Estimation of Object Dimension using Image Processing

Project Team

Sl. No.	Reg. No.	Student Name
1.	16ETCS002124	Shridhar Hegde
2.	16ETCS002122	ShivaKumar M
3.	16ETCS002112	Santosh G
4.	16ETCS002116	Sai Bharadwaj

Supervisors: Mr. Prakash P
Ms. Pallavi R Kumar

May – 2019

B. Tech. in Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY
M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES
Bengaluru -560 054

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

*This is to certify that the Project titled "Estimation of Object Dimension using Image Processing" is a bonafide work carried out in the Department of Computer Science and Engineering by **Mr. Shridhar Nagesh Hegde** bearing Reg. No. **16ETCS002124** in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

May – 2019

Mr. Prakash P	**Ms. Pallavi R Kumar**
Assistant Professor	**Assistant Professor**

Supervisors

Dr. P.V.R. Murthy	**Dr. M. Arulanantham**
Professor and Head – Dept. of CSE	**Professor and Dean-FET**

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

*This is to certify that the Project titled "Estimation of Object Dimension using Image Processing" is a bonafide work carried out in the Department of Computer Science and Engineering by **Mr. Shivakumar M** bearing Reg. No. **16ETCS002122** in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

May – 2019

Mr. Prakash P	**Ms. Pallavi R Kumar**
Assistant Professor	**Assistant Professor**

Supervisors

Dr. P.V.R. Murthy	**Dr. M. Arulanantham**
Professor and Head – Dept. of CSE	**Professor and Dean-FET**

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

This is to certify that the Project titled "Estimation of Object Dimension using Image Processing" is a bonafide work carried out in the Department of Computer Science and Engineering by ***Mr. Santosh G*** *bearing Reg. No.* ***16ETCS002112*** *in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

May – 2019

Mr. Prakash P
Assistant Professor

Ms. Pallavi R Kumar
Assistant Professor

Supervisors

Dr. P.V.R. Murthy
Professor and Head – Dept. of CSE

Dr. M. Arulanantham
Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

This is to certify that the Project titled "Estimation of Object Dimension using Image Processing" is a bonafide work carried out in the Department of Computer Science and Engineering by ***Mr. Sai Bharadwaj*** *bearing Reg. No.* ***16ETCS002116*** *in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

May – 2019

Mr. Prakash P
Assistant Professor

Ms. Pallavi R Kumar
Assistant Professor

Supervisors

Dr. P.V.R. Murthy
Professor and Head – Dept. of CSE

Dr. M. Arulanantham
Professor and Dean-FET

Declaration

Estimation of Object Dimension using Image Processing

The project work is submitted in partial fulfilment of academic requirements for the award of B. Tech. Degree in the Department of Computer Science and Engineering of the Faculty of Engineering and Technology of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

Sl. No.	Reg. No.	Student Name	Signature
1.	16ETCS002124	Shridhar Hegde	
2.	16ETCS002122	ShivaKumar M	
3.	16ETCS002112	Santosh G	
4.	16ETCS002116	Sai Bharadwaj	

Date: 22 May 2019

Acknowledgements

The successful completion of this project could not have been possible without the guidance of **Mr. Prakash P** and **Ms. Pallavi R Kumar** our mentors, whose guidance was priceless.

We extend our sincere gratitude to everyone who guided us, cleared our doubts and supported us throughout the project either directly and indirectly.

We humbly thank **Dr. P.V.R. Murthy** and **Dr. M. Arulanantham** for their constant, unwavering support.

We thank our family and friends who have supported us in carrying out this project and gave us a moral support.

Summary

Engineers and Scientists constantly build physical scientific models & need to ascertain its dimensions. It is cumbersome to carry around measuring instruments. This problem was the motivating factor for this project. Applying the knowledge of Image Processing can solve this problem if one can obtain the pictures of the objects customarily using cellphone camera. So, we intend to build a system that takes the images of objects as input and gives their dimensions as output.

This project when implemented with a higher accuracy has a huge scope in the science and academia. Engineers and Scientists can use it to measure their physical models, colleges & universities can use it to grade students on the basis of their models etc. Python programming language is used to implement the project and “OpenCV” is the library is used to pre-process the image. Threshold and Canny methods are used to identify the edges of the object. Then, Reference Object Method & “imutils” library is used to find the dimensions of the object.

The outcome of the project is satisfactory. It has overall a good accuracy in measuring objects and can be used in academia. The system is not yet so sophisticated that it can be used in science as experiments need readings which are very close to the real values. The project was successfully able to cover all the objectives that was devised in the beginning of the project. The team aims to take up the project for future works and make it more accurate.

Table of Contents

List of Tables

List of Figures

Abbreviation and Acronyms

DIP	Digital Image Processing
FR	Functional Requirements
mm	Milli meter
NFR	Non-Functional Requirements
PPM	Pixel Per Metric

1. Introduction

Science is not a 21st century thing but can be dated back to 3500 to 3000 BCE (Lindberg & David 2007). Scientists and Engineers have adopted various methods to research from a very long time and building physical models is one of them. Measuring the dimensions of these models is a crucial task and has to be done with utter care. Colleges, Universities and other educational institutions also make students build models and often need to grade them based on their models by measuring them.

Accurate measurements are necessary only in certain fields or domains. For example, a carpenter who is building thigs using wood needs a precision of say 0.5 mm. If the accuracy is reduced significantly, the furniture built would be terrible. But 0.5 mm precision is totally insignificant for more sophisticated domains like building a car engine or other parts. Now the precision needed boils down to 0.01 mm or less.

So, a lot of experimental physicists are focused on measuring things more accurately, discovering new things and confirm theoretical ideas. This important task of measuring has led to the development of sophisticated measuring devices. Tons of Nobel prizes were awarded to essentially better measurements, e.g., Rudolf Mössbauer, Albert Abraham Michelson, Polykarp Kusch etc.

1.1 Motivation

Often, teachers and professors have to measure models of hundreds of students and grade them accordingly which is both tiring and prone to errors. There is a huge need of using technology to automate this task. The reason is that to grade students in a university, the precision required resembles precision for woodworking mentioned earlier and the accuracy obtained by automating the task also resembles the precision needed for the task. This gives a major reason to go ahead with the idea of developing such tool. If such a tool is developed it will not only help the academia industry but other common and day to day tasks that people carry out. The developed tool would aid people to measure common objects like a piece of wood or some metal whenever needed without the help of any kind of instruments.

1.2 Introduction to the problem

1.2.1 Pretext

The standard measuring devices and tools may be sophisticated but there are errors while measuring due to various reasons varying from instrumental errors to human errors. Also, carrying these instruments around is cumbersome. Even if the device is somehow disassembled and carried, some of these devices are complex in structure and takes good amount of effort and expertise to set up.

This amount of effort can be put up but only for situation where high precision and accurate measurements are needed. There are domains where not so accurate measurements are not needed like woodworking as mentioned earlier. There are many other such domains where not so precise measurements are not required yet there is a need to buy these fancy measuring devices and carry them around.

1.2.2 Scope of the project

The project has a very good scope in the field of academia which was the major initial motivation. Schools, Colleges and Universities can opt to use this tool. A customised web or mobile application can be designed using the current project as an interface if needed. The teachers and professors can use that tool to analyse the models of the students and get its dimensions with just a digital image(picture) of the models.

The developed tool is not only restricted to schools and colleges. Any domain where objects have to be measured can use the tool. The only criteria for using it would be the amount of accuracy that the domain expects from the tool. If the accuracy of the tool is nearly same as the precision required for measuring objects in the particular domain, then it can be readily used in those domains. Some common examples are warehouses where the dimension of the package or object have to be measured, construction industry where the height of the structure has to be measured etc.

The usage and scope of the project increases with the increasing accuracy that the developed tool provides. There is a huge need for such tool in the world and with high accuracy the tool can even be used in science for experimenting and measuring scientific models.

1.2.3 Development of solution

The goal here is to find the dimension of the object using its image. To do this there has to be a system that manipulates takes the image as input, uses mathematical operations and does computations to give the dimensions of the object in the image as output. In order to do to this, there has to be a technique using which we can extract information from images and use the data for further computations. This can be done using Image Processing. Specifically, Digital Image Processing deals with such problems.

The field of digital image processing has become widespread and is used extensively as possible. Applying this knowledge to solve the problem at hand can solve it one and for all. Open Source programming languages like Python have lots of image processing and manipulation libraries like “openCV”, “imutils”, “simpleCV”, “scikit-image”, “pillow” etc.

There are various research papers which gives mathematical insights that are in line with the objectives of the current project and using all these resources the project is built.

1.3 Organization of the report

The report is organized into different chapters where each chapter explaining one part of the project. When the reader finishes the whole report, they will have a clear and intuitive understanding of what the project is, what are the previous research works around it, how it is designed, the implementation, how the system is test and analysed and suggestions for the future work.

Each of the chapters along with its short summary is given in table 1.1.

Table 1.1 Summary of all the chapters in the report

Sl. No	Chapter Title	Summary
1.	Introduction	The first chapter introduces the problem at hand to the readers. Motivation for the project, pretext, scope of the project and short explanation about development of the solution is given. At the end, this chapter contains the organization of the report which had a short summary of every chapter in the report.
2.	Digital Image Processing	DIP is the main concept using to implement the solution that is proposed to the problem at hand. This chapter contains various programming languages and tools that can be used to implement DIP, various libraries in the chosen programming language (Python), scholarly documents which aids the project along with citation.
3.	Aim and Objectives	The title, aim and objectives of the current project is explained here. The Methods and Methodologies that

		are adopted to satisfy the objectives are also explained in this chapter.
4.	Problem solving	This is the core part of the report. In this chapter, the design of solution is drafted, the explanation to the actual implementation is explained along with testing the end product. This chapter contains the code that is developed to solve the problem.
5.	Results	In this chapter, the results obtained after implementing the project and testing is explained along with suitable explanations. Inference is drawn as to where the system can be used with the amount of accuracy obtained in the result.
6.	Project costing	The total amount of investment that is done on the project during the course of development is given here.
7.	Conclusions and Suggestions for future work	Final thoughts on the project from the team and suggestions for future work on the same project on how to improve the results is mentioned here.

2.Digital Image Processing

Digital Image Processing has become widespread in the recent times due to the advent of the sophisticated smartphone cameras. Large number of people use Image Processing in their daily life either directly or indirectly when they take selfie and apply filters to it, crop a picture or increase its brightness, contrast or photoshop themselves with their favourite celebrities!

In computer science, Digital Image Processing is the use of computer algorithms to perform image processing on digital images (Chakravorty 2018). An image is a two-dimensional signal defined by the mathematical function f (x, y) where x and y are the two co-ordinates horizontally and vertically. The image which is a mathematical function is manipulated using mathematical operations so as to change its structure or derive some useful information from it.

2.1 Background Theory/Previous works

2.1.1 Depth and Geometry from a Single 2D Image Using Triangulation (Saliah & Malik, 2012)

In this paper, the authors present a novel method for computing depth of field and geometry from a single 2D image. The presented technique, unlike the ones that existed at the time of publishing the paper, measures the absolute depth of field and distances in the scene from single image only using the concept of triangulation. The algorithm devised by the authors requires minimum inputs such as camera height, camera pitch angle and camera field of view for computing the depth of field and 3D coordinates of any given point in the image. In addition, the presented method can be used to compute the actual size of an object in the scene (width and height) as well as the distance between different objects in the image.

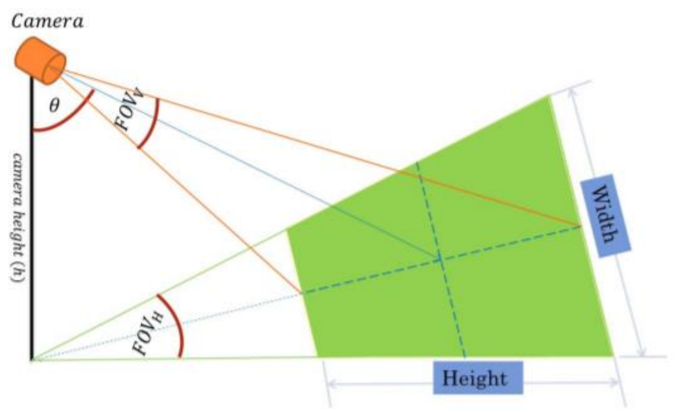


Figure 2.1 Typical camera setup for surveillance cameras (Source: https://urlzs.com/6Z1g)

As shown in figure 2.1, the paper takes into account how the cameras are set up and uses basic principles of physics pertaining to triangulation to compute the size of the objects.

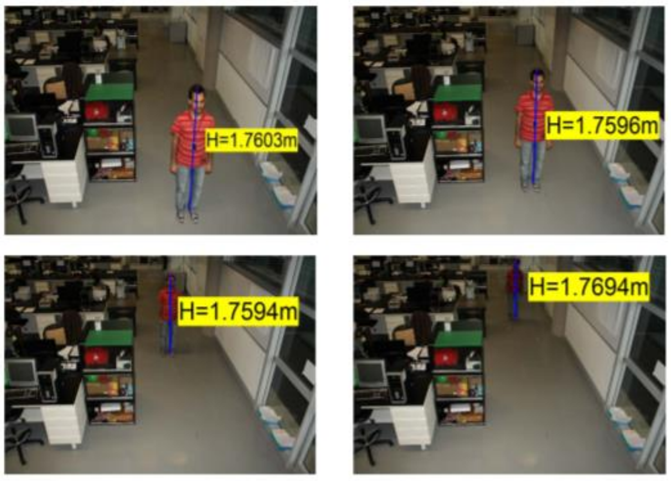


Figure 2.2 Capturing the image of the target (Source: https://urlzs.com/p4gP)

As shown in figure 2.2, the devised algorithm can be used to measure targets such as a human being itself and gives results with good precision. The proposed methodology, according to the authors, can be implemented in high impact applications such as distance measurement from mobile phones, robot navigation and aerial surveillances.

2.1.2 **How to measure the size of an object using a camera** (Johnson 2010)

This article is published in a blog written by Forrest Johnson. The author wanted to do a fun project of measuring some object but was reluctant to climb up and measure it. Thus, the author undertook this project.

The author took a picture of something that is measurable and then made a version of that situation in Unity. The camera is placed at a set distance away from the doorway and then the image is taken. Then the dimension of the doorway and distance to the camera is measured which is illustrated in Figure 2.1.

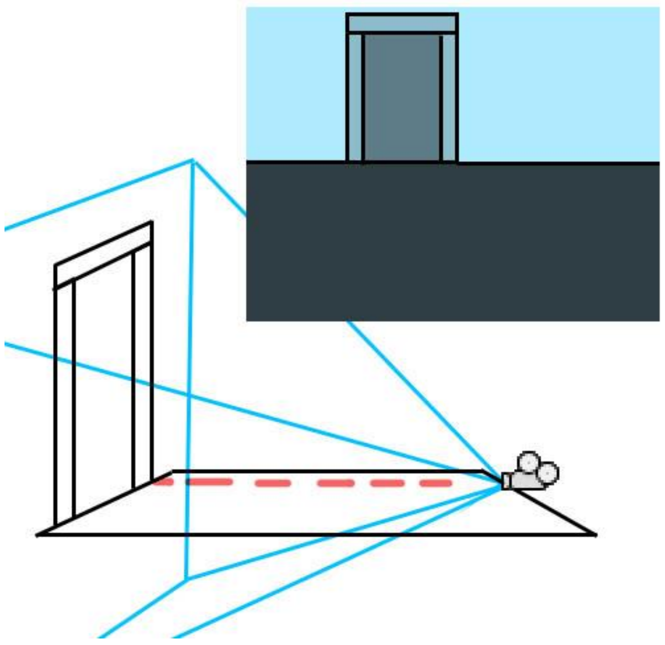


Figure 2.3 Capturing the image of the target (Source: https://urlzs.com/p4qP)

The same process is done with another object, say a cube or another doorway. Then a similar step is carried out but in backwards. The object is photographed, the distance to the object measured, and then the scene is reconstructed inside unity with the photo as a background and a cube representing the object of interest.

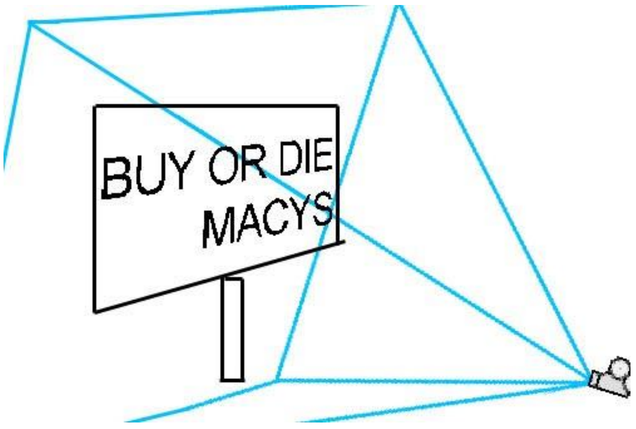


Figure 2.4 Reconstructing the scene inside unity (Source: https://urlzs.com/Pq5p)

Then, the scale of the cube is adjusted until it matches the photo, and the resulting scale is the measurement.

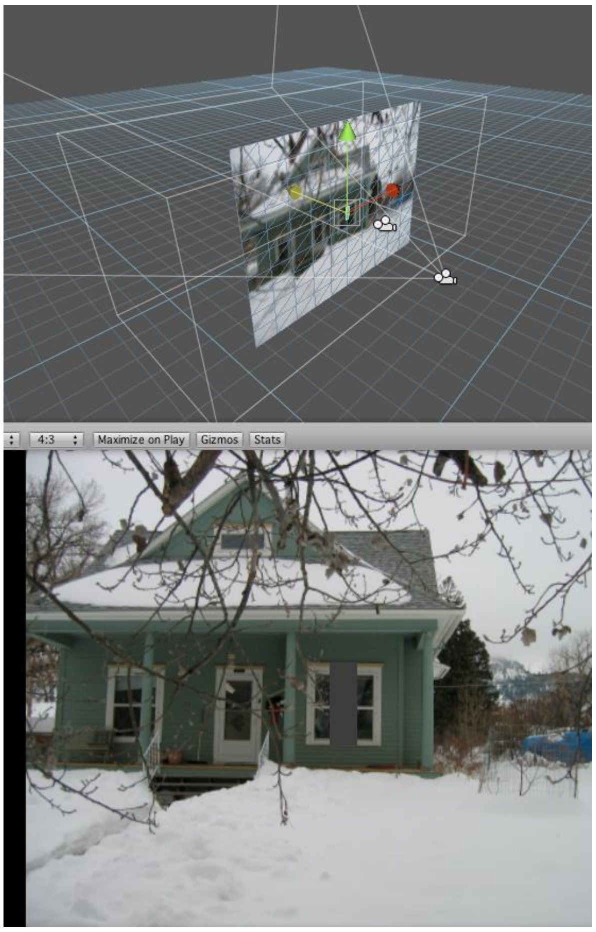


Figure 2.5 Measuring the dimension of the house (Source: https://urlzs.com/qUuL)

That window that the author measured is 77 inches tall and the camera method measured ~79 inches. That is 2.5% inaccuracy. The blog gives some more implementation ideas which is beyond the scope of this report and thus not mentioned here.

2.1.3 A fast and accurate approach for computing the dimensions of boxes from single perspective images (Fernandes et al., 2006)

This paper describes an accurate method for computing the dimensions of boxes directly from perspective projection images acquired by cameras. The approach is based on projective geometry and computes the box dimensions using data extracted from the box silhouette and from the projection of two parallel laser beams on one of the imaged faces of the box.

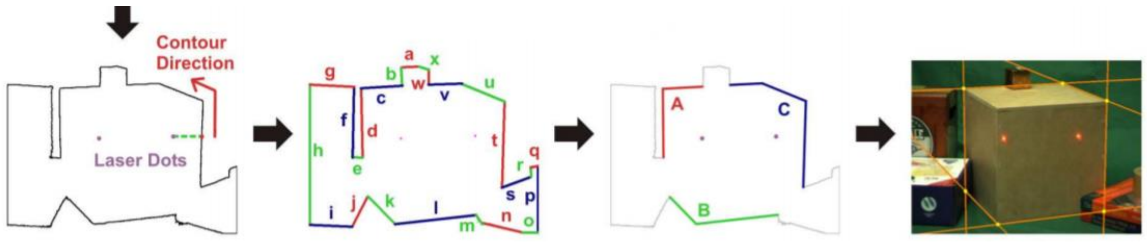


Figure 2.6 Illustration of proposed method using lasers and contour segmentation (Source: http://www.scielo.br/pdf/jbcos/v12n2/03.pdf)

In order to identify the box silhouette, the authors have developed a statistical model for homogeneous-background-colour removal that works with a moving camera, and an efficient voting scheme for the Hough transform that allows the identification of almost collinear groups of pixels.

The paper demonstrates the effectiveness of the proposed approach by automatically computing the dimensions of real boxes using a scanner prototype that implements the algorithms and methods described in the paper.

2.1.4 **Edge detection using Image Processing** (Meysenburg 2016)

In the current article, he author has explained how to use OpenCV functions to apply edge detection to an image. In edge detection, the boundaries or edges of objects in an image are found by determining where the brightness of the image changes dramatically. Edge detection can be used to extract the structure of objects in an image. If interested, one can also get information about the number, size, shape, or relative location of objects in an image. Edge detection all enables focusing on the parts of the image most helpful, while ignoring parts of the image that will not help in the process.

For example, once the edges of the objects in the image is found (or once we have converted the image to binary using thresholding), that information can be used to find the image contours. This procedure is explained in the article by the author. With the contours, one can do things like counting the number of objects in the image, measure the size of the objects, classify the shapes of the objects, and so on.

2.1.5 **Single-image shadow detection and removal using paired regions** (Guo et al., 2011)

In the current paper, the authors address the problem of shadow detection and removal from single images of natural scenes. The methods used in the paper are different from traditional methods that explore pixel or edge information. This paper employs a region-based approach. In addition to considering individual regions separately, the authors predict relative illumination conditions between segmented regions from their appearances and perform pairwise classification based on such information.

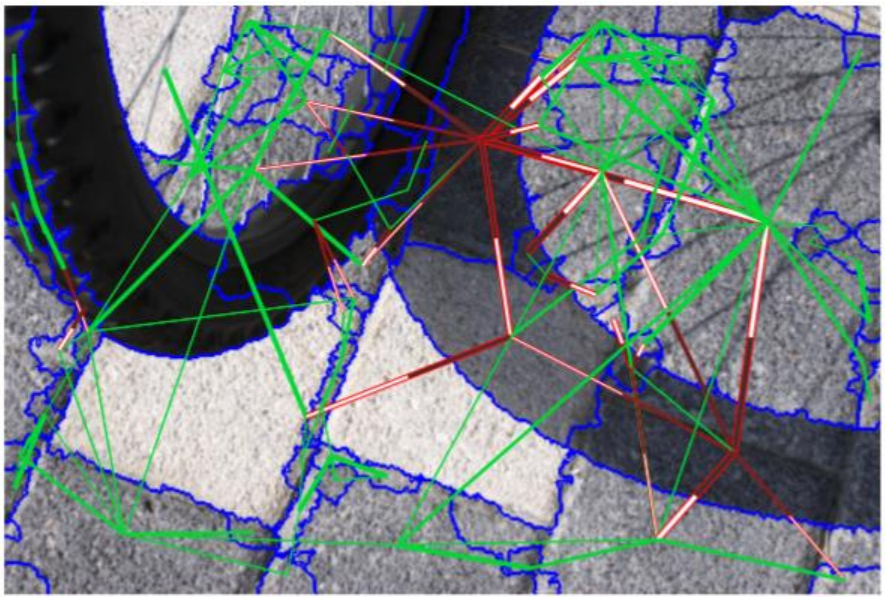


Figure 2.7 Illumination relation graph (Source: http://dhoiem.web.engr.illinois.edu/publications/cvpr11_shadow.pdf)

Classification results are used to build a graph of segments, and graph-cut is used to solve the labelling of shadow and non-shadow regions. Detection results are later refined by image matting, and the shadow free image is recovered by relighting each pixel based on our lighting model. We evaluate our method on the shadow detection dataset given by Jiejie et al., 2010.

2.2 Inference drawn from previous works

Conducting literature survey about the previous research and projects that focused on similar theories gives immense insight on the project that is being carried out and gives an opportunity to introspect the requirements. Such surveys many times makes the researcher realise that similar projects that can satisfy the requirements already exist and can be used freely to solve the problem.

That is not the case for the current project. After thorough research, it was found that there are several projects exists which can vaguely achieve what the current project is about. But coming to the specific details, those solutions cannot solve the problem that the current project aims to solve. The project was motivated by the problem faced by scientists, engineers and universities to measure the physical models.

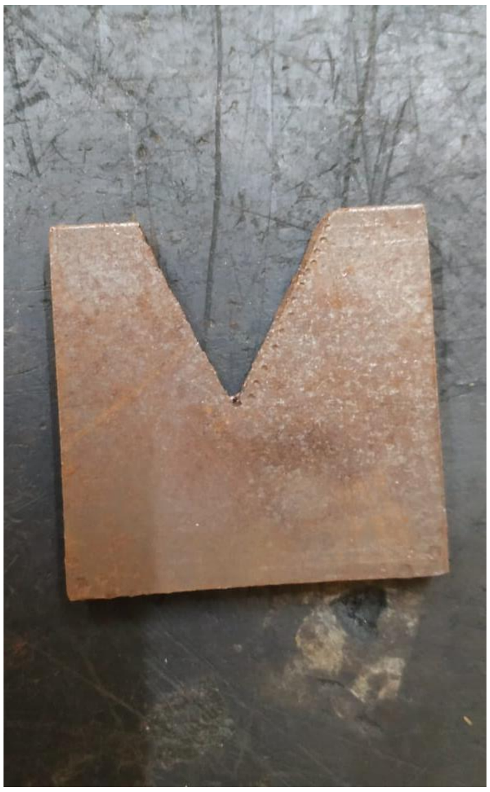


Figure 2.8 Typical model developed at RUAS Mechanical Laboratory

The figure 2.8 shows a typical model developed in a laboratory by students. Universities need to measure each edge and then grade all the students based upon the reading. The task becomes very hectic for a large number of students and the current project comes into picture. Previous solutions that were found in research can only identify and measure homogenous shapes i.e. objects with only one shape. Since the object in figure 2.8 contains two shapes, the current project needs to be developed such that it can measure such objects.

2.3 Possible merits and demerits of the developed system

2.3.1 Merits

The project if implemented would have huge application in science and academia. It would aid researchers to measure their models. Schools, colleges and universities can use the developed system to measure and grade models developed by students. The project would become the first of its kind to find a place in academia. It can be used by common people also for day to day applications such as measuring the dimension of a window in a building, finding dimension of common objects like wood etc.

2.3.2 Demerits

The project needs to be developed with utter care. If the project has errors that go unnoticed, it would have a very bad impact on the academia and science if used in those areas. The system should have a very good accuracy in measuring the objects and the obtained results must be precise. This is a major concern while developing the project.

3. Aim and Objectives

This chapter contains the title and aim of the current project. Then the objectives that were devised at the beginning of the project are listed. Then the methods and methodologies adopted to satisfy the objectives.

- **Title**
    - ❖ Development of a system to estimate the object dimension using Image Processing
- **Aim**
    - ❖ To develop a system to estimate the dimensions of a given object using Image Processing techniques
- **Objectives**
    - ❖ To conduct literature survey on image processing, depth and geometry prediction using triangulation
    - ❖ To analyze the literature survey and derive the requirements for building system to find the dimensions of an object in an image
    - ❖ To design a mathematical model for depth analysis and to measure dimensions of an object
    - ❖ To implement the designed mathematical model using appropriately chosen programming languages and tools
    - ❖ To test and validate the system built for different cases considered and incorporate necessary changes
    - ❖ To document the report by unifying all the results and outcomes

- **Methods and Methodology/Approach to attain each objective**

Table 3.1 Methods and Methodologies adopted to satisfy the objectives of the project

Objective No.	Statement of the Objective	Method/ Methodology	Resources Utilised
1	To conduct literature survey on image processing, depth and geometry prediction using triangulation	To conduct literature survey on - Parallax method to compute the dimensions of object when the viewpoint changes - Reference object method, scale of a reference object to compute the dimensions of the object - Image pre-processing and processing, Object Segmentation in an image	
2	To analyze the literature survey and derive the requirements for building system to find the dimensions of an object in an image	- Compare different methods found in the literature survey to find the dimensions of the object in an image - Based on the analysis, find the functional and non-functional requirements - Check feasibility for the identified requirements	

3	To design a mathematical model for depth analysis and to measure dimensions of an object	- A high-level block diagram is designed to identify primary functionalities of the system - A mathematical model to compute the dimensions from a theoretical aspect is built - A low-level block diagram is then designed to represent the functions that inputs an image, processes the image, separates the object from background and outputs the dimensions of the object	
4	To implement the designed mathematical model using appropriately chosen programming languages and tools	- Identify the appropriate programming language and tools (mostly free and/or open source) to use based on the design and requirements - Develop system that accepts an image as input, processes it, identifies various objects in it, compute and returns the	

		dimensions of the object	
5	To test and validate the system built for different cases considered and incorporate necessary changes	- The system built will be tested for different cases and find exceptions and anomalies - Unit test, integration and functional tests will be made to ensure the product works well in different layers/levels - Comparison is made between the actual dimensions of an object and the dimensions given by the system - The errors and the functionality test case failures (if any) are identified and changes are incorporated	
6	To document the report by unifying all the results and outcomes	- A report will be made illustrating the need for the project, introduction of the project and the related works in this domain so far - The various designs made, implementation,	

		testing and results are reported - Demonstrate the working of the product built with sample image as input to the system - The performance is compared with other existing models and efficiency, error rate is reported	

4. Problem Solving

This section of the report discusses the design and implementation of the project. Initially, the system overview, modelling of the system and assumptions are discussed. Later in this section; choice of programming languages, tools, testing and analysis of the implementation is discussed.

4.1 System Overview

The input for the system being built is a raw image with a clear background with a standard object towards the left of the image. Pre-processing of the image is then done which helps in identifying the objects in the image and computing the dimensions. A standard reference model towards the left of the image is significant here. Since the dimensions of the standard model is known, the size of the standard model is computed and other objects are scaled with the same ratio as the reference model. Finally, the dimensions of the other objects are made available to the user.

4.2 Functional and Non-Functional Requirements

The Functional requirements (FRs) and the Non-Functional Requirements (NFRs) are identified before the design phase of the system is identified.

Functional Requirements for the system are:

FR1: System should read the image given as input

FR2: System should identify the objects from the image read

FR3: System should identify all the edges for objects present in image

FR4: System should identify corners for each edge of each object in image

FR5: System should calculate dimensions of each edge of these objects, Euclidean distance in the image

FR6: System should calculate the approximate dimensions based on dimensions of reference object

FR7: System should produce an image as output showing the identified edges and dimensions of input image

Non-Functional Requirements for the system are:

NFR1: System shall produce the output image accurately and quickly

NFR2: System shall take input image of any size

NFR3: System shall be available in any platform to be executed

NFR4: System shall allow only one image as input with any number of objects in it

The FRs and NFRs are identified for the system and feasibility of the requirements are analysed. On careful analysis, it can be concluded that the requirements identified for the system is feasible and implementable.

4.3 Design

There are various dependencies in the FRs identified for the system. In order to build the system as per the requirements, a proper flow of the tasks needs to be designed.

A flowchart is designed representing the flow of the system shown in Figure 4.1. The flow here represents a high-level design which is just a simple design representing the system overview. It starts with providing an image to the system, which is pre-processed before proceeding with finding edges and the dimensions of those edges. The dimensions are then finally output to the user.

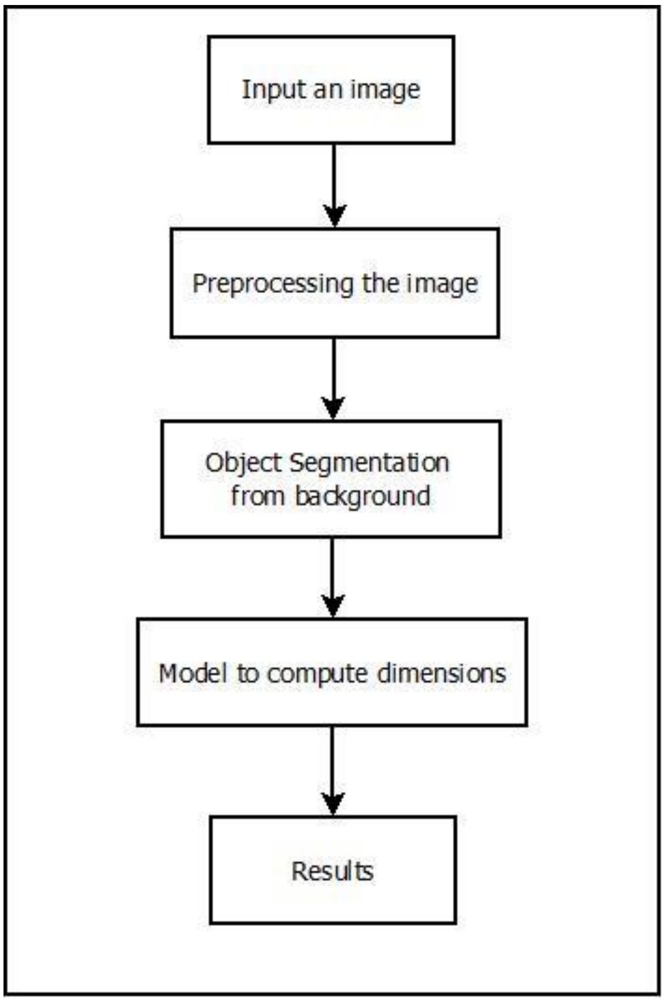

Input an image
Preprocessing the image
Object Segmentation from background
Model to compute dimensions
Results

Figure 4.1 High Level Block Diagram

While the high-level design represents an abstract overview of the system, it is less helpful to begin the implementation with this flowchart. The high level design shows pre-processing of the image. However, it does not tell anything about what needs to be done. Therefore, a low-level block diagram is designed which will aid in the implementation phase. The low-level design is shown in Figure 4.2.

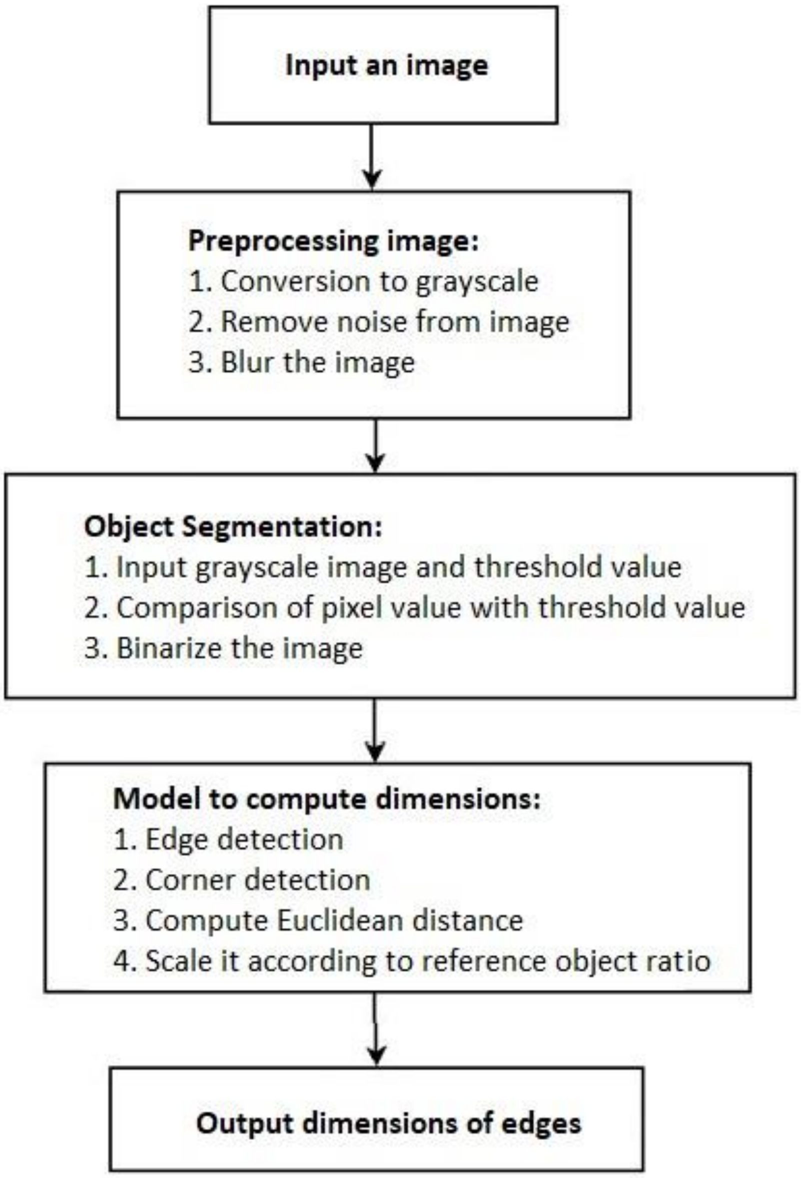

Input an image
Preprocessing image:
1. Conversion to grayscale
2. Remove noise from image
3. Blur the image
Object Segmentation:
1. Input grayscale image and threshold value
2. Comparison of pixel value with threshold value
3. Binarize the image
Model to compute dimensions:
1. Edge detection
2. Corner detection
3. Compute Euclidean distance
4. Scale it according to reference object ratio
Output dimensions of edges

Figure 4.2 Low Level Block Diagram

4.4 Choice of programming language and tools

A proper programming language and libraries/tools has to be chosen considering various parameters like project-cost, efficiency, simplicity, performance, et cetera.

Choice of programming Language:

- Three programming languages; Python, C++/C#, Matlab, come very close considering different parameters for image processing.
- Python is free-to-use with well-defined libraries like OpenCV, SimpleCV, scikit-image, et cetera. Matlab needs a license which would increase the project cost.
- Both the languages are comparable in terms of performance as both are based on dynamic programming which involves fast coding and slow processing. The cost factor puts Python on the top.
- Python code, generally, is slower than C++. But for the modules needed in this project, OpenCV, it is quite different. Python's OpenCV is just a wrapper around the original C/C++ OpenCV code. So when a function is called in OpenCV python, the underlying C/C++ code runs. So, the performance is the same. However, Python brings in simplicity and portability.

Hence, Python is chosen as the programming language to build the system.

Choice of libraries/tools:

- OpenCV is one of the most widely used libraries for computer vision applications.
- OpenCV-Python is not only fast, since the background consists of code written in C/C++, but it is also easy to code and deploy (due to the Python wrapper in the foreground). This makes it a great choice to perform computationally intensive computer vision programs.
- Alongside OpenCV, other libraries like Numpy, imutils and distance are used for other calculations in the system.

4.5 Image processing techniques

The image processing techniques used in the system are listed below:

4.5.1 Conversion to Grayscale

It is done since it is very efficient to apply thresholding on a grayscale image. Otherwise, the RGB values of a pixel has to be compared with a threshold tuple of RGB values. This is not an efficient way. Hence, it is converted to grayscale. Also, contouring works best with black and white images which is a result of thresholding. Conversion to grayscale is accomplished using the method from cv2 (OpenCV) library.

4.5.2 Blurring

The object whose dimension is to be found need not always have a clear surface. Especially in cases where the blocks provided in mechanical workshops, which are so often prone to rust and often comes with irregularities. Various methods are available to blur an image like simple blur, Gaussian blur, Median blur, et cetera.

4.5.3 Thresholding

The image input to this is a grayscale image with each pixel value between 0-255. It basically sets each pixel to 0 if the pixel value is below a threshold and 1 if it is above the threshold, thereby producing a binary image. The problem with common thresholding methods like adaptive threshold, binary thresholding is that threshold value varies with images. However, Otsu's threshold technique provides a way to dynamically determine the threshold value for each image which is used here.

4.5.4 Canny Edge Detection

The objects are identified, but for further processing only the outline of the objects is needed. Canny edge detector method in OpenCV provides this functionality.

4.5.5 Dilation and Erosion

Dilation and eroding of the image is done which helps in closing in on minute gaps and approximating Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries.

4.5.6 Getting all the contours

After all the pre-processing is done, the contours are detected from this image. This is done using the function cv2.findContours provided by the OpenCV library. These contours are then sorted from left to right. In this process, smaller contours representing the noises are ignored.

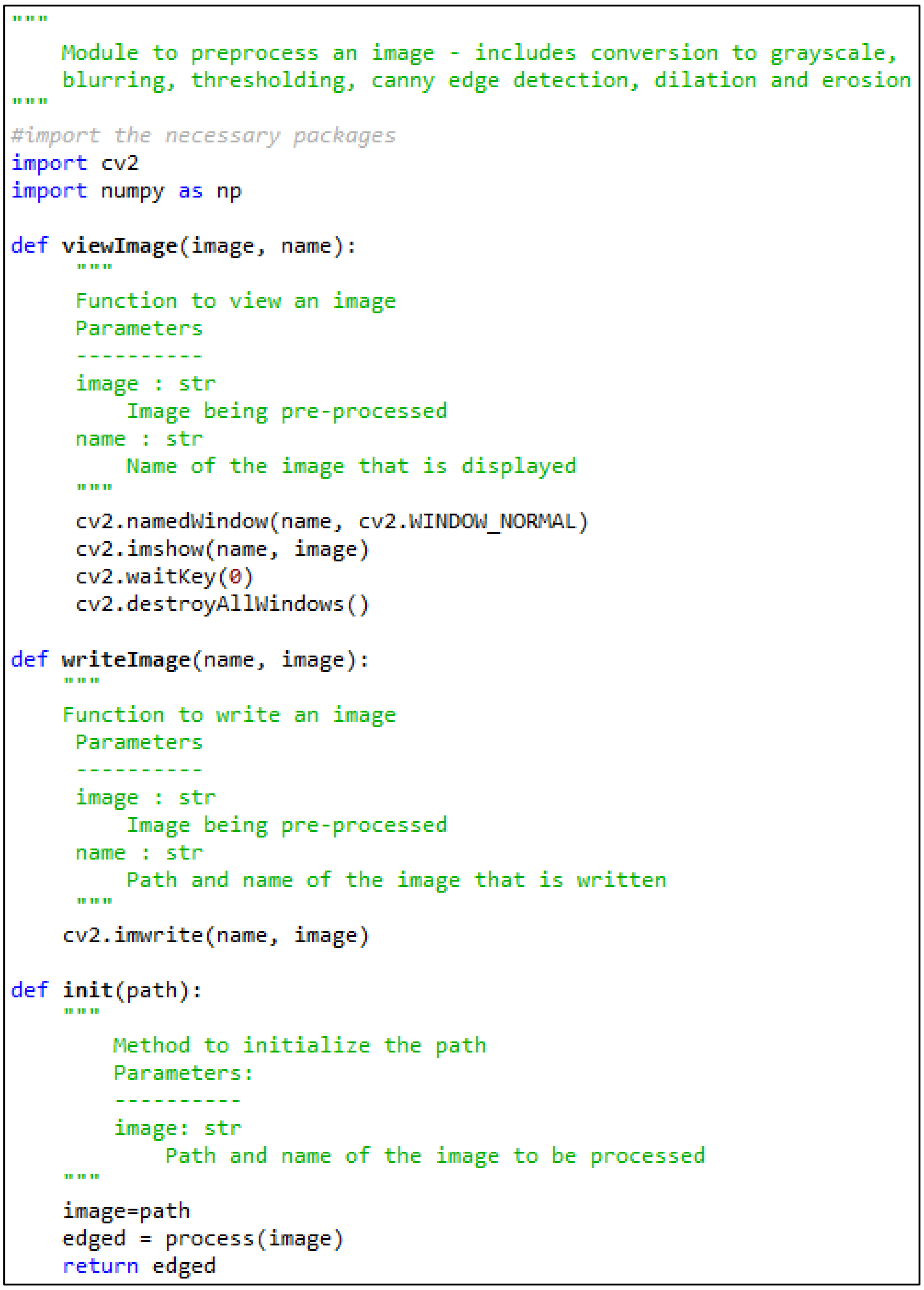
4.5.7 Check for convexity

A few contours may have a few edges not properly recognized, say the edge of a square is cut and pushed inwards. The convexHull function in OpenCV fixes this and provides a proper square.

4.5.8 Approximating the polygon

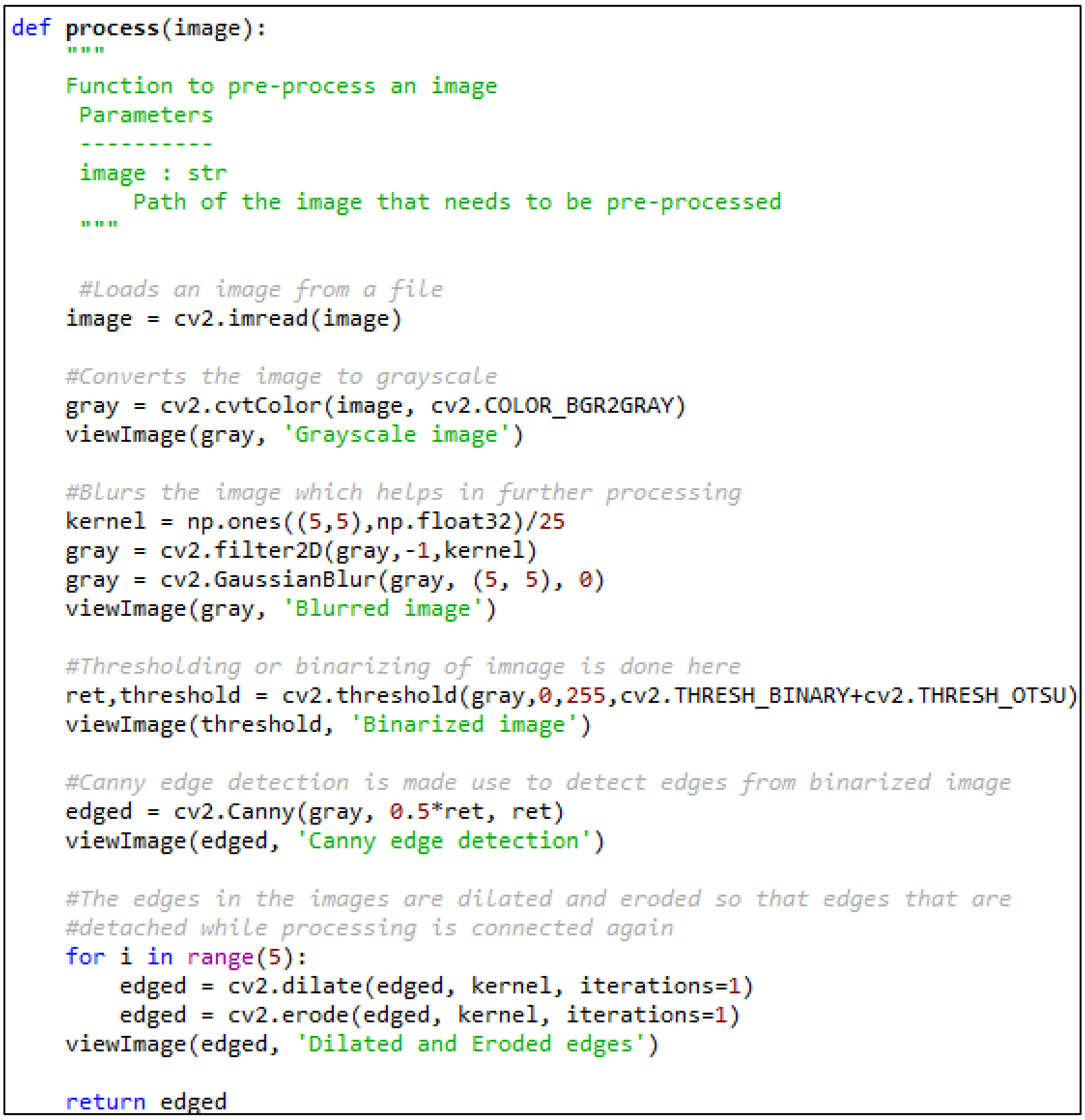
The contours determined thus far has far too many vertices, which might just differ by a pixel or two. All these unwanted vertices must be removed and this done using the approxPolyDp method provided by OpenCV. This approximates a polygon using the Ramer-Douglas-Peucker algorithm. It then returns a list of vertices that represents an object in the image, using which dimensions are then computed.

4.6 Implementation



```
"""
    Module to preprocess an image - includes conversion to grayscale,
    blurring, thresholding, canny edge detection, dilation and erosion
"""
#import the necessary packages
import cv2
import numpy as np

def viewImage(image, name):
     """
     Function to view an image
     Parameters
     ----------
     image : str
         Image being pre-processed
     name : str
         Name of the image that is displayed
     """
     cv2.namedWindow(name, cv2.WINDOW_NORMAL)
     cv2.imshow(name, image)
     cv2.waitKey(0)
     cv2.destroyAllWindows()

def writeImage(name, image):
    """
    Function to write an image
     Parameters
     ----------
     image : str
         Image being pre-processed
     name : str
         Path and name of the image that is written
     """
    cv2.imwrite(name, image)

def init(path):
    """
        Method to initialize the path
        Parameters:
        ----------
        image: str
            Path and name of the image to be processed
    """
    image=path
    edged = process(image)
    return edged
```

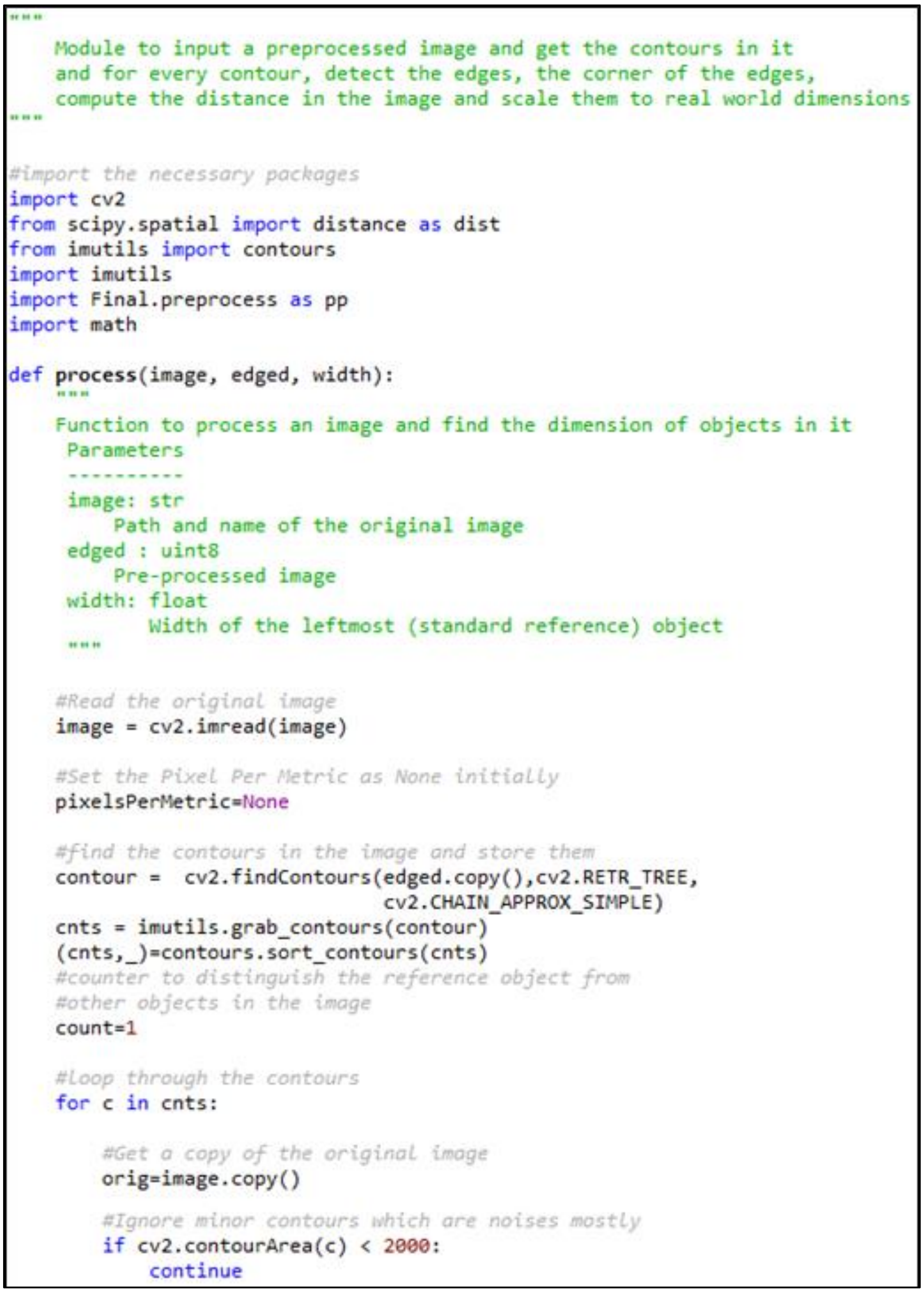
Figure 4.3 – Implementation of preprocess.py



```
def process(image):
    """
    Function to pre-process an image
     Parameters
     ----------
     image : str
         Path of the image that needs to be pre-processed
     """

     #Loads an image from a file
    image = cv2.imread(image)

    #Converts the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    viewImage(gray, 'Grayscale image')

    #Blurs the image which helps in further processing
    kernel = np.ones((5,5),np.float32)/25
    gray = cv2.filter2D(gray,-1,kernel)
    gray = cv2.GaussianBlur(gray, (5, 5), 0)
    viewImage(gray, 'Blurred image')

    #Thresholding or binarizing of imnage is done here
    ret,threshold = cv2.threshold(gray,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    viewImage(threshold, 'Binarized image')

    #Canny edge detection is made use to detect edges from binarized image
    edged = cv2.Canny(gray, 0.5*ret, ret)
    viewImage(edged, 'Canny edge detection')

    #The edges in the images are dilated and eroded so that edges that are
    #detached while processing is connected again
    for i in range(5):
        edged = cv2.dilate(edged, kernel, iterations=1)
        edged = cv2.erode(edged, kernel, iterations=1)
    viewImage(edged, 'Dilated and Eroded edges')

    return edged
```

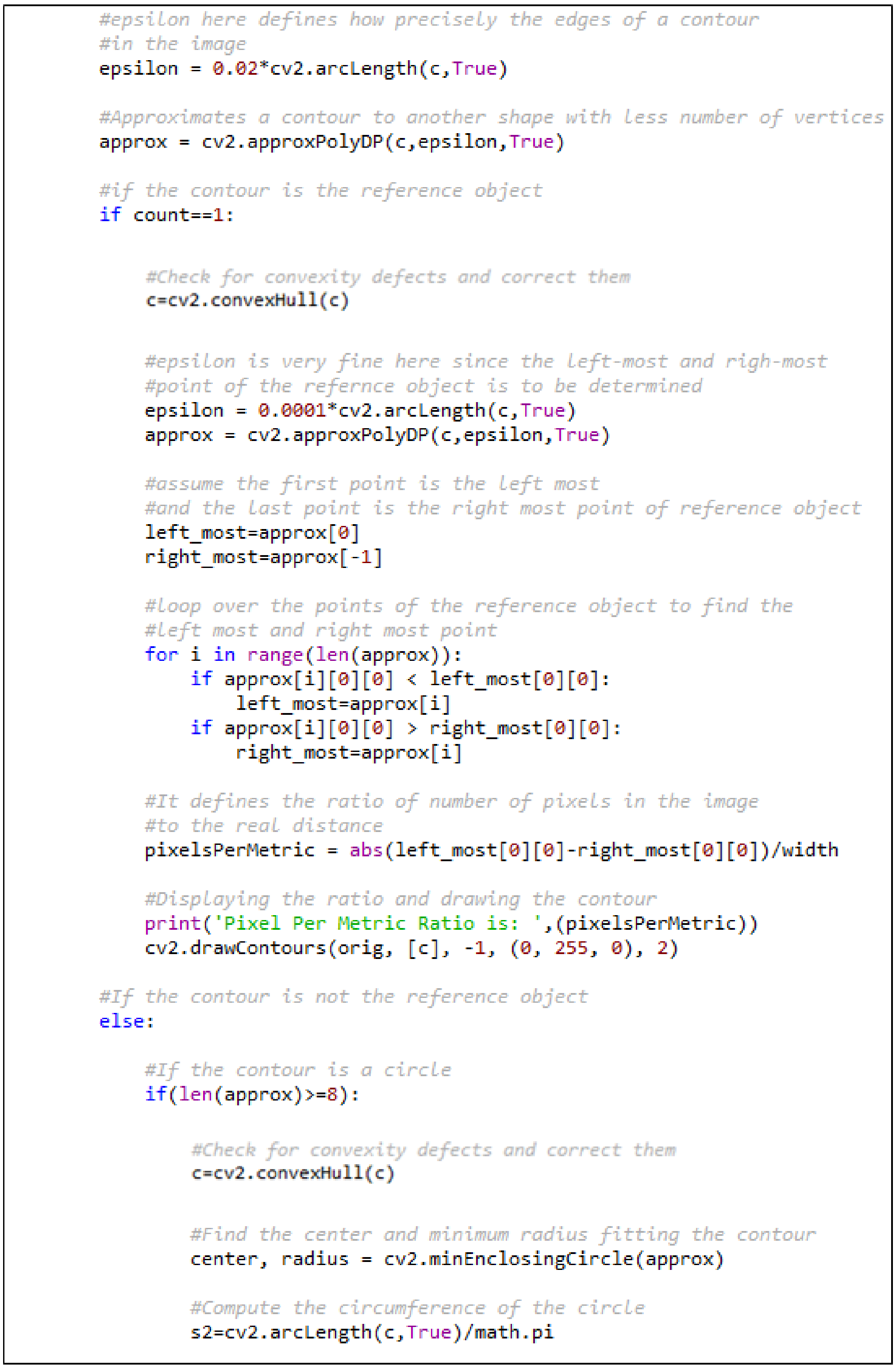
Figure 4.4 – Implementation of preprocess.py – 2

The preprocess.py module here does all the pre-processing of the image. It inputs the path of the image, reads the image, converts to grayscale, blurs it and binarizes the image which helps in object segmentation. Edges are identified in this image using Canny edge detection method, which is followed by dilation and erosion of the image to yield better results.



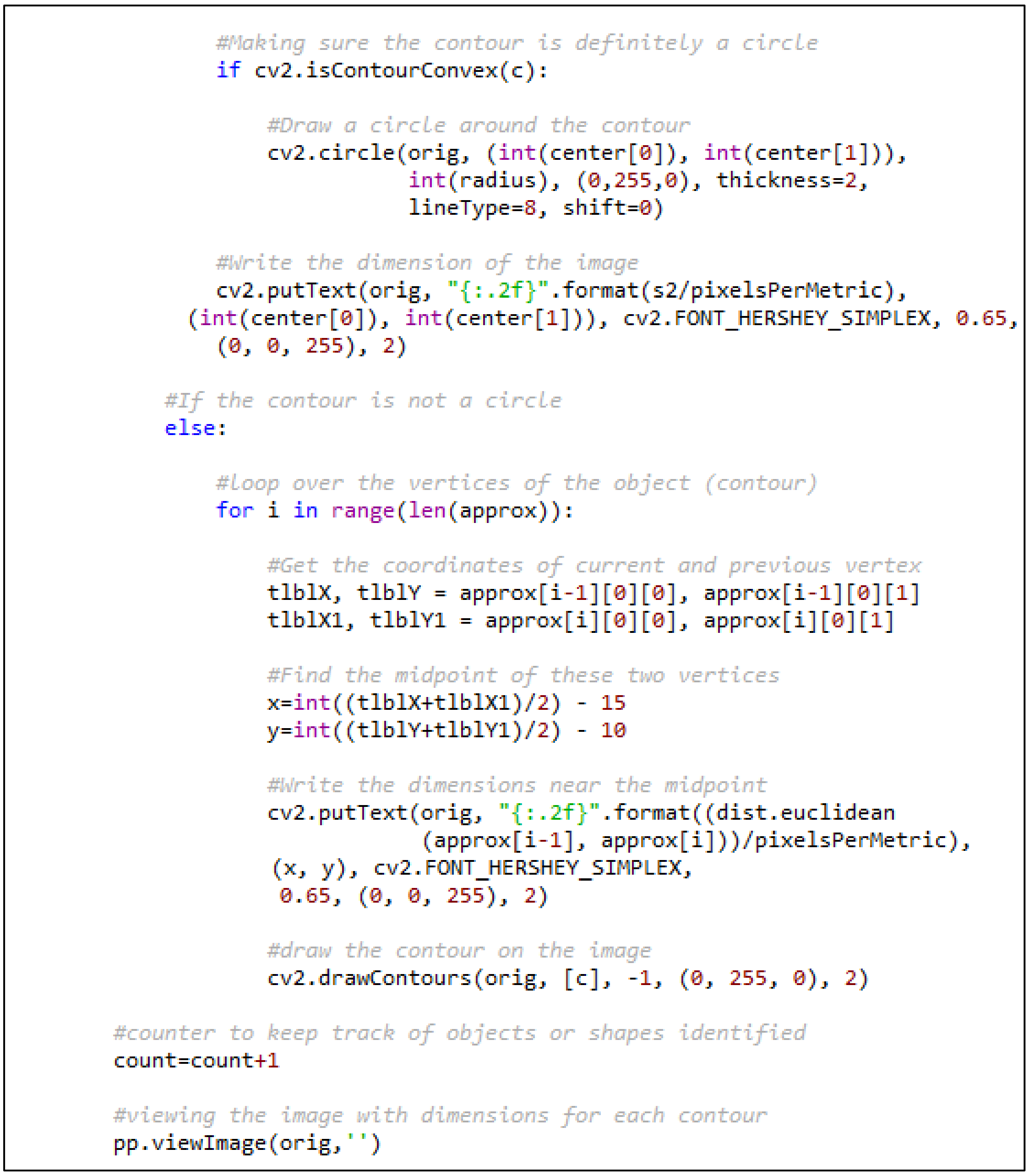
```
"""
    Module to input a preprocessed image and get the contours in it
    and for every contour, detect the edges, the corner of the edges,
    compute the distance in the image and scale them to real world dimensions
"""

#import the necessary packages
import cv2
from scipy.spatial import distance as dist
from imutils import contours
import imutils
import Final.preprocess as pp
import math

def process(image, edged, width):
    """
    Function to process an image and find the dimension of objects in it
     Parameters
     ----------
     image: str
         Path and name of the original image
     edged : uint8
         Pre-processed image
     width: float
            Width of the leftmost (standard reference) object
     """

    #Read the original image
    image = cv2.imread(image)

    #Set the Pixel Per Metric as None initially
    pixelsPerMetric=None

    #find the contours in the image and store them
    contour =  cv2.findContours(edged.copy(),cv2.RETR_TREE,
                                cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(contour)
    (cnts,_)=contours.sort_contours(cnts)
    #counter to distinguish the reference object from
    #other objects in the image
    count=1

    #Loop through the contours
    for c in cnts:

        #Get a copy of the original image
        orig=image.copy()

        #Ignore minor contours which are noises mostly
        if cv2.contourArea(c) < 2000:
            continue
```

Figure 4.5 – Implementation of findDimensions.py



```
        #epsilon here defines how precisely the edges of a contour
        #in the image
        epsilon = 0.02*cv2.arcLength(c,True)

        #Approximates a contour to another shape with less number of vertices
        approx = cv2.approxPolyDP(c,epsilon,True)

        #if the contour is the reference object
        if count==1:

            #Check for convexity defects and correct them
            c=cv2.convexHull(c)

            #epsilon is very fine here since the left-most and righ-most
            #point of the refernce object is to be determined
            epsilon = 0.0001*cv2.arcLength(c,True)
            approx = cv2.approxPolyDP(c,epsilon,True)

            #assume the first point is the left most
            #and the last point is the right most point of reference object
            left_most=approx[0]
            right_most=approx[-1]

            #loop over the points of the reference object to find the
            #left most and right most point
            for i in range(len(approx)):
                if approx[i][0][0] < left_most[0][0]:
                    left_most=approx[i]
                if approx[i][0][0] > right_most[0][0]:
                    right_most=approx[i]

            #It defines the ratio of number of pixels in the image
            #to the real distance
            pixelsPerMetric = abs(left_most[0][0]-right_most[0][0])/width

            #Displaying the ratio and drawing the contour
            print('Pixel Per Metric Ratio is: ',(pixelsPerMetric))
            cv2.drawContours(orig, [c], -1, (0, 255, 0), 2)

        #If the contour is not the reference object
        else:

            #If the contour is a circle
            if(len(approx)>=8):

                #Check for convexity defects and correct them
                c=cv2.convexHull(c)

                #Find the center and minimum radius fitting the contour
                center, radius = cv2.minEnclosingCircle(approx)

                #Compute the circumference of the circle
                s2=cv2.arcLength(c,True)/math.pi
```

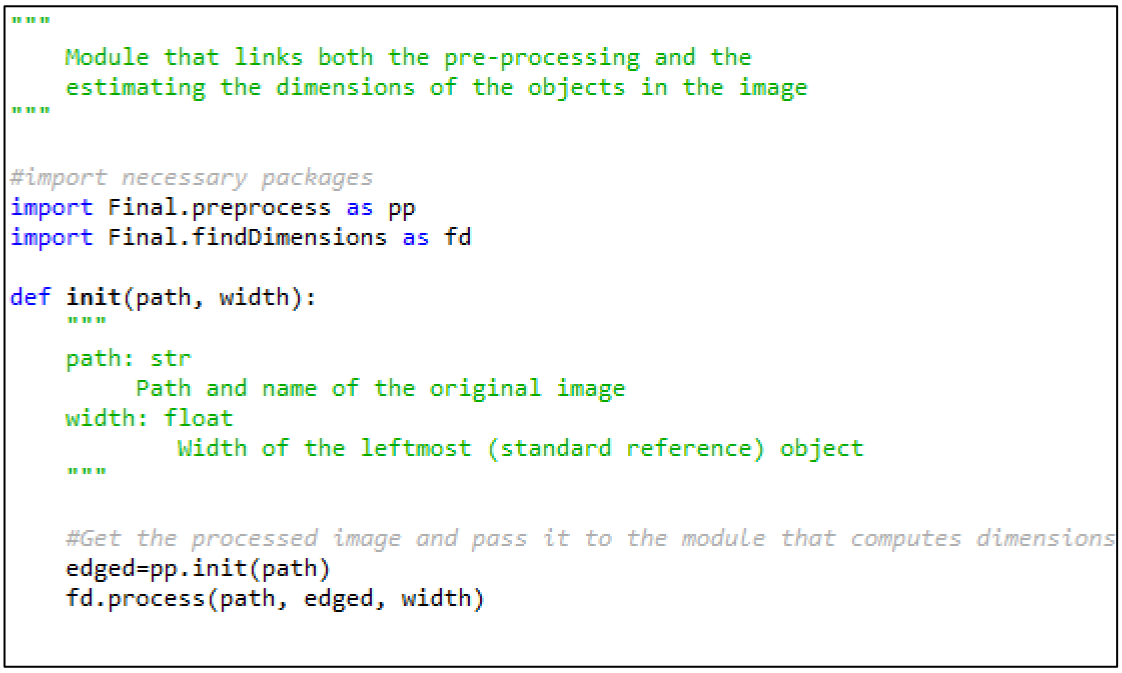
Figure 4.6 – Implementation of findDimensions.py – 2



```
                #Making sure the contour is definitely a circle
                if cv2.isContourConvex(c):

                    #Draw a circle around the contour
                    cv2.circle(orig, (int(center[0]), int(center[1])),
                               int(radius), (0,255,0), thickness=2,
                               lineType=8, shift=0)

                #Write the dimension of the image
                cv2.putText(orig, "{:.2f}".format(s2/pixelsPerMetric),
              (int(center[0]), int(center[1])), cv2.FONT_HERSHEY_SIMPLEX, 0.65,
                (0, 0, 255), 2)

            #If the contour is not a circle
            else:

                #loop over the vertices of the object (contour)
                for i in range(len(approx)):

                    #Get the coordinates of current and previous vertex
                    tlblX, tlblY = approx[i-1][0][0], approx[i-1][0][1]
                    tlblX1, tlblY1 = approx[i][0][0], approx[i][0][1]

                    #Find the midpoint of these two vertices
                    x=int((tlblX+tlblX1)/2) - 15
                    y=int((tlblY+tlblY1)/2) - 10

                    #Write the dimensions near the midpoint
                    cv2.putText(orig, "{:.2f}".format((dist.euclidean
                                (approx[i-1], approx[i]))/pixelsPerMetric),
                    (x, y), cv2.FONT_HERSHEY_SIMPLEX,
                     0.65, (0, 0, 255), 2)

                    #draw the contour on the image
                    cv2.drawContours(orig, [c], -1, (0, 255, 0), 2)

        #counter to keep track of objects or shapes identified
        count=count+1

        #viewing the image with dimensions for each contour
        pp.viewImage(orig,'')
```

Figure 4.7 – Implementation of findDimensions.py – 3

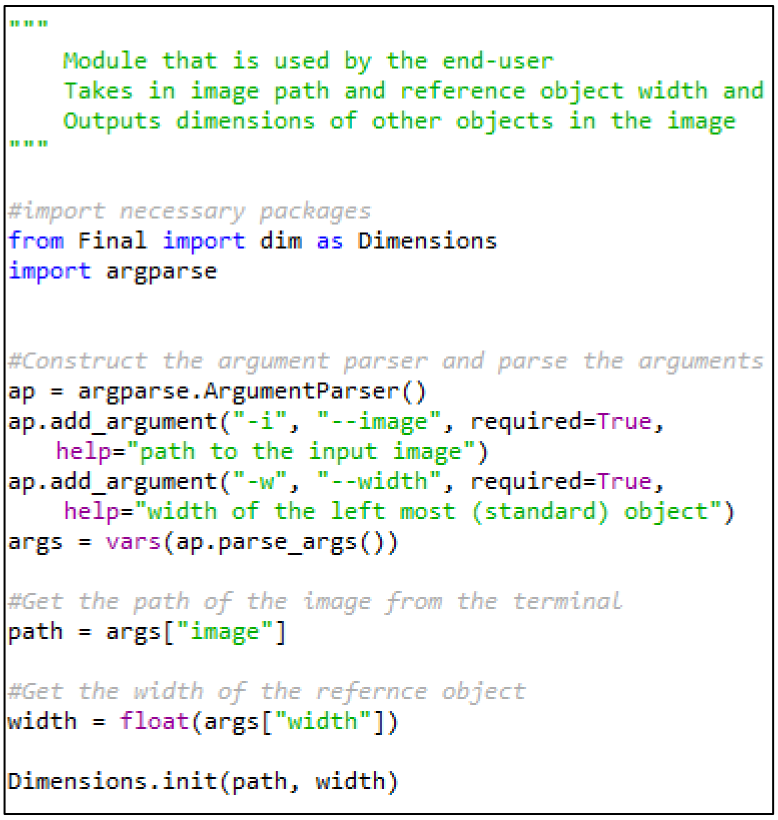
The findDimensions.py module takes in the binarized image from preprocess.py and finds the contours in it. For each contour identified, it is approximated to a polygon and its vertices are noted down. Euclidean distance between consecutive vertices are determined and scaled to the pixels per metric (PPM) ratio. PPM ratio is the ratio of actual dimensions of the reference object to its dimension computed by the system in the image. The dimensions are then displayed on the image.



```
"""
    Module that links both the pre-processing and the
    estimating the dimensions of the objects in the image
"""

#import necessary packages
import Final.preprocess as pp
import Final.findDimensions as fd

def init(path, width):
    """
    path: str
         Path and name of the original image
    width: float
            Width of the leftmost (standard reference) object
    """

    #Get the processed image and pass it to the module that computes dimensions
    edged=pp.init(path)
    fd.process(path, edged, width)
```

Figure 4.8 – Implementation of dim.py

The dim.py module integrates the pre-processing and finding the dimension functionality into one and the whole code is made available as a package named “Final”.

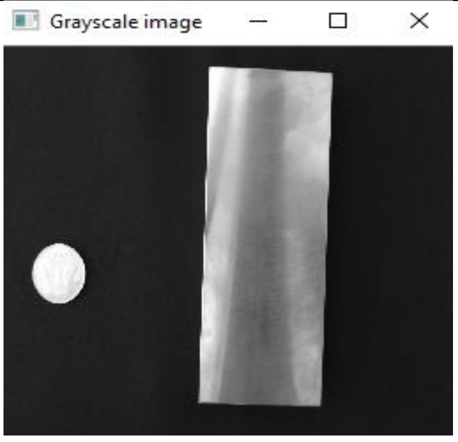
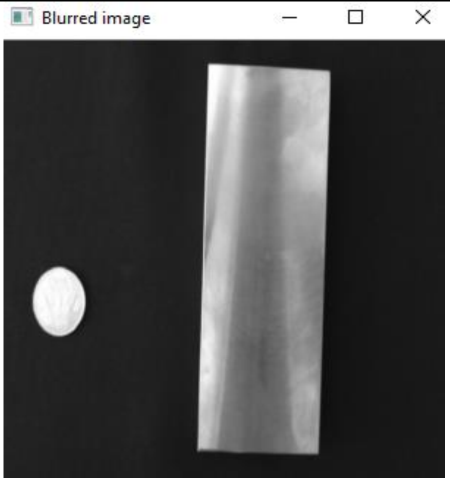
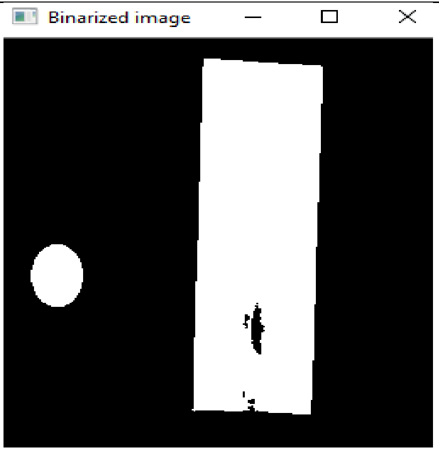


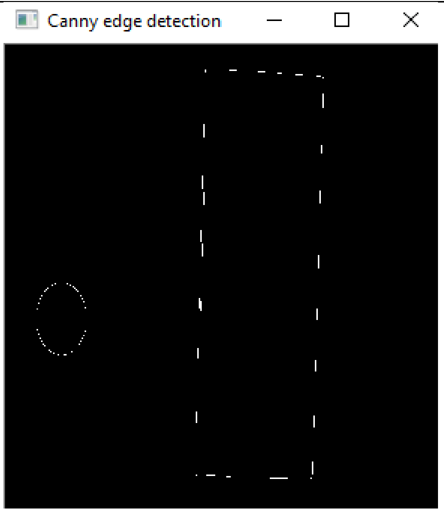
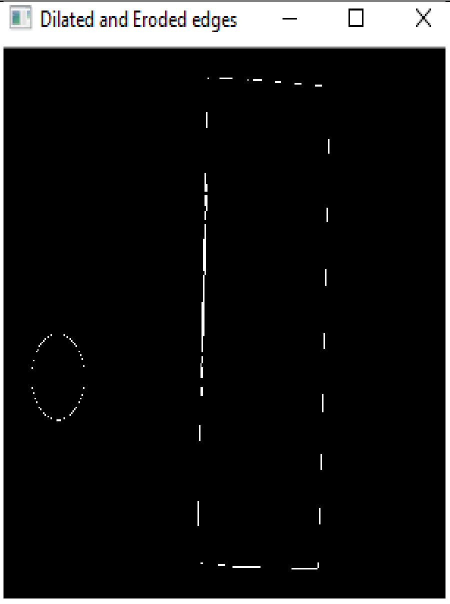
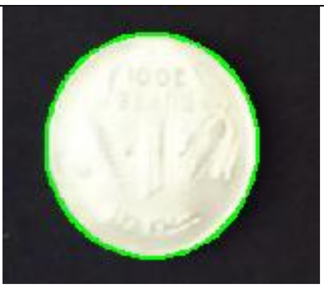
```
"""
    Module that is used by the end-user
    Takes in image path and reference object width and
    Outputs dimensions of other objects in the image
"""

#import necessary packages
from Final import dim as Dimensions
import argparse


#Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
   help="path to the input image")
ap.add_argument("-w", "--width", required=True,
    help="width of the left most (standard) object")
args = vars(ap.parse_args())

#Get the path of the image from the terminal
path = args["image"]

#Get the width of the refernce object
width = float(args["width"])

Dimensions.init(path, width)
```

Figure 4.9 – Implementation of User.py

The User.py module is the python code which the end-user has to run on the terminal. The user has the run the python file with the image path and width of the reference object as arguments. These arguments are parsed and sent to dim.py

4.7 Testing

Test Case	Description	Input	Expected Output	Actual output	Result
1.	Conversion to grayscale	Colour Image	A grayscale image		**Pass**
2.	Blur the image	Grayscale image	Blurred image		**Pass**
3.	Binarize the image	Blurred image	Complete black and white image		**Pass**

Test Case	Description	Input	Expected Output	Actual output	Result
4.	Canny edge detection	Threshold image	Image with edges of objects		**Pass**
5.	Dilation and Erosion	Edge detected image	Enhanced edges		**Pass**
6.	Get the contours	Image with edges detected	Objects with contours around it		**Pass**

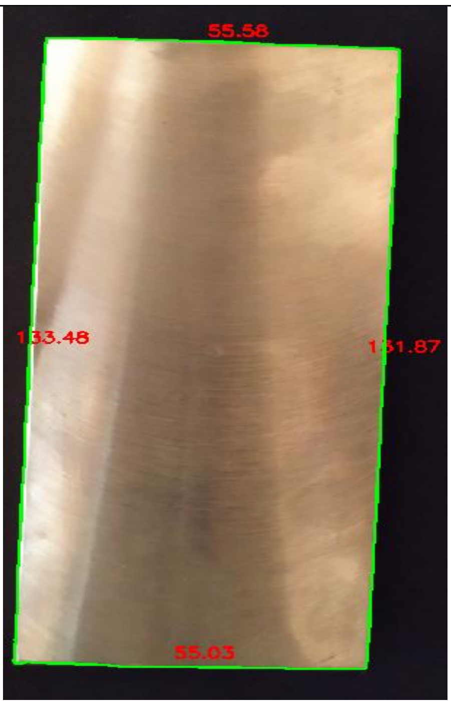
Test Case	Description	Input	Expected Output	Actual output	Result
7.	Show dimensions of objects	Image with edges detected	Object with its dimensions		**Pass**

Table 4.1 Unit test cases

4.8 Analysis

The different image processing techniques discussed in Section 4.5 are significant in the process of estimating the dimensions of the object in the image. The results produced are image dependent. The better the image, better results are given. It is as good as the image. The main criteria is that the image should be taken from a 90° from the plane containing the objects. Also, the background should not have any edges and must be distinguishable from the object easily. All these would lead to better results and improve the efficiency. The whole code is made available as a package, so the user has to just run one file and pass arguments to get results.

5. Results

In this part of the report results are shown i.e., output of our developed python program is shown when an image given as an input.

Many industries develop products everyday which have dimensions and dimensions of these products are to be measured by hand in most of the industries, for each product developed. They use very high precision Vernier calliper or ball micrometre to measure dimensions of these products.

After visiting an industry where such products are developed, for a reference their high precision measuring instrument was used to measure one of their product's dimensions. The actual real-world dimensions of the product are shown in below images (All dimensions shown in images are in mm)

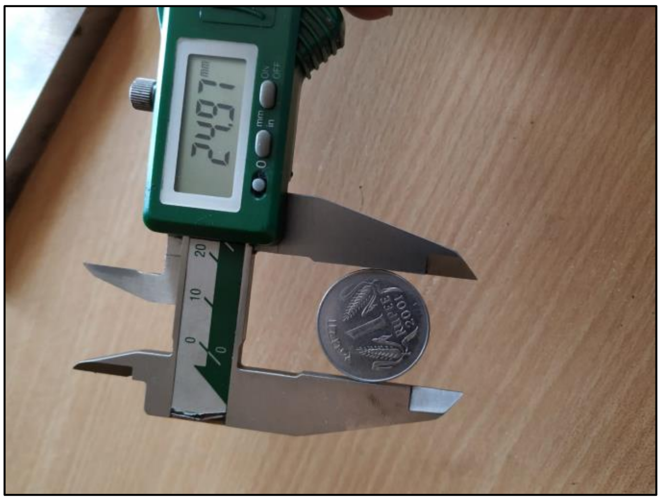


Figure 5.1 Showing real dimensions of reference object (a standard Indian one-rupee coin) which is 24.97 mm

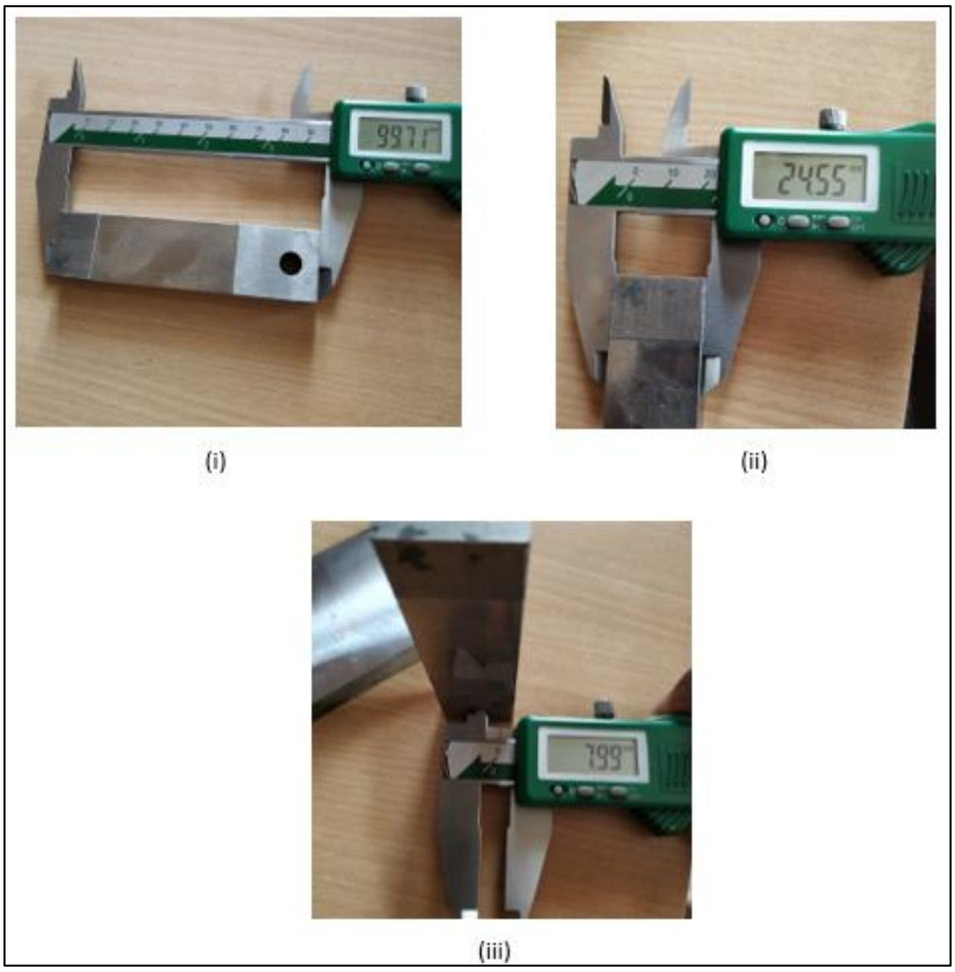


Figure 5.2 Showing real dimensions of a product which was developed in an industry we visited (i) its length was 99.71 mm (ii) its breath was 24.55 mm and (iii) diameter of hole is 7.99 mm

Now, an image of this product next to our reference object (standard Indian one-rupee coin) was taken with a phone (Xiaomi Mi A2) camera holding it in our bare hands.

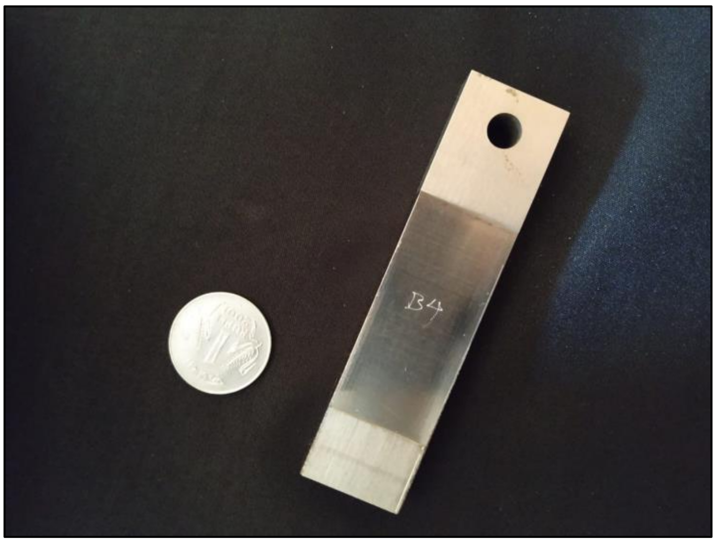
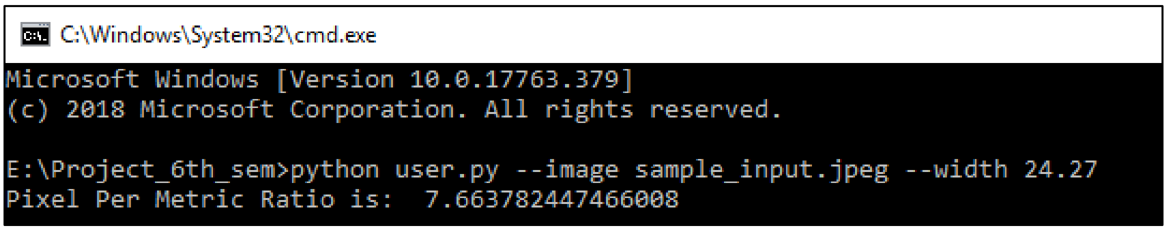


Figure 5.3 Input image given to developed python program

This image undergoes pre-processing where it will be converted to grayscale, grayscale image is then blurred now this image undergoes thresholding where background is separated from the objects and then edges are detected after thresholding is done.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

E:\Project_6th_sem>python user.py --image sample_input.jpeg --width 24.27
Pixel Per Metric Ratio is:  7.663782447466008
```

Figure 5.4 In this, location of the image to be uploaded and width (in this case diameter of one-rupee coin) of reference object is given as prerequisite

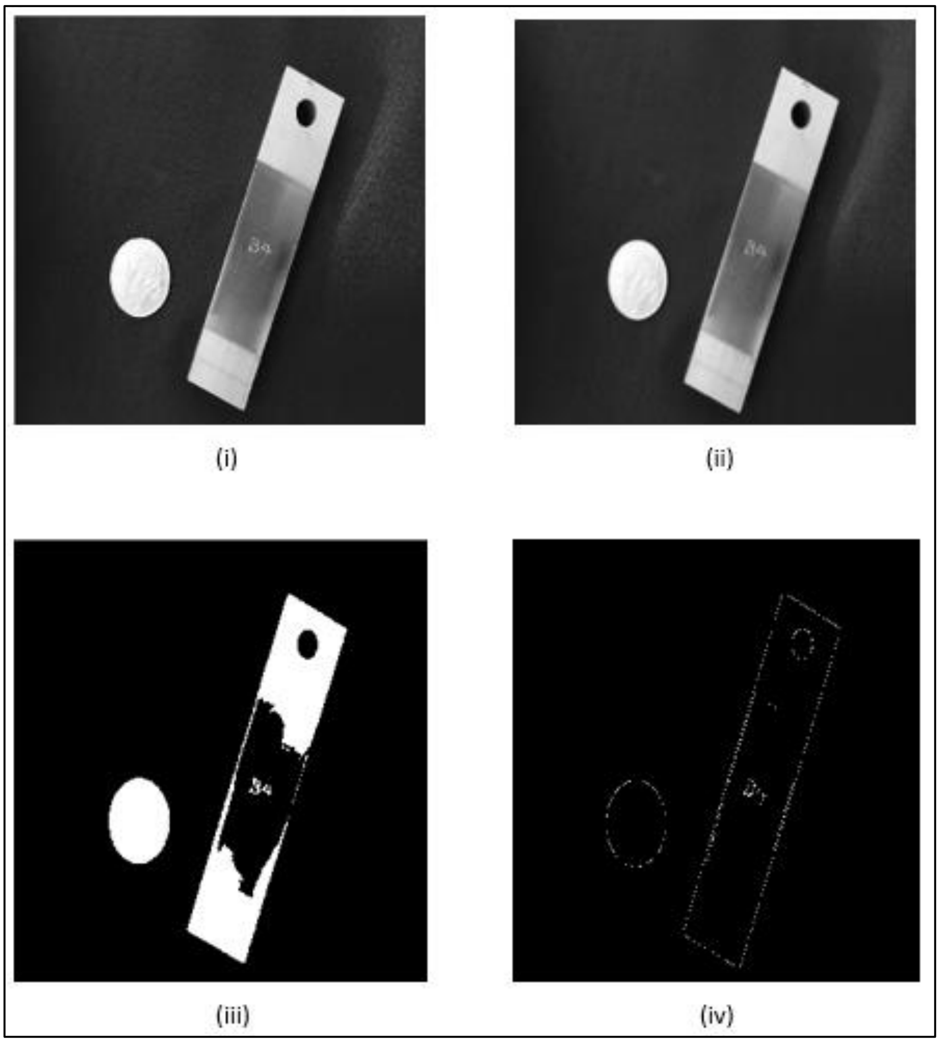


Figure 5.5 showing the stages of pr-eprocessing done on the input image

(i) image after converting it to grayscale (ii) image after blurring the grayscale image

(iii) thresholding the blurred image (iv) identified edges of threshold image

After the above pre-processing is done to the image, program identifies the leftmost object in the image as a reference whose real dimensions are already known and given as input to the program.

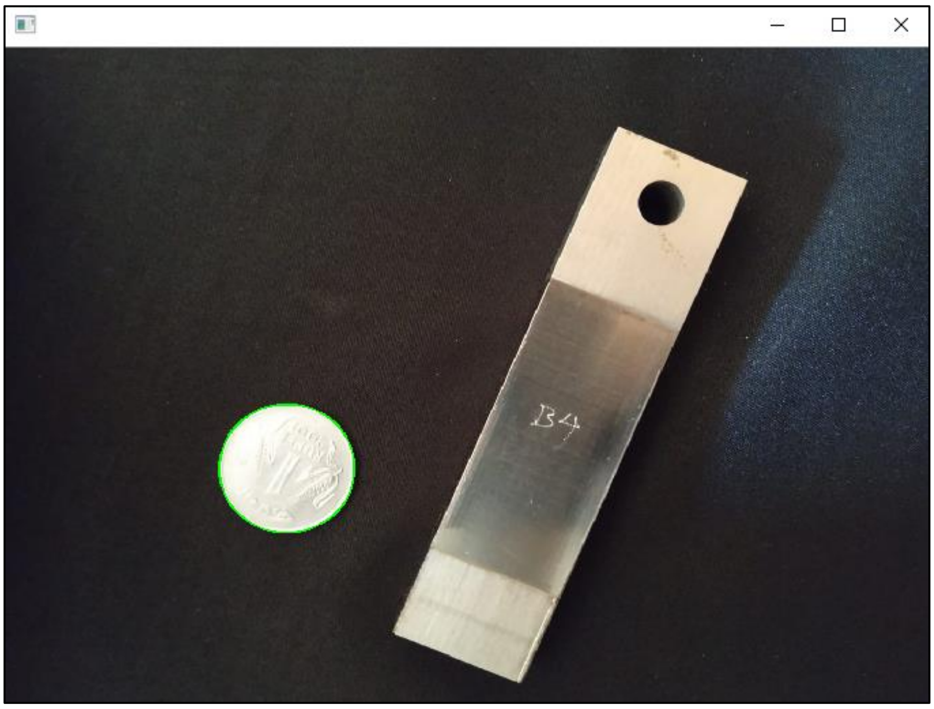


Figure 5.6 Showing that reference object is identified from the image given

Based on given dimensions of reference object pixels per metric ratio is calculated from this ratio approximate dimensions of object in input image are calculated and shown in an output image

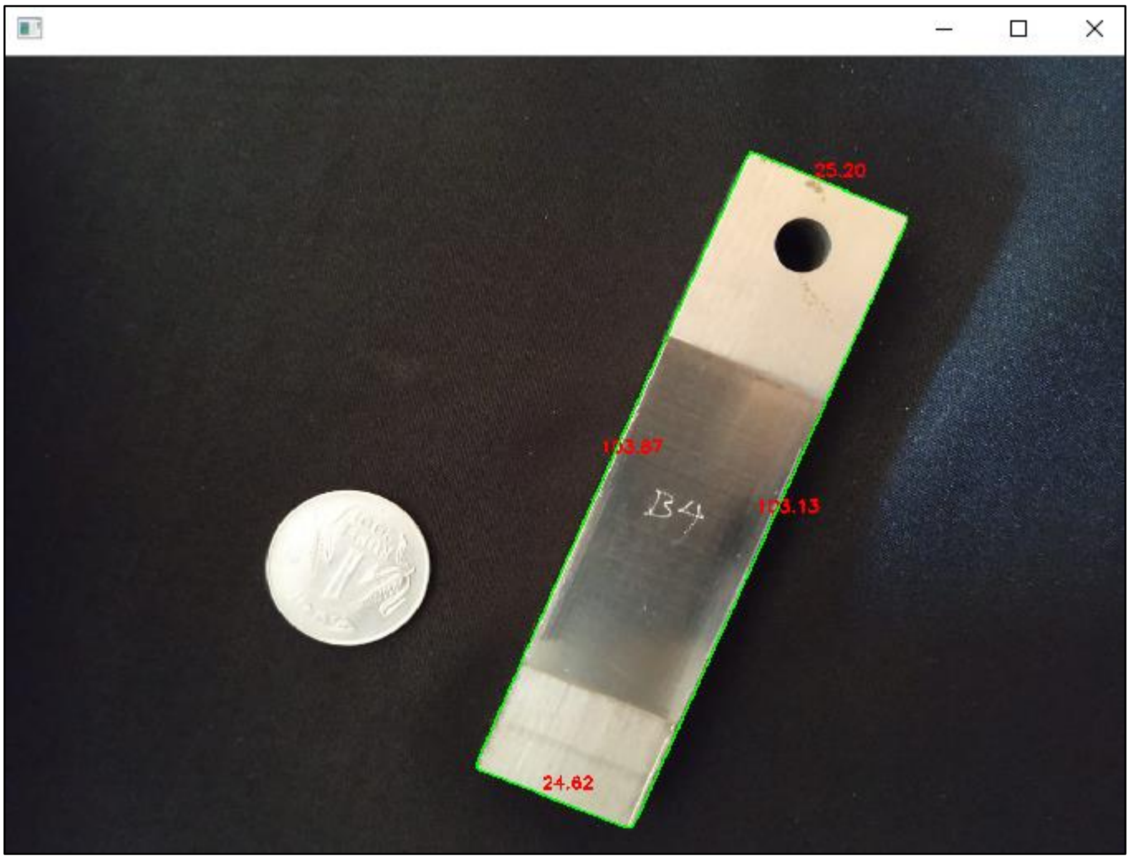


Figure 5.7 Output image showing the length and breadth of the object in mm, length is shown as 103.13 mm and breadth is shown as 24.62 mm (considering the closest values)

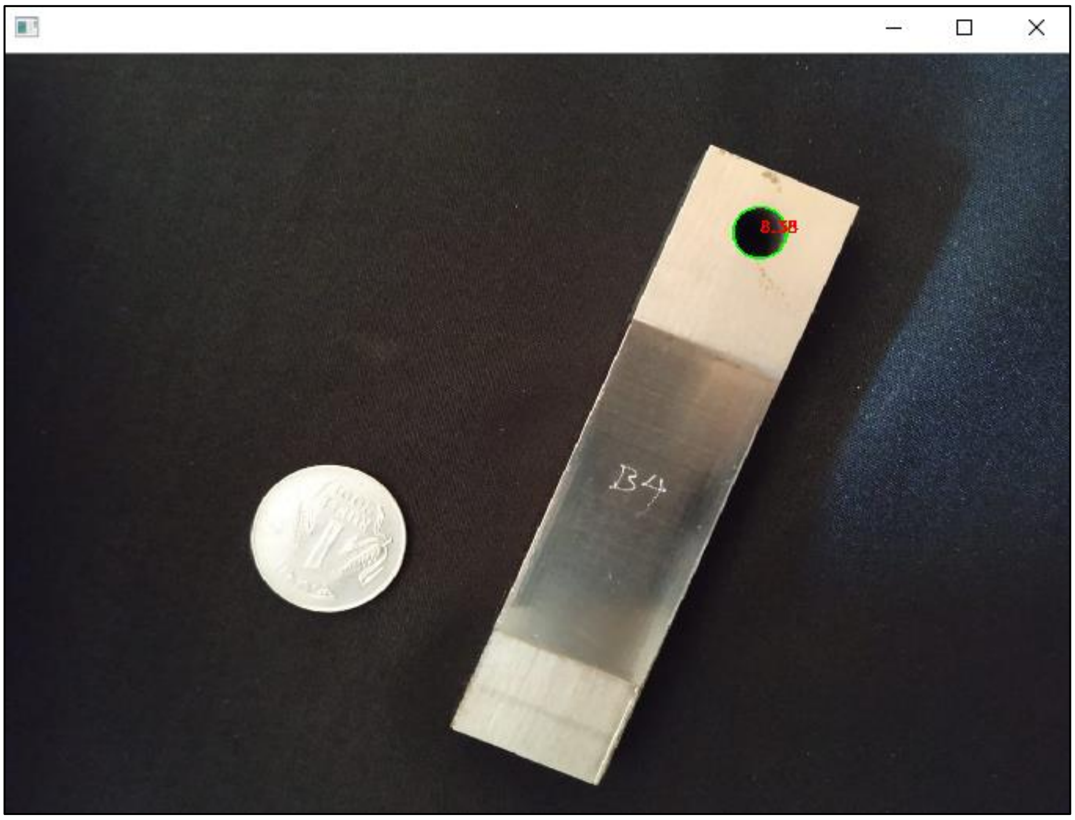


Figure 5.8 Output image showing diameter of hole drilled into object in mm, diameter is shown as 8.38 mm

Table 5.1 comparing the real dimensions and dimensions obtained from output image

Real Dimensions	Output dimensions (from image processing)	Difference	Accuracy
99.71 mm	103.87 mm and 103.13 mm	4.16 mm and 3.42 mm	96.3 %
24.55 mm	24.62 mm and 25.20 mm	0.07 mm and 0.65 mm	98.5 %
7.99 mm	8.38 mm	0.39 mm	95.3 %

What happens if the image is not taken perpendicular to the object:

While taking the image to find the dimensions one should make sure the camera lens is facing exactly perpendicular if not the result i.e., the dimensions obtained will not be so accurate. This will be explained briefly by showing some sample images.

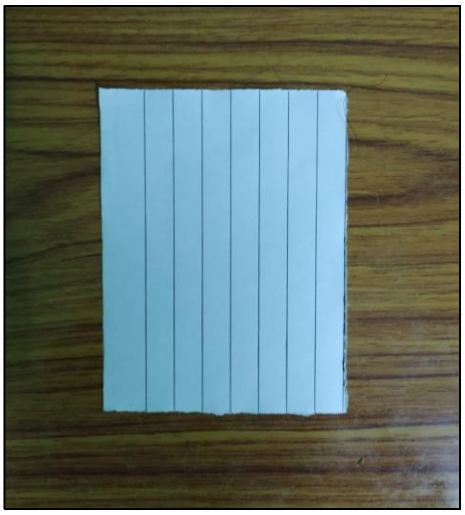


Figure 5.8 Image taken exactly perpendicular to the object (piece of paper) shows its original shape as a rectangle

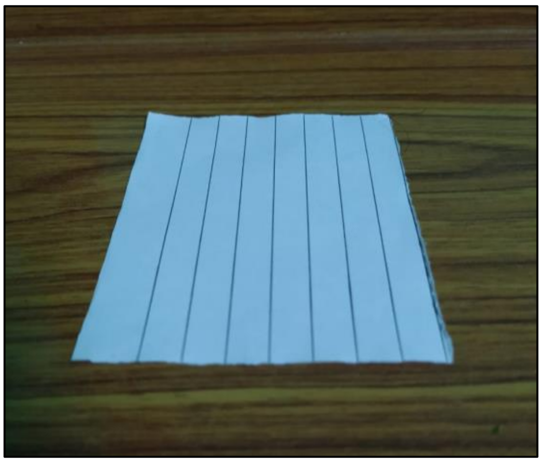


Figure 5.9 Image taken with some angle to the same object (piece of paper) shows that the rectangle shape is now somewhat looking like trapezoid

From the above images shown one can conclude that shape of the objects appears different from different angles similarly in images they look like both have different dimensions though being the same object in both the images

Our main objective in this project was to find approximate dimensions of objects just by using simple image clicked from our smartphone and using some image processing methods available which, we have done with accuracy level of nearly, greater than or equal to 94 percent as depicted in table 5.1

6. Project Costing

In the current chapter, cost incurred during the development of the project is mentioned.

Table 6.1 Project costing table

Component	Cost
Laptops	4 laptops× ₹ 50,000 / laptop **= ₹ 2,00,000/-**
Human Effort	$4\ people\ (4\ hours \times 2.5\ days) \times 16\ weeks$ $=$ 640 Man Hours 640 hours × ₹ 200 per hour **= ₹ 1,28,000/-**
Total Cost	**₹ 3,28,000/-**

7. Conclusions and Suggestions for Future Work

In this part of the report final conclusions are drawn for our project work and also discuss some methods to improve our accuracy in measuring the dimensions of object.

After going through some of the research papers and some articles based on image processing, many possible ways to calculate the dimensions of object from an image were found. Two of them were very easy and efficient to implement, one was to know the focal distance between the object and the camera while taking the image and focal length of the lens of camera being used to take the image beforehand, using this data the dimensions of the object can be calculated by some simple knowledge on trigonometry. In another method, (the one which is chosen for this project work) while taking the image of an object to calculate its dimensions a reference object should also be present in that image whose dimensions are known beforehand and now by knowing this one can calculate the number of pixels occupied within that specified dimensions of reference object leading to PPM ratio using which the dimensions of the object are calculated and given as output.

OpenCV, imutils and scipy are the most necessary libraries in the project. OpenCV library was used in the project for pre-processing of the input image, imutils library was used to find the contours from the pre-processed image and finally scipy library was used to find the Euclidean distance of each edge identified from the image. All these libraries are free to use and are mostly open source.

Now, after dividing this distance with already calculated PPM ratio gives the dimensions of the object.

Suggestions for future work:

- Presently the program works better when the image is taken in black background
- While taking the image shadows and reflections interfere, which result in difficulty to identify objects from the background correctly

The program can be improved by developing a workaround in the above mentioned problems which is our future work in terms to improve our project

References

❖ Chakravorty, Pragnan. "What Is a Signal? [Lecture Notes]," IEEE Signal Processing Magazine, vol. 35, no. 5, pp. 175-177, Sept. 2018. https://doi.org/10.1109/MSP.2018.2832195

❖ Fernandes, Leandro A. F., Oliveira, Manuel M., Silva, Roberto da, & Crespo, Gustavo J. (2006). “A fast and accurate approach for computing the dimensions of boxes from single perspective images”. Journal of the Brazilian Computer Society, 12(2), 19-30. https://dx.doi.org/10.1007/BF03192392

❖ Guo, R, Dai, Q & Hoiem, DW 2011, Single-image shadow detection and removal using paired regions. in 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011., 5995725, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, pp. 2033-2040. https://doi.org/10.1109/CVPR.2011.5995725

❖ Jiejie. Zhu, Kegan. G. G. Samuel, Syed. Masood, and Marshall. F. Tappen. (2010). “Learning to recognize shadows in monochromatic natural images”.[Online]. Available at: http://www.cs.ucf.edu/~mtappen/pubs/cvpr10_shadow.pdf [Accessed: 27 Feb. 2019].

❖ Johnson, Forrest. (2010). “How to measure the size of an object using camera” [*Online*]. *Journal.* Available at: https://web.archive.org/web/20131123073926/http://forestjohnson.blogspot.com/2010/01/how-to-measure-size-of-object-using.html [Accessed: 24 Feb. 2019].

❖ Lindberg, David C. (2007). "Science before the Greeks". The beginnings of Western science: the European Scientific tradition in philosophical, religious, and institutional context (Second ed.). Chicago, Illinois: University of Chicago Press. pp. 1–27. ISBN 978-0-226-48205-7.

- Meysenburg, Mark. (2016). Image Processing: Edge Detection. [online] mmeysenburg.github.io. Available at: https://mmeysenburg.github.io/image-processing/08-edge-detection/ [Accessed 26 Mar. 2019].