Estimation of Object Dimension using Image Processing

Shridhar Hegde[1]; Santosh G[2], Shivakumar M[3]
Department of Computer Science and Engineering
Ramaiah University of Applied Sciences, Bengaluru, India

Abstract — ***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.

Keywords - dimensions; camera; python; opencv; imutils; canny method; reference object method

I. INTRODUCTION

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 day to day tasks that people carry out.

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.

II. RELATED WORKS

In literature, some algorithms have been proposed to find the dimensions of an object using its 2D/3D image. In this paper [1], the technique used measures 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.

Reference [2] chooses a reference plane in the image, which is usually the ground floor. The elements in the scene are first segmented to provide the texture maps. Finally, element positions on the reference plane and distances from that plane (i.e. heights, if the ground plane is chosen as the reference) are estimated. This estimation mostly follows the single-view metrology algorithms suggested in [3], and could be performed in an uncalibrated framework, in which no estimation of camera pose or the internal parameters is required.

This paper [4] 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.

III. DESIGN

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.

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.

A flowchart is designed representing the flow of the system shown in Figure 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.

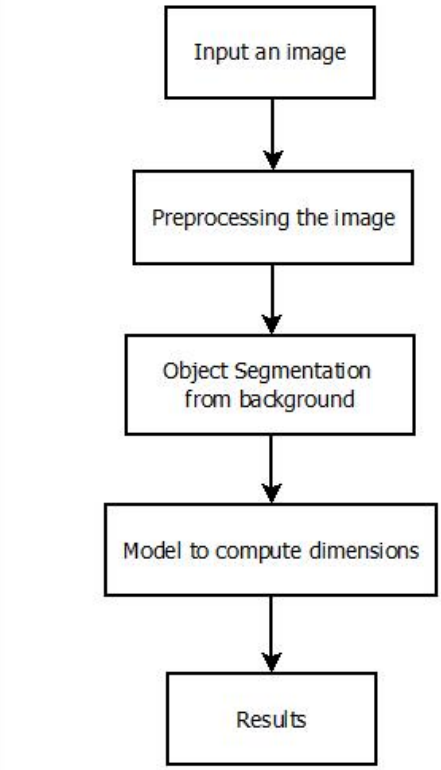

Figure 1. High level block diagram

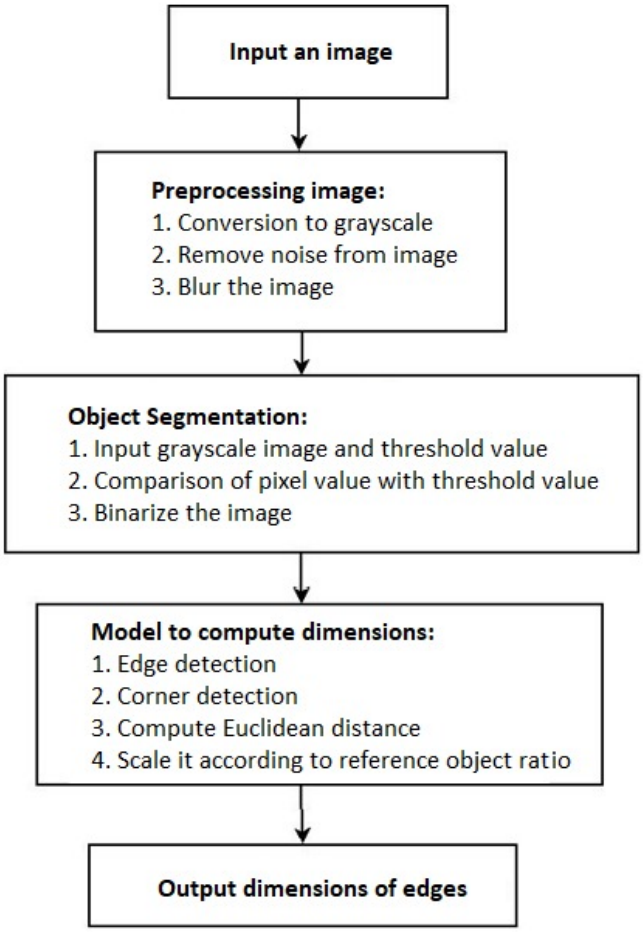

Figure 2. Low level block diagram

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 2.

A. Choice of programming language and libraries/tools

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

1) 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.

2) 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

B. Image Processing Techniques

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

1) Converstion 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.

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 etc.

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) 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.

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.

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.

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.

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.

IV. RESULTS

Many industries develop products every day 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 calipers or ball micrometer 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).

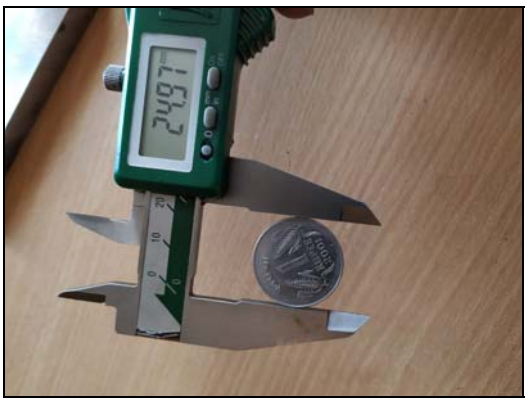

Figure 3. Showing real dimensions of reference object (a standard Indian one-rupee coin) which is 24.97 mm

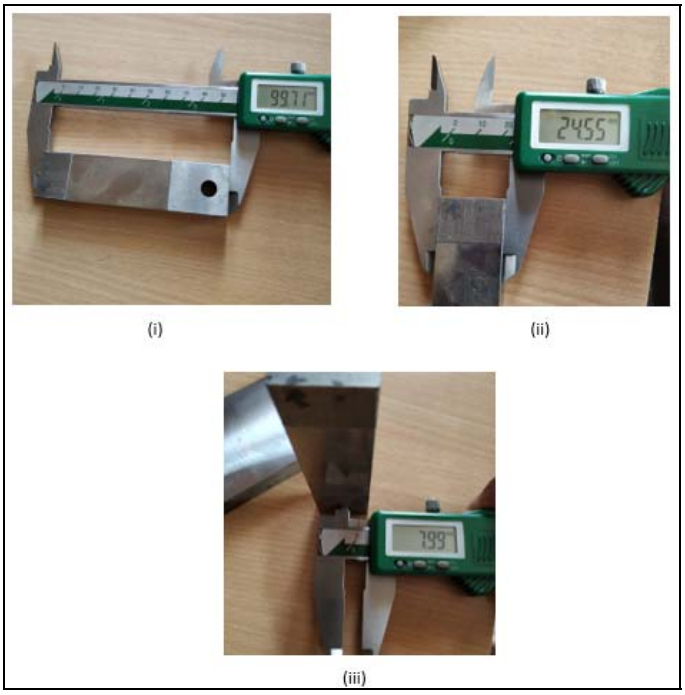

Figure 4. Showing real dimensions of a product which was developed in an industry we visited (i) its length was 99.71 mm (ii) its breath was 24.55 mm and (iii) diameter of hole is 7.99 mm

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. Input image given to developed python program

```
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 6. 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 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.

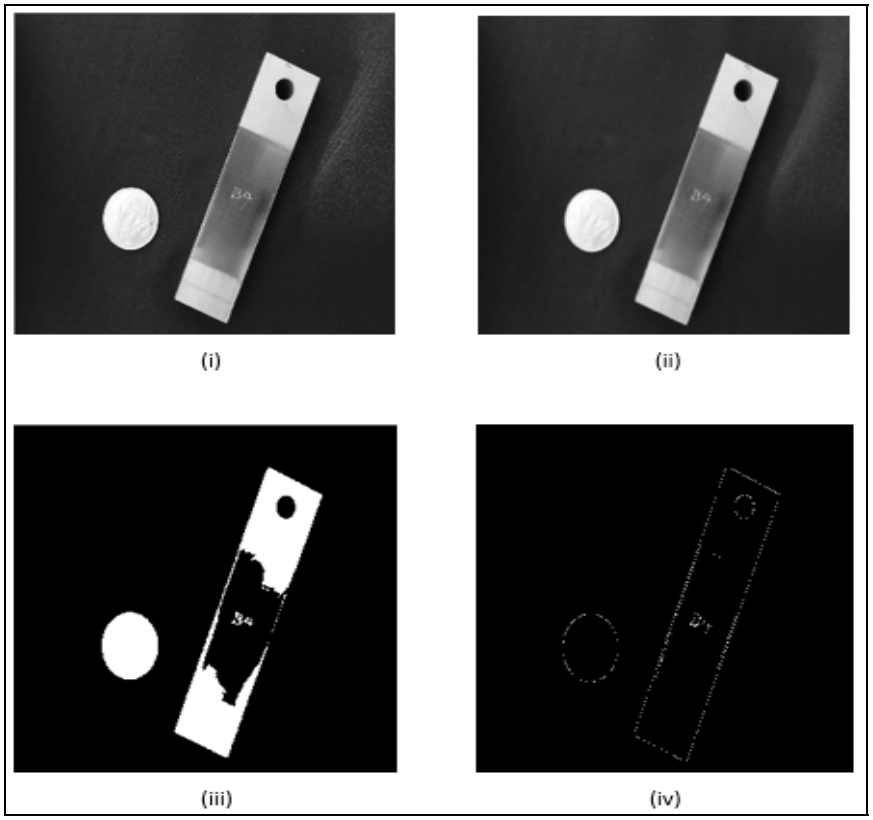

Figure 7. Showing the stages of pre-processing done on the input image (i) image after converting it to grayscale (ii) image after blurring the grayscale image (iii) thresholding the blurred image (iv) identified edges of threshold image

After the pre-processing is done to the image as shown in figure 7, program identifies the leftmost object in the image as a reference whose real dimensions are already known and given as input to the program.

The program first identifies the reference object which is placed on the left side of the image and then moves on to the target object which is on the right side. The dimensions of the target object are then shown in the output terminal.

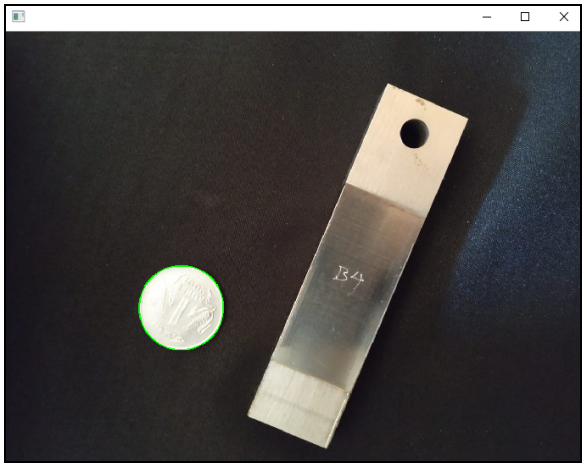

Figure 8. Showing that reference object is identified from the image given

Based on given dimensions of reference object, pixels per metric ratio is calculated from the reference object's actual size and the pixels it occupies in the image. Using this ratio, approximate dimensions of object in input image are calculated and shown in an output image.

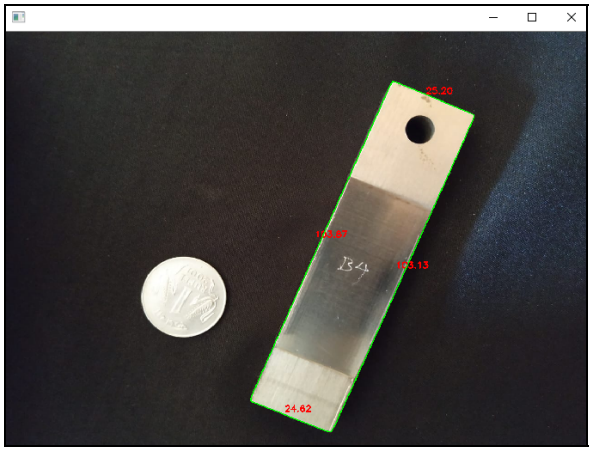

Figure 9. Output image showing the length and breadth of the object in mm, length is shown as 103.13 mm and breadth is shown as 24.62 mm (considering the closest values)

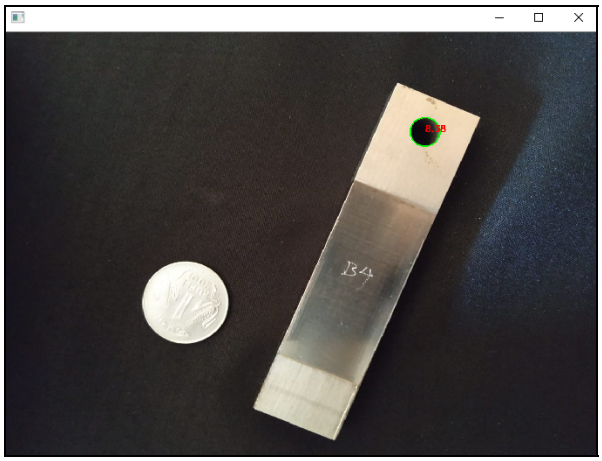

Figure 10. Output image showing diameter of hole drilled into object in mm, diameter is shown as 8.38 mm

TABLE I. COMPARING THE REAL DIMENSIONS AND DIMENSIONS OBTAINED FROM OUTPUT IMAGE

Real Dimensions	Output Dimensions	Difference	Accuracy
99.71mm	103.87mm and 103.13mm	4.16mm and 3.42mm	96.3%
24.55mm	24.62mm and 25.20mm	0.07mm and 0.65mm	98.5%
7.9mm	8.38mm	0.39mm	95.3%

Referring to table 1, one can see that the error is of the order of millimeters with best accuracy of over 98%. 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 has been done with accuracy level of greater than 95 percent as depicted in table 1.

The different image processing techniques discussed in Section 3.5 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 are 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.

V. CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

A. *Conclusion*

After going through some of the research papers and articles based on image processing, many possible ways to calculate the dimensions of object from an image were found. Two of them were 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.

B. Suggestions for future work

Current issues with this work include:

1) Works better when the image is taken in black background, otherwise, the model is prone to errors

2) While taking the image, shadows and reflections interfere; this results in difficulty to identify objects from the background correctly

The program can be improved by developing a workaround in the above-mentioned problems.

REFERENCES

[1] Ali, Yasir & Malik, Aamir. (2012). Depth and Geometry from a Single 2D Image Using Triangulation. Proceedings of the 2012 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2012. 511-515. 10.1109/ICMEW.2012.95

[2] Shih-Yu Sun, "An Implementation of Single-View Metrology" [Online]. Available at: https://pdfs.semanticscholar.org/2a4b/290d10b49b35f9fe61cda7aeeba259be5f38.pdf [Accessed: 27 March. 2019]

[3] A. Criminisi, "Single-view metrology: Algorithms and applications," Lecture notes in computer science, 2002, p. 224–239.

[4] 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