Project Report
Adobe GenSolve Hackathon R2
(Curvetopia)

Team members:

Om Saran	20cs02007@iitbbs.ac.in	8750931785
Yatin Dhiman	yatindhiman603@gmail.com	6284725893
Ayush Kumar	ayushkr7130@gmail.com	8875231058

Content:

1. Training Data Set Generation

We began by using **Inkscape(open-source vector graphics editor)** to draw various curves and shapes, which were then exported as SVG files. These SVG files were uploaded to an open-source **Coordinator** tool available on GitHub. Within the coordinator, we adjusted the point density and radius to meet our specific requirements. Finally, we downloaded the processed data as CSV files, which contained the x and y coordinates of all points along the curves. This dataset formed the basis for training our models.

2. Model Training:

2.1 For open curves:

Feature Extraction for Open Curves

In this section, we detail the process of feature extraction for open curves. We calculated the curvature at each point along the curve and used this feature as input for our machine learning models, specifically **Random Forest** and SVM. The results showed an accuracy of 80% with SVM and 100% with Random Forest, demonstrating the effectiveness of curvature as a distinguishing feature in our models.

SVM	Random Forest
80% Accuracy	Almost 100% Accuracy

2.2 For Closed Curves:

CNN Model for Closed Curves, Symmetry, and Occlusion

In this section, we describe the process of training our models for closed curves, symmetry detection, and occlusion handling. We began by converting the 1D array of x and y coordinates into their nearest rounded integers, then resampled this data into a 2D Numpy array. This 2D array was used as input for our CNN model, which utilized the **VGG16 architecture (224x224x3)**. The model achieved an accuracy range of 85-90%, demonstrating its effectiveness in handling these tasks**.**

VGG16's use of small 3x3 convolutional filters and a deep architecture enables it to capture intricate details and hierarchical features in images, making it exceptionally effective for image classification tasks.

3. Workflow:

Task 1 : Regularization

3.1 CSV File Import:

Begin by importing a CSV file that contains data for various curves. The file has four columns:

- **Column 1**: Identifies the curve number.
- **Column 2:** Zero value.
- **Column 3**: Contains the X coordinates of the curve.
- **Column 4**: Contains the Y coordinates of the curve.

Curve Number	NULL	X Coordinate	Y Coordinate

3.2 Curve Splitting:

Use a custom function to analyze the curves based on a threshold angle. This function splits the curves into segments wherever there's a significant change in direction.

3.3 Open Curve Classification:

Import an open curve classifier model that processes these segments. This model specifically identifies and regularizes straight lines within the curve data.

3.4 Closed Curve Classification:

Segments that are not straight lines are passed to a closed curve classifier. This model uses a depth-first search (DFS) algorithm to detect and regularize common geometric shapes such as circles, rectangles, ellipses, regular polygons, and stars.

3.5 Final Output:

Curves that are not detected by either classifier are being output as they are, without modification. The curves that are successfully classified are shown after regularization.

Task 2 : Symmetry Detection

The aim was to detect any of three types of line of symmetry : vertical, horizontal, and diagonal, in the closed shape present in the given input. Hence, we implemented **3 individual CNN models** as described above, each for detecting each type of symmetry.

Output : The closed figures which are detected to have a particular type of symmetry are shown in the output with respective type of line of symmetry labeled on top.

Task 3 : Occlusion removal

In this task, we were supposed to complete a regular curve which has been occluded by another object. There are 2 types of Occlusions : connected and disconnected and our approach covers only the connected occlusion.

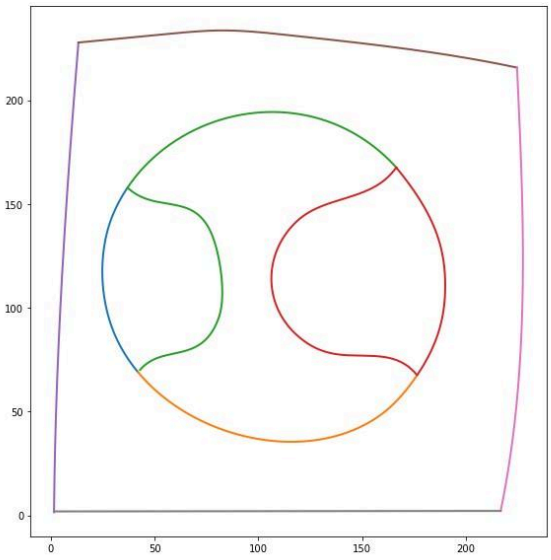
Again we started by generating a small dataset and training a **CNN Model** as described above. We built a classification model to detect closed curves that are regular shapes but occluded. The model accuracy remains to be around 75-80 % since the dataset generated was manual and hence not large and robust.

Output : On successful identification, the regular occluded curve is completed and regularized.

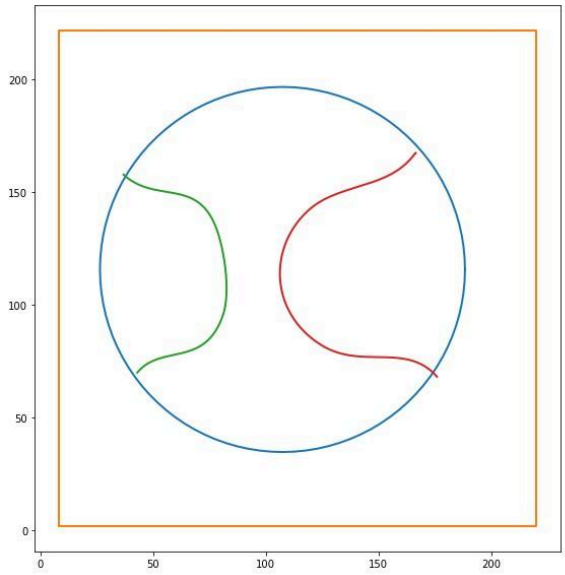
4. Sample Outputs of our model for each Task :

4.1. Regularization :

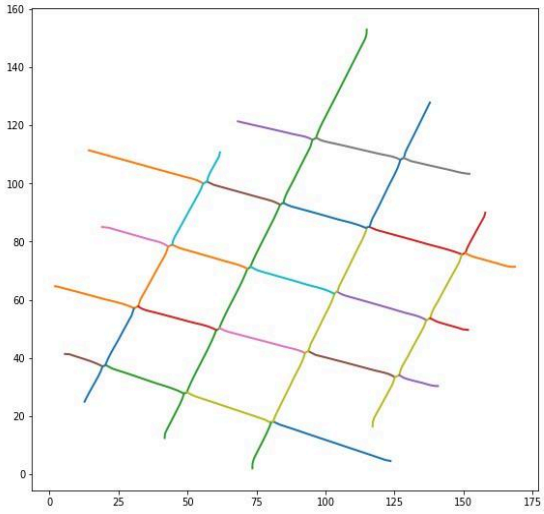
Input figure :



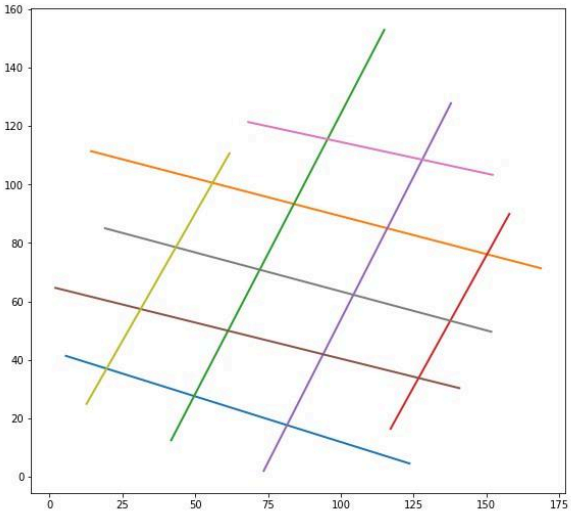
Output Figure :



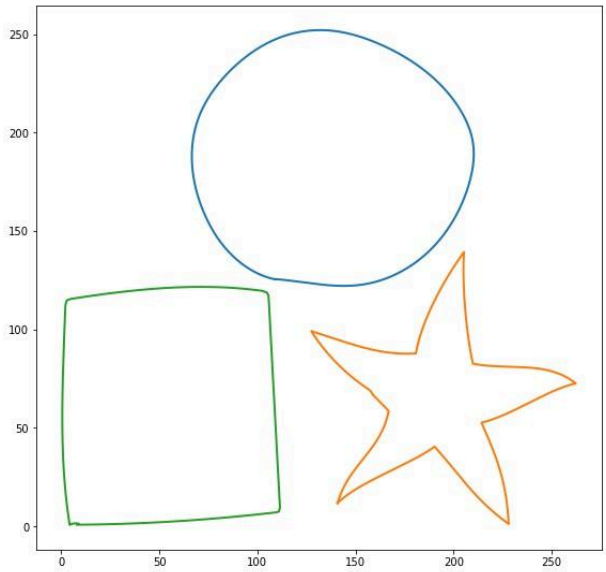
Input figure :



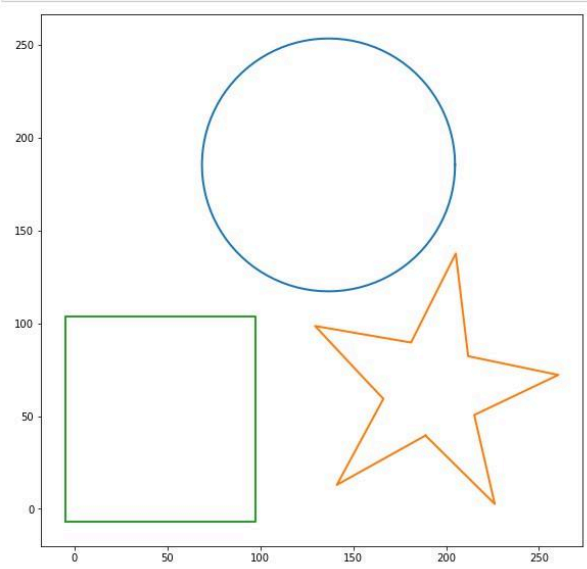
Output Figure :



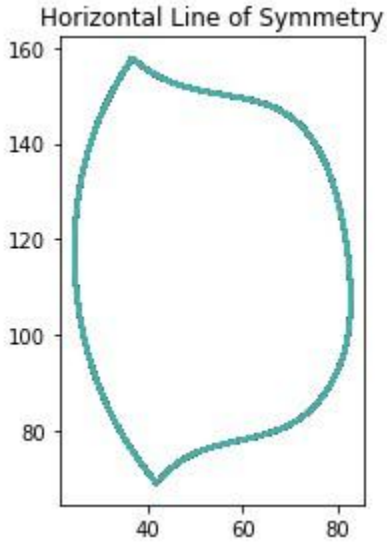
Input figure :

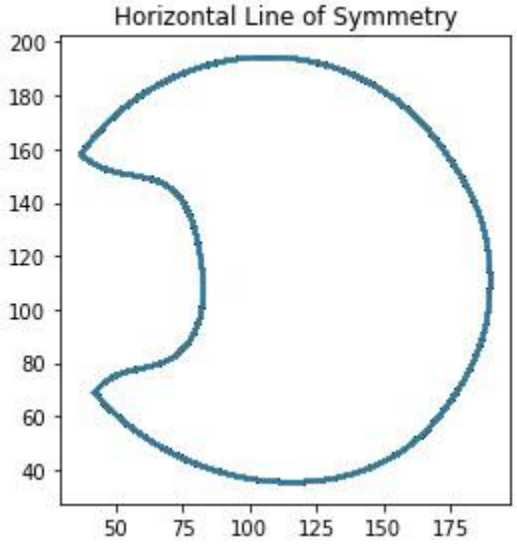


Output Figure :

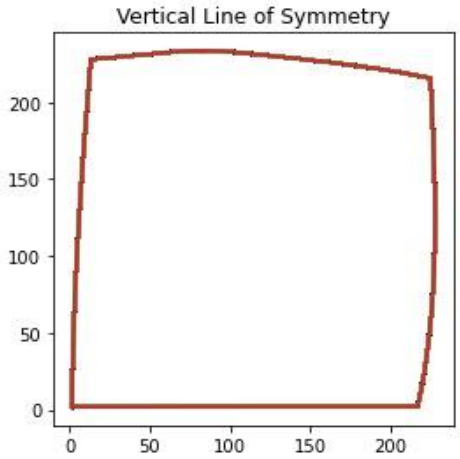


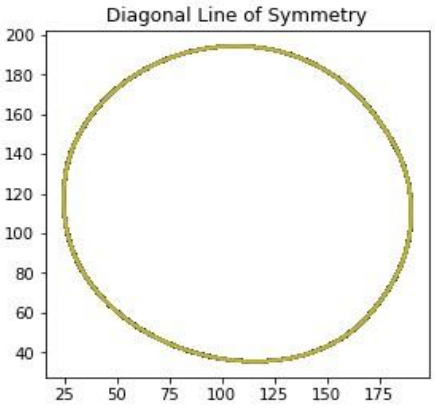
4.2. Symmetry Detection :

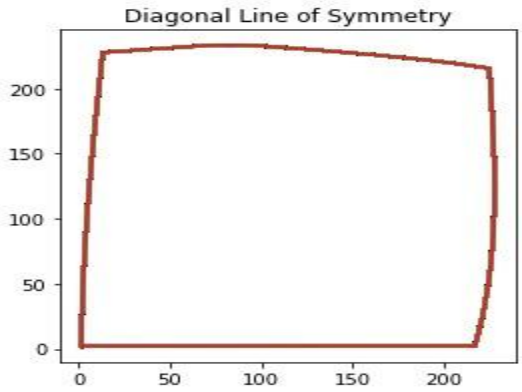




4.3 :

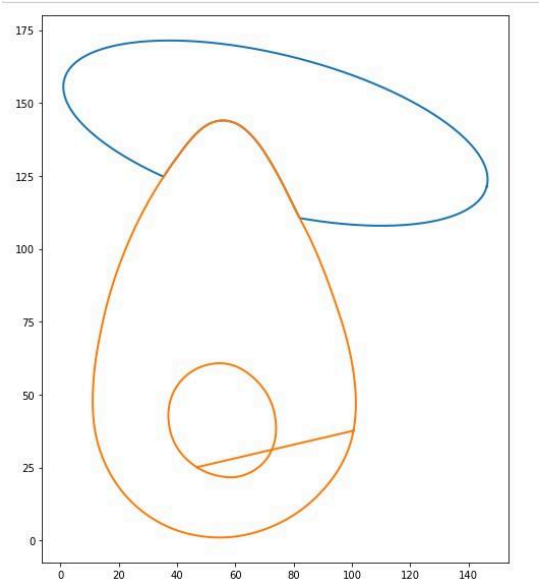






4.3. Occlusion :

Input Figure :



Output Figure :