20MCA245 MINI PROJECT

SYNOPSIS

Name of Student: Abdul Hakeem. P

Roll No: 1

Title: 2D Graph Interpretation and 3D Modeling for Polar and Parametric Functions

Synopsis Approved by Guide: Yes / No

Name of Guide: Dr. Reena Murali

Signature of Guide with date:

Name of Coordinator: Dr. Reena Murali

Signature of Coordinator:

1. Title of the Project

2D Graph Interpretation and 3D Modeling for Polar and Parametric Functions

2. Problem Statement

Traditional CAD tools require significant technical expertise and lack the ability to semantically interpret mathematical graphs. Specifically, curves such as polar and parametric functions are rich in visual and structural traits but are not easily mapped to meaningful 3D forms using standard modeling platforms.

This project proposes a system that semantically interprets 2D mathematical graphs — such as rose curves, spirals, and Lissajous figures — and converts them into functionally or aesthetically valuable 3D models. The system bridges the gap between mathematics and design by automatically detecting curve characteristics and mapping them to real-world applications using STL exportable models.

3. Objectives

- To semantically interpret 2D mathematical graphs (polar and parametric)
- To detect traits like symmetry, lobes, closure, and periodicity
- To classify curves (e.g., rose, spiral, Lissajous) using visual geometry
- To generate 3D models that are either functional (e.g., stamps) or aesthetic (e.g., pendants)
- To export models in .stl format
- To provide natural-language feedback on how graph traits affect 3D design
- To enable real-time graph preview and STL rendering in the browser

4. Proposed System

The proposed system enables users to convert 2D graphs of mathematical expressions — especially polar and parametric functions — into meaningful 3D printable models through a structured, automated pipeline. The process begins with the user inputting a mathematical expression, such as a polar equation like $r = \cos(3\theta)$, into the web interface built using React. The system then uses this expression to generate a 2D graph using mathematical libraries such as SymPy and NumPy, and visualizes it in the frontend using tools like Matplotlib (as static image).

Once the graph is plotted, the system performs graph interpretation, where it analyzes key geometric traits including symmetry, number of lobes, periodicity, and whether the curve is closed or open. Based on these detected traits, the system classifies the curve into a known type (e.g., rose curve, spiral, or Lissajous figure) and determines its potential application, such as a pendant, keychain, or educational tool.

Both the 2D graph and the 3D STL model are displayed in the browser. The 2D graph preview helps users visually understand the shape of the mathematical function, while the 3D model is rendered interactively using Three.js within the React UI. Finally, the system provides a natural-language explanation that describes how the mathematical features of the input graph influenced the final 3D design. This complete pipeline supports interactive learning, creativity, and functional modeling for users with or without prior CAD experience.

5. Required Technologies

- **Frontend:** React
- **Backend:** Flask
- **3D Modeling Tools:** Trimesh, Three.js, STLLoader.js

6. Expected Outcome

- A fully functional web app that transforms 2D mathematical curves into .stl files for 3D printing
- A browser-based STL viewer using React and Three.js for real-time 3D model preview
- Curve classification and educational explanations on how mathematical traits influence design
- Downloadable and printable 3D model output without requiring OpenSCAD

7. Tentative Work Plan

- **Week 1:** Literature survey, finalize scope and tools
- **Week 2:** UI design using React and setup Flask backend

- **Week 3:** 2D graph plotting and trait detection
- **Week 4:** Curve classification logic
- **Week 5:** 3D model extrusion using Trimesh and STL export
- **Week 6:** STL preview integration in React using Three.js
- **Week 7:** Natural-language explanation generation
- **Week 8:** Testing, debugging, and UI polishing
- **Week 9:** Final report writing and submission