A Project Report

On

STREAMLIT APPLICATION FOR IMAGE PROCESSING

BY

ANKIT SINGH

2022A4PS1433H

Under the supervision of

KUNDAN KUMAR SINGH

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF

ME F376: DESIGN PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(MAY 2025)

CONTENTS

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to the Mechanical Department at BITS Pilani Hyderabad Campus for allowing me to work on this design project.

Without their outstanding support and constant encouragement, the successful execution of the "Streamlit application for image processing" project would not have been feasible.

I would like to express my sincere gratitude to Prof. Kundan Kumar Singh, my wonderful mentor and advisor, for his unfailing tolerance, drive, limitless passion, and profound knowledge. His leadership and unflinching support have been essential to our endeavour at every stage.

Through the supervision and assistance of Mr. Kundan Kumar Singh and the Mechanical Department, I have had the opportunity to delve deeper into this fascinating profession and make a positive impact.

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled "**Streamlit application for image processing**" submitted by Mr. Ankit Singh (ID No. 2022A4PS1433H) in partial fulfillment of the requirements of the course ME F376, Design Project Course, embodies the work done by him under my supervision and guidance.

Date: **(Kundan Kumar Singh)**

BITS- Pilani, Hyderabad Campus

ABSTRACT

The use and demand of streamlit in the field of data science is ever-evolving and plays a critical role in demonstrating results effectively. This project focuses on building a streamlit application which showcasing the output and results of an computer vision-based tool for automated dimensional measurements which was build using python and opencv for image processing. But streamlit focuses majorly on the output aspect and how can we make an interactive dashboard which is easy to use.

In this project we are using many streamlit and other libraries such as numpy, matplotlib, PIL and streamlit_image_coordinates for making the web application collectively interactive and easy for the user to execute the results. Different libraries have different purposes such as image loading and unloading, plotting graphs, conversion of dimensions to find the area and being able to choose the coordinates directly through the web application.

This domain has immense potential in the field of information and technology as with the recent updates the scale of interactive web applications is only going to increase which are really good to display your plots, graphs and results.

INTRODUCTION

Streamlit has revolutionized modern front end applications as the ease of building it and the effectiveness of how interactive dashboards have helped a lot of employees in their presentations. We use streamlit library along with many inbuilt libraries for specific features such as selection of coordinates, ploting graphs, converting and calculating to give results. Also the UI interface of the application is very good which also makes it user friendly in a way.

Here we build this web application on the already built code which was on image processing of the tool which calculates the dimensions and provides the final area of the object in the selected dimension and selected length. This project was built using libraries like cv2, numpy, scipy.spatial and scipy.optimize which was used to select the outline of the image to accurately calculate the area of the image through image processing with few additional features which gives us a line, polygon, best-fit circle, best-fit ellipse and many more shapes live plotting with their area and other dimensions.

Functions used in building the streamlit application are to being able to upload image and displaying it on the application. Then being able to select the coordinates of reference line and giving length measurements and dimension. Also, while selecting the surface area coordinates, giving an option to being able to select unlimited number of points till it forms a polygon or any other shape then based on the selected points it provides the calculated area.

Most importantly being able to give results in an interactive dashboard makes it easier to understand and produce better results.

EXPLANATION

This code is divided into 6 segments which all have different functions and need to executed without error to produce desired results. The segments are as follows:

1. **Importing required libraries and dependencies:**

```
import streamlit as st
from streamlit_drawable_canvas import st_canvas
import numpy as np
import cv2
import math
import pandas as pd
from PIL import Image
```

- **Streamlit: for building a web application**
- **Numpy: for numerical operations**
- **Cv2: library used for computer vision attributes to show images**
- **PIL: for handling image uploads.**
- **Streamlit_drawable_canvas: a package that allows users to click on an image and get the pixel coordinates.**
- **Math: for solving and calculating all areas and dimensions**
- **Pandas: used for understanding the selecting coordinates and then giving appropriate data.**

2. Required calculations and coordinate selection:

```python
def calculate_distance(p1, p2):  4 usages
    return np.linalg.norm(np.array(p1) - np.array(p2))

def calculate_polygon_area(points):  1 usage
    x, y = zip(*points)
    return 0.5 * abs(
        sum(
            x[i] * y[(i + 1) % len(points)]
            - x[(i + 1) % len(points)] * y[i]
            for i in range(len(points))
        )
    )

def calculate_internal_angles(points):  1 usage
    angles = []
    for i in range(len(points)):
        p0 = np.array(points[i - 1])
        p1 = np.array(points[i])
        p2 = np.array(points[(i + 1) % len(points)])
        v1, v2 = p0 - p1, p2 - p1
        cosang = np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
        angles.append(math.degrees(math.acos(np.clip(cosang, -1,  a_max: 1))))
    return angles
```

- **We define the calculate_distance,calculate_polygon_area,calculate_internal_angles and many more functions for all mathematical calculations.**
- **Then with respect to the selected coordinates and selected shape the respective functions get run as a call function.**

3. Building UI and interactive click buttons/controls:

```python
st.set_page_config(layout="wide")
st.title("🔪 Interactive Image Measurement Tool")

# Persist background in session
if 'bg_img' not in st.session_state:
    st.session_state.bg_img = None

# Image upload
uploaded = st.file_uploader("🖼️ Upload an image", type=["jpg","jpeg","png"])
if not uploaded:
    st.stop()

base = np.array(Image.open(uploaded).convert("RGB"))
h, w = base.shape[:2]
if st.session_state.bg_img is None or st.session_state.get('uploaded') != uploaded:
    st.session_state.bg_img = base.copy()
    st.session_state.uploaded = uploaded

# Sidebar controls
st.sidebar.header("Settings")
shape = st.sidebar.selectbox("Shape to measure:", ["Line","Polygon","Best-Fit Circle","Best-Fit Ellipse"])
ref_val = st.sidebar.number_input(
    "Reference length:", min_value=0.000001, value=1.0,
    format="%.9f", step=0.000001
)
unit = st.sidebar.selectbox("Unit:", ["mm","cm","inches","pixels"])
if st.sidebar.button("Clear All"):
    st.session_state.bg_img = base.copy()
```

This is to build an interactive dashboard for a seamless experience by adding features like option to upload image and select all shapes and dimensions.

4.Using interactive canvas function and set limits :

```python
# Interactive canvas
canvas_res = st_canvas(
    background_image=Image.fromarray(st.session_state.bg_img),
    height=h, width=w,
    drawing_mode="point",
    stroke_color="#FF0000",
    stroke_width=5,
    update_streamlit=True,
    key="canvas",
)

# Process clicks and annotate
if canvas_res.json_data and canvas_res.json_data.get("objects"):
    pts = [(int(o['left']), int(o['top'])) for o in canvas_res.json_data['objects']]
    if len(pts) >= 2:
        ref_pts = pts[:2]
        shape_pts = pts[2:]

        # calculate scale
        pix = calculate_distance(ref_pts[0], ref_pts[1])
        if pix < 1e-6:
            st.warning("⚠️ Reference points too close. Re-select two distinct points.")
            st.session_state.bg_img = base.copy()
            st.experimental_rerun()
        scale = ref_val / pix
```

- **We here from the canvas library make it possible to select coordinates directly from clicking on the image. Also plotting the shapes live as we select the coordinates.**
- **Also we provide limits to select the reference length with their limit in decimal value input.**
- **Then also adding features like labelling of points and also live display of measured distance.**

5.Building tables to display all results in a tabular format:

```python
# prepare tables
tables = {}

# draw shape and compute tables
if shape == "Line" and len(shape_pts) >= 2:
    a, b = shape_pts[0], shape_pts[1]
    d = calculate_distance(a, b) * scale
    cv2.line(img, a, b, color: (255,255,0), thickness: 2)
    cv2.putText(
        img,
        text: f"{d:.2f}{unit}",
        org: ((a[0]+b[0])//2, (a[1]+b[1])//2),
        cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.6, color: (255,255,0), thickness: 2
    )
    tables['Line Length'] = [f"{d:.4f} {unit}"]

elif shape == "Polygon" and len(shape_pts) >= 3:
    sides = []
    for i in range(len(shape_pts)):
        p1, p2 = shape_pts[i], shape_pts[(i+1)%len(shape_pts)]
        cv2.line(img, p1, p2, color: (255,0,0), thickness: 2)
        s = calculate_distance(p1, p2) * scale
        sides.append(s)
        mid = ((p1[0]+p2[0])//2, (p1[1]+p2[1])//2)
        cv2.putText(
            img,
            text: f"{s:.2f}{unit}",
            mid,
            cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, color: (255,255,0), thickness: 1
        )
```

After selection of coordinates and building the required shape we display all the calculated results in a tabular format for better understanding of the results. It will run and display the table of the shape which was selected and then generate the table using a defined call function.

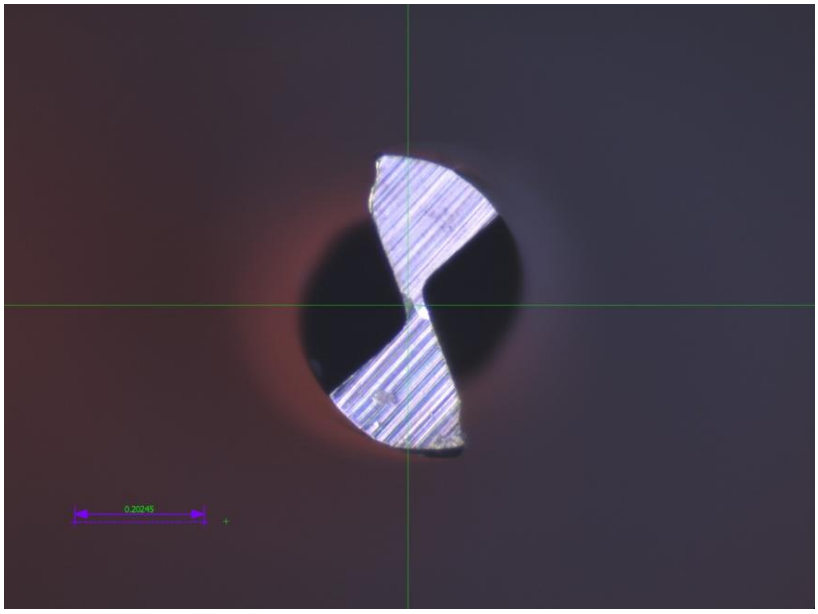
6.Create an option to know distance between any points:

```
# distance between any two points
if len(pts) >= 2:
    st.subheader("Distance Between Selected Points")
    p1 = st.selectbox("First point:", labels)
    p2 = st.selectbox("Second point:", [l for l in labels if l != p1])
    i1, i2 = labels.index(p1), labels.index(p2)
    sd = calculate_distance(pts[i1], pts[i2]) * scale
    dist_tbl = pd.DataFrame({f"{p1} → {p2}":[f"{sd:.4f} {unit}"]})
    st.table(dist_tbl)
```

This additional feature gives us an option to know distance between any two selected points on the image.

CONCLUSION

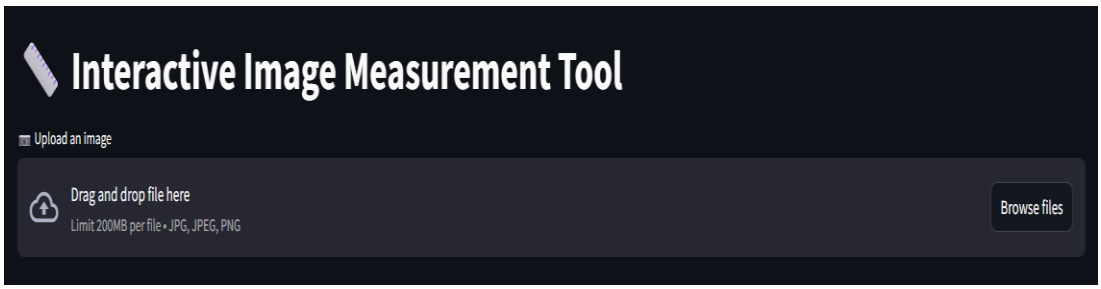
1. **The image using which the code was run was in JPG file format:**



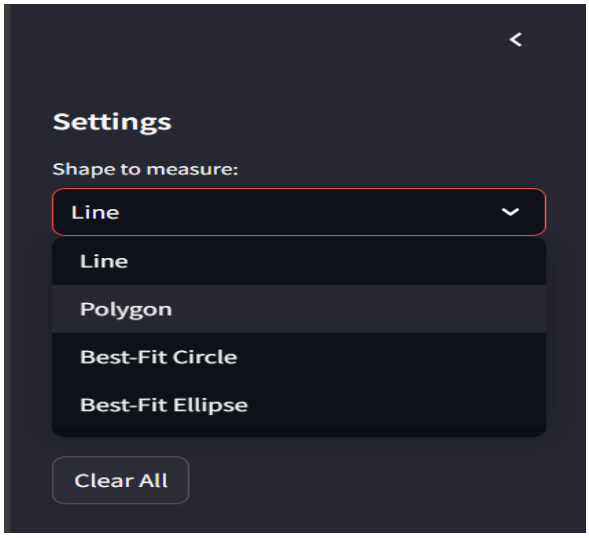
2. **we run the streamlit application through our terminal**

```
(.venv) PS C:\Users\Admin\Desktop\image process> streamlit run final.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.16.12.73:8501
```

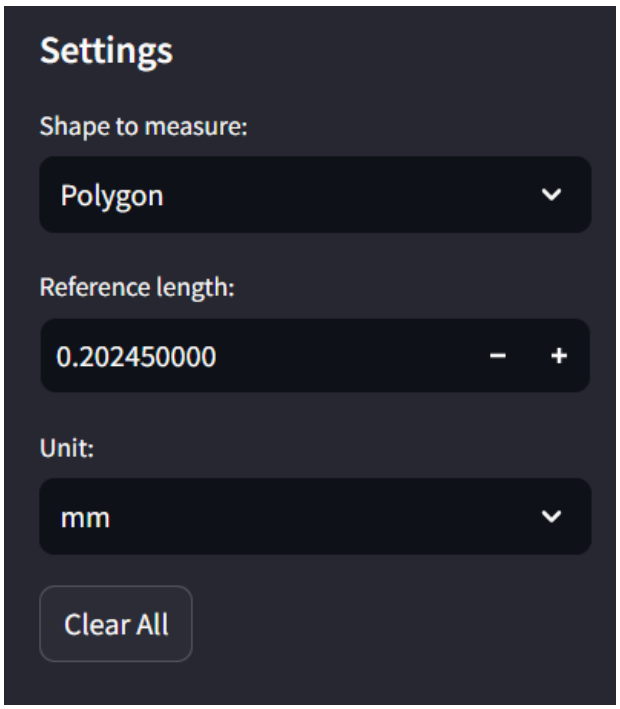
3. **upload the image in the area provided**



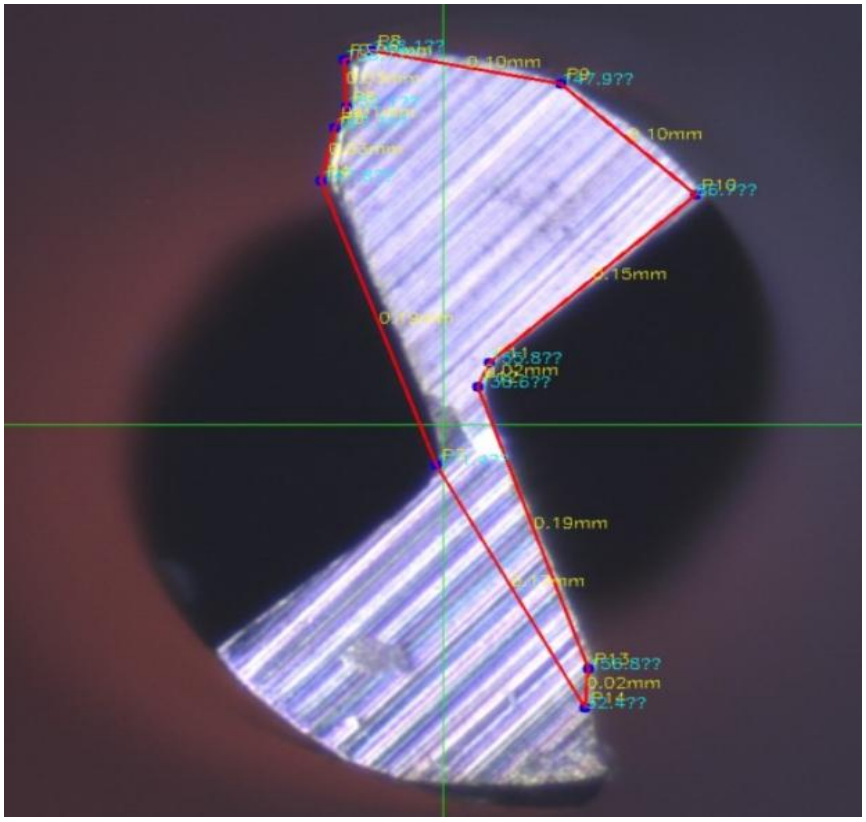
4. **Select the required shape to calculate and display:**



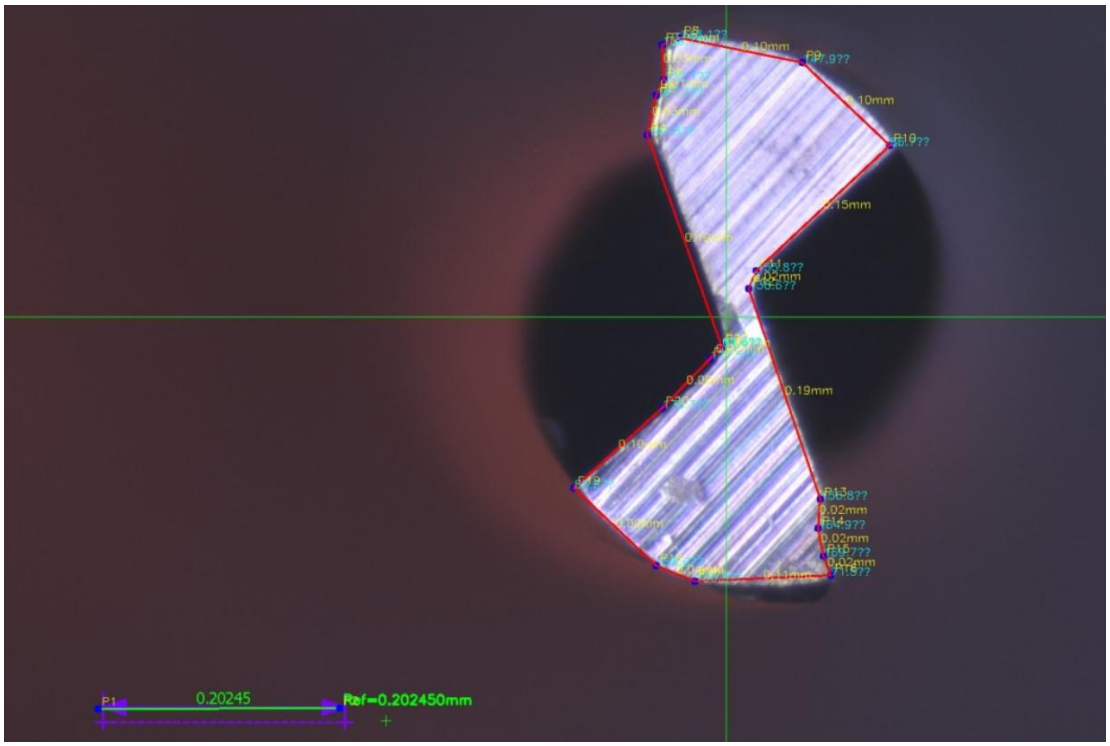
5. **Select the reference length and give its measurement with units:**



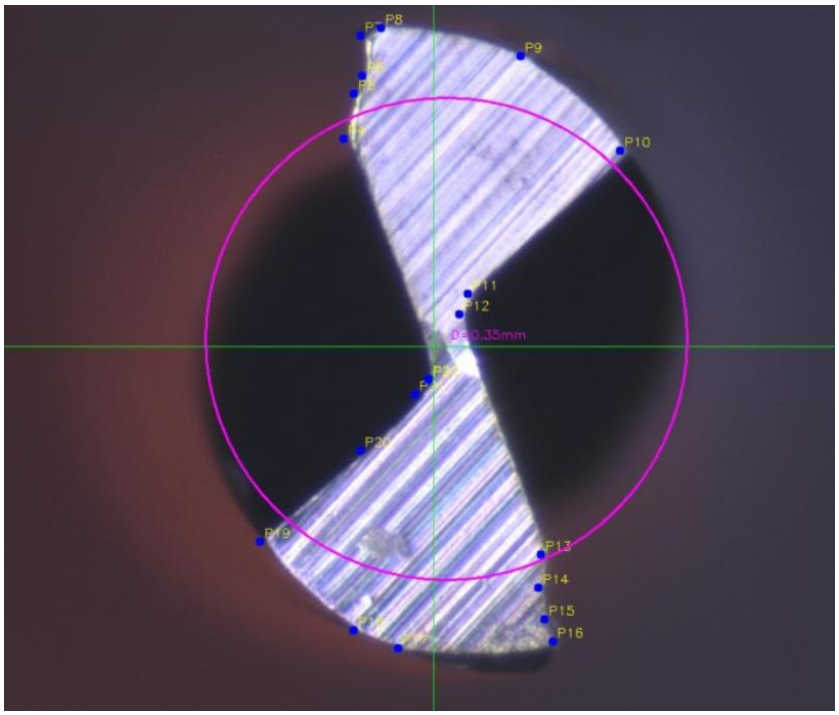
6. **Live demonstration of polygon or any shape formation on the same image itself in which we select the coordinates**

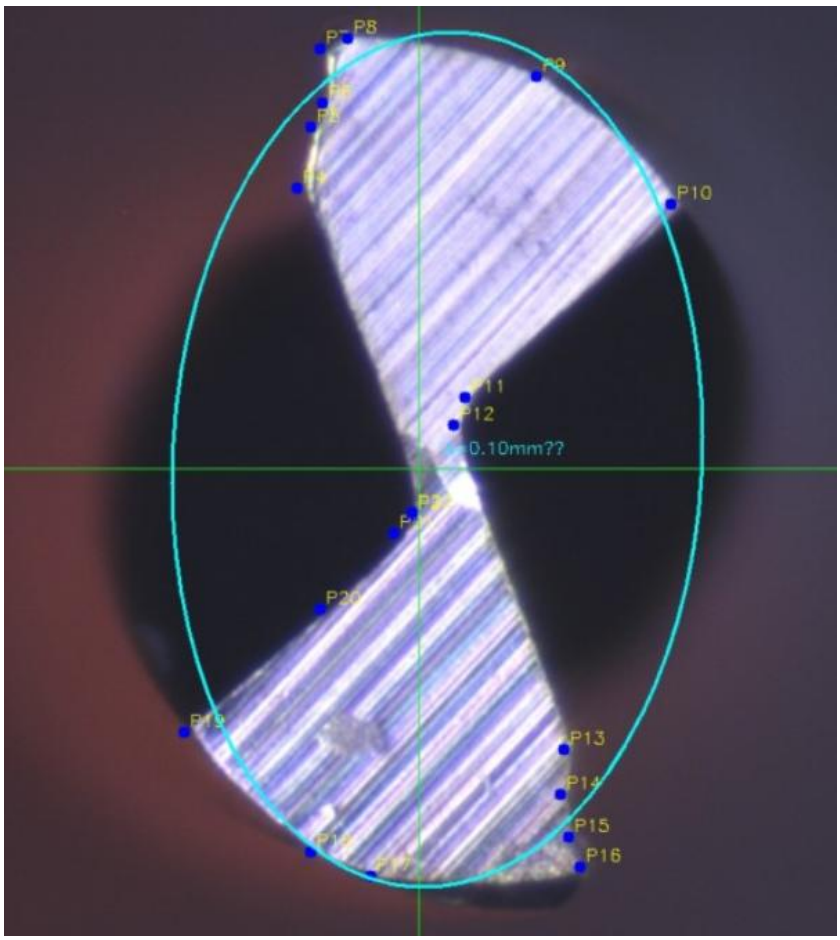


7. **Final selected polygon image:**



8. **We can also demonstrate best-fit circle and bestfit ellipse with respect to the coordinates selected for the polygon earlier.**





9. Showing calculated result outputs in a tabular format

- **polygon**

Polygon Side Lengths

	Side 1	Side 2	Side 3	Side 4	Side 5	Side 6	Side 7	Side 8	Side 9	Side 10	Side 11	Side 12	Side 13	Side 14	Side 15	Side 16	Side 17	Side 18	Side 19	Side 20
0	0.1881 mm	0.0340 mm	0.0145 mm	0.0294 mm	0.0160 mm	0.1045 mm	0.1008 mm	0.1534 mm	0.0162 mm	0.1861 mm	0.0244 mm	0.0235 mm	0.0174 mm	0.1138 mm	0.0350 mm	0.0946 mm	0.0995 mm	0.0574 mm	0.0150 mm	0.0012 mm

Internal Angles

	∠1	∠2	∠3	∠4	∠5	∠6	∠7	∠8	∠9	∠10	∠11	∠12	∠13	∠14	∠15	∠16	∠17	∠18	∠19	∠20
0	19.40°	147.84°	167.29°	152.10°	108.12°	148.07°	147.88°	86.68°	155.80°	138.58°	156.80°	164.89°	169.68°	71.48°	155.49°	158.60°	85.60°	176.32°	177.51°	41.63°

Area

	Area
0	0.0548 mm^2

- **best-fit circle**

Diameter

	Diameter
0	0.3537 mm

Area

	Area
0	0.0982 mm^2

- **best-fit ellipse**

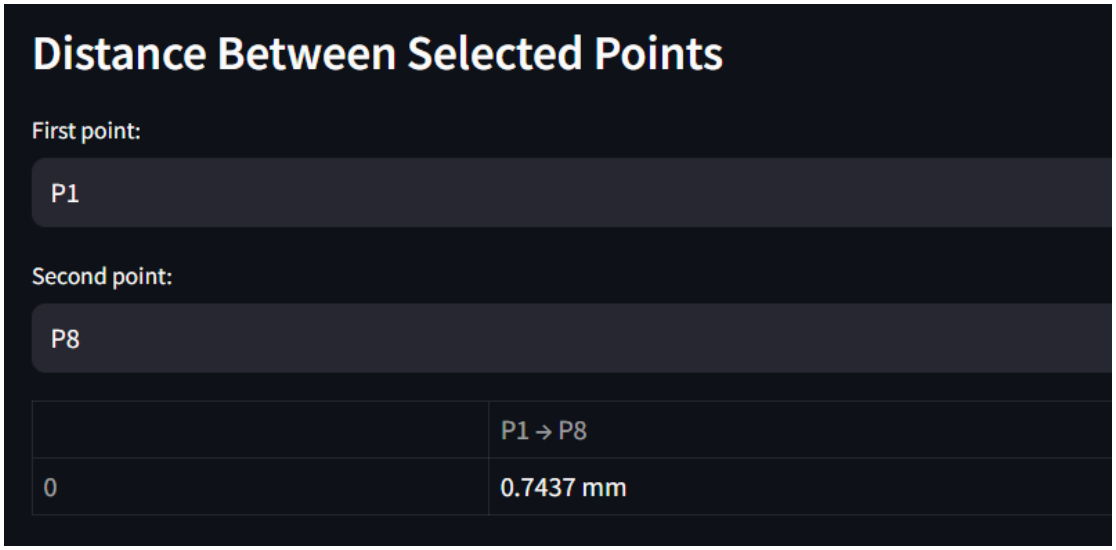
Axes

	Major Axis	Minor Axis
0	0.2874 mm	0.4649 mm

Area

	Area
0	0.1049 mm^2

10.Finally measuring distance between any two selected points:



11.Changes required in the coding environment to run this are

- **Having all the libraries mentioned installed.**
- **Setting the streamlit to 1.40.0 version and for that you can use the command "pip install streamlit==1.40.0"**
- **Also focus on these versions and libraries the you can directly run the command "streamlit run file_name.py" to access the streamlit dashboard.**

REFERENCE

1. **https://docs.streamlit.io/get-started/fundamentals/main-concepts**
2. https://numpy.org/learn/
3. https://gist.github.com/omairaasim/a9b835ceb0777af7ef9a223921c6d7b1
4. https://pypi.org/project/streamlit-drawable-canvas/

1- For syntax and understanding of building streamlit application
2- Understanding and application of numpy library
3- Application of cv2 library for plotting of coordinates and finding the area live as we select coordinates on the image itself.
4- Application of streamlit drawable canvas is selecting points directly from streamlit and locating the points as well.