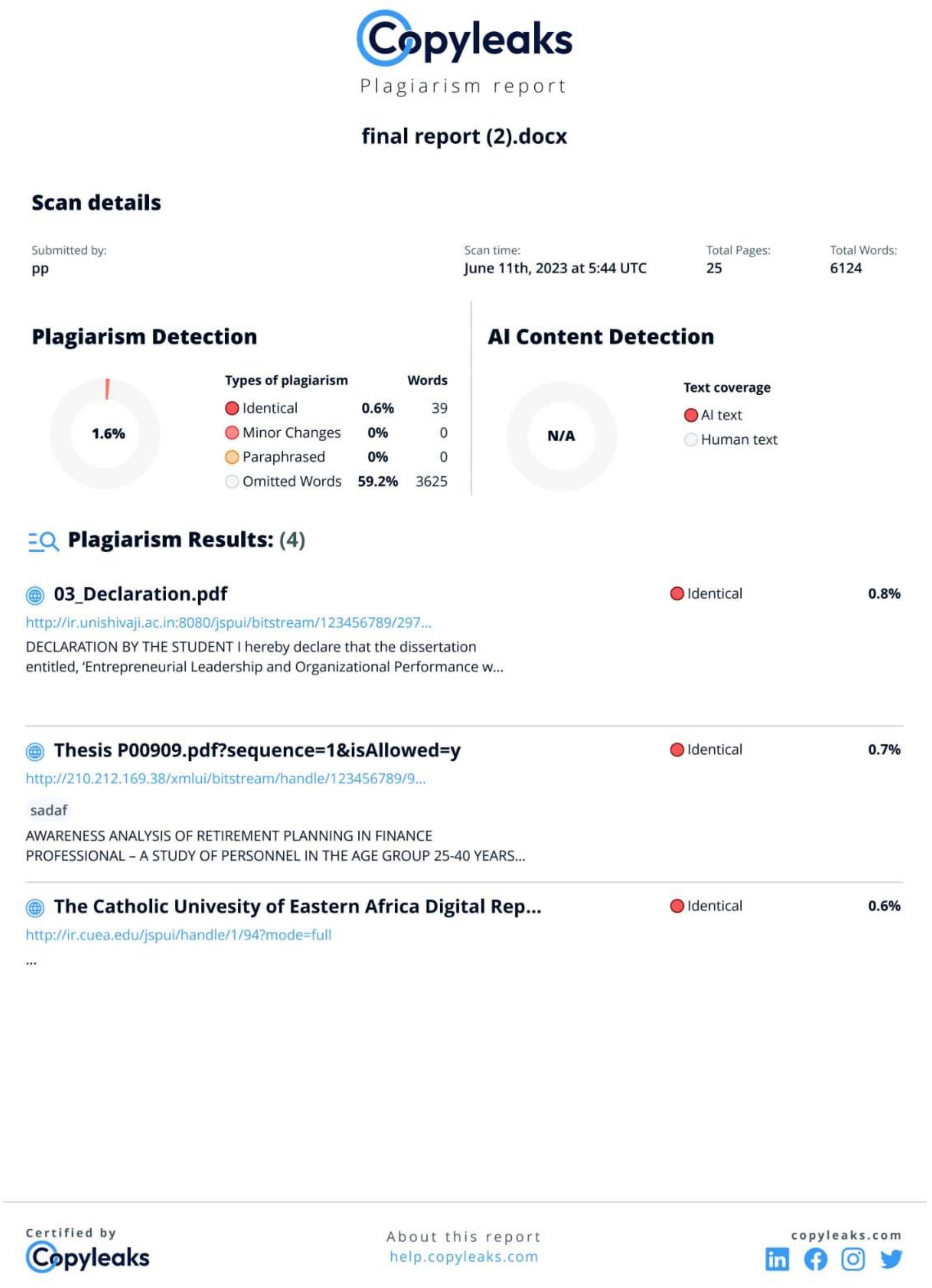
Load the image: Load the image that you want to inspect using OpenCV's imread() function.

Perform visual inspection : Apply visual inspection techniques to the loaded image. This could include object detection, image segmentation, or feature extraction.

Display the results: Display the results of the visual inspection using OpenCV's imshow() function or matplotlib's plot() function.

Overall, visual inspection of objects using OpenCV requires some basic knowledge of Python and image processing techniques. By following the above installation guide and user manual, users can easily perform visual inspection on their own images.

11.Plagiarism Report :



Copyleaks
Plagiarism report

final report (2).docx

Scan details

Submitted by:	Scan time:	Total Pages:	Total Words:
pp	June 11th, 2023 at 5:44 UTC	25	6124

Plagiarism Detection

1.6%

Types of plagiarism		Words
● Identical	0.6%	39
● Minor Changes	0%	0
● Paraphrased	0%	0
○ Omitted Words	59.2%	3625

AI Content Detection

N/A

Text coverage
● AI text
○ Human text

Plagiarism Results: (4)

03_Declaration.pdf ● Identical **0.8%**
http://ir.unishivaji.ac.in:8080/jspui/bitstream/123456789/297...
DECLARATION BY THE STUDENT I hereby declare that the dissertation entitled, 'Entrepreneurial Leadership and Organizational Performance w...

Thesis P00909.pdf?sequence=1&isAllowed=y ● Identical **0.7%**
http://210.212.169.38/xmlui/bitstream/handle/123456789/9...
sadaf
AWARENESS ANALYSIS OF RETIREMENT PLANNING IN FINANCE PROFESSIONAL – A STUDY OF PERSONNEL IN THE AGE GROUP 25-40 YEARS...

The Catholic Univesity of Eastern Africa Digital Rep... ● Identical **0.6%**
http://ir.cuea.edu/jspui/handle/1/94?mode=full
...

Certified by Copyleaks
About this report
help.copyleaks.com
copyleaks.com

Fig 11.1

12.References :

Muthukrishnan.R and M.Radha “Edge Detection Techniques for image Segmentation” International Journal of Computer Science & Information Technology (IJCSIT) Vol3, No 6, Dec 2011.

Mane, Sanjana Sanjay; Yangandul, Chaitra Gajanan (2016). [IEEE 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT) - Bangalore, India (2016.7.21-2016.7.23)] 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT) - Calculating the dimensions of an object using a single camera by learning the environment.

Wenhao Wang, “Attentive WaveBlock: Complementarity-enhanced Mutual Networks for Unsupervised Domain Adaptation in Person Re-identification and Beyond”, Published in Journal of Latex Class Files, August 2015, pp.1-14

You Only Look Once: Unified, Real-Time Object Detection, Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

Dellen, Babette, and Iván Andrés Rojas Jofre, " measurement of object with a consumer depth camera based on structured infrared light." In Proceedings of the 16th Catalan Conference on Artificial Intelligence, poster session, pp. 1-10. 2013