CRANFIELD UNIVERSITY

ASSIGNMENT – VI

MACHINE VISION FOR ROBOTICS

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING

Mr. Balasekhar Chandrasekaran Kannan

S359368

M.Sc. Robotics

Academic Year : 2021 – 2022

Faculty : Dr. Zeeshan Rana

February 2022

1. ABSTRACT

The use of computer vision and related technologies to industrial automation is known as machine vision. One of these applications, automated visual inspection, can be utilised to solve a variety of challenges in industry. Customized solutions are required for industrial applications, which are constrained by a number of factors. The last stage of integrating a vision system into an industrial process is difficult due to the variability of the process; the design of a machine vision system must take this variability into consideration (Fernandes, Moreira, & Mata, 2011). This study discusses dimension checking stage of industrial process where a system is developed to measure the objects instantaneously with minimum error rate.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

2. INTRODUCTION

Computer vision place a very important role in metrology applications in today’s world. Vision based systems are being implemented in various sectors for fast and accurate calculations of object’s dimensions. Various sectors like aerospace are heavily dependent on metrology since they act under very less tolerances. Any deviation from these values can be very unforgiving, especially during operational stage. Several MRO companies are dependent of perfecting their measurement standards in order to improve their efficiency of inspection. They are focussed on reducing the downtimes of their inspection process to increase the operational times of the aircraft. With every operational minute, the performance of the aircraft deteriorates and beyond certain limit the parts start to show signs of wear and tear. Initial stage of inspection is purely visual to evaluate the dimensional validity of parts. This is where metrology is helpful. The parts are measured with instruments that are calibrated with international standards.

With MRO (Maintenance, repair and overhaul) companies looking to optimise the process, the physical measurements of parts would take significant time out of entire overhaul time. The topic of vision systems is introduced here which has the capacity to give instantaneous measurements. It is level of accuracy that determines the feasibility of the system to be used for implementation. The very reason can be associated with all the sectors that depend on any time of metrology in their process.

The problem statement and objective of this study is given in the next section. Section –5 covers the description of the system that is detailed in this study. The evidence of experimentation done to analyse the flexibility of the system is given in section-6. This involves changing various parameters such as lighting condition, mounting positions etc. The results obtained from these various experimentation procedures are given in section-7 under the heading system assessment and analysis. Results and future work is written in section-8.

3. PROBLEM STATEMENT

We have explained the importance of metrology in today’s world. It is necessary to perform measurement operations with high accuracy. But the current tools used for measurement are incapable of achieving these targets. We have turned our attention to image processing to achieve this goal. It is easy to capture an image real time of an object and it can be done even in places where human cannot reach. The current disadvantages associated with conventional methods of measurements are instruments error, slow speed, accessibility to the part to be measured, lack of standard references etc. Most of this can be addressed by using image processing.

Every industry is trying to optimise their metrology systems in order to improve the longevity of their products in operation. The problem system deals in developing a vision based object measurement system which has high accuracy and is dependable enough to be implemented to tackle real-world issues. The dependability is associated with the flexibility of the system-ability to operate with same accuracy under different lighting conditions, changing orientation of the objects, mounting position and orientation of the vision camera and noise caused due to environmental changes.

3.1. OBJECTIVE

The objective of the system proposed in the section 5 is

- To measure the objects with highest level of accuracy
- Compare accuracy in different mounting conditions of web cam and with different references
- Provide dynamic calculation and to measure as many complex objects as possible

4. SYSTEM REQUIREMENTS

The video camera used for capturing the image is the only hardware requirement of the system. We are using a Canyon 10XSUPP webcam for our purpose. It has a built-in lamp and microphone with a suction base for easy mounting. This particular webcam uses CMOS sensor technology and has a lens of 1.3 mega pixels. The maximum video resolution output is 1280*1024 with maximum frame capture rate of 15 fps. It has a USB 2.0 and a type A 4-pin for connecting to desktops. Dimensions of the camera are 78*245*68 with all measured in millimetres. The nominal weight of the webcam is 0.158kg (CNR-WCAM613G, n.d.).

The entire code of this project is written in Visual Studio 2019 IDE using C++. OpenCV package was utilised in implementing the image related transformations. OpenCV (Open Source Computer Vision Library) is a free software library for computer vision and machine learning. OpenCV was created to provide a common infrastructure for computer vision applications and to help companies incorporate machine perception more quickly. OpenCV is a BSD-licensed product, meaning its open-source making it simple for businesses to use and change the code.

Artificial lamps might be necessary if the test region is a lowly lit area. This experiment is done with four light sources which shines light on the platform from all directions.

5. DESCRIPTION OF THE SYSTEM

The following section gives the detailed view of how the system works in order to provide us with accurate measurement. The main approach of this design to be as flexible as possible regardless of the distance of camera from object, lighting conditions and measurement of the reference used.

5.1. IMAGE FILTERING

A web-cam is setup for capturing data for real-time measurement. An object of this instance is created and frame information is stored in a matrix. An error message is displayed if the frame is empty. It is fairly simple to obtain the pixel measurements of the objects in question by using the tools provided by OpenCV library package. But it is the reference against which it is calculated that gives the metric value of the objects. Before fixing the reference for relative measurement, the frames are pre-processed using various filters for feature extraction. The filtering methods applied are conversion of BGR to greyscale, Gaussian blur, edge detection, dilate and erode. Blurring is done with kernel size of five is performed to remove noise from the image and canny edge detector is used to obtain features. Dilate is used to join the features that are not a closed figure as a result of lighting conditions and subsequent erode operations gives us with only required features. The parameters of these functions can be varied to suit the lighting conditions the program is running under.

5.2. REFERENCE MEASUREMENT

The system uses the platform used for measuring the objects as the reference. The actual measurements of the platform used for measuring are entered prior to getting frames from video camera. All the contours available in the image are obtained from the processed frame using findContours function available in OpenCV package. Since the platform on which the objects are being kept will be detected as the largest contour of the frame, a for loop which iterates over all the detected contours along with a nested if-else loop is used to get the index of the largest contour found in the image. A minimum area rectangle is drawn for this contour and the width and height features of this rectangle is used to find the pixels per inch value of the frame for both x and y scale. Our reference object has now server two purpose of acting as the platform for objects to be kept and also as the medium to convert pixel information to actual dimensions. The reason for adapting this method is to improve the scalability of the objects that can be measured with this program and also to eliminate the video set-up constraints such as height, angle etc. The only requirement for the program to work is that the frames should cover all of the platform completely, it should be the largest contour in the frame and the user should know its measurement as accurately as possible. Alternatively, this system would also work if the user knows the measurement of the largest contour in the frame, regardless of it acting as the platform or not. (Fig-4)

5.3. SHAPE DETECTION

The next step after this is to measure the objects present in the frame. The word objects and contours are used interchangeably since objects are represented as contours after feature extraction. Shape detection is performed to recognise the object’s shape. By using approxPolyDP function which gives the minimum number of points required to define the contour, we can identify the shape. If size of the output from the function is three, then the shape is triangle, if it is four then rectangle or a square and the if the number of points is above four, it can be anything between higher order polygons or a circle.

5.4. OBJECT MEASUREMENT

Depending on the shape detected, certain instructions are provided to display their measurements. The length of all sides are displayed if triangle is detected, width and height are shown if rectangle is detected and radius is displayed for a circle. The methodology explained so far can be used to measure objects dynamically and the parameters of the system such as accuracy, time taken for measurement, reliability under different mounting heights etc is discussed in the next section of the report. Despite experimentation with mounting setups, there are few certain conditions that has to be met with the set up before calculating the fillet radii. The setup needs to be properly setup for getting accurate measurement of the fillet radii. The code has the algorithm to detect the radii accurately but it will ultimately depend on the set up. The orientation of the platform and the object should be either perfectly horizontal or vertical. The camera should be mounted in such a way that skewness is eliminated completely.

6. EVIDENCE OF EXPERIMENTATION

The experiment was conducted with so many different setups to test the versatility of the system under different conditions. The main objective of the developed system is to improve scalability of the system and perform the required task with same level of accuracy.

There are 2 types of parameters varied in order to check the flexibility of the system. The objects to be measured will be of different size from a coin to a chessboard etc. The objective of the proposed system is to be flexible enough to measure both of the objects with same level of accuracy. The mounting height of the webcam will vary to accommodate the objects depending on its size.

The experimentation was conducted on three different heights which are shown in the figures 1-3. The different mounting stations allow us to study scalability of the system.

The platforms on which objects are measured can be varied as the mounting height differs can also be changed according to our convenience. The variables for this part of the experimentation were the dimension of the platform used as reference and the orientation with which it is used. Both horizontal and vertical orientations of platform was tried as part of the experiment for single height setup. The dimensions of the platform was varied to fit into the focus of the camera when height was varied. The reliability of system under different conditions was validated by comparing the results provided for different reference scale. The results are provided in the next section which discusses the results obtained and the efficiency of the system.

To capture the dynamic capability and the time taken to calculated the dimension, a draw test was conducted on our system. Shapes were hand drawn in our platform which is a cut A4 sheet and they were measured by the model. The values are noted down and is compared against the actual values which are obtained after measurement with proper instruments.

The system was also tested against a completely new system to which the filtering or any operation is not optimised for in any whatsoever. This is done to identify the changes required to implement to the code for importability.

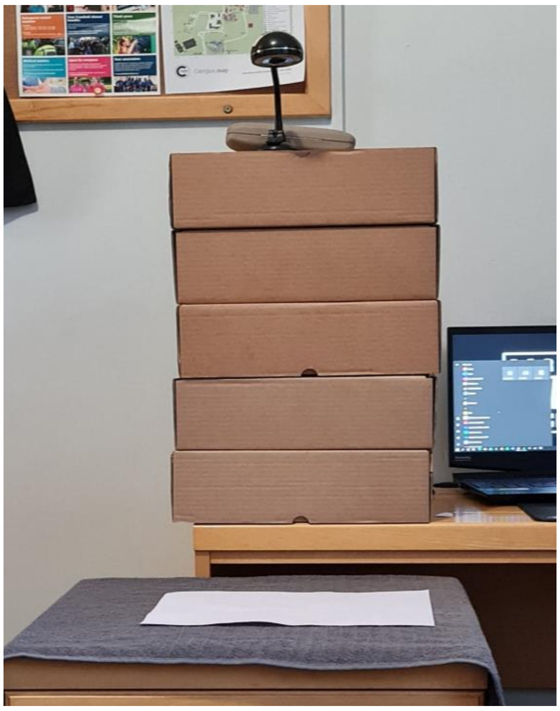

Figure 1 : Set up: Max height

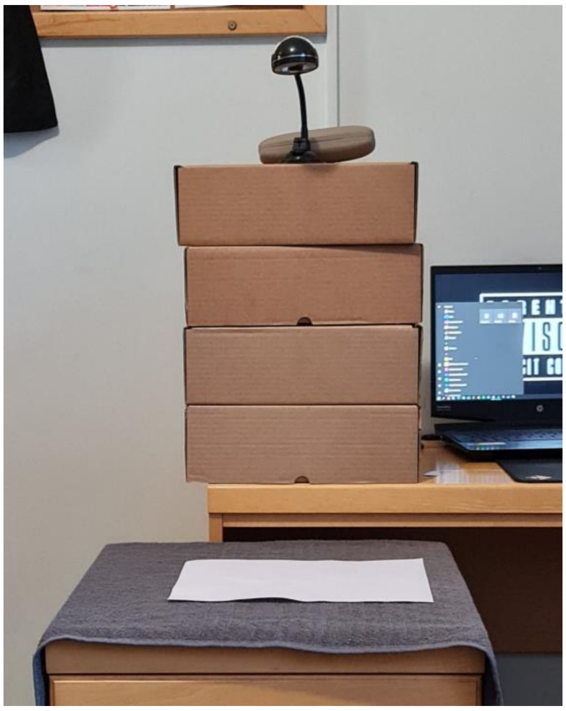

Figure 2 : Setup: Med height

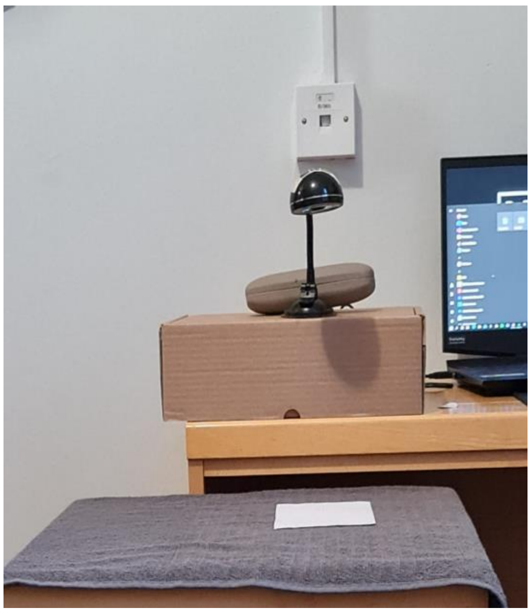

Figure 3 : Setup: Low height

7. SYSTEM ASSESSMENT AND ANALYSIS

The results of the experimentation conducted is given here. Before that, the measurements that can be measured and the methodology behind it is explained. For triangle, the dimensions of all the three sides are displayed to the user. The dimensions of the minimum area rectangle that captures the rectangle or square objects perfectly was obtained and their actual dimension after applying scaling factor is displayed. Little geometry was involved in getting the dimension of the circle. The bounding rectangle captures the outlines of circles in the frames and its diagonal is calculated. The radius of the circle can be obtained by applying the relation, radius=diagonal/sqrt(8).

The system is assessed by comparing the dimensions obtained from the system versus actual measurement obtained after measurement with appropriate tools. The actual measurements of the objects that are used are given in the table below.

Object	Width(mm)	Height(mm)
Card-1(Matte Finish)	54	85
Card-2 (Glossy Finish)	54	85
Card-3(Platform colour)	53	85
Rectangle Sheet	59	80
Pound Coin	Radius	11.5
2-Pound Coin	Radius	14

Table 1 : Actual dimensions of Objects

The objective of using 2 cards of similar dimension is to check the surface properties of the objects and its effect of the lighting conditions present. The glossy finish reflects a lot more light as compared to the others which can affect the system during feature extraction process. Another card which has the similar colour as the platform is used to prove the concept that the reference should always be the biggest contour available in the frame and not necessarily the platform. Rectangle sheet of different dimension was given for variance. To check for accuracy of circular objects, a pound and two pound coin was used.

The trial of dimension measurement was done for 10 times with the same setup was done and its mean value was obtained. They are compared against actual values to calculate the accuracy of the system. The values are taken at different time intervals with different lighting conditions to simulate uncertainty to the system. The mean values obtained for different objects are given in table-2 and the accuracy of the system is measured against it.

Object	Mean Width (mm)	Mean Height (mm)	Error Width (in %)	Error Height (in %)
Card-1(Matte Finish)	54.66	85	1.2	0
Card-2(Glossy finish)	52.75	87	2.3	2.3
Card-3(Platform colour)	51.5	83.5	2.8	1.7
Rectangle Sheet	57	79	3.3	1.2
Pound Coin	Radius	11.2	Radius Error	2.6
2-Pound Coin	Radius	13.15	Radius Error	6

Table 2 : Error Percentage of measurement calculation

The above table has given enough proof that the accuracy of the system is high with maximum deviation of 2.8 error percentage.

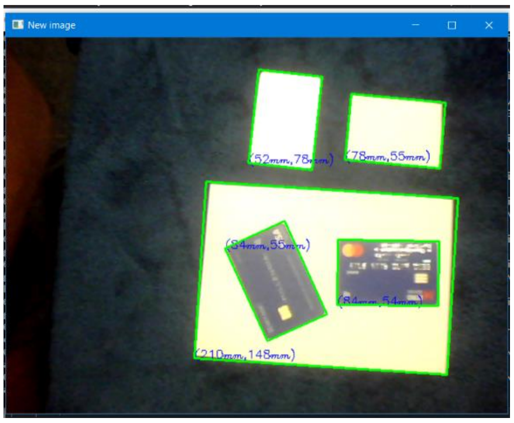

Figure 4 : Set up-1

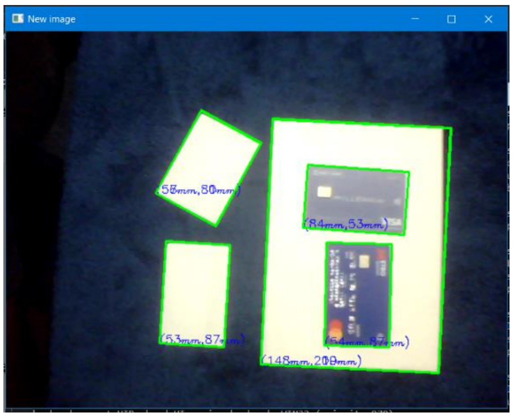

Figure 5 : Set up-2

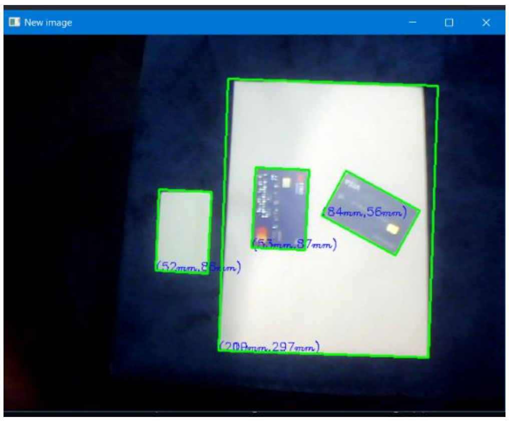

Figure 6 : Set up-3

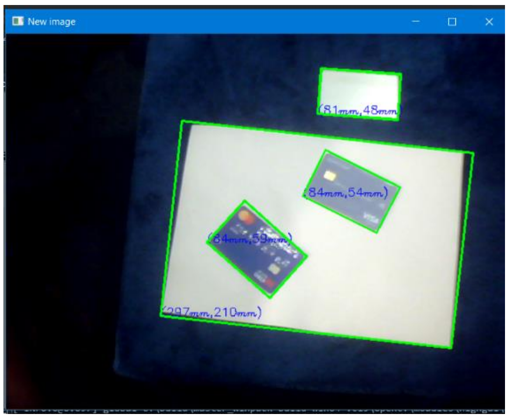

Figure 7 : Set up-4

The measurements of the cards were checked in 4 different setups. Set up 1 and 2 uses a standard A4 sheet as a reference in two different orientations and set up 3 and 4 uses half A4 sheet as reference in horizontal and vertical orientation. The below table shows the values of dimensions obtained at different set up for comparison. The first 2 setups are designed to scale bigger objects and next two for medium sized objects.

Dimension	Set-up 1	Set-up 2	Set-up 3	Set-up 4
Card-1 Width	53	55	54	56
Card-1 Height	84	84	84	84
Card-2 Width	54	54	53	59
Card-2 Height	84	87	87	84
RS Width	57	56	61	57
RS Height	82	79	82	81

Table 3 : Dimension Comparison in different setup

Two setups are designed for finding the dimensions of smaller scale objects like coins etc. The setups use a platform of white sheet of dimensions 59*80 millimetres.

Dimension	Set-up 1	Set-up 2
Pound coin Radius	10	10
2-Pound coin Radius	13	13

Table 4 : Radii Comparison in multiple Setup

The experimentations for the result shown in table-4 was done with extreme lighting condition where the dimensions are affected by glare on the coin.

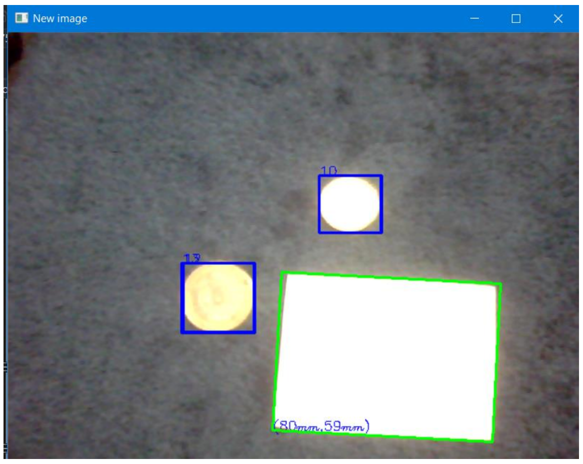

Figure 8 : Small Scale Set up-1

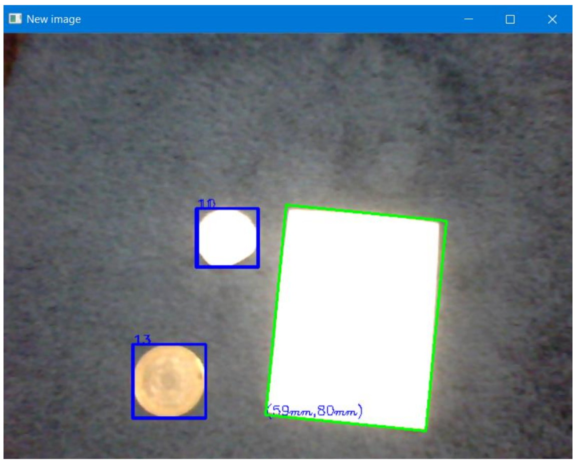

Figure 9 : Small Scale Set up-2

In order to properly check the dynamic capabilities, the system was tested by drawing multiple figures on the platform and obtaining the measurement. These shapes were them measured with proper tools and values were compared.

Shape Drawn	Measured Value	Actual Value
Circle	Radii=6 mm	Radii=6 mm
Rectangle	w=12 mm, h=9 mm	w=11 mm, h=9 mm

7.1. BIAS CORRECTION

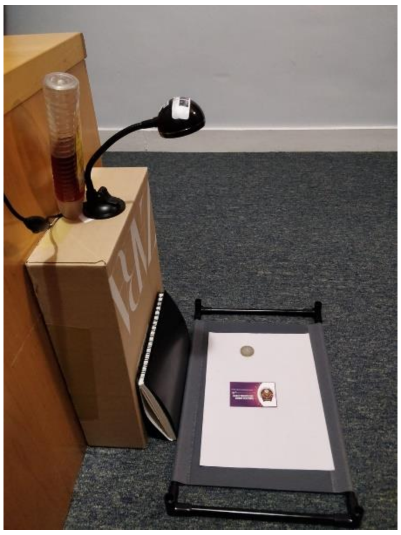

Figure 10 : Setup used for Bias Checking

We cannot completely ascertain that a system works perfectly unless if it is tried on different systems. Here we can consider the experimentation tried on the different set up provided as the validation set of the system. But unless and until we have tried it out on a test, we cannot be sure of the optimisation done. The set up given in figure-10 was used as the test set. The results obtained after running the program once is summarised as part of the study.

The test set up was located in a well lit environment with light intensity higher than what the program was optimised for. Detection of contours were difficult on glossy objects. However on the objects that was detected, the program was able to give accurate dimension. This has proven than the existing program can be imported with minimal parameter changes to work in a different environment.

8. RESULTS AND FUTURE WORK

The study was able to provide a competent system that can measure the objects dynamically under various stress conditions. The system was assessed under various parameters and its results are provided. The feasibility of the system to measure was studied as part of extreme tests provided in evidence of experimentation. The drawbacks of the system was lighting conditions that impeded the creation of complete contours of the figure. Dynamic change of filtering properties applied on the image would be the future study required for implementation of the system in various sectors. Research on camera and pixel density and ability to detect small objects even when mounted at great height. A thresholding study should be done to nullify the noise produced in the background which would improve the ease of use.

9. REFERENCES

[1] (n.d.). Retrieved from OpenCV: https://opencv.org/

[2] CNR-WCAM613G. (n.d.). *Webcam CANYON*. Retrieved from Gaming.Canyon.eu: https://canyon-ru.translate.goog/product/cnr-wcam613g/?_x_tr_sl=ru&_x_tr_tl=en&_x_tr_hl=en&_x_tr_pto=sc

[3] Fernandes, A. O., Moreira, L. F., & Mata, J. M. (2011). Machine vision applications and development aspects. *9th IEEE International Conference on Control and Automation (ICCA).*

10. APPENDIX

```
/***********OBJECT MEASURMENT***********/
/* This program is written to calculate*/
/* the dimensions of various objects.  */
/***************************************/




//The header files required for the
//functions used are given below

#include <opencv2/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <opencv2/imgcodecs.hpp>
#include <math.h>

using namespace std;
using namespace cv;

/********************This is the main part of the program*************************/
/*The following function is responsible for getting image input from vision system,*/
/*perform image filter operations, shape detection and finally measurment operation*/
/********************************************************************************/

int main(int argc, char** argv)
{
    // Create an object for video capturing
    VideoCapture cap;

    // Give access of webcam to the video object and give it scaling constraints
    cap.open(1);
    cap.set(3, 1920);
    cap.set(4, 1080);

    // Initialise all the necessary variable for the program to work
    Mat img, img1;
    float area, peri;
    double valuex, valuey, fillet;
    int frames = 1000;
    int key;

    // Introduction about the program for the user
    cout << "*********************WELCOME*************************\n";
    cout << "This system measures the dimensions of the obejcts\n\n\n ";
    cout << "Please enter the dimensions of the platform used for measurement\n";

    cout << "NOTE: Make sure these measurements are accurate since they will be used as reference\n";
    cout << "Enter the width of your region of interest: ";
    cin >> valuex;
    // This value is used as reference in scaling of pixels in x-direction
    cout << "Enter the height of your region of interest: ";
    cin >> valuey;
    // This value is used as reference in scaling of pixels in y-direction
```

```cpp
    cout << "Do you want to find fillet radius (1 if yes/ 0 if no)";
    cin >> fillet;
    //check with user if conditions are met for fillet area

    // checking of the videocapture object created is working and connected to webcam
    if (!cap.isOpened())
    {
        std::cout << "error: could not grab a frame" << std::endl;
        exit(0);
        // Error meassage is sent and program is closed if connection is absent
    }


    for (int i = 0; i < frames; i++)
    {
        // Image is captured and stored in a Matrix
        cap >> img;
        img1 = img;

        // The Following steps performs image filtering operations for feature extraction

        cvtColor(img, img, COLOR_BGR2GRAY); //color conversion from bgr to greyscale
        GaussianBlur(img, img, Size(5, 5), 1);  //blurring to remove noise from the image
        Canny(img, img, 100, 100);  //edge detection to identify the features
        dilate(img, img, Mat(), Point(-1, 1), 2); //used to create closed figures from features
        erode(img, img, Mat(), Point(-1, 1), 2);  //erode the extra thickness

        imshow("Image", img);

        key = waitKey(50);
        if (key == 'c') //An hotkey is provided to quit the program anytime the user wants to
        {
            exit(0);
        }

        //Initialising variable to identify the platform region
        vector<vector<Point> > contours;
        vector<Vec4i> hierarchy;
        int max_index{}, maxarea = 0;
        float scale_x,scale_y;

        //finds all the contours in the image and returns them along with heirarchy
        findContours(img, contours, hierarchy, RETR_TREE, CHAIN_APPROX_SIMPLE);

        //Condition to go to next frame when no contours are detected
        if (contours.size() == 0)
            continue;

        //This part loops through all the contours in the image and gives the index of maxmimum area contour
        for (int i = 0; i < contours.size(); i++)
        {
            int area = contourArea(contours[i]); //finds the area of contour

            if (area > maxarea) //compares the area with previous contours
            {
                max_index = i; //stores the index of contour with max area
                maxarea = area;
            }
        }

        //The properties of platform is obtained in this part
        RotatedRect minRect_max = minAreaRect(contours[max_index]);   //finds the minimum area rectangle that covers the contour

        //the ppi value of the current frame is updated to the system
        scale_x = valuex / minRect_max.size.width; //x-scaling of the image
        scale_y = valuey / minRect_max.size.height;//y-scaling of the image

        //Initialising variables to find the shape of the contours detected
        vector<vector<Point> > conpoly(contours.size());
        vector<Rect> boundRect(contours.size());
        int pts;

        //This loop is dedicated to find the shape and dimension of the objects
        for (int i = 0; i < contours.size(); i++)
        {
            int area = contourArea(contours[i]); //calculates the contour area

            // contours may be detected from noises which needs to be eliminated before proceeding
            if (area > 1000) //Only significant contours are considered after this
            {
                float peri = arcLength(contours[i], true); //calculates the perimeter of the contour
                approxPolyDP(contours[i], conpoly[i], 0.02 * peri, true); //approximates the contours to the shape of standard polygon
                pts = conpoly[i].size(); //determine the no of points required for approximation


                if (pts == 3) //Traingle objects can be detected with only 3 points
                {
                    // displays the lines that make up the triangle along with their dimensions
                    for (int j = 0; j < pts-1; j++)
                    {
                        line(img1, conpoly[i][j], conpoly[i][(j + 1) % 4], Scalar(0, 255, 0), 2); //function to draw a line
                        double res = scale_x*cv::norm(conpoly[i][j]-conpoly[i][(j + 1) % 4]); // calculates the distance between 2 points
                        putText(img1,to_string(int(res)), Point(conpoly[i][j].x, conpoly[i][j].y+20), FONT_HERSHEY_DUPLEX, 0.5, Scalar(255, 0, 0), 1); //enters text in image matrix
                    }
                }


                else if (pts == 4) //Rectangle or Square can be detected with 4 points
                {
                    RotatedRect minRect = minAreaRect(contours[i]); //detectes the minimum area rectangle for the contour
                    Point2f rect_points[4];
                    minRect.points(rect_points); //extracts the points that make up the rectangle

                    // the following lines calculates the dimensions of object in pixel value and converts it into actual dimensions
                    float width = minRect.size.width; //calculates the width of bounding rectangle
                    float height = minRect.size.height; //calculates the height of bounding rectangle
                    float real_width = scale_x * width; //actual width in millimeters
                    float real_height = scale_y * height; //actual height in millimeters


                    string r_width = to_string(int(real_width));
                    string r_height = to_string(int(real_height));

                    string part1 = "(";
                    string part2 = "mm,";
                    string part3 = "mm)";


                    string final = part1 + r_width + part2 + r_height + part3; //combining all the information to display
                    putText(img1, final, Point(rect_points[0].x, rect_points[0].y +20), FONT_HERSHEY_DUPLEX, 0.5, Scalar(255, 0, 0), 1);
                    putText(img1, "Quadilateral", Point(rect_points[0].x, rect_points[0].y + 40), FONT_HERSHEY_DUPLEX, 0.5, Scalar(255, 0, 0), 1);

                    // this loops prints the points as lines that make up the rectangle
                    for (int j = 0; j < pts; j++)
                    {
                        line(img1, rect_points[j], rect_points[(j + 1) % 4], Scalar(0, 255, 0), 2);

                    }

                    if (fillet == 1)
                    {
                        //Radii caluclation of the fillet
                        Point center = minRect.center; //center point of the rectangle
                        float fillet_radius, area_diff;
                        area_diff = (real_width * real_height) - (area * scale_x * scale_y); //area difference between shape and rectangle that defines it
                        fillet_radius = sqrt(area_diff / 0.8584); //fillet radius
                        putText(img1, to_string(fillet_radius), center, FONT_HERSHEY_DUPLEX, 0.5, Scalar(255, 0, 0), 1); //prints radii in center point
                    }


                }

                else if (pts > 6) //For circle and higher order polygons
                {
                    //get the rectangle that bounds the object and print them
                    boundRect[i] = boundingRect(conpoly[i]);
                    rectangle(img1, boundRect[i].tl(), boundRect[i].br(), Scalar(0, 255, 0), 2);

                    //Calculate the radius of the circle
                    double width = scale_x*cv::norm(boundRect[i].tl()-boundRect[i].br());
                    int radius = width / sqrt(8); //calculating the radius in actual values
                    string name = "radius:";
                    string final = name + to_string(radius);
                    putText(img1, final, Point(boundRect[i].tl().x, boundRect[i].tl().y-20), FONT_HERSHEY_DUPLEX, 0.5, Scalar(255, 0, 0), 1);

                }

            }
        }
        imshow("New image", img1); //This displays the image with the dimensions


    }
}
```