# Geometric Parameter Measurement of Optical Fiber by Image Processing

**Senior Design Project Report**


**Group Members:**

Muhammad Aasher Naeem (UW-18-MTS-BSc-012)

Muhammad Rana Areeb Sajid (UW-18-MTS-BSc-016)

**Supervisor:**

Dr. Shaukat Ali

**Department of Mechatronics Engineering**
**Wah Engineering College**
**University of Wah**
**2022**


# Geometric Parameter Measurement of Optical Fiber by Image Processing

**Senior Design Project Report**

Submitted to the Department of Mechatronics Engineering
In partial fulfillment of the requirements
For the degree of
Bachelor of Science in Mechatronics Engineering
2022

**Supervisor:**

Dr. Shaukat Ali

**Submitted by:**

**1**- Muhammad Aasher Naeem
UW-18-MTS-BSc-012

**2**- Muhammad Rana Areeb Sajid
UW-18-MTS-BSc-016

| Project Id | 02 | Number of Group Members | 02 |
|---|---|---|---|
| Project Title | Geometric Parameters Measurement of Optical Fiber by Image Processing | | |
| Supervisor | Dr. Shaukat Ali | | |

| Sr. No | Name & Registration No. | E-mail | Mobile Phone |
|---|---|---|---|
| 1 | Muhammad Aasher Naeem (UW-18-MTS-BSc-012) | UW-18-MTS-BSc-12@wecuw.edu.pk | +92 333 1562227 |
| 2 | Muhammad Rana Areeb Sajid (UW-18-MTS-BSc-016) | UW-18-MTS-BSc-16@wecuw.edu.pk | +92 302 8673727 |

_____ _____

Dr. Shaukat Ali Muhammad Aasher Naeem

_____

Muhammad Rana Areeb Sajid

**Checklist**

- Number of pages attached with report ____.
- We have enclosed the softcopy of this report along script created by our self.
- Our supervisor has attested the attach document.
- We confirm to state that this project is free from any type of plagiarism and misuse of copyright material.

**Department of Mechatronics Engineering**
**Wah Engineering College, University of Wah,**
**Wah Cantt**

# UNDERTAKING

It is certified that project work with title "Geometric Parameters Measurement of Optical Fiber by Image Processing" is done by our own. No portion of the work in this dissertation has been submitted in support of another award or qualification at this institution, where material has been used from other sources it has been properly acknowledged.

**Name:** Muhammad Aasher Naeem

**Registration #:** UW-18-MTS-BSc-012

| **Signature:** | | |
|---|---|---|
| **Date:** | | |

**Name:** Muhammad Rana Areeb Sajid

**Registration #:** UW-18-MTS-BSc-016

| **Signature:** | | |
|---|---|---|
| **Date:** | | |

# CERTIFICATE OF APPROVAL

It is certified that I have checked the project "**Geometric Parameter Measurement of Optical Fiber by Image Processing**" presented and demonstrated by:

Muhammad Aasher Naeem (UW-18-MTS-BSc-012)

Muhammad Rana Areeb Sajid (UW-18-MTS-BSc-016)

and approved it.

_____

Dr. Shaukat Ali

**Department of Mechatronics Engineering**
**Wah Engineering College, University of Wah**
**Wah Cantt, Pakistan**

# ACKNOWLEDGEMENT

Start with the Name of Allah, the Most Beneficent, the Most Merciful. We would like to give special thanks all of those who have given us the chance to complete this project. We are extremely thankful to our FYP supervisor, Dr. Shaukat Ali who gave us continuous guidance, assistance, and inspiration to continue work efficiently and obtain promising goals. We would especially like to thanks our family for their infinite support and encouragement. We really appreciate the guidance provided by other faculty members of Mechatronics Engineering Department to strengthen our technical and presentation skills through their feedback and advice.

# ABSTRACT

The fundamental aim of the project is to develop an Algorithm to measure the geometric parameters of Optical Fiber which includes cladding diameter of fiber and counting number of cores in a fiber by utilizing Image Processing technique. Python language is used for the development of the algorithm. The Optical Fiber is placed in a light controlled environment with a reference object, here its image will be taken using a 2D COMS technology camera. The lightning environment is used to remove unnecessary noise in our image. After taking the image the steps performed in our computer's Jupyter Notebook which is used as our python environment are following: image acquisition, filter out image noise, edge detection, extraction of contours in our image, sorting the contours in such a way that it can detect our reference object which is placed in the left most location and measure the parameters of our fiber accordingly. Experiment results show that algorithm accuracy meets the needs.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1 – INTRODUCTION AND LITERATURE REVIEW

## 1.1 Introduction:

### 1.1.1 Inspection:

An inspection is a process that involves measuring, examining, testing, or gauging one or more qualities of a product and comparing the findings to stated requirements in order to determine if that characteristic is conformed to[1]. The quality inspection is performed in two categories:

1. Manual Inspection
2. Machine Vision

Manual Inspection:

Manual inspection is the practice of visually inspecting units during assembly using human operators who follow a standard operating procedure (SOP). Human inspectors frequently use their eyes or a microscope to evaluate specific regions for quality.

Machine Vision:

Machine vision systems are a collection of technologies that undertake image-based inspections and measurements, including sensors, microscopes, and cameras. A machine vision system is a piece of software that allows a computer to inspect, evaluate, and recognize static or moving images. It is the combination of image processing hardware and software specially used for the quality inspection of an object.

### 1.1.2 Fiber Optics:

Optical fibre is a data transmission device that uses light pulses that travel down a long fibre, which is commonly constructed of plastic or glass[1]. The fibres are designed to aid in the propagation of light together with the optical fibre, depending on the power and transmission distance requirements.

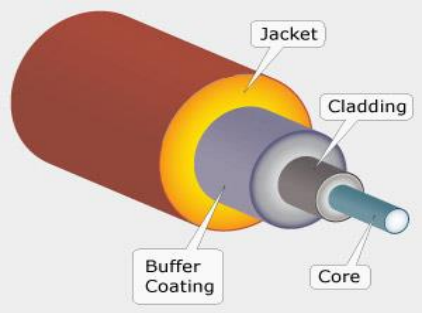



*Figure 1.1 Fiber Optics*

Optical fiber is classified as following:

- Material used (Glass & Plastic)
- Refractive index (Step index Fibers & Graded Index Fibers)
- Mode (Single mode & Multi mode)

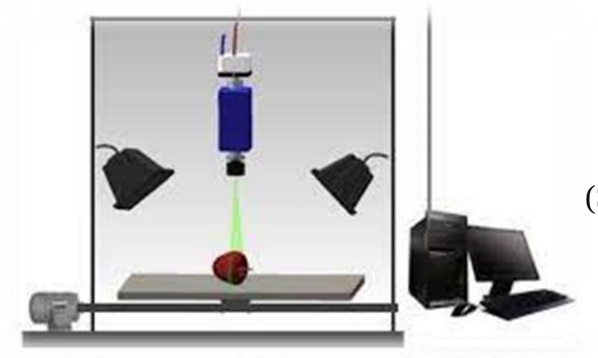
## 1.2 Project Motivation:

Industries all around the world have shifted several parts of their Quality inspection from being manual to being automatic. In industry optical fiber is manufactured and gauged manually. The Fiber optics is used for high-performance data networking over long distance. In inspection cladding diameter and number of cores are gauged manually which's time consuming. It is also very difficult to measure manually because of their small size. So, to calculate these geometric parameters automatically research is being taken to develop a machine vision system which is used to measure and count its required parameters in less amount of time without any errors.

## 1.3 Practical Examples:

### 1.3.1 Automatic Food Inspection:

Food Inspection is quite important but manual inspection is time consuming process. The researchers created a computer vision-based system to assess the quality of vegetables and fruits. as shown in Figure 1.2. Shape difference, size and fungal attacks on fruits and vegetables has observed in automatic inspection. The researchers noticed that the intensity of light had an effect on the outcomes. The long-time biological processes observed by the vision system that humans are able to observe due to limitation of visible electromagnetic spectrum[2].



(Source: https://www.fanuc.eu/repeatability)

*Figure 1.2 Automatic Food Inspection*

### 1.3.2 The Automobile Industry:

One of the first industries to use machine vision for quality control was this one. Hundreds of inspection locations are common in automotive production lines. In-process monitoring, individual part/component monitoring, and inspection at several crucial locations throughout the assembly line are among the inspection points. Manufactured parts/components must be checked to ensure that their dimensions are exact and that they are aligned properly for assembly[3]. Designing systems for the automotive sector presents numerous obstacles, including inspection of multiple aspects of a part, multiple complicated parts, and inspection of various materials. Inspection of many features such as surfaces, forms, holes, slots, bolts, and studs are required when assembling large things such as vehicle bodywork and their subassemblies. Each component could have its own set of capabilities. Consider the part shown in Figure 1.3. As you can see, there are a lot of things to look at. The number and size of the holes, for example, must be consistent. The overall shape and proportions must also be in accordance with the requirements. Only then can this part fit appropriately into the subassembly or assembly. Multiple inspection cycles, one for each feature, would be required to inspect each feature individually. This would be a lengthy procedure. It could be possible to create a more efficient, but more expensive, approach that employs many cameras to take multiple measurements at the same time.

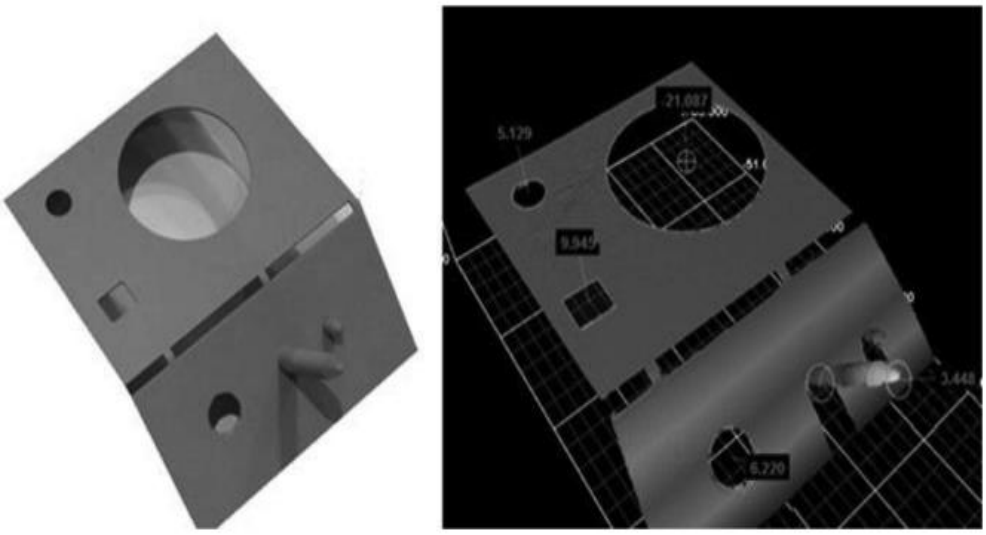


*Figure 1.3 Sample automobile parts*

(Source: QUALITY CONTROL IMP2.pdf)

### 1.3.3 Spur Gear Automatic Inspection System:

Gears are used most widely for power transmission applications. Proper gear inspection and quality maintenance is necessary. As manual process takes lot of time, the researchers presented their work for automation of the gear inspection process. Vision system had developed as shown in Figure 1.4, the system inspired the world to adopt automatic inspection rather than manual gauging processes. Camera placed at 90 degree that inspect gears according to the developed algorithm. The results of the gear inspection are amazing; the result time is much reduced as compared to manual inspection, and the accuracy is improved[4].

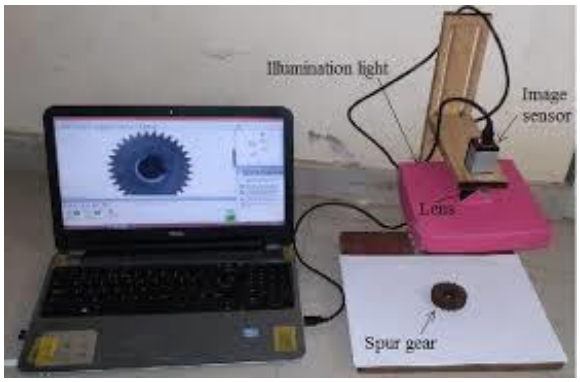



Figure 1.4 Spur Gear Automatic Inspection System

### 1.3.4 Shaft Parts Measurement System:

A computer vision-based system for measuring shaft parts without physical contact has been created[4]. It involves basic image processing step, filtering, edge detection, opening, closing and contouring. System consists of a data processing device i.e., computer and an imaging sensor for image acquisition as shown in Figure 1.5. Sample part placed on the workbench; the base of the system is illuminated with light to avoid noise as shown in Fig.7. This system shows repeatable error of 0.01mm.

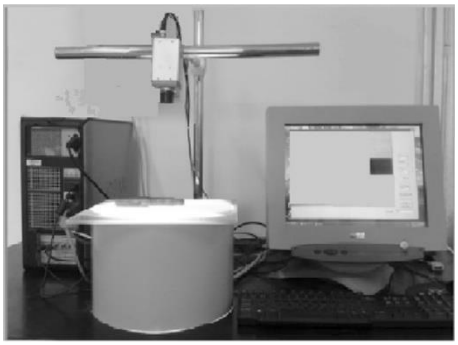


*Figure 1.5 Shaft Parts Measurement System*

### 1.3.5 Counting Calories in Meal:

In this study, researchers developed a mobile camera-based calories and nutrition measurement system for the diet conscious persons. Flow chart of the system shown in Figure 1.6, Person takes a picture of his/her food with the thumb touching the food. Thumb is used as reference to determine the area/volume of the food. SVM algorithm is used to train and identify the different foods placed in a plate. Calories and Nutritional values are obtained from national and international health organizations. Based on the calculated volume and nutrition values, researchers calculate the good results about the number of calories and nutrition in the food.

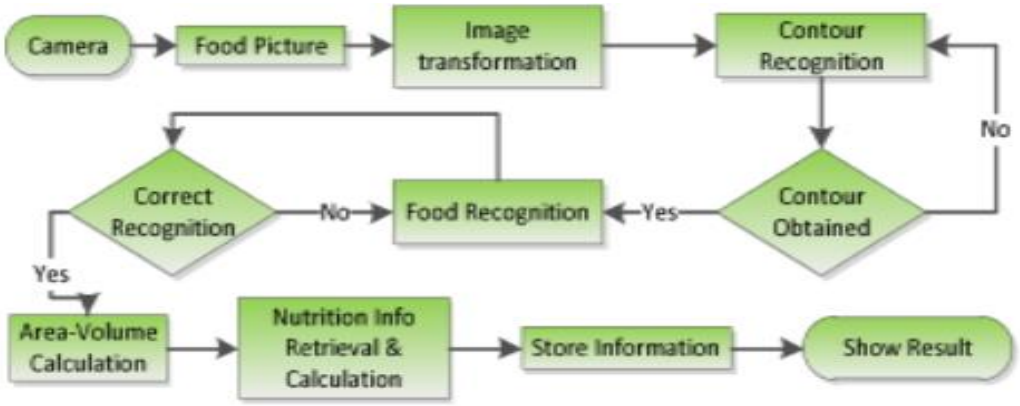



*Figure 1.6 System to count calories in Meal*

### 1.3.6 Automatic Coordinate Measurement Machine:

A coordinate measuring machine (CMM) measures the dimensions of a part by coordinate processing and recording technique. They are capable to automatically measure the target, store the processed data, obtain geometric dimension and tolerance measurements[5]. Automatic CMM has a built-in control system to operate the probe for acquiring the data points of the part. Physical contact between the probe and part has made to obtain measurements as shown in Figure 1.7. So, this is time consuming process as compared to contactless measuring systems.

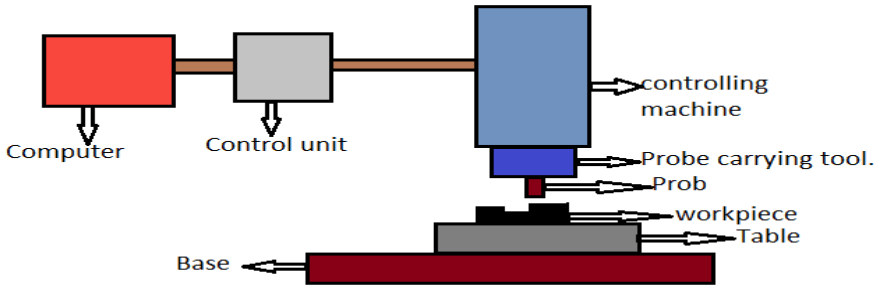



*Figure 1.7 Automatic Coordinate Measurement Machine*

### 1.3.7 Machine Vision based LED Inspection:

Most applications use Light Emitting Diodes (LEDs). LEDs are manufactured in bulk; the unavoidable defects in LED's are: missing components, orientation issues, inverse polarity problems, missing gold wires, mouse bites, and surface stains[5]. Due to their small size and bulk production, their manual inspection is quite time-consuming. Researchers developed a machine vision-based system to automatically detect the defects. Proto-type of a system in shown in Figure 1.8. The system is trained with the defective LED's images. During training each image has to pass through a set of the following operations: thresholding, closing, affine transformation, and inspection region specification as shown in Figure 1.8. Then the developed system that runs in real time, take the image of led as input, preprocess the image, segment the inspection region, then compared it with the trained data and display results. Experimentation shows 95% accuracy in detecting the defects[6].


*Figure 1.8 Machine Vision based LED inspection*

# CHAPTER 2 – METHODOLOGY

## 2.1 Introduction

In literature review, we have seen manual inspection system and machine vision system. For this research, we selected a computer vision system that enables computers to acquire images and retrieve information from them, just as humans do, in order to automatically inspect Geometrical parameters of Optical Fiber. It consists of two parts: experimental setup and software system.

The experimental setup contains:

- Optical Fiber Sample
- Image Acquisition Device (Camera)
- Controlled Environment.
- Data Processing Unit

The software system contains:

- Video Frames Extraction
- Frame/ Image Calibration Image Processing
- Image Processing
- Dimensions Measurement Algorithm

## 2.2 Optical Fiber Sample

This research aims to measure cladding diameter and the number of cores of the optical fiber. The test object is a small piece of optical fiber as a sample. It can have several measurable parameters, but here we are only concerned with two.

## 2.3 Image Acquisition Device

In this study, a CMOS technology camera is used as an image acquisition device. Camera Model used is Logitech C290e which take real world image and directly convert it into array of numbers.

The image is taken live using python in jupyter-lab for further processing of the acquired image of our fiber.

Logitech C290e dimensions (including fixed mounting clip):

Height: 43.3 mm,

Width: 94 mm,

Depth: 71 mm,

Cable length: 1.5 m,

Weight: 162 g.

Technical Specification:

Multiple resolutions

1080p/30fps (up to 1920 x 1080 pixels)

720p/30fps (up to 1280 x 720 pixels)

Camera mega pixel: 3

Focus type: Autofocus

Lens type: Glass

Built-in mic: Stereo

Mic range: Up to 1 m

Diagonal field of view: 78°

Digital zoom: 1.2 x

USB connectivity: USB-A plug-and-play

RightLight™ 2 technology for clarity in various lighting environments, even low light

Attachable privacy shutter

Image acquisition is a process of capturing real-world view and convert it into array of numbers that can be easily manipulated on computer as shown in Figure 2.1.

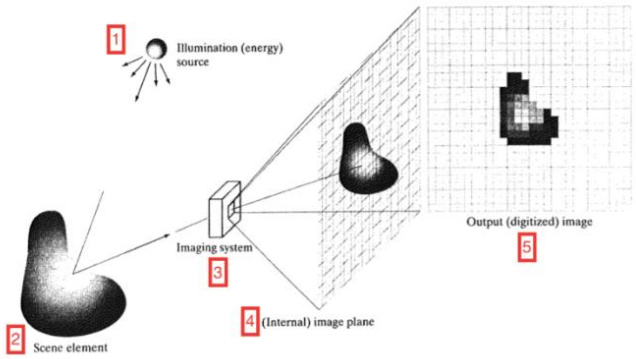



*Figure 2.9 Image Acquisition*

## 2.4 Experimental Setup:

In computer vision, controlled environment is required for efficient performance of the system. Excessive light and reflections can introduce noise, which can be harmful in the accuracy of the measurement systems[6]. Hardware setup of this research consists of a wooden box that is covered with white paper, shown in Figure 2.3. The purpose of the white paper is to provide a controlled environment i.e., free of dust particles and differentiate between object and background. Box dimensions are 31.50*21*11 cm, shown in Figure 2.4. A rectangular black object i.e., dimensions 1.0*1.6 cm is placed on the left side of the fiber to get reference for the measurements. A stand made inside the box is used to place camera and move it forward or backward as required is shown in Figure 2.2.

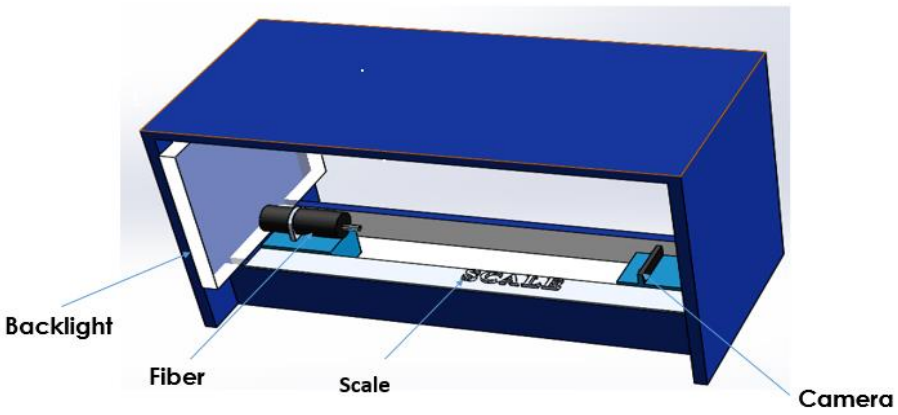



*Figure 10 Project 3D Design*

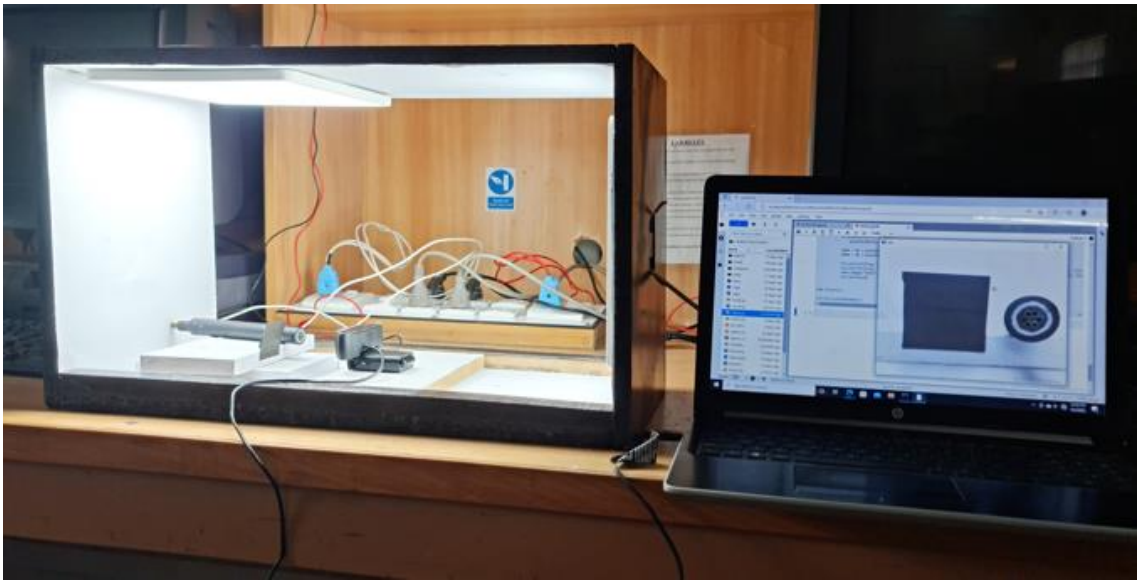


*Figure 11.3 Actual Project Image*

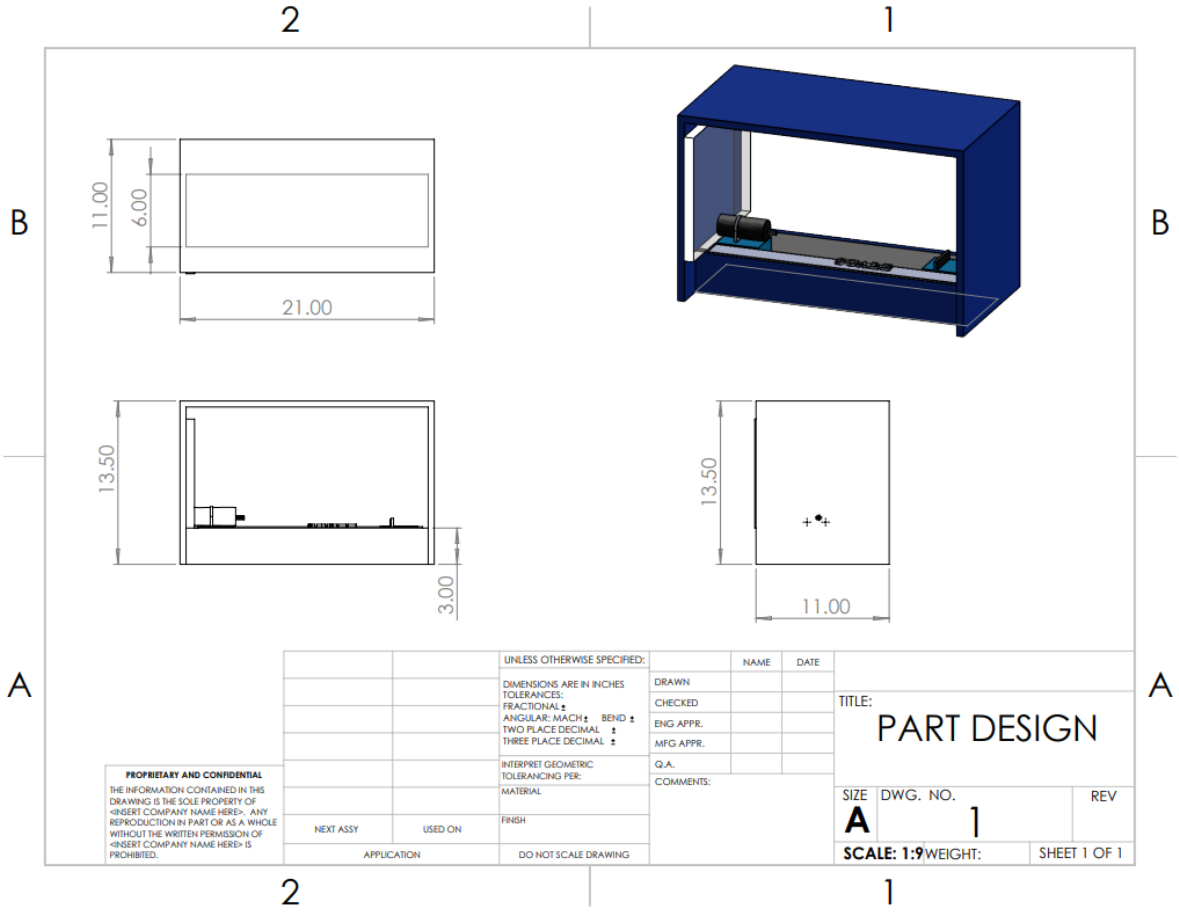



*Figure 12 Isometric Views*

## 2.5Lighting Controlled Environment:

Lighting controlled environment in our project represent the illumination techniques we used to remove noise i.e., shadow. The white paper is used to cover inner setup of our hardware. So, the illumination technique does not perform reflection in our image acquisition step. The two illumination techniques we used are following: Front Lighting & Back Lighting.

**Front Lighting**

In Front lighting, a ring light illuminates the object, nearly parallel to the optical axis of the camera. The image appears mottled and non-uniform.

*Figure 13 Front Lighting*

**Back Lighting**

When back lighting creates dark outlines against a bright background, it provides quick contrast. The most popular applications are detecting the presence or absence of holes and gaps, part placement and orientation, and object measurement. If precise (subpixel) edge detection is required, a monochromatic light, such as red, green, or blue, with light control polarization is frequently used.

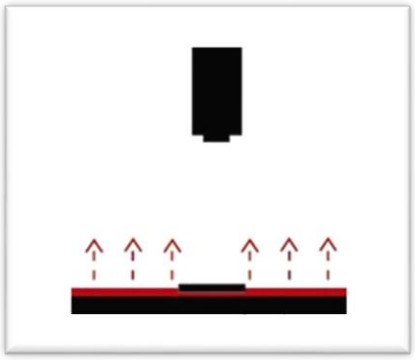


*Figure 14 Back lighting*

## 2.6 Requirements for Image Acquisition and Analysis

Following requirements are need to be fulfilled for experimental testing:

- Fiber sample should be at the right side because camera will detect reference object first which is placed on left side.
- Fiber sample should be placed parallel the reference object.
- Light should be good enough and shadow is not acceptable.
- Background other than the reference object should be white or close to white color.

## 2.7 Data Processing Unit

Personal laptop is used write algorithm for computer vision-based system. The laptop has following specification:

- Ram (4GB)
- Processor (intel CORE i3, 8th Generation)

System Requirements:

Compatibility

Windows 8 or later

macOS 10.10 or later

Chrome OS

USB-A port

## 2.8 Software/ Libraries for Algorithm Development

- Jupyter Notebook
- Python
- OpenCV
- NumPy
- Utilis

## 2.9 System Block Diagram

Data acquisition device needs to be calibrated once to eliminate errors that occur due to manufacturing flaws. After calibration, Image acquisition process takes place. Image processing functions are applied to remove noise, reduce size of image for fast processing, opening and closing to eliminate gaps and to get contour points. Then dimension measurement algorithm applied on contour points to calculate and display experimental results. System block diagram is shown in Figure 2.7 and detailed steps of block diagram are shown Figure 2.8.

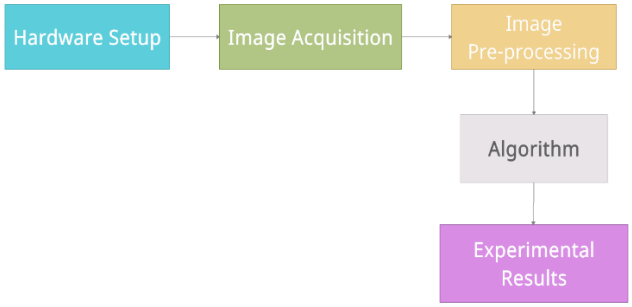



Figure 15 System Block Diagram

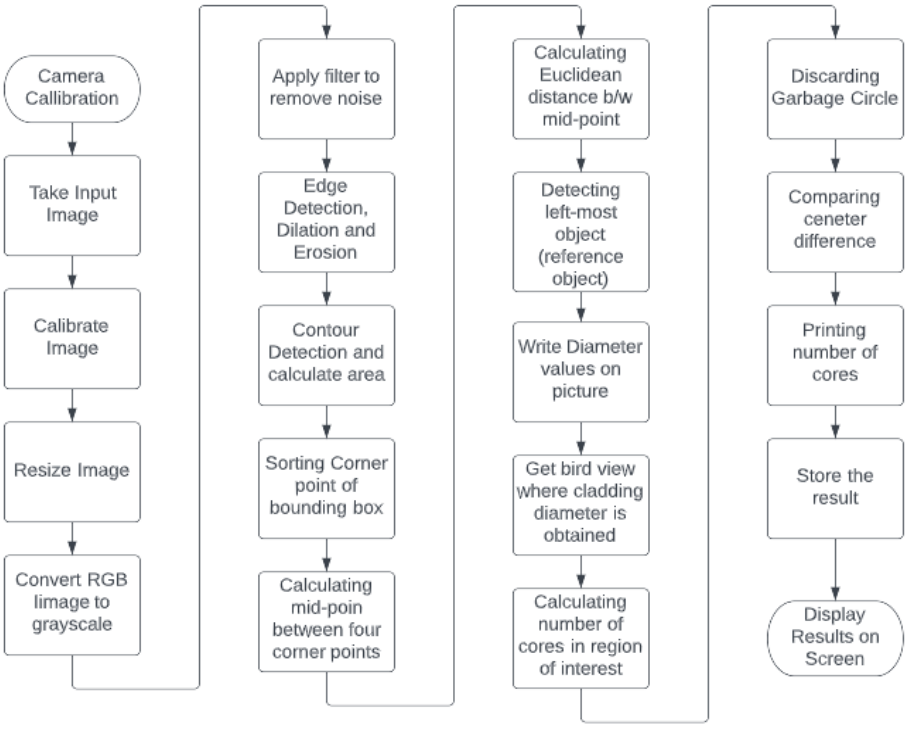



*Figure 16 Algorithm steps*

## 2.10 Camera Calibration

The process of identifying intrinsic and extrinsic camera parameters to characterize the mapping between 3D reference coordinates and 2D camera coordinates is known as geometric camera calibration[7]. These parameters help to remove the radial and tangential distortions of a camera. Tangential distortion occurs when the lens and camera sensor is not perfectly parallel as shown in Figure 2.9. Radial distortion makes the straight lines appear curved[8]. This error occurs due to lens spherical shape. It refracts the rays of light entered from the corner. The radial and tangential distortion results shown in Figure 2.10. Camera calibration parameters are used to remove distortions from an image. The camera calibration approach compensates for physical camera flaws in software.

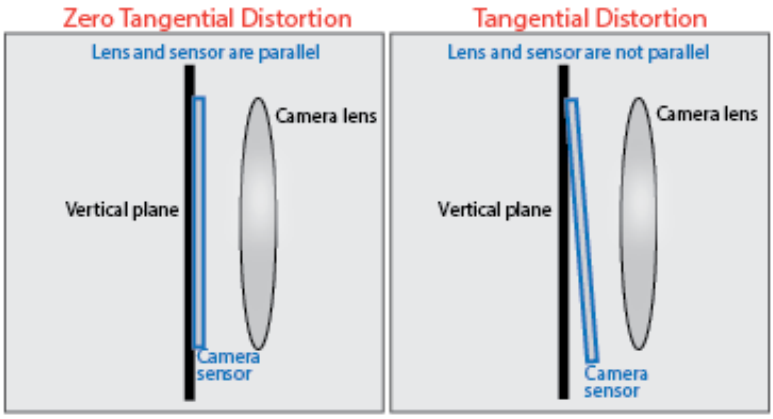



*Figure 17 Tangential distortion*

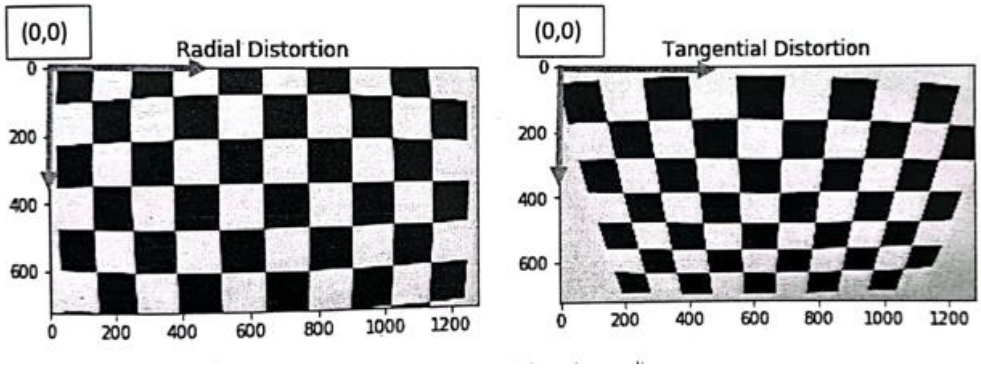



*Figure 18 radial and tangential distortion results*

## 2.11 MATLAB Camera Calibration Toolbox

Camera Calibration is done using MATLAB camera calibration toolbox. A 11x8 checkerboard pattern is used for camera calibration process with squares size equals to 25 mm as shown in Figure 2.11. Almost 30 images of pattern have taken from the camera at different angles. MATLAB toolbox processes these captured images and provides the required intrinsic and extrinsic parameters as shown in Figure 2.12.

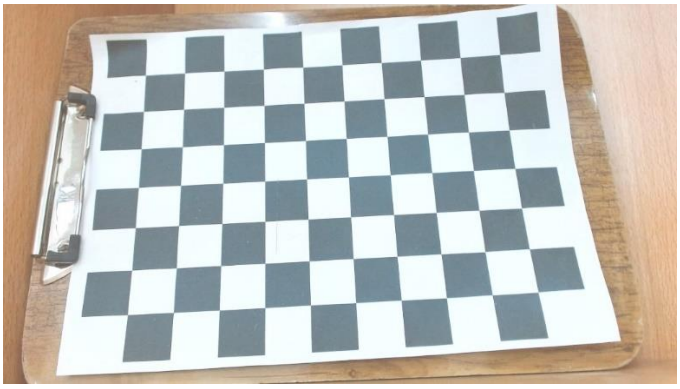


*Figure 19 11x8 checkerboard pattern*

```
Params =

  cameraParameters with properties:

   Camera Intrinsics
                    IntrinsicMatrix: [3×3 double]
                        FocalLength: [1.5486e+03 1.5517e+03]
                     PrincipalPoint: [994.1563 581.2879]
                               Skew: 0
                   RadialDistortion: [0.0607 -0.2225]
                TangentialDistortion: [0 0]
                          ImageSize: [1080 1920]

   Camera Extrinsics
                   RotationMatrices: [3×3×20 double]
                  TranslationVectors: [20×3 double]

   Accuracy of Estimation
              MeanReprojectionError: 0.3423
                 ReprojectionErrors: [28×2×20 double]
                  ReprojectedPoints: [28×2×20 double]
```

*Figure 202 Intrinsic and Extrinsic parameters*

### 2.11.1 Colored Frame Extraction and Conversion to Grayscale:

Video consists of 30 fps (frames per second). So first we extracted individual frames, and then applied intrinsic and extrinsic parameters for radial and tangential distortion corrections. Each frame is considered as single image of resolution 1920x1080p. The frame consisting of 3 colors BGR, was converted it to Grayscale (single color) in order to reduce the processing time and to save the disk space. OpenCV built-in function cv2.cvtColor is used for color space conversion. The result is displayed in Figure 2.13.

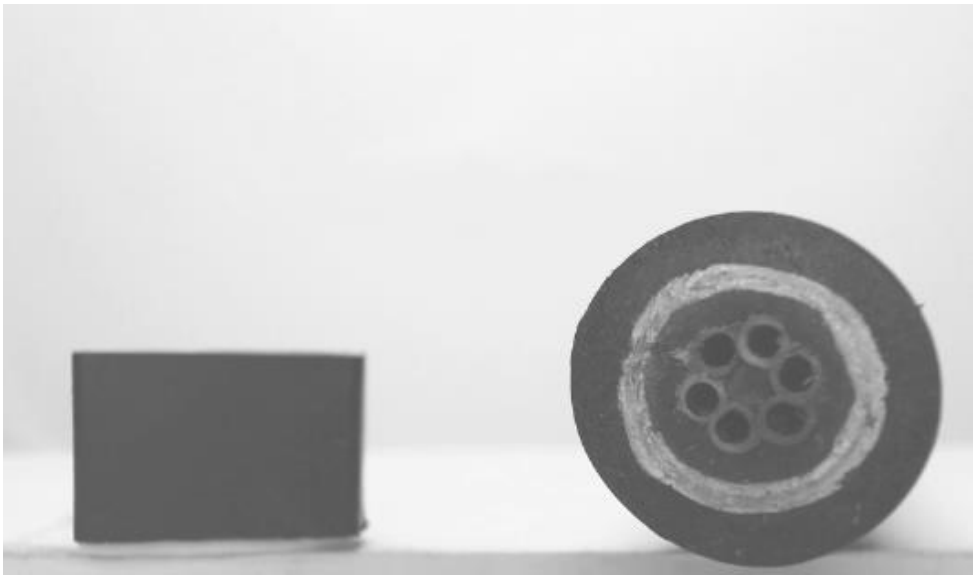


*Figure 21 Colored Frame Extraction and Conversion to Grayscale*

### 2.11.2 Image Thresholding:

The simplest thresholding method swap each pixel of the image with a black pixel if the image intensity $I_{i,j}$ is less than the fixed value called the threshold, or a white pixel if the pixel intensity is greater than that threshold. In some circumstances, though, the threshold should be selected manually by the user, there are cases where the user wants the threshold to be automatically set by an algorithm. If this is the case, the threshold should be the "best" threshold in the sense that the brighter things considered to be part of the foreground and the darker objects considered to be part of the background should be separated into two classes. Many types of automatic thresholding methods exist, the most widely used and famous being Otsu's method. In our case we are using TRUNC thresholding type with Otsu’s method in order to automate the threshold values.

THRESH_TRUNC
Python: cv.THRESH_TRUNC

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

The above representation shows if the source of our image is greater than the mentioned threshold than give the threshold value to the image otherwise set the source image value in our program. The threshold image of our optical fiber is shown in Figure 2.14.

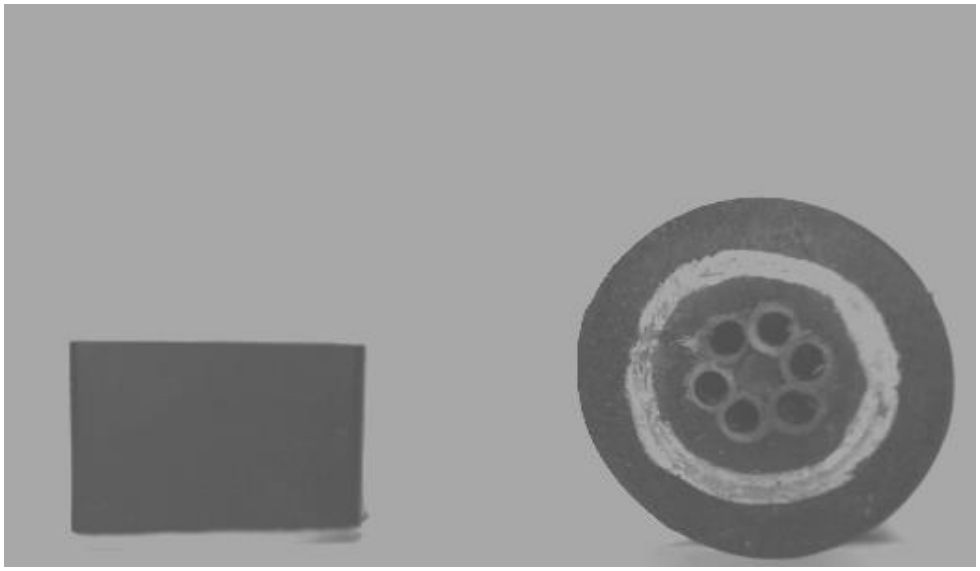


*Figure 22 Image Thresholding*

### 2.11.3 Filtration

A digital image is made up of pixels and has countable number of digital values. In processing, the important task is to remove the noise from the image. Filtration is the process for the removal of noise from the image. In this study, Gaussian filter, two-dimensional filter, has been used to filter the frames from live stream, controlled environment has reduced lot of noise but during shuttering process every time little noise was added in the captured frame. The type of noise is Gaussian noise, defined as the noise created by bad lightning or high temperature. Gaussian kernel has the from

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

Where x and y represent the horizontal and vertical distances of the kernel that convolve over the image. Convolution is the process to move a small window with specific values/ numbers, called kernel, over all the pixels of an image as shown in Figure 2.15. At every pixel, some mathematical operations have been performed that involves the neighbors of the concerned pixel as well for the sake of noise removal.

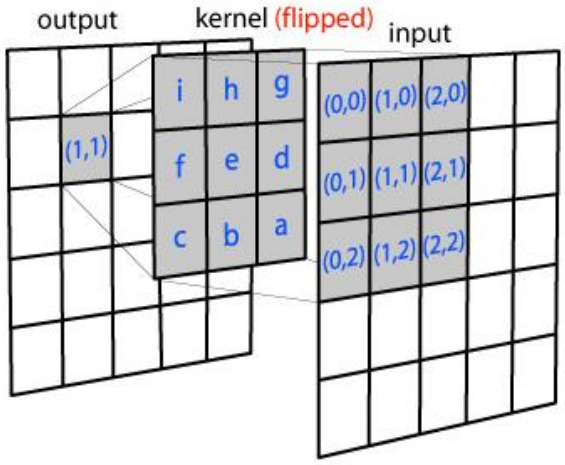



*Figure 23 Kernel process*

Gaussian Blur OpenCV function is used with sigma value '1' and 7x7 kernel for filtration process of Figure 2.14 and then result is shown in Figure 2.16.

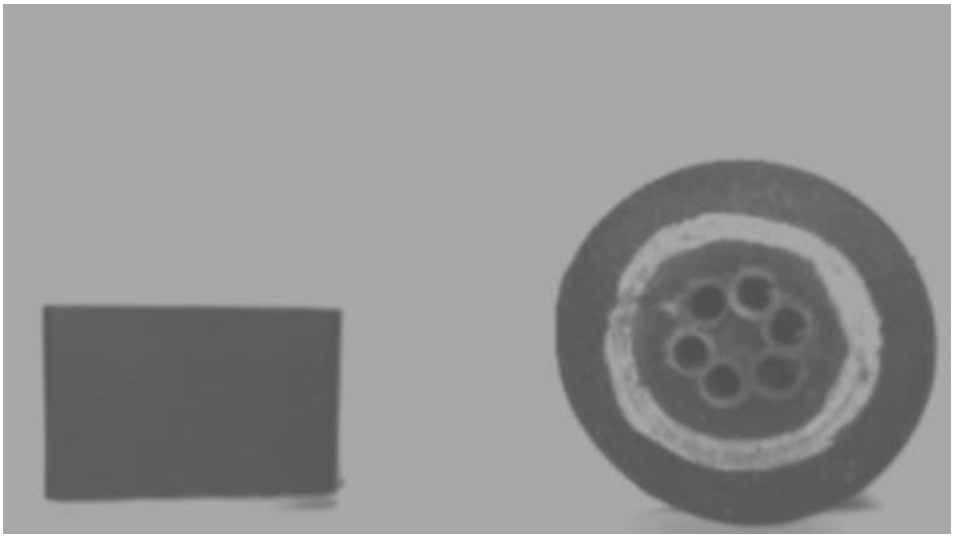


*Figure 24 Applied Gaussian Blur*

### 2.11.4 Edge Detection:

Edge detection encompasses a wide range of mathematical techniques aimed at detecting edges and curves in digital images where the image brightness abruptly changes or, more formally, has discontinuities. The same problem of finding discontinuities in one-dimensional signals is known as step detection and the problem of finding signal discontinuities over time is known as change detection. Edge detection is a crucial tool in image processing, machine vision, and computer vision, especially for feature recognition and extraction. he Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.

The Canny edge detection algorithm can be divided into five steps:

- First, apply Gaussian filter, it will smooth the image and remove the noise
- Second, Find the intensity gradients of image
- Third, to get rid of spurious response to edge detection, apply gradient magnitude thresholding or lower bound cut-off suppression
- Fourth, to determine potential edges, apply double threshold
- Lastly, Track edge by hysteresis: by suppressing all the other edges that are weak and not connected to strong edges, finalize the detection of edges.

The below images in Figure 2.17 shows the progression of a frog image through above five steps.

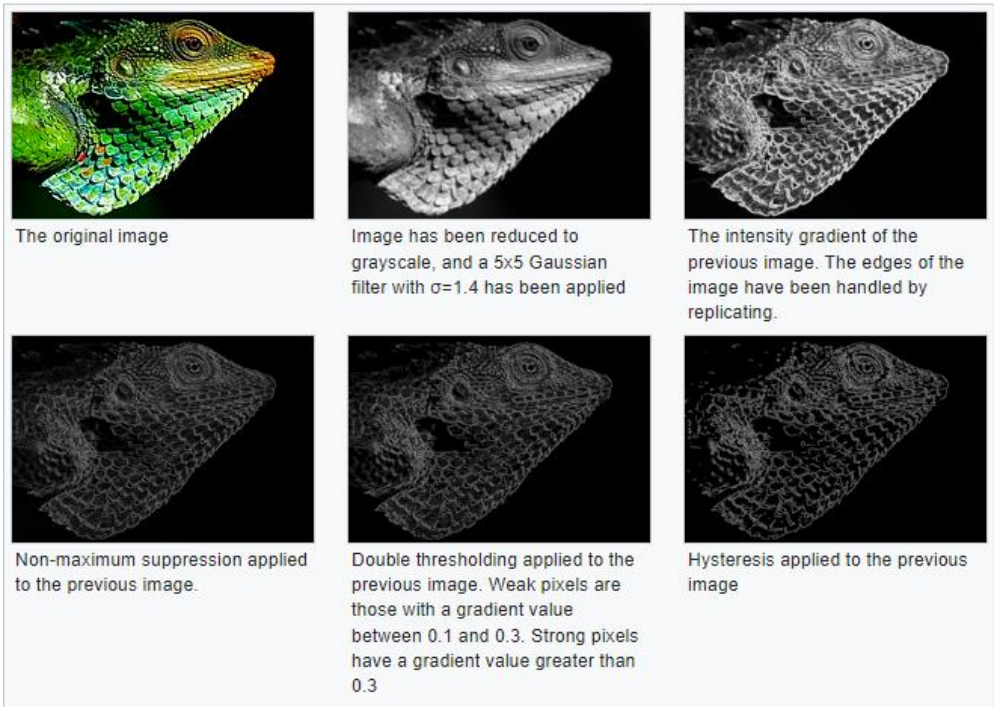



*Figure 25 Progression of Canny edge detection*

Technique that classifies and

highlights the abrupt discontinuities in digital image is known as edge detection. In this study, Canny edge detection is used to identify the edges. The algorithm is implemented with the help of OpenCV library. Canny edge detector separates the noise form the original image before tracking the edges. The algorithms steps are listed below:

- Detection of edge with low error rate, it means that the detection should accurately catch as many edges as possible shown in the image
- The edge point detected from the operator should accurately localize on the center of the edge.
- A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

Canny edge detector applied with threshold [40,40] to get the edge as shown on Figure 2.18.

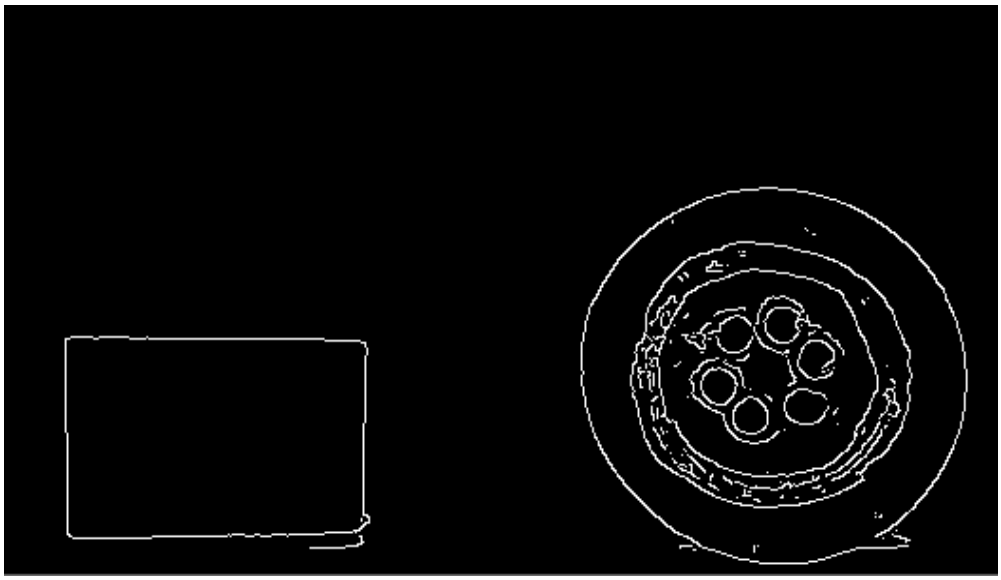


*Figure 26 Canny edge detector applied*

### 2.11.5 Erosion and Dilation

Image characteristics enhancement for the better analysis is necessary. After edge detection, binary image would be obtained, so it is possible that image contains few edges. that are very faint. To strengthen those and eliminate gaps, the dilation and erosion morphological operations are being used. Dilation adds pixels to the edges, while erosion removes pixels from the edges.

Two iterations of dilation with kernel 3x3 are implemented and then two iterations of erosion with kernel 3x3 implemented as shown in Figure 2.19 and Figure 2.20 respectively.

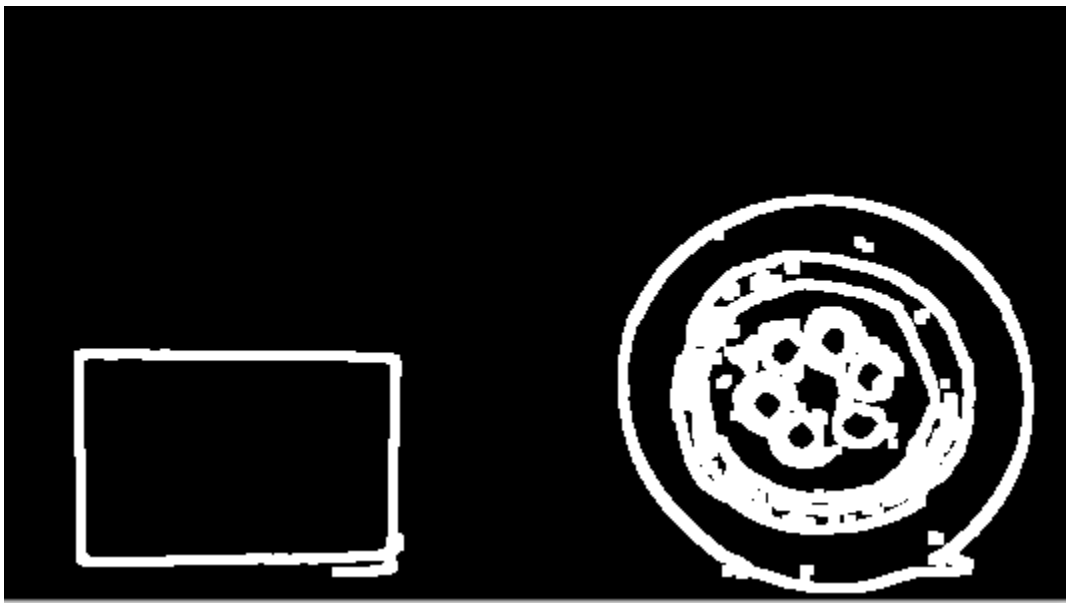


*Figure 27 Two iterations of Dilation performed*

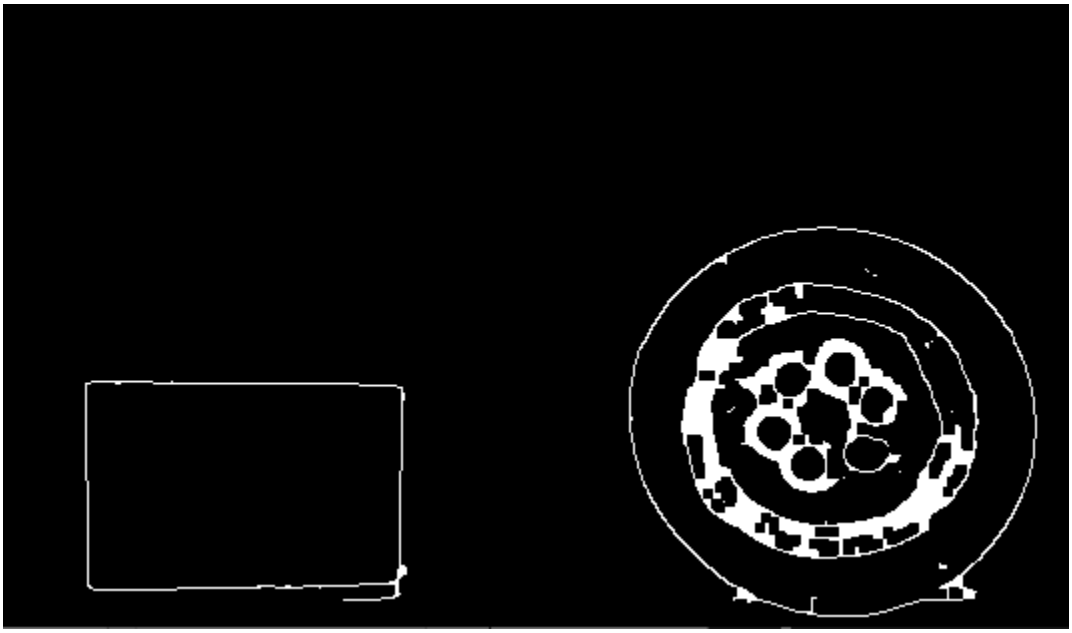


*Figure 28 Two iterations of Erosion performed*

### 2.11.6 Contour Detection:

Contours are made by connecting the border points of the objects in the picture. Analyzed in terms of shape and metrology. The contour detection is very useful. The contour approach is used to extract the reference object and fibre optics from the image. OpenCV arranges all the contours present in an image based on its area, so in this study a constraint has been added that camera is always at a predetermined distance from the fiber and reference object. This constraint helps us to calibrate the system easily. We calculated the area of the outer circle of the fiber once with the help of contour and then used it as a check point in algorithm, contours of objects smaller than the biggest circle contour is always rejected.

To do contours detection OpenCV provide a function called Find Contours which intent to find contours in the image. Of course, to some treatment should be applied to the picture in order to get a good contours detection. We are using bounding box to be drawn on our contour image which have four corner points. Our algorithm main working is based on the sorting of these corner point in such a way that it can detect the reference object first. The bounding box drawn on our contours is shown in Figure 2.21.

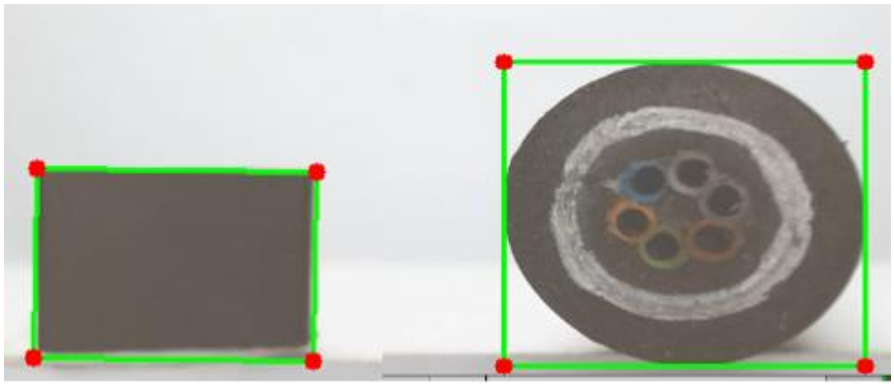


*Figure 22921 Bounding Box*

### 2.11.7 Ordering Coordinates Points Clockwise:

To order our points clockwise first, we define our order point's function which requires only a single parameter the list of points that we want to order. Then sorts these points based on their **'x-values'**. Given the sorted x-Sorted list, we apply array slicing to grab the two left-most points along with the two right-most points. The left-most points will thus correspond to the top-left and bottom-left points while right-most will be our top-right and bottom-right points the main point is

to figure out which is which. If we sort our left-most points according to their y-value, we can derive the top-left and bottom-left points, respectively.

We can then use some geometry to identify the bottom-right and bottom-left positions. We may use the Pythagorean Theorem to calculate the Euclidean distance between the top-left and right-most points by using the top-left point as an anchor. The hypotenuse is the longest side of a right-angled triangle, according to the definition of a triangle. As a result, if we use the top-left point as our anchor, the bottom-right point will have the greatest Euclidean distance, allowing us to extract both the bottom-right and top-right points.

Finally, it returns a NumPy array representing our ordered bounding box coordinates in top-left, top-right, bottom-right, and bottom-left order and after sorting the corner point the algorithm will first detect the left-most object which is our reference object the sorting is shown in Figure 2.22.

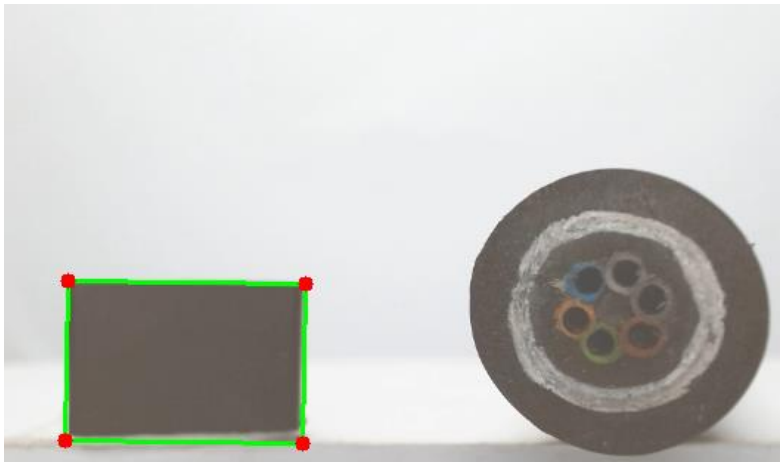


*Figure 2.30 Left-most object detected*

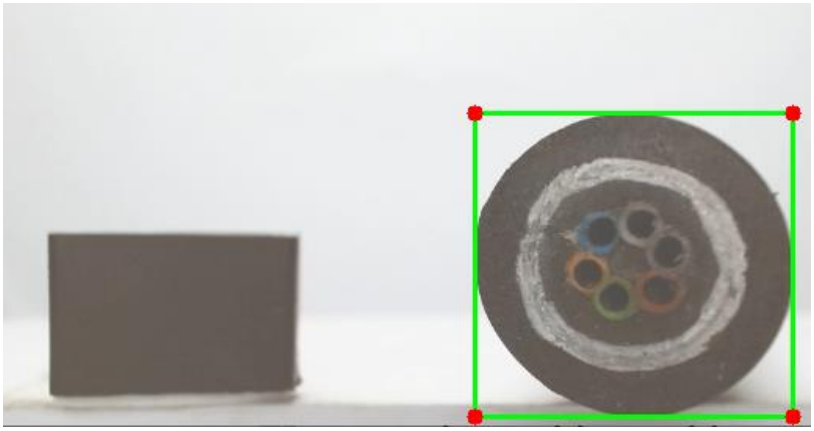


*Figure 2.313 Main object*

### 2.11.8 Dimension Measurement Algorithm

We developed an algorithm that uses the sorted corner points in order to measure the required parameters of the fiber. The algorithm is used to measure the size of any object accordingly. We need to establish a ratio that measures the number of pixels per metric in order to quantify the size of an object.

To find the size of an object in an image, first we need to perform a "calibration" using a reference object, by performing this calibration it does not matter how far our camera is from the object. Our reference object should have two important properties:

- **Property #1:** This object's dimensions (in terms of width or height) should be known in a measurable unit e.g., centimeter, inches etc.
- **Property #2:** We should be able to locate this reference object in an image quickly, either by its location (for example, the reference object should always be at the top-left corner of an image) or by its appearance (like being a distinctive color or shape, unique and different from all other objects in the image). Our reference should be individually identifiable in some way in either situation.

We will be utilizing the left most placement for the detection of the reference object. After uniquely identifying our reference object, we can sort our object contours from left-to-right, to grab the reference object and use it to define our pixels metric ratio, which we define as:

$$\text{Pixels\_Metric} = \frac{\text{Object} - \text{Width}}{Known\ Width}$$

Let's suppose the known width of our reference object is 1.5 cm. Now, suppose that our object width is computed be 150 pixels wide.

The Pixels_Metric is therefore:

$$\text{Pixels\_metric} = 150\text{px} / 1.5 = 100\ \text{px}$$

Thus, shows that there are around 100 pixels per every 1.5 centimeters in our image. Using this ratio, we can calculate the size of objects in the image.

Now, to formulate the above logic in our image we first calculate the mid-point of our four sorted corner points which are top-left, top-right, bottom-right, bottom-left. After finding the mid points

between the four corner points, we will than calculate the Euclidean distance between these mid-points[9].

$$d(p,q) = \sqrt{(p_1 - q_1)2 + (p_2 - q_2)2}$$

After getting the Euclidean distance between the mid-points we can get the cladding diameter of our fiber automatically. Now, the next step in our algorithm is to calculate the number of cores in our fiber. In order to get the number of results we will first perform a check to get the perspective view of our fiber image if the diameter is greater than equal to 1 cm and less than equal to 1.2 cm. The 'and' represents the AND operator in our check. By performing this check, we will get the region of interest of our image where only core are present. Now, again we will pre-process our ROI image using grayscale, median blur filter, and edge detection and finally drawing contours on our process ROI image. Before drawing contours on our ROI image, we will again mention a check in which we will specify that if our contours corner is greater than 6 than draw a contour on our image. Corners greater than 6 means it is a circle as three corner represents triangle, 4 points represent square. So, 6 points obviously represents the circle.

## 2.12 Result of Computer Vision System

The algorithm's output is to display results on the screen. The result will be shown in centimeters, known width of our reference object is 1.5 centimeter. The result will be shown on the screen as our output image. A bounding box and mid-point lines are drawn on our output image on which our measurement is marked in centimeters. The result of diameter calculations is shown in Figure 2.24 and Figure 2.25 below.

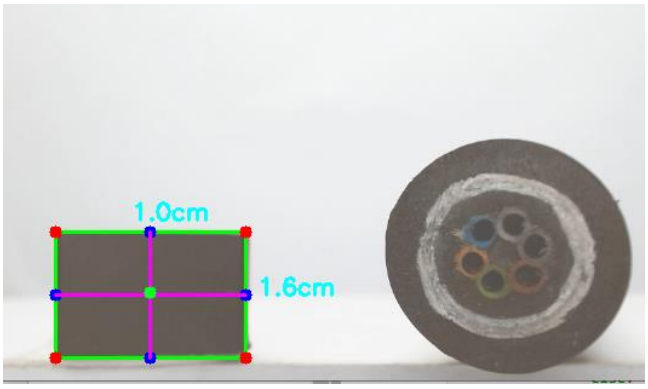



*Figure 2.32 Reference object's diameter*

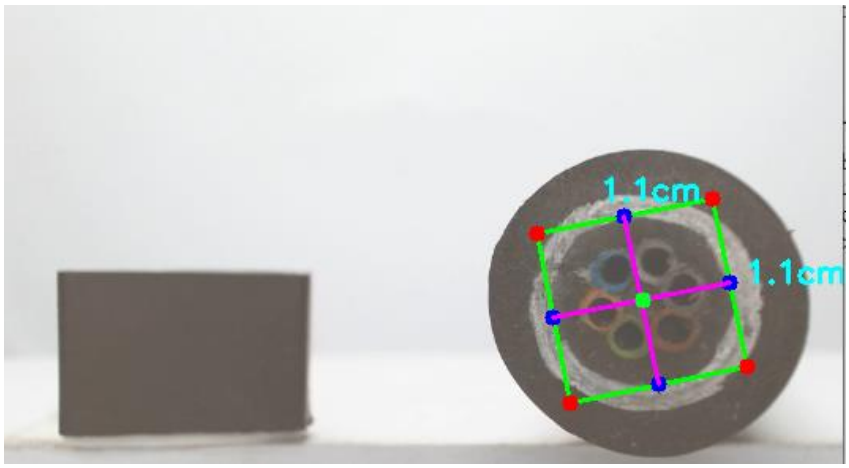



*Figure 2.33 Calculated cladding diameter*

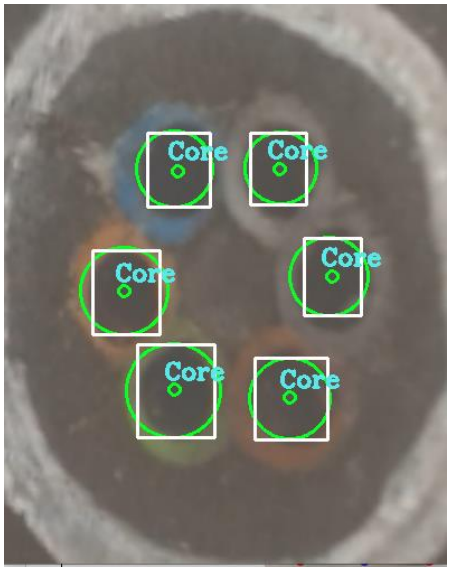



*Figure 2.34 Detection of cores*

# CHAPTER 3 – RESULT AND DISCUSSION

## 3.1 Acquired Image Result as a Function of Sigma

In this section, we used the acquired image as shown in Figure 3.1 We made few changes to the value of sigma, which is the coefficient of variance in image smoothening, and results were observed.

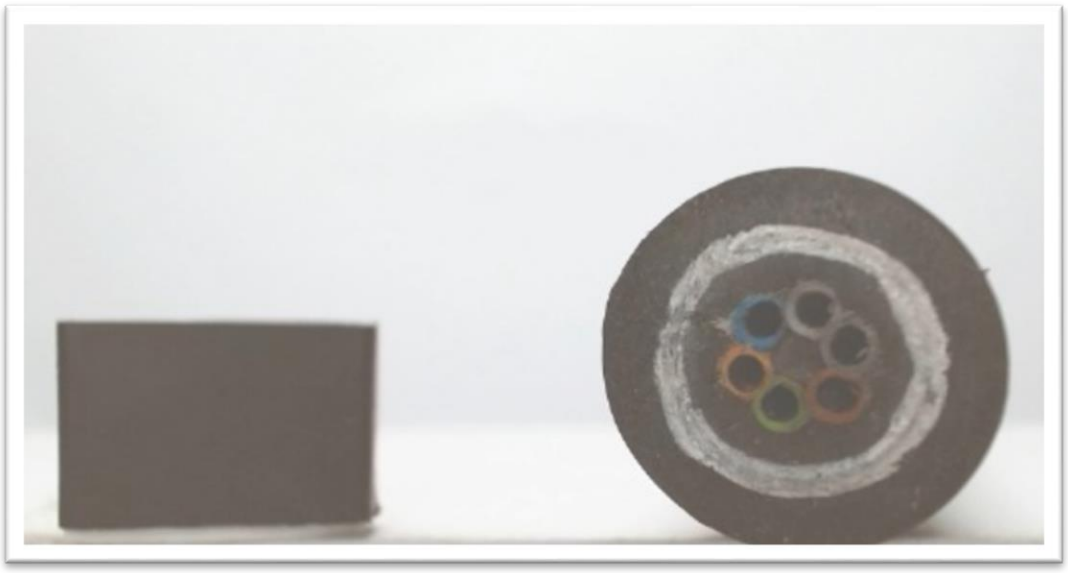


*Figure 35 Input Image*

Initial settings are as follows:

- Measurement Reference: black rectangular object i.e., Dimensions 16x10mm
- Image Resolution: 891x630
- Gaussian Kernel: 7x7
- Canny Threshold: [50, 50]
- Dilation and Erosion Kernel: 7x7, Iterations = 2

### 3.1.1 First test image with sigma = 0.5

Except the value of sigma, all other parameters were kept the same. The sigma value was set to 0.5 and the results is as shown in Figure 3.2.

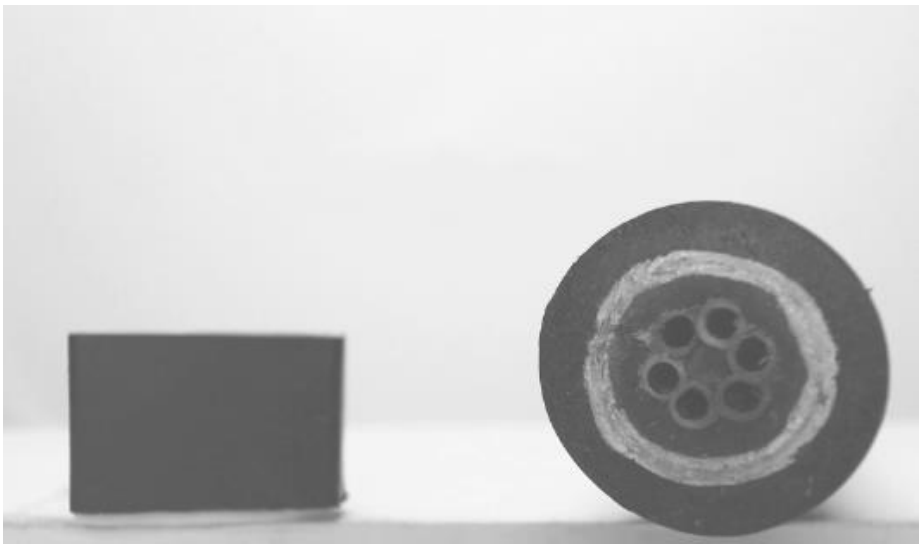


*Figure 36 Measurements result with sigma = 0.5*

### 3.1.2 Second test image with sigma = 1.0

The value of sigma was changed to 1 and all other parameters were kept as the previous one. The result is as shown in Figure 3.3.

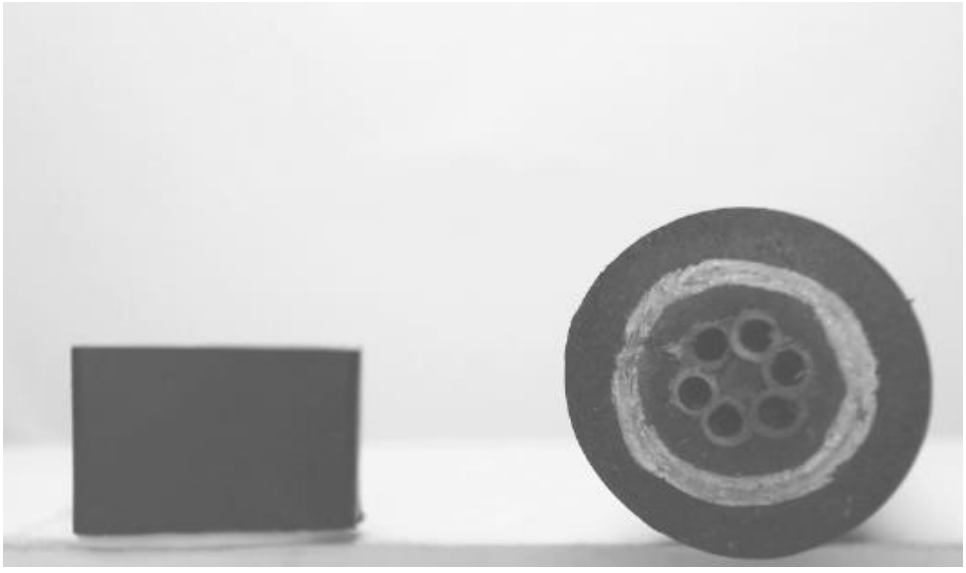


*Figure 37 Measurements results with sigma = 1.0*

### 3.1.3 Third test image with sigma = 1.5

For sigma =1.5, The result is as shown in Figure 3.4 keeping all other parameters same as previous one.

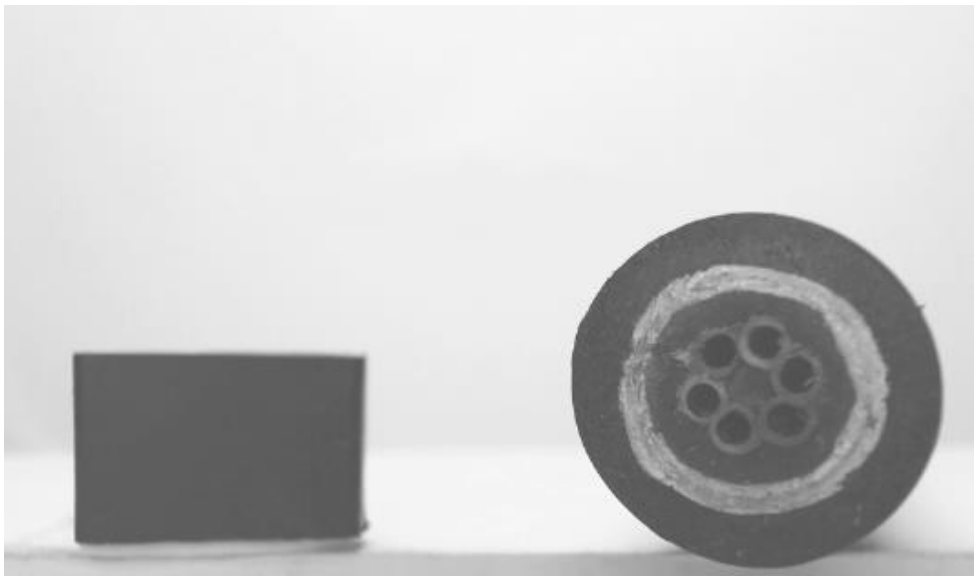


*Figure 38 Measurements results with sigma = 1.5*

### 3.1.4 Fourth test image with sigma = 2.0

All other parameters were kept as initial one, except the value of Sigma. The result is as shown in Figure 3.5 for sigma = 2.0.

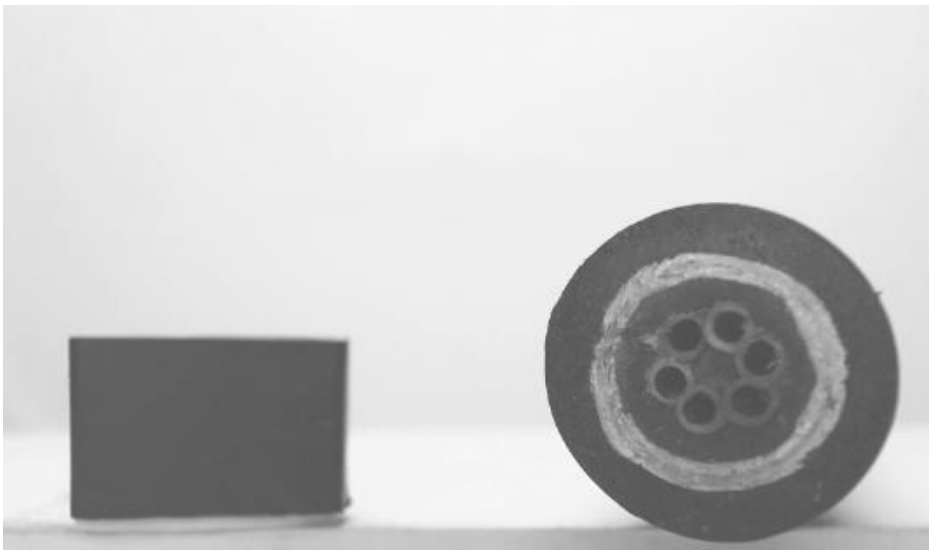


*Figure 39 Measurements results with sigma = 2.0*

### 3.1.5 Discussion on Results as Function of Sigma

From section 3.1.1 to 3.1.4, only the sigma value was altered; all other values remain constant. For four sigma values (0.5, 1.5, and 2), measurements were recorded and are shown in the table

*Table 1 Change in percentage error with sigma values*

| Dimensions | Percentage Error | | | |
|---|---|---|---|---|
| | Sigma = 0.5 (First Test) | Sigma = 1.0 (Second Test) | Sigma = 1.5 (Third Test) | Sigma = 2.0 (Fourth Test) |
| L1 | 2.5 | 0.9 | 0.5 | -1.2 |
| L2 | 4.5 | 2.3 | 1.7 | -1.5 |
| D1 | 0.9 | 0 | 0 | 0 |
| D2 | 1.2 | 0 | 0 | -0.9 |

below.

- The L1 and L2 are the length of our reference object L2 is the known dimension we have given to our object which is 16 mm and L2 is calculated with reference to that known length. We see that L1 and L2 give the actual value between sigma (1 to 1.5) because before sigma = 1 it detects some noise around the reference object and after sigma = 1.5 the edges become faint which misses useful information.
- L2 is a small length, i.e., very small change in measured value with respect to actual value produces large percentage error for L2, while L1 is a big length i.e., it has small percentage error.
- L2 is a small length, so its percentage error is high.
- Diameter D1 and D2 shows positive percentage errors. The reason is the resolution of the image.
- By changing the sigma value, we observed that the coefficient of variance increases and decreases the smoothness of the image, due to which the number of points detected at the edge changes.
- At sigma = 1, the results are very near to the actual ones because for larger values of sigma, edges get faint.

## 3.2 Single image results of Function of Scale Value

In this section, we used the acquired image as shown in Figure 3.6. We change the scale value, which is the resolution of processing image and results were observed.

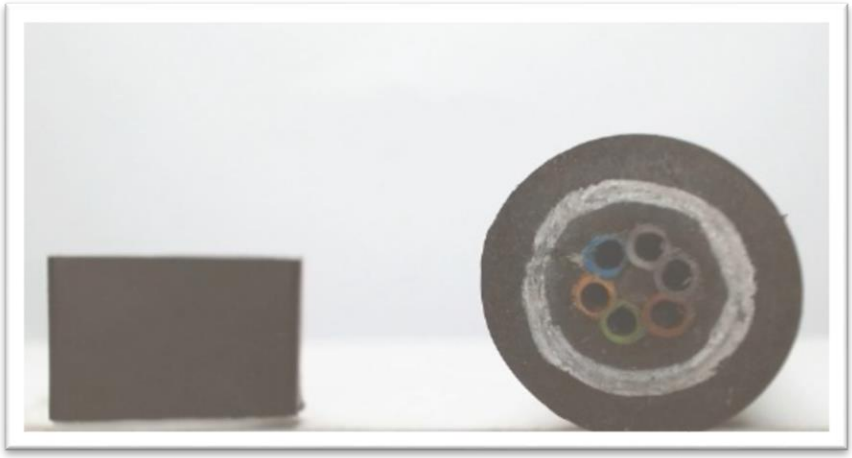


*Figure 40 Acquired image as results of Function of Sigma*

Initial settings are as follows:

- Measurement Reference: Rectangular Object i.e., Dimensions 16x10mm
- Image Resolution: 4000x1844
- Sigma = 1.3
- Gaussian Kernel: 9x9
- Canny Threshold: [50, 50]
- Dilation and Erosion Kernel: 5x5, Iterations = 3

Discussion on Results as a Function of Scale Value

In these tests, the scale value is changed, all other factors remain constant. For seven scale values (2, 3, 4, 5, 6, 7 and 8), values the is measurements were recorded. The following table shows the recorded measurements for each scale value.

Change in scale means changing the resolution of the image, which affects the number of pixels in an image. The resolution of the image, which affects the number of pixels in an image. The resolution of the processed image, i.e., the warped image, for each test is shown in the last row of Table 2.

*Table 2 Changes in percentage error with scale values.*

| Dimensions | Percentage Error | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Test 1 Scale = 2** | **Test 2 Scale = 3** | **Test 3 Scale = 4** | **Test 4 Scale = 5** | **Test 5 Scale = 6** | **Test 6 Scale = 7** | **Test 7 Scale = 8** |
| L1 | 1.6 | 1.7 | 0.7 | 0.4 | 0.8 | 1.2 | 0.8 |
| L2 | -0.8 | 2.9 | 3 | 1 | 2.7 | 2.9 | 2.7 |
| D1 | 0 | 0 | 2.5 | 2 | 1.7 | 1.4 | 2.5 |
| **D2** | 0 | 0 | 0 | 2.9 | 0 | 2 | 1.8 |
| **Resolution** | 594x420 | 891x630 | 1188x840 | 1485x1050 | 1782x1260 | 2079x1470 | 2376x1680 |

From Table 2, it can be seen that for smaller scale values, the error in diameter D1 and D2 measurements Zero, while for higher scale values, errors start to be introduced in the diameter D1 and D2 measurements and increases with increase in scale value. Also, the number of pixels increases on curve for each diameter.

With increase in scale value, the increase in percentage error of diameter measurements due to the presence of points on curve. As the resolution increases the points on curve increase. Increasing the scale value i.e., resolution, - decreases the negative percentage error in the measurement because it detects more points for each measurement curves.

## Counting Number of Cores:

After getting out desired cladding diameter which is 11 mm, we performed a check in order to get our region of interest. As, we know cladding is used in fiber in order to hold the cores of fiber. So, we used cladding diameter for our check and perform warp perspective in order to get our region of interest. In order to find our ROI, we used the coordinates of our bounding box which was drawn on our fiber's cladding. After getting the region of interest, we than again performed image pre-processing step which includes:

- Gray-Scale Conversion
- Applying Median Blur to the image
- Applying threshold to the image
- Applying Canny edge detection
- Draw Center in our region of interest as a reference
- Now taking center as a reference we used fitting circle technique in order to find the circle in our region of interest.
- As a result, we have found the number of cores in our fiber.

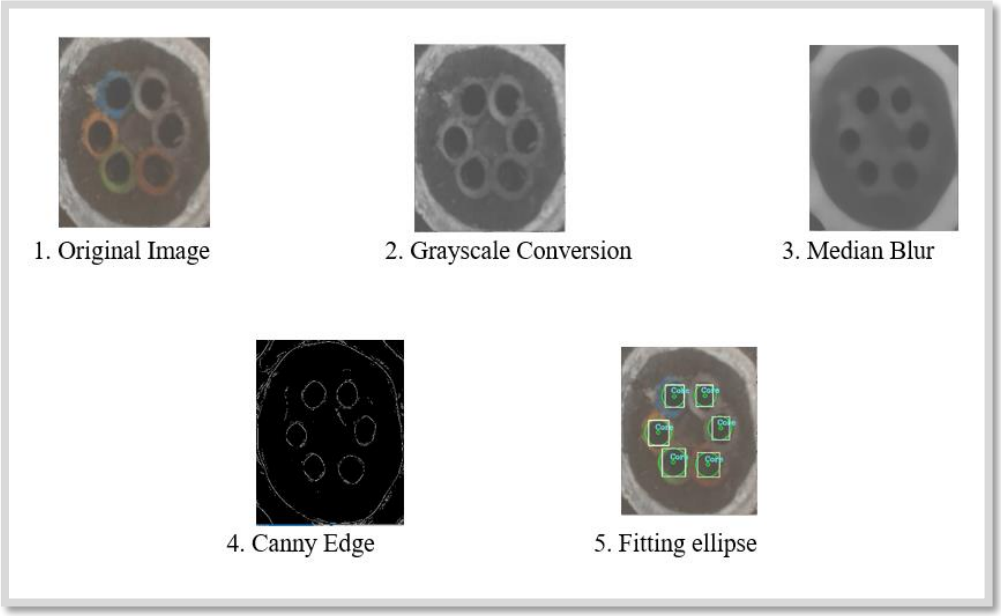



*Figure 41 Steps to count Number of Cores*

## 3.3 Live Stream Video Measurement Results

In this section, we describe how we do the real time video processing, extract frames and process them to find out the dimension real time as shown Table 3.

Initial settings are as follows:

- Measurement Reference: black rectangular object i.e., Dimensions 16x10mm
- Video Resolution: 1920x1080
- Gaussian Kernel: 9x9
- Frames per second: 30 fps
- Sigma = 1.5
- Canny Threshold: [40, 80]
- Dilation and Erosion Kernel: 5x5, Iterations = 3

Note: All setting will remain same for real time processing and measurement.

*Table 3 Fluctuation of percentage error in live stream*

| Dimensions | Percentage Error | | |
|---|---|---|---|
| | First Test | Second Test | Third Test |
| L1 | 2 | -0.3 | 0.4 |
| L2 | -1.5 | -1.9 | -0.1 |
| D1 | 3.3 | 1.7 | 1.7 |
| D2 | 0 | 3.3 | -1.2 |

Discussion on Live Stream Results:

Measurement percentage errors in results are fluctuating due to the addition of noise in each frame during shuttle process. The processing time for each image is about 40 milliseconds. For each the new frame, all the images processing steps are repeated. The points detected vary for each frame due to changes in light condition. For the same scale and sigma value, the result varies. Diameter results show high repeatability, as seen in Table 3. Lengths have a noticeable difference in percentage error due to detection of excessive points at sharp transitions.

# CHAPTER 4 – CONCLUSION

In this research, we designed and developed a vision-based system for measuring geometric parameters of optical fiber. The system is tested in a controlled environment using a 2D camera used for real-time video acquisition and dimension measurements. It measures the cladding diameter and calculate the number of cores of fiber with good accuracy. i.e., measurement errors are less than 1% for the calculated diameter. Measurements obtained from this system have negligible human error.

For this work, we concluded that real-time vision-based measurement systems provide results in less time. Our developed system is vision-based and its measurement method is contactless, so wear and tear in the system does not occur and therefore it gives good life time to the system. Our developed system is capable of processing 10 frames per second, which means it can process 10 optical fibers in one second and that is a great achievement. Its capability of automatically measuring the required parameter reduces effort, money and time. The result of this system depends on the environment and light conditions and that is why to reduce noise, we tested it in a controlled environment. Weak light and also strong light have an effect on the results, as observed in the shutter process. The light's intensity as well as the object's shadow manipulates results and effects measurement accuracy. We applied proper light conditions as well as a suitable background and a reference object to accurately measure the required parameters. From the results, it is concluded that the system's regular calibration is required for constant accuracy. Tests of a single image of known resolution, under the same light, but with different filtering, edge detection, opening and closing conditions, gives different results. So, our calibration is done with the hit and trial method which helps to get appropriate setting for our parameters. The edge points detected during image processing steps define the accuracy of the system. Optimizing the image processing parameters results in an increase in system accuracy. Another factor which impacts the measurement accuracy is the image resolution.

In live video stream testing, under the same light conditions, we observed fluctuations in our measurements. These fluctuations were due to noise added during the shutter process. So, we conclude that in real time processing, there will be a slight compromise in precision of

measurements. Because the created algorithm has a minimal level of complexity, it can handle the parameter measuring work quickly.

# REFERENCES


[1] Oreilly.com. 2022. *O'Reilly Media - Technology and Business Training*. [online] Available at: <https://www.oreilly.com/>.

[2] Science, I., 2022. *IRJMETS*. [online] IRJMETS International Research Journal of Modernization in Engineering Technology and Science. Available at: <https://www.irjmets.com/>.

[3] www.slideshare.net. 2022. *SlideShare.net*. [online] Available at: <https://www.slideshare.net/>.

[4] Scribd. 2022. *Discover the Best eBooks, Audiobooks, Magazines, Sheet Music, and More | Scribd*. [online] Available at: <https://www.scribd.com/>.

[5] David, R., 2022. *Robin David*. [online] Robindavid.fr. Available at: <http://www.robindavid.fr/>.

[6] Coursehero.com. 2022. *Course Hero | Make every study hour count*. [online] Available at: <https://www.coursehero.com/>.

[7] PyImageSearch. 2022. *PyImageSearch - You can master Computer Vision, Deep Learning, and OpenCV.*. [online] Available at: <https://pyimagesearch.com/>.

[8] B. G. Batchelor, Book Review: Digital Image Processing, vol 17, no 1980.

[9] MATLAB, “No Title.” https://www.mathworks.com/help/vision/ug/camera-calibration.html.

# APPENDIX A

---

## Code for Vision based System:

```
import numpy as np
import imutils
import cv2


ref_width = 1.5 # in cm
def order_points(pts):
    # sort the points based on their x-coordinates
    xSorted = pts[np.argsort(pts[:, 0]), :]


    # grab the left-most and right-most points from the sorted
    # x-roodinate points
    leftMost = xSorted[:2, :]
    rightMost = xSorted[2:, :]


    # now, sort the left-most coordinates according to their
    # y-coordinates so we can grab the top-left and bottom-left
    # points, respectively
    leftMost = leftMost[np.argsort(leftMost[:, 1]), :]
    (tl, bl) = leftMost
```

```
    # now that we have the top-left coordinate, use it as an
    # anchor to calculate the Euclidean distance between the
    # top-left and right-most points; by the Pythagorean
    # theorem, the point with the largest distance will be
    # our bottom-right point
    D = dist.cdist(tl[np.newaxis], rightMost, "euclidean")[0]
    (br, tr) = rightMost[np.argsort(D)[::-1], :]

    # return the coordinates in top-left, top-right,
    # bottom-right, and bottom-left order
    return np.array([tl, tr, br, bl], dtype="float32")

def four_point_transform(image, pts):
    # obtain a consistent order of the points and unpack them
    # individually
    rect = order_points(pts)
    (tl, tr, br, bl) = rect

    # compute the width of the new image, which will be the
    # maximum distance between bottom-right and bottom-left
    # x-coordiates or the top-right and top-left x-coordinates
    widthA = np.sqrt(((br[0] - bl[0]) * 2) + ((br[1] - bl[1]) * 2))
    widthB = np.sqrt(((tr[0] - tl[0]) * 2) + ((tr[1] - tl[1]) * 2))
```

```
maxWidth = max(int(widthA), int(widthB))

# compute the height of the new image, which will be the
# maximum distance between the top-right and bottom-right
# y-coordinates or the top-left and bottom-left y-coordinates
heightA = np.sqrt(((tr[0] - br[0]) * 2) + ((tr[1] - br[1]) * 2))
heightB = np.sqrt(((tl[0] - bl[0]) * 2) + ((tl[1] - bl[1]) * 2))
maxHeight = max(int(heightA), int(heightB))

# now that we have the dimensions of the new image, construct
# the set of destination points to obtain a "birds eye view",
# (i.e. top-down view) of the image, again specifying points
# in the top-left, top-right, bottom-right, and bottom-left
# order
dst = np.array([
    [0, 0],
    [maxWidth - 1, 0],
    [maxWidth - 1, maxHeight - 1],
    [0, maxHeight - 1]], dtype="float32")

# compute the perspective transform matrix and then apply it
M = cv2.getPerspectiveTransform(rect, dst)
warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
```

```
    # return the warped image
    return warped

def sort_contours(cnts, method="left-to-right"):
    # initialize the reverse flag and sort index
    reverse = False
    i = 0

    # handle if we need to sort in reverse
    if method == "right-to-left" or method == "bottom-to-top":
        reverse = True

    # handle if we are sorting against the y-coordinate rather than
    # the x-coordinate of the bounding box
    if method == "top-to-bottom" or method == "bottom-to-top":
        i = 1

    # construct the list of bounding boxes and sort them from top to
    # bottom
    boundingBoxes = [cv2.boundingRect(c) for c in cnts]
    (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes),
                                key=lambda b: b[1][i], reverse=reverse))
```

```
    # return the list of sorted contours and bounding boxes
    return cnts, boundingBoxes
def read_and_preproces(image, canny_low= 50, canny_high = 60, blur_kernel=9,
d_e_kernel=3):

    #image = cv2.resize(image, (500, 500))
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (blur_kernel, blur_kernel), 0)
    edged = cv2.Canny(gray, canny_low, canny_high)
    return image, edged

def midpoint(ptA, ptB):
    return ((ptA[0] + ptB[0]) / 2, (ptA[1] + ptB[1]) / 2)
def get_distance_in_pixels(orig, c):
    box = cv2.minAreaRect(c)
    box = cv2.boxPoints(box)
    box = np.array(box, dtype="int")
    box = perspective.order_points(box)
    cv2.drawContours(orig, [box.astype("int")], -1, (0, 255, 0), 2)

    (tl, tr, br, bl) = box
    (tltrX, tltrY) = midpoint(tl, tr)
    (blbrX, blbrY) = midpoint(bl, br)
```

```
    (tlblX, tlblY) = midpoint(tl, bl)
    (trbrX, trbrY) = midpoint(tr, br)
    cv2.line(orig, (int(tltrX), int(tltrY)), (int(blbrX), int(blbrY)), (255, 0, 255), 2)
    cv2.line(orig, (int(tlblX), int(tlblY)), (int(trbrX), int(trbrY)), (255, 0, 255), 2)

    dc_W = dist.euclidean((tltrX, tltrY), (blbrX, blbrY))
    dc_H = dist.euclidean((tlblX, tlblY), (trbrX, trbrY))
    return dc_W, dc_H, tltrX, tltrY, trbrX, trbrY

def find_object_in_pix(orig, edge, area_threshold):
    cnt,hierarchy = cv2.findContours(edged, cv2.RETR_CCOMP,
cv2.CHAIN_APPROX_SIMPLE)
    print(hierarchy)

    image_internal = np.zeros(edged.shape)

# Iterate through list of contour arrays
    n=0
    for i in range(len(cnt)):
    # If third column value is NOT equal to -1 than its internal
        if hierarchy[0][i][3] == 24:

        # Draw the Contour
```

```
            cv2.drawContours(image_internal, cnt, i, (255,255,255), 3)
            cv2.imshow("core",image_internal)
            n=n+1
    print("Number of Cores", n-1)

    cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    (cnts, _) = contours.sort_contours(cnts)
    P = None
    print(_)

    for c in cnts:
        if cv2.contourArea(c) < int(area_threshold):
            continue
        dc_W, dc_H, tltrX, tltrY, trbrX, trbrY = get_distance_in_pixels(orig, c)

        if P is None:
            P = ref_width / dc_H
            dr_W = ref_width
            dr_H = ref_width
        else:
                dr_W = dc_W * P
```

```
        dr_H = dc_H * P


    cv2.putText(orig, "{:.1f} cm".format(dr_H), (int(tltrX - 15), int(tltrY - 10)),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                (0, 0, 255), 1)
    cv2.putText(orig, "{:.1f} cm".format(dr_W), (int(trbrX + 5 ), int(trbrY)),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                (0, 0, 255), 1)
  return orig


cap = cv2.VideoCapture(0)
cap.set(10,160)
cap.set(3,1920)
cap.set(4,1080)


while True:
  success,img = cap.read()
  image, edged = read_and_preproces(img)
  image = find_object_in_pix(image, edged,10)


  cv2.imshow('window', image)
  if cv2.waitKey(0) == ord('q'):
    cv2.destroyAllWindows()
```

# Geometric Parameter Measurement of Optical Fiber by Image Processing

ORIGINALITY REPORT

| 13% | 12% | 3% | 8% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.pyimagesearch.com<br>Internet Source | 3% |
| 2 | en.wikipedia.org<br>Internet Source | 2% |
| 3 | www.coursehero.com<br>Internet Source | 1% |
| 4 | wikimili.com<br>Internet Source | 1% |
| 5 | Submitted to Southampton Solent University<br>Student Paper | 1% |
| 6 | Submitted to University of Limerick<br>Student Paper | 1% |
| 7 | www.robindavid.fr<br>Internet Source | 1% |
| 8 | Submitted to Manipal International University<br>Student Paper | 1% |
| 9 | www.scribd.com<br>Internet Source | 1% |

| | | |
|---|---|---|
| 10 | Nur Intan Raihana Ruhaiyem, Noor Shariah Ismail. "Chapter 14 Evidence-Based of Improved Electron Tomogram Segmentation and Visualization Through High-Pass Domain Kernel in Bilateral Filter", Springer Science and Business Media LLC, 2021<br>Publication | <1 % |
| 11 | Submitted to University of Sheffield<br>Student Paper | <1 % |
| 12 | Computer Vision, 2014.<br>Publication | <1 % |
| 13 | docplayer.net<br>Internet Source | <1 % |
| 14 | Submitted to University of Huddersfield<br>Student Paper | <1 % |
| 15 | Submitted to University of Wales, Bangor<br>Student Paper | <1 % |
| 16 | www.iapb.org<br>Internet Source | <1 % |
| 17 | www.ucc.ie<br>Internet Source | <1 % |
| 18 | emrlibrary.gov.yk.ca<br>Internet Source | <1 % |
| 19 | Submitted to Sim University<br>Student Paper | <1 % |

| | | |
|---|---|---|
| 20 | Submitted to Guru Nanak Dev Engineering College<br>Student Paper | <1 % |
| 21 | core.ac.uk<br>Internet Source | <1 % |
| 22 | mechomotive.com<br>Internet Source | <1 % |
| 23 | Submitted to Sheffield Hallam University<br>Student Paper | <1 % |
| 24 | ar.in.tum.de<br>Internet Source | <1 % |
| 25 | epdf.pub<br>Internet Source | <1 % |
| 26 | medium.com<br>Internet Source | <1 % |
| 27 | Paul Reilly, Elizabeth Shanahan, Steven Staudaher. "An implementation of microdiagnostics on the ECLIPSE® MV/8000", ACM SIGMICRO Newsletter, 1980<br>Publication | <1 % |
| 28 | digitalassets.lib.berkeley.edu<br>Internet Source | <1 % |
| 29 | www.irjmets.com<br>Internet Source | <1 % |

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 3 words |
| Exclude bibliography | On | | |