Dissertation on

"An Entity to 3D model Prototyping from a Photo"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE20CS390B – Capstone Project Phase - 2

Submitted by:

NIKITA S PATGAR	**PES1UG21CS825**
RAVALLU NIKHIL RAJAREDDY	**PES1UG21CS830**
TITEERSHA GHATAK CHOWDHURY	**PES1UG21CS834**
VAISHNAVI K	**PES1UG21CS838**

Under the guidance of

Prof. Nitin.V.Pujari
Dean of IQAC
PES University

August - December 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'An Entity to 3D model Prototyping from a Photo'

is a bonafide work carried out by

Nikita S Patgar	**PES1UG21CS825**
Ravallu Nikhil Rajareddy	**PES1UG21CS830**
Titeersha Ghatak Chowdhury	**PES1UG21CS834**
Vaishnavi K	**PES1UG21CS838**

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE20CS390B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period August- December 2023. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature **Prof.Nitin.V.Pujari** Dean of IQAC	Signature Dr. Mamatha H R Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **“An Entity to 3D Model Prototyping from a Photo”** has been carried out by us under the guidance of Prof. Nitin.V.Pujari, Dean of IQAC, Internal Assurance Cell and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester August - December 2023. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1UG21CS825	**Nikita S Patgar**	_____
PES1UG21CS830	**Ravallu Nikhil Rajareddu**	_____
PES1UG21CS834	**Titeersha Ghatak Chowdhury**	_____
PES1UG21CS838	**Vaishnavi K**	_____

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Nitin.V.Pujari, Department of Computer Science and Engineering, PES University, for his continuous guidance, assistance, and encouragement throughout the development of this UE20CS390B - Capstone Project Phase – 2.

I am grateful to the project coordinator, Dr Priyanka H, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Mamatha H R, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

Innovative features extraction and modeling techniques revolutionize how 2D facial images are transformed into 3D prototypes. It uses sophisticated algorithms to capture and recreate the smallest details of facial features, such as expressions, contours, and textures. In addition to aesthetic replication, this capability offers an additional dimension of depth and accuracy to facial representations.

After the creation of these highly accurate 3D models, the project takes an important step towards ensuring practical utility by converting them to G-code. This programming language acts as a link between the virtual and real worlds, allowing 3D prototypes to be seamlessly integrated into a variety of applications. G-code translation enables specialists across industries to bring these 3D models to life, from influencing the design of consumer products with lifelike features to facilitating realistic architectural representations and boosting characters in video game production. By combining powerful image processing techniques.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

This project is centered around the conversion of human faces from photographs into three-dimensional (3D) models. The software, designed with advanced recognition capabilities, not only identifies faces in a photo but also meticulously categorizes them into distinct folders. Each folder is named after the individual, culminating in an organized compilation of facial profiles, primed for further development.

The key feature of this project is its capacity to transform these chosen faces into virtual 3D models. The journey of these models doesn't end here; they are subsequently converted into Stereolithography (STL) files, a specific format that is compatible with 3D printing technology.

Finally, to bring these models to life, the project incorporates the use of a third-party tool, 'Cura Engine', which facilitates the generation of G-code. This code serves as the final step in the 3D printing process, guiding the movement of the printer to actualize the 3D models.

This project stands at the intersection of facial recognition, 3D modeling, and 3D printing technologies, ultimately enabling the transformation of faces from mere photographs into 3D objects.

CHAPTER-2

PROBLEM STATEMENT

This project's goal is to create a comprehensive system for face recognition, 3D modelling. The system is initially tasked with scanning an input image, identifying and recognizing faces within it, and storing each recognized face in a distinct folder built based on the individual's face identification. This step involves implementing facial detection and recognition algorithms, as well as organizing and storing the discovered faces for easy retrieval.

The system expands its functionality to allow users to select a recognized face from the extracted set, building on its face recognition capability. Upon selection, the system begins converting the selected face into a detailed 3D virtual model. This involves gathering facial traits and translating them into a full 3D model, which adds realism to the digital models.

The project's last phase involves converting the 3D virtual model into STL (Stereolithography) and G-code formats. These formats are critical for digital fabrication techniques like 3D printing because they ensure interoperability with a variety of production equipment. This research addresses the multidimensional task of translating 2D facial photos into physical 3D things by easily combining face recognition, 3D modelling, and digital fabrication, giving applications in varied domains such as personalized manufacturing and creative design.

CHAPTER-3

LITERATURE REVIEW

3.1 FACE DETECTION AND RECOGNITION

3.1.1 Face Detection and Recognition Using Machine Learning

Face recognition is a rapidly growing and exciting field with numerous applications. Over the years, many face recognition algorithms have been developed. In this work, we chose the HOG (Histogram of Oriented Gradient) based face detector over other machine learning algorithms such as the Haar Cascade because it gives more accurate results. We employ CLAHE (Contrast Limited Adaptive Histogram Equalisation) for pre-processing during the recognition phase, followed by HOG, a widely used approach for feature extraction. HOG features have been retrieved from both the training and test images. Finally, for categorization, we employ SVM (support vector machine). SVM is used to categorise the HOG features. Pre-processing techniques are used to eliminate noise, improve contrast, and balance illumination. This study's findings demonstrate the risks and benefits of improving facial recognition accuracy.

Pre-processing is highly valued in the realm of image processing. Histogram equalisation is an image processing technique for adjusting the intensity of an image. As a result, the contrast of an image improves. It could be explained using a histogram. The histogram is said to be equalised when the image uses all of the grey levels equally. The intensities in the histogram are so uniformly distributed. CLAHE is an upgraded variant of AHE.

The initial stage of calculation is calculating the gradient values. The first technique uses one-dimensional derivative masks in both the vertical and horizontal axes. The second step in obtaining HOG properties is binding by orientation. According to the amount of data provided in

the gradient computation, each pixel within the cell casts a weighted vote for a histogram channel based on the orientation. The channels are spread over a range of 0 to 3600 or 0 to 1800, depending on whether the gradient is specified, and the cells can be either outspread or rectangular in shape.

3.1.2 Face Recognition and Identification using Deep Learning Approach

The face is the most important factor in recognising someone. Even identical twins have distinct facial characteristics. Identification and facial recognition are essential to distinguish between distinct people. A face recognition system is a method for verifying a person's identity that uses biometrics. Face recognition technology is widely used in a variety of applications today, including home security systems, criminal identification, and phone unlocking methods. This approach is more secure because it only requires a facial image to function rather than a key or card. Face detection and face identification are the two methods that make up a person recognition system.

This paper presents the notion of constructing and building a face recognition system using deep learning and OpenCV in Python. Face recognition can be performed using deep learning, and considering the task's high accuracy, this method appears acceptable.

The Haar Cascade classifier is a Haar feature-based cascade classifier that is used in face detection. A Haar Cascade is, in essence, a classifier that is used to identify the object for which it was trained from the source. The Haar Cascade is generated by superimposing a positive image over a sequence of negative photographs. Typically, training is performed on a server and in stages. Increase the number of steps for which the classifier is trained, and use high-quality pictures, and you'll get better results.

TensorFlow is the framework used in the system classifier section. A classifier is trained to recognise items. Training a classifier to get better takes a long time. The classifier improves as more

training runs are completed. The proposed face recognition system requires three days to train. By prolonging the training period, the loss can be reduced even further while the precision is increased.

3.1.3 Face Detection Techniques: A Review

The method by which computerised systems can recognise a face simply by looking at it is referred to as "face detection" in one form of biometric technology. Biometrics, digital cameras, and social tagging are examples of popular applications. Face detection and identification research has grown in recent years. In this study, we studied numerous face detection algorithms and tested them using the MATLAB software.

We want to recognise a collection of geometric traits such as nose width and length, mouth position, chin shape, and so on from a facial image. This set of features is then compared to those of known individuals. A suitable metric, such as Euclidean distance (identifying the nearest vector), can be used to discover the closest match.

The advantage of using geometrical features as the foundation for face recognition is that recognition may be achieved even with noisy images and poor resolutions.

3.2 IMAGES TO 3D MODEL PROTOTYPE

3.2.1 From Images to 3D Models

The paper focuses on the importance of computer graphics for 3-D (3D) models. Computer graphics enable the creation of realistic views of the 3D environment. However, in order to generate these visuals, the world must first be graphically represented. There are several approaches available. Laser range scanners and other technologies that shine light on an object and collect 3D data by monitoring the reflection are probably the most well-known. The simplest way to

understand how a 3D shape might be created is to consider how humans naturally see the 3D world. Depth can be felt by seeing a three-dimensional world from two slightly different viewpoints.

First, the relative motion between successive photos must be recovered. This strategy goes hand in hand with finding comparable picture features—that is, image points that come from the same 3D feature—between these photos. Following that is the process of recovering the camera's motion, calibration, and 3D structure of the features. This procedure is divided into two stages. The reconstruction initially has a projective skew, which means that parallel lines are not parallel, angles are inaccurate, lengths are either too long or too short, and so on. As a result, no a priori calibration exists.

To reconstruct these locations as well, the next stage entails attempting to match all of the image pixels in a picture with pixels in surrounding images. The reconstruction mentioned in the preceding phrase has only a sparse set of 3D points (only a few features are initially taken into account).

3.2.2 Image based 3D Modelling

This study discusses the key difficulties and potential solutions for creating 3D models from terrestrial images. Close-range photogrammetry has long dealt with either manual or automatic picture measurements for accurate 3D modelling. Although image-based modelling is still the most thorough, inexpensive, portable, adaptive, and widely-used technology, 3D scanners are becoming a frequent source of data input in many application fields today. This paper provides the entire process for 3D modelling from terrestrial image data, taking into consideration alternative methodologies and assessing all steps.

IBR stands for image-based rendering. This precludes the production of a geometric 3d Model, but it may be considered a valuable technique for creating virtual views for specific items

and under specific camera motions and scene conditions (Shum and Kang, 2000). IBR creates new viewpoints of 3D environments directly from source images. The method is based on either having a perfect understanding of the camera positions or using automatic stereo matching, which requires a large number of closely spaced images in the lack of geometric data.

Modelling based on range. This method immediately collects an object's 3D geometric information. It is capable of producing highly exact and detailed representations of most shapes and is based on pricey (at least for the time being) active sensors. These sensors are nonetheless expensive, designed for certain applications or ranges, and affected by the surface's reflecting qualities. They must have some grasp of each specific technology's capabilities within the defined range, and the resultant data must be filtered and adjusted. Most systems only focus on 3D geometry capture, providing only a monochromatic intensity value for each range value.

3.2.3 Creation of prototype 3D models using RAPID PROTOTYPING

The paper is about using RAPID PROTOTYPING to produce 3D models. New technologies are being integrated into everyday life. A good example is the use of FDM (fused deposition modelling) technology, which primarily employs thermoplastics to create 3D models. Fast prototyping technologies were previously exclusively available to particular sorts of corporations, research organisations, and universities. In recent years, technologies such as FDM and STL (Stereolithography) have become more affordable for smaller organisations and individuals. The centrepiece of this chapter is the RepRap (replicating rapid prototype), of which the extended version is the Prusa i3 printer.

The first technique used was stereolithography (SLA - Stereolithography). It sparked the development of later iterations of 3D printing prototyping techniques. The STL (Stereolithography) format has become the industry standard for model processing. All CAD programmes used in engineering practice, as well as various commercial and non-commercial modelling tools, already

support it. As the major component, laser-cured resin was used. After application and curing, adhesive-treated laminated paper or film was used, which was then laser-shaped to the appropriate shape.

SLS is the most commonly utilised technique since it makes models in a variety of solid materials and uses the material entirely. The item's high price and massive size are disadvantages. Consumption costs are relatively high per kilogramme of material consumed. This is not deciding in a business where strength and optimal material use are the most critical issues. If someone wishes to use a 3D printer for personal purposes, they should pursue a different path. This individual has modest purchasing requirements, low energy requirements, high dependability, and low consumable material expenses.These standards are particularly compatible with FDM (Fused Deposition Modelling) technology, in which the thermoplastic is extruded through the nozzle, cooled, and then the layers are formed. The full model is made as a result.When selecting the printer, the following factors were taken into account: printing area, cost of consumables, construction resistance, and potential for improvement.

3.3 IMAGE TO G-CODE GENERATION

3.3.1 Image-based Contouring and G-code Generation for Additive Manufacturing

A novel method for obtaining G-codes for additive manufacturing (AM) utilising voxel-based models is proposed in this research. During the pre-processing stage, the suggested technique divides the foreground voxels into skin and interior portions. The model is then separated into layers along the z-axis, and each layer is hatched one at a time to generate the G-codes required to construct the desired result. Before processing a layer, the proposed approach groups all non-zero pixels into clusters. The toolpaths are then traced and translated into G-codes for each cluster to be

printed. The skin and internal sections are built by using 3D image morphological procedures on the input model, which is treated as a 3D picture.

As a result, the pre-processing method has been greatly simplified.The suggested approach employs a texture-mapping technique to implement the hatching of the 2D layers. Users can change the angles and resolutions of the infilling patterns by utilising geometrical transforms. As a result, the hatching technique becomes more adaptive and effective. The proposed method also employs a novel strategy to encoding toolpaths during the hatching process, resulting in less G-codes created. The compression of the G-code programme improves the efficiency of the additive manufacturing (AM) process.

Each cluster is treated separately when creating the toolpaths and G-codes. A cluster undergoes two stages of hatching. In the first step, the cluster's skin region pixels are rasterized to generate the G-codes that print the cluster boundary. In the second stage, the inside component of the cluster is hatched by producing G-codes using a texture-mapping method and filling it in according to prescribed patterns.

3.3.2 A G-Code Generator for Volumetric Models

Slicers are used in layered manufacturing (LM) to convert input geometric models into G-codes. Surface models are the sole input data that standard slicers allow. Volumetric models mustbe translated into polygonal representations to fit the data format of the slicers. Geometric errors and increasing computational costs ensue. In this research, we show how an efficient slicer can generate G-codes for volumetric models.

The recommended slicer involves several stages. In the first step, the input model is positioned to maximise stability while decreasing height. Support structures are required for some LM modalities to prevent printed products from collapsing during the printing process. The

suggested slicer's second stage entails the identification and classification of overhangs, as well as the generation of support structures to aid the input model. Our slicer separates the skin and internal portions of the input model, then slices the input model into a stack of 2D photos for the subsequent computation. Following that, a contouring technique is employed to generate toolpaths that create a high-quality skin surface.

The input model's skin should be densely printed in an LM process to provide a smooth surface, while the interior space is filled in accordance with a predetermined pattern to shorten the printing process and lessen the end product's weight. To carry out the remaining computations of the G-code generation process, the skin and inner portions must be separated. Our G-code generator continuously erodes the foreground voxels to create the skin region. The number of erosions affects the thickness of the skin region. As there are more erosions, the skin region thickens. In this work, the amount of erosions is left up to the users. Many clusters of skin and inner pixels may form in each slice, depending on the shape of the input model. We must utilise a connected-component labelling strategy to cluster the foreground pixels. The toolpaths are created by hatching each of these clusters separately.

3.3.3 G-code Modeling for 3D Printer Quality Assessment

This paper describes our efforts to analyse infill mistakes since the quality of the infill affects the structural integrity of the component. Even if a part looks okay from the outside, the infill can make it structurally sound. Furthermore, once a part has been printed, the infill is usually hidden. So, when the item is being printed, the infill must be monitored-code. Parser is a numerical control (NC) programming language that facilitates the manufacture of goods by instructing the motors of computerised machine tools where to go, how rapidly to move, and what path to take.The OpenGL parser only used two of the possible G-code commands. The two commands are G0 and

G1. The remaining commands either setup the printer, such as altering the bed temperature or height, or switch the coordinate system from absolute to relative placement.The first step is to erase every single comment. G-code has two alternative ways of representing comments. The first method employs a semicolon. The parser will look for the semicolon at the end of a line. If that happens, everything after the semicolon will be eliminated. The second approach for distinguishing a comment is to use brackets. The parser searches the rest of the line for the closing parenthesis after detecting the initial parenthesis. After locating the ending parenthesis, it will remove the substring containing the comment. The second stage of extraction is determining whether a command is a G0 or G1. To accomplish this, it examines the line's opening character. The next character in the queue is checked to see if it is a "G." It moves on to the next line if the initial character is not a "G." If the second character is a "0," the rest of the characters on the line are passed to a function that executes the G0 command. If the second character is a "1,," the rest of the line is passed to a function for the G1 command.

CHAPTER-4

PROJECT REQUIREMENT SPECIFICATION

4.1 Project Overview

4.1.1 Project Purpose

Develop a software system that can convert human faces from photographs into 3D models and generate G-code for 3D printing.

4.1.2 Project Scope

The system should be able to handle a variety of input image formats, including JPEG, PNG, and BMP. The generated 3D models should be realistic and compatible with 3D printing systems.

4.1.3 Project Objectives

1. Identify and extract facial features from input images.
2. Generate virtual 3D models from recognized faces.
3. Convert virtual 3D models into STL files.
4. Extract G-code for 3D printing.

4.2 Product Vision

The product vision is to create a user-friendly software system that can seamlessly convert human faces from photographs into 3D models and generate G-code for 3D printing. The system should be efficient and easy to use.

4.3 Operating Environment

- Hardware platform: A desktop or laptop computer with at least an Intel Core i5 processor, 8GB RAM, GPU and a dedicated graphics card.
- Operating System: Windows 10 or macOS 10.14 or later is required.
- Software Components: Python 3.7 or later, OpenCV, TensorFlow, NumPy, and the Flask web framework are required.

4.4 Functional Requirements

1. Face Recognition: Identify and extract facial features from input images.
   Categorize recognized faces into distinct folders named after individuals.
2. 3D Model Generation:Generate virtual 3D models from recognized faces.
   Ensure realistic representation of facial features, including eyes, nose, and mouth.
3. STL File Generation:Convert virtual 3D models into STL files.
   Ensure accurate representation of facial structures.
4. G-code Extraction:Extract G-code from STL files.
   Provide instructions for 3D printers for optimal layer-by-layer construction.

4.5 Non-Functional Requirements

1. Performance:The system should efficiently process input images and generate 3D models.
   The system should be able to handle multiple simultaneous conversions.
2. Usability:The user interface should be intuitive and easy to use.

The system should provide clear instructions for uploading images and managing conversions.

3. Reliability:The system should be stable and reliable.

It should be able to handle unexpected input images and errors.

4.6 Project Deliverables

- Software Application: A fully functional software application that can convert human faces from photographs into 3D models and generate G-code for 3D printing.
  User documentation for the software application.
- 3D Models: A collection of virtual 3D models generated from input images.
  STL files corresponding to the generated 3D models.
- G-code: G-code generated from STL files for 3D printing.

4.7 Success Criteria

The project will be considered successful if it meets the following criteria:

1. The software application is able to convert human faces from an image into 3D models and generate G-code for 3D printing.
2. The generated 3D models are realistic and compatible with 3D printing systems.
3. The user interface is intuitive and easy to use.
4. The system meets all performance, usability, and reliability requirements.

CHAPTER-5

SYSTEM DESIGN

5.1 System architecture

The system will be a three-tier architecture consisting of the following components:

1. Presentation Tier
2. Application Tier
3. File organization

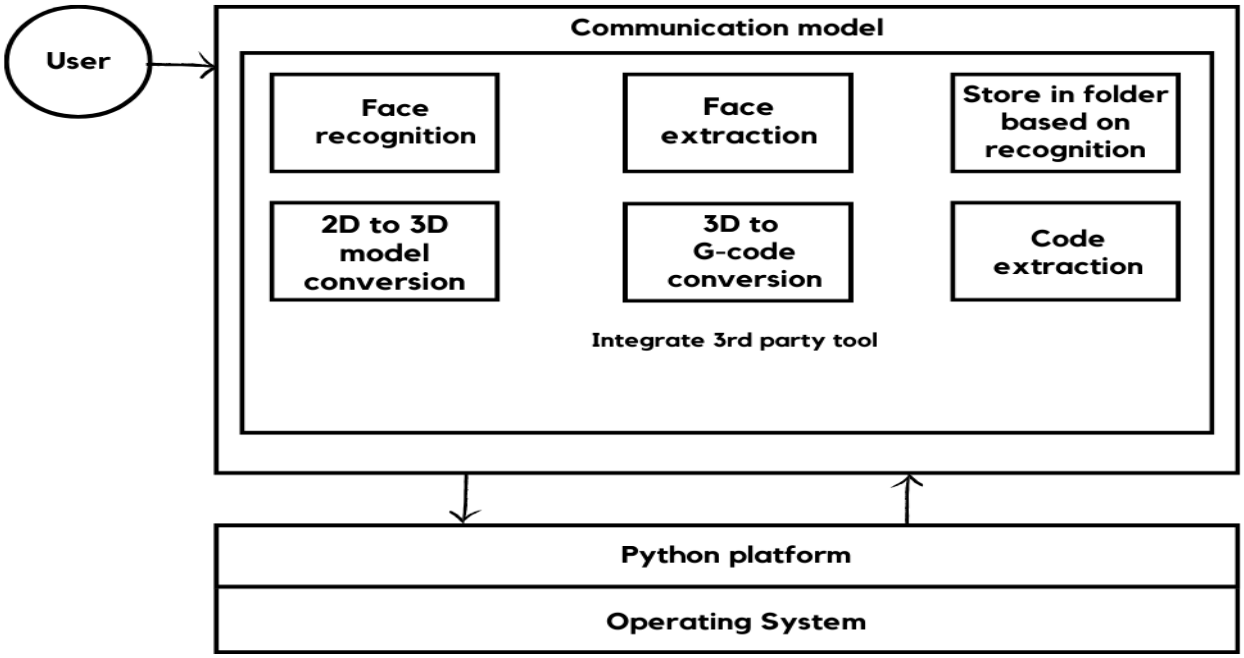

Figure 1: High level Diagram

5.1.1 Presentation tier

The presentation tier will be responsible for handling user interactions and displaying the results of the system. It will be implemented as a web application using a modern front-end framework such as React. The presentation tier will incorporate the following elements:

- User Interface (UI): A user-friendly interface for uploading images, displaying generated 3D models and G-code, and providing a good user experience.
- Image Upload: A mechanism for users to upload images containing faces for processing.
- 3D Model Visualization: A 3D model viewer to display the generated 3D models, allowing users to manipulate and inspect the models from different perspectives.
- G-code Download: An option for users to download the generated G-code for 3D printing.

5.1.2 Application tier

The application tier will contain the core business logic of the system, including the face recognition algorithm, 3D model generation, STL file generation, and G-code extraction. It will be implemented as a REST API using a programming language such as Python. The application tier will include the following modules:

- Face Recognition Module: Responsible for identifying and extracting facial features from input images. It will utilize a pre-trained face recognition model such as Dlib or OpenCV.
- 3D Model Generation Module: Responsible for generating virtual 3D models from recognized faces. It will employ a 3D modeling library PyMesh.
- STL File Generation Module: Responsible for converting virtual 3D models into STL files. It will leverage an STL file generation library such as PyMesh.
- G-code Extraction Module: Responsible for extracting G-code from STL files. It will utilize a G-code generation tool such as CuraEngine.

5.1.3 File organization

The system will use a structured directory structure to organize the files. For example, the following directory structure could be used:

```
root
├── images
│   ├── person1
│   │   ├── image1.jpg
│   │   └── image2.jpg
│   ├── person2
│   │   ├── image1.jpg
│   │   └── image2.jpg
│   └── ...
├── 3d_models
│   ├── person1.obj
│   ├── person2.obj
│   └── ...
├── stl_files
│   ├── person1.stl
│   ├── person2.stl
│   └── ...
└── gcode_files
    ├── person1.gcode
    ├── person2.gcode
    └── …
```

File Naming: Each file will be named using a unique identifier and a descriptive extension. For example, an image file could be named person1_image1.jpg, a 3D model file could be named person2.obj, an STL file could be named person3.stl, and a G-code file could be named person 4.gcode

5.2 System Integration

The system components will be integrated using a REST API. The presentation tier will make HTTP requests to the application tier to perform the face recognition, 3D model generation, STL file generation, and G-code extraction operations.

5.3 System Workflow

The following is a detailed workflow of the system:

- User Upload: The user uploads an image to the system through the presentation tier's image upload interface.
- Face Recognition: The presentation tier sends an HTTP request to the application tier's face recognition module, providing the uploaded image as input.
- Facial Feature Extraction: The face recognition module processes the image, identifying and extracting facial features using the pre-trained face recognition model.
- 3D Model Generation: The application tier sends the extracted facial features to the 3D model generation module.
- Virtual 3D Model Creation: The 3D model generation module utilizes the facial features to generate a corresponding virtual 3D model.
- STL File Generation: The application tier sends the virtual 3D model to the STL file generation module.
- STL File Conversion: The STL file generation module converts the virtual 3D model into an STL file, compatible with 3D printing systems.
- G-code Extraction: The application tier sends the STL file to the G-code extraction module.
- G-code Generation: The G-code extraction module processes the STL file, generating the corresponding G-code instructions for 3D printing.

5.4 Design description

5.4.1 Use case Diagram

A use case is an approach for identifying, clarifying, and arranging system needs in system analysis. A use case is a collection of desirable sequences of interactions between systems and users in a specific environment that are tied to a specific purpose. The function generates a document that details all of the steps a user takes to accomplish an activity.

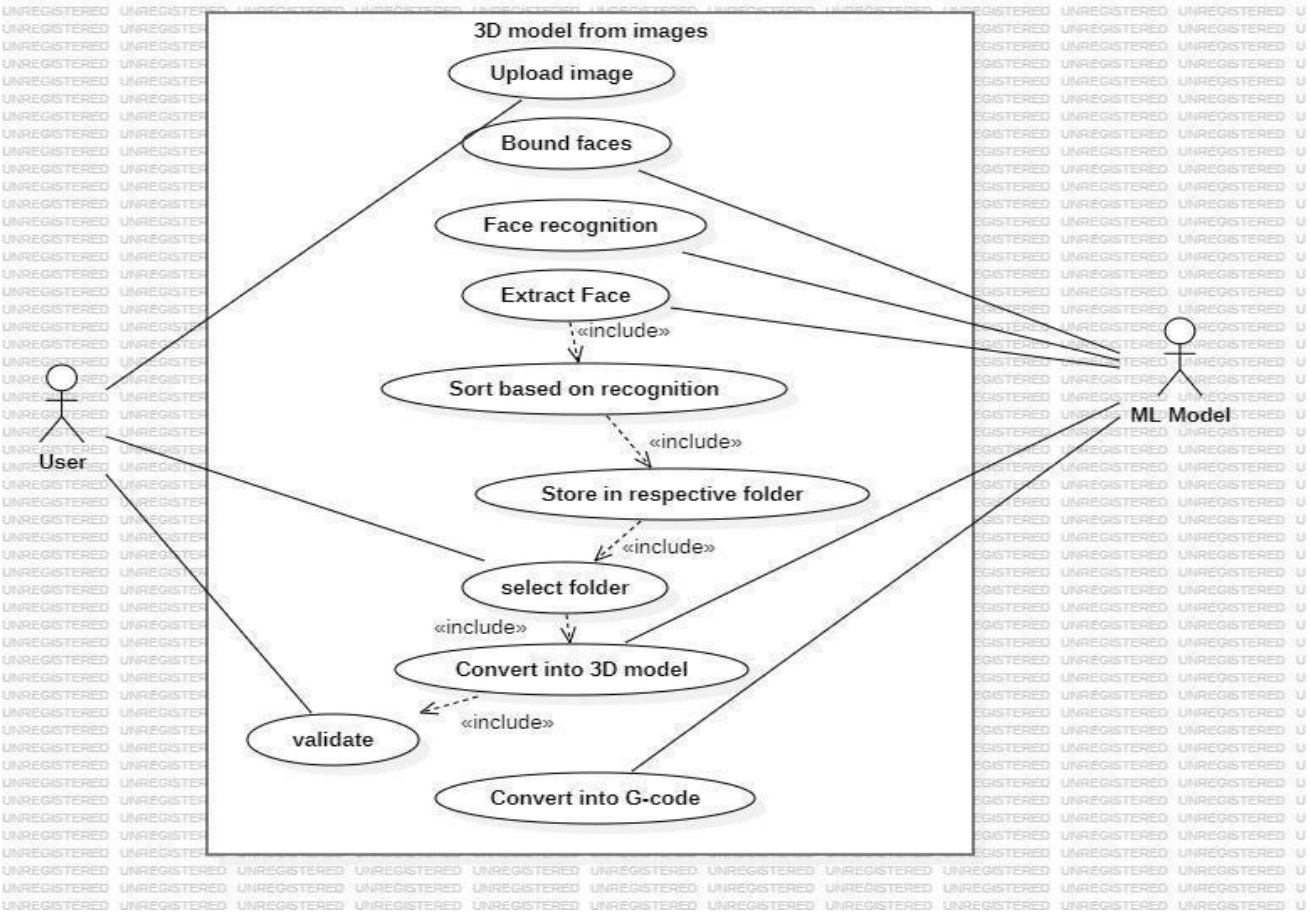

Figure 2: Use case diagram

CHAPTER-6

PROPOSED METHODOLOGY

The methodology for converting a 2D face from an image to a 3D virtual prototype involves leveraging computer vision and facial recognition algorithms to extract key facial features from the 2D image. Subsequently, these features are used to generate a three-dimensional representation, employing techniques such as 3D modeling and mesh reconstruction to create a lifelike virtual prototype of the original face.

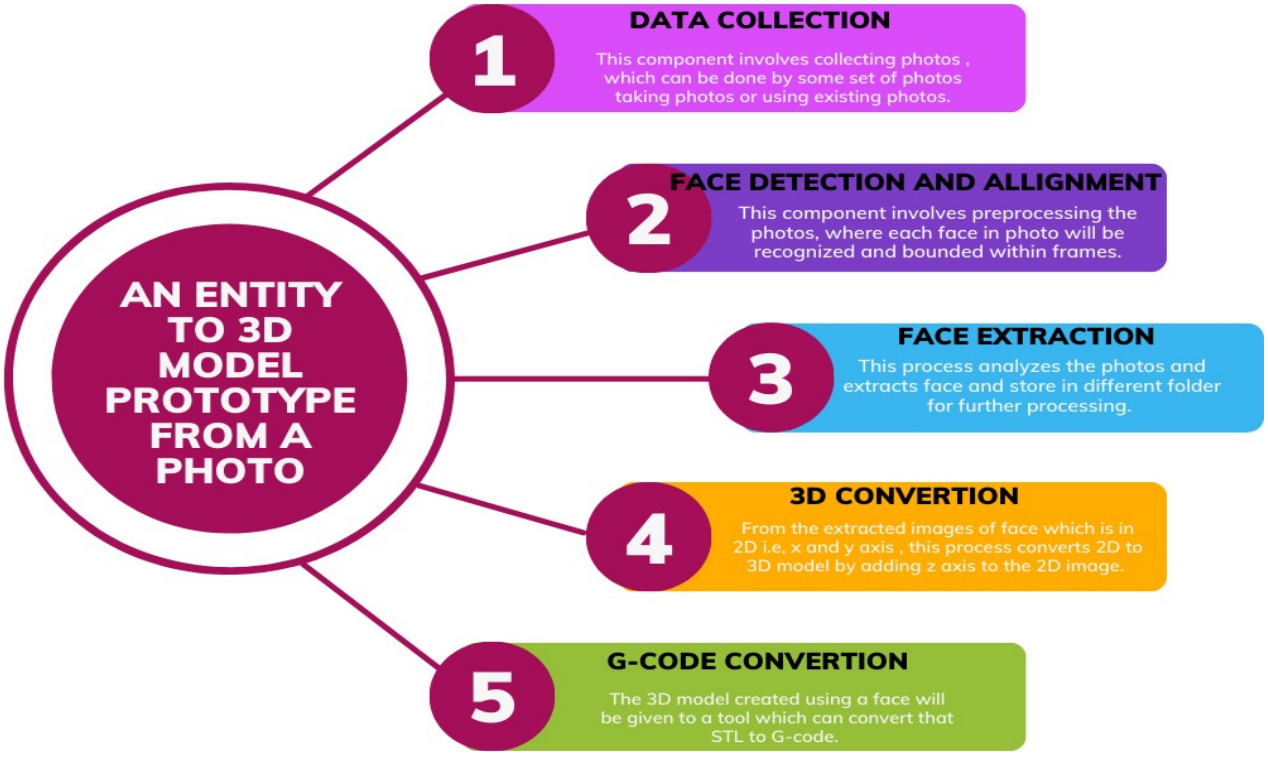

Figure 3: Working method of “An Entity to 3D model prototype from a photo”

Basically, first we made use of Multi-task Cascaded Convolutional Networks (MTCNN) which is a crucial tool for face detection and feature extraction from the input photos, enabling the identification and transformation of facial features into 3D models.

The tools used are:

- Matplotlib is being used for data visualization and the display of images and graphs, which aids in the analysis and representation of the 3D modeling process.
- Keras and TensorFlow are pivotal components for implementing a Convolutional Neural Network (CNN) model in Python. Keras provides a high-level interface for building and training neural networks, while TensorFlow serves as the deep learning framework that powers the underlying computations.
- NumPy (Numerical Python) is a powerful Python library specifically designed for working with arrays, matrices, and other numerical data structures. It provides efficient operations, a wide range of data types, and advanced mathematical functions, making it an essential tool for scientific computing, data analysis, machine learning, and various computational fields.

Here we have used few libraries which is very crucial for running the code:

- Flask is used to create a web application that allows users to upload images, and it processes these images using face recognition.
- Axes3d is included in the 3D plotting toolkit of Matplotlib. It is used to produce 3D graphs and charts. It lets you work with data in three dimensions and offers tools for adding axes (x, y, and z) to your graphs.
- OS is an operating system-dependent functionality in Python that uses the OS library. It enables you to interact with the Linux, Mac, or Windows operating system that Python is executing on. It has functions for managing directories and file paths, initiating and terminating processes, and reading and writing to the file system.

- Werkzeug is used in our code because it is a utility library for building web applications in Python, and Flask relies on Werkzeug for various web-related tasks.
- Pillow, a powerful image processing library in Python, is used in your code for a specific task related to image handling. Pillow's Image class is used to create an image object from the NumPy array representing the extracted face. The save method of the Image class is then used to save the face image to a specified file path.
- face_recognition is an open-source Python library for face recognition and facial feature analysis. It is built on top of the popular computer vision library Dlib and provides a user-friendly interface for working with facial data in images and videos. It supports face recognition by comparing the encodings of known faces with those of detected faces. This is useful for identifying or verifying individuals.
- CuraEngine is a command-line interface (CLI) for the Cura slicing software. Cura itself is a popular open-source 3D printing slicer that allows you to prepare 3D models for printing. CuraEngine is the core slicing engine used by Cura to generate G-code instructions for 3D printers.

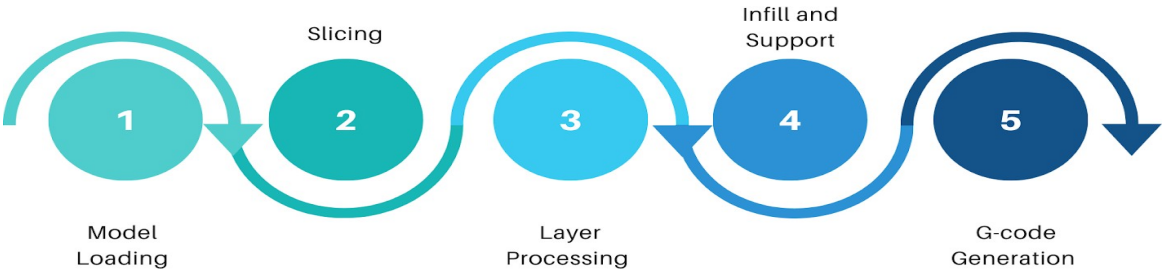

Figure 4: Workflow of Cura Engine

6.1 Data collection and preprocessing

1. Library Import: Import necessary libraries: Flask, request,werkzeug,face_recognition,and Image from PIL.
2. Configuration: Define the upload folder and allow file extensions for managing incoming images.
3. Initialization: Create empty lists to store known face encodings and corresponding names for recognition.
4. Face Recognition: Utilize the face recognition library to recognize and extract faces from the uploaded images.
5. Image Processing: Process the extracted faces using PIL for further analysis and modeling.
6. Data Storage: Organize the recognized face encodings and names for subsequent steps in the process.

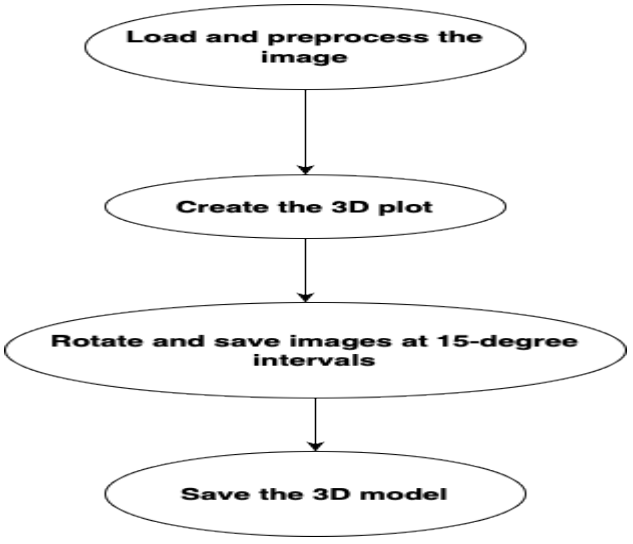

Figure 5: Flowchart

CHAPTER-7

IMPLEMENTATION AND PSEUDOCODE

7.1 Face extraction code

```
app = Flask(_name_)
app.template_folder = 'templates'

UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

known_face_encodings = []
known_face_names = []

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def load_known_faces():
    # Load known faces from the 'output' directory
    output_folder = 'output'
    for person_folder in os.listdir(output_folder):
        person_folder_path = os.path.join(output_folder, person_folder)
```

```
        if os.path.isdir(person_folder_path):
            known_face_names.append(person_folder)

            face_image_path = os.path.join(person_folder_path, f"{person_folder}_1.jpg")

            # Check if the face image file exists before attempting to load it
            if os.path.exists(face_image_path):
                face_image = face_recognition.load_image_file(face_image_path)
                face_encoding = face_recognition.face_encodings(face_image)[0]
                known_face_encodings.append(face_encoding)

def find_person(face_encoding):
    # Compare the input face encoding with known face encodings
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    for i, match in enumerate(matches):
        if match:
            return known_face_names[i]
    return None

def extract_faces(image_path, output_folder):
    image = face_recognition.load_image_file(image_path)
    face_locations = face_recognition.face_locations(image)
    face_encodings = face_recognition.face_encodings(image, face_locations)

    for i, (face_location, face_encoding) in enumerate(zip(face_locations, face_encodings)):
        person_name = find_person(face_encoding)
```

```
        if person_name is None:
            # If the person is not recognized, prompt the user for a name
                person_name = input(f"Enter the name for this face (person_{len(known_face_names) + 1}): ").strip() or f"person_{len(known_face_names) + 1}"
            known_face_names.append(person_name)
            known_face_encodings.append(face_encoding)

        person_folder = os.path.join(output_folder, person_name)

        if not os.path.exists(person_folder):
            os.makedirs(person_folder)

        output_path = os.path.join(person_folder, f"{person_name}_{i + 1}.jpg")
                                pil_image = Image.fromarray(image[face_location[0]:face_location[2], face_location[3]:face_location[1]])
        pil_image.save(output_path)

# Define a Flask route for uploading images, processing them using face recognition, and displaying success or error messages in the HTML template.
@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        if 'file' not in request.files:
            return render_template('index.html', error='No file part')
```

```
        file = request.files['file']

        if file.filename == '':
            return render_template('index.html', error='No selected file')

        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)

            output_folder = 'output'
            extract_faces(file_path, output_folder)

            return render_template('index.html', success='File uploaded and faces extracted!')

    return render_template('index.html')

if __name__ == '__main__':
    load_known_faces()
    app.run(debug=True)
```

7.2 2Dto3D conversion code

```
def create_stl(x, y, z, output_folder, filename, height_factor=2):
  z_scaled = np.clip(z * height_factor, a_min=None, a_max=z.max())
  faces = []
  for i in range(x.shape[0] - 1):
    for j in range(x.shape[1] - 1):
      z_idx = lambda i, j: i * y.shape[1] + j
      faces.append([z_idx(i, j), z_idx(i + 1, j), z_idx(i + 1, j + 1)])
      faces.append([z_idx(i, j), z_idx(i + 1, j + 1), z_idx(i, j + 1)])

  vertices = np.column_stack((x.ravel(), y.ravel(), z_scaled.ravel()))
  face_array = np.zeros(len(faces), dtype=mesh.Mesh.dtype)
  for i, f in enumerate(faces):
    for j in range(3):
      face_array['vectors'][i][j] = vertices[f[j], :]

  model_mesh = mesh.Mesh(face_array)
  model_mesh.save(os.path.join(output_folder, f'{filename}.stl'))



def main1():
  image_path = "/Users/titeershaghatakchowdhury/Desktop/final/uploads/nikita_3.jpg"
  output_folder = "/Users/titeershaghatakchowdhury/Desktop/final/output"
  os.makedirs(output_folder, exist_ok=True)

  image = Image.open(image_path).convert('L')
```

```
image = ImageOps.autocontrast(image)

face_locations = face_recognition.face_locations(np.array(image))
if face_locations:
  top, right, bottom, left = face_locations[0]
  face_image = image.crop((left, top, right, bottom))
  image_array = np.array(face_image)
else:
  raise Exception("No face found in image.")

nose_center = (face_image.size[0] // 2, face_image.size[1] // 2)
y, x = np.ogrid[:image_array.shape[0], :image_array.shape[1]]
x, y = np.meshgrid(x - nose_center[0], y - nose_center[1])

create_stl(x, y, image_array, output_folder, '3d_stl', height_factor=0.3)

# Function to simulate the rotation of the 3D plot around the x-axis
def rotate(ax, angle):
  ax.view_init(elev=angle, azim=0) # Rotate around the x-axis

# Load the image and convert it to grayscale
image = Image.open(
    "/Users/titeershaghatakchowdhury/Desktop/final/uploads/nikita_3.jpg"
).convert('L')
image_array = np.array(image)
```

```
# Find the nose center, assuming it is at the center of the image
nose_center = (image_array.shape[1] // 2, image_array.shape[0] // 2)

# Find the base of the neck (assuming it is at the bottom of the image)
neck_base = image_array.shape[0]

# Create x and y coordinates with the nose as the middle point
x = np.arange(image_array.shape[1])
y = np.arange(
    neck_base) # y coordinates from the top to the base of the neck
x, y = np.meshgrid(x - nose_center[0], y - nose_center[1])

# Use the pixel values as z-coordinates and scale the depth by 0.05 (reduced depth scaling)
z = image_array * 0.025

# Create the figure and axis for the plot in 3D with increased size
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Set up the folder for saving images
output_folder = "/Users/titeershaghatakchowdhury/Desktop/final/output"
os.makedirs(output_folder, exist_ok=True)

# Plot the surface with improved quality settings
surf = ax.plot_surface(x,
                       y,
```

```
                z,
                rstride=1,
                cstride=1,
                cmap=plt.cm.gray,
                linewidth=0,
                antialiased=True,
                shade=True)

    # Rotate and capture images at 15-degree intervals around the x-axis up to 90 degrees
    for angle in range(-90, 91, 15):
      rotate(ax, angle)
      filename = os.path.join(output_folder, f'rotation_{angle}.png')
      plt.savefig(filename, dpi=1200)  # Save with higher resolution

    # Save the 3D model with increased dpi
    model_filename = os.path.join(output_folder, '3D_model.png')
    plt.savefig(model_filename, dpi=1200) # Save with higher resolution
    plt.show()


  if __name__== "_main_":
    main1()
```

CHAPTER-8

RESULT AND DISCUSSION

The project successfully demonstrated the ability to convert human faces from photographs into 3D models and generate G-code for 3D printing. The system achieved the following key results:

1. Face Recognition: The face recognition algorithm effectively identified and extracted facial features from input images.

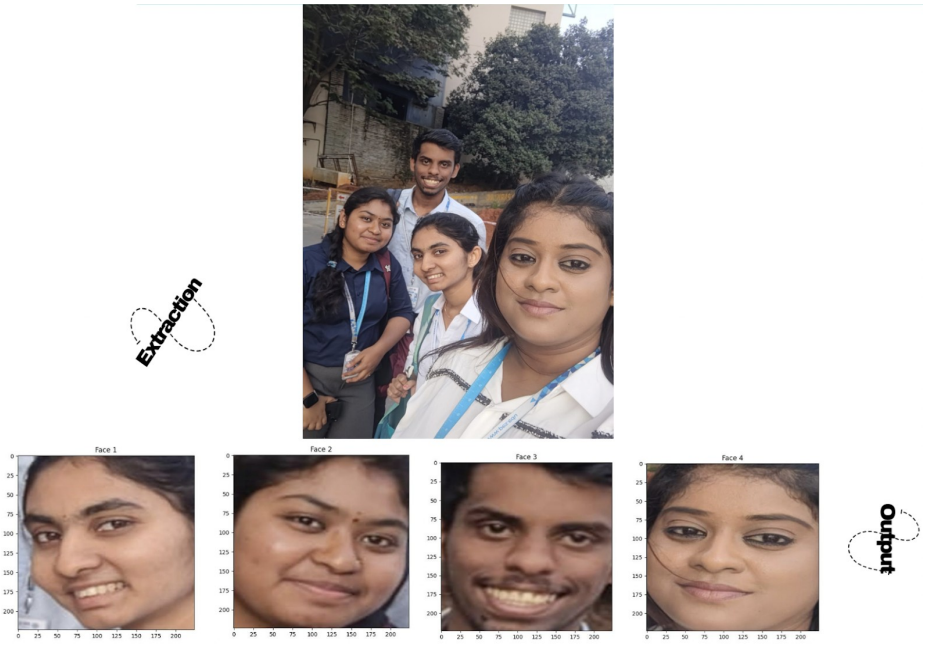

Figure 6: Showcase of Face recognition and extraction

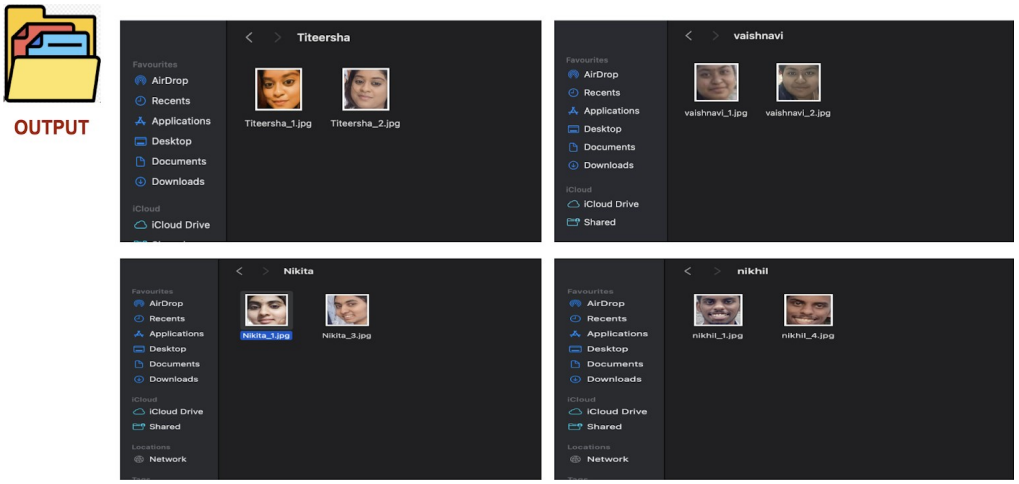

Figure 7: Extracted Faces stored in respective folder leading to “Collection of faces”

2. 3D Model Realism: The virtual 3D models generated from the recognized faces exhibited a realistic representation of facial feature.

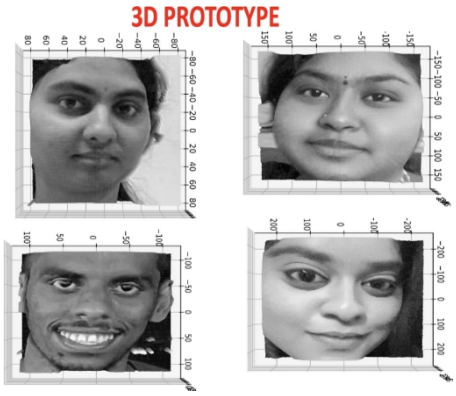

Figure 8: 3D Prototype of faces extracted

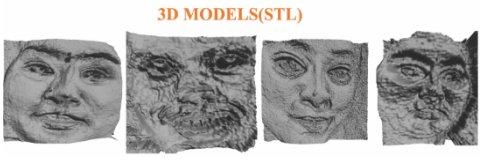

Figure 9: 3D models in STL Format

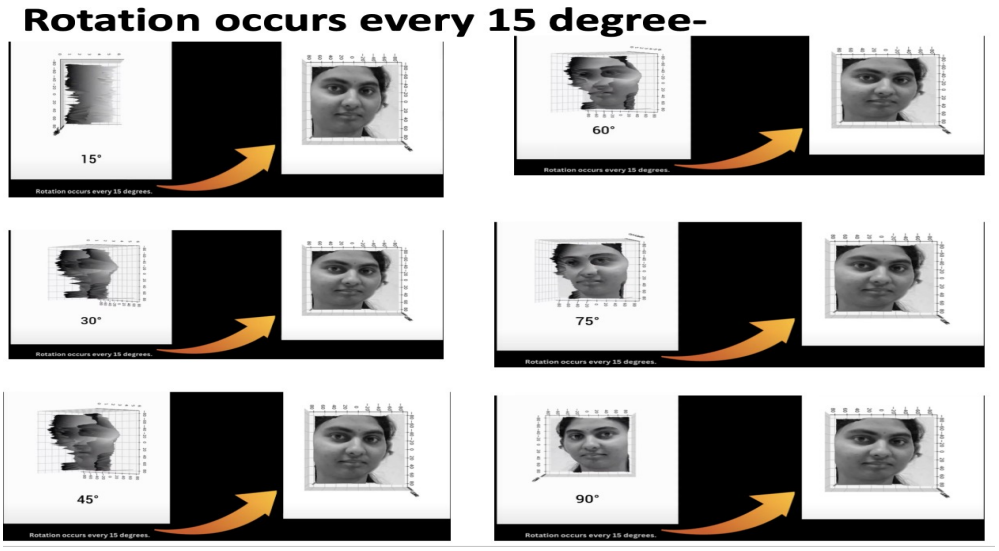

Figure 10: Rotation shown at each 15 degree

3. STL File Generation: The extension of virtual 3D models to STL files was successful, ensuring compatibility with 3D printing systems.

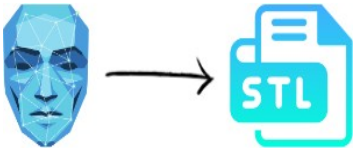

Figure 11: Conversion of 3D model to STL File

G-code Extraction: The G-code generation using the third-party tool, Cura Engine, was seamlessly integrated into the system. The extracted G-code provided accurate instructions for 3D printers, ensuring optimal layer-by-layer construction of the prototypes.

Figure 12: STL to G-code Generation

4. User Interaction and Interface: The user interface facilitated easy uploading of images, and the system efficiently organized recognized faces into folders named after individuals.

The results of the project demonstrate the feasibility of converting human faces from photographs into 3D models for various applications, including personalized avatars, medical visualizations, and creative design. The system's ability to generate G-code for 3D printing further expands its potential by enabling the physical creation of 3D replicas of human faces.

CHAPTER-9

CONCLUSION AND FUTURE WORK

The project of converting human faces from photographs into 3D models has successfully demonstrated the feasibility of this technology and its potential for various applications. The system effectively identifies and extracts facial features from input images, generates realistic virtual 3D models, converts 3D models into STL files compatible with 3D printing systems, and extracts G-code for precise layer-by-layer construction of 3D-printed replicas. The user-friendly interface further enhances the accessibility and usability of the system.

Future work can focus on enhancing the capabilities and applications of this project. Areas of future work include refining the face recognition algorithm, improving 3D model detail and realism, expanding G-code compatibility.These future directions hold immense promise for advancing the technology of 3D face reconstruction and its applications across various domains.

REFERENCES/BIBLIOGRAPHY

[1] Jha, Medha et al. “Face Recognition: Recent Advancements and Research Challenges.” *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (2022): 1-6.

[2] Beymer, "Face recognition under varying pose," 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 1994, pp. 756-761, doi: 10.1109/CVPR.1994.323893.

[3] Adjabi, I.; Ouahabi, A.; Benzaoui, A.; Taleb-Ahmed, A. Past, Present, and Future of Face Recognition: A Review. Electronics 2020, 9, 1188.

[4] G. M. Zafaruddin and H. S. Fadewar, "Face recognition: A holistic approach review," 2014 International Conference on Contemporary Computing and Informatics (IC3I), Mysore, India, 2014, pp. 175-178, doi: 10.1109/IC3I.2014.7019610.

[5] A. N. Razzaq, R. Ghazali and N. K. El Abbadi, "Face Recognition – Extensive Survey and Recommendations," 2021 International Congress of Advanced Technology and Engineering (ICOTEN), Taiz, Yemen, 2021, pp. 1-10, doi: 10.1109/ICOTEN52080.2021.9493444.

[6] Nath, Raktim & Kakoty, Kaberi & Bora, Dibya & Welipitiya, Udari. (2021). Face Detection and Recognition Using Machine Learning. 43. 194-197.

[7] KH Teoh *et al* 2021 *J. Phys.: Conf. Ser.* **1755** 012006

[8] Kaur, Jashanpreet & Akanksha, & Singh, Harjeet. (2018). Face detection and Recognition: A review.

[9] Pollefeys, Marc & Van Gool, Luc. (2002). From images to 3D models. Commun. ACM. 45. 50-55. 10.1145/514236.514263.

[10] Remondino, Fabio & El-Hakim, Sabry. (2006). Image-based 3D Modelling: A Review. The Photogrammetric Record. 21. 269 - 291. 10.1111/j.1477-9730.2006.00383.x.

[11] Harušinec, Jozef & Suchánek, Andrej & Loulová, Mária. (2019). Creation of prototype 3D models using RAPID PROTOTYPING. MATEC Web of Conferences. 254. 01013. 10.1051/matecconf/201925401013.

[12] Ueng, Shyh-Kuang & Huang, Hsuan-Kai & Liu, Zen-Yu. (2019). Image-based Contouring and G-code Generation for Additive Manufacturing.

[13] Ueng, Shyh-Kuang & Huang, Hsuan-Kai & Huang, Hsin-Cheng. (2019). A G-Code Generator for Volumetric Models. Applied Sciences. 9. 3868. 10.3390/app9183868.

[14] Welander, Tyler & Marsh, Ronald & Amin, Md Nurul. (2018). G-code Modeling for 3D Printer Quality Assessment.

[15] A. A. Khleif and A. A. Shnawa, "Vision system aided 3D object reconstruction and machining using CNC milling machine," 2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA), Wasit - Kut, Iraq, 2018, pp. 249-254, doi: 10.1109/ICASEA.2018.8370990.

[16] P. Ji, L. Wang, D. -X. Li and M. Zhang, "An automatic 2D to 3D conversion algorithm using multi-depth cues," 2012 International Conference on Audio, Language and Image Processing, Shanghai, China, 2012, pp. 546-550, doi: 10.1109/ICALIP.2012.6376677.

[17] Manbae Kim, "2D-to-3D conversion using color and edge," 2014 International SoC Design Conference (ISOCC), Jeju, Korea (South), 2014, pp. 171-172, doi: 10.1109/ISOCC.2014.7087681.

[18] K. Yamada and Y. Suzuki, "Real-time 2D-to-3D conversion at full HD 1080P resolution," 2009 IEEE 13th International Symposium on Consumer Electronics, Kyoto, Japan, 2009, pp. 103-106, doi: 10.1109/ISCE.2009.5156848.

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Sl.No	*Value*	*Description*
1	*Entity*	*An object or thing that exists in the real world.*
2	*3D model*	*A three-dimensional representation of an object or space created using computer software.*
3	*Prototype*	*A preliminary version of a product, often used for testing and evaluation before final production.*
4	*Photo*	*A visual image captured using a camera or other device.*
5	*3D*	*3-Dimensional*
6	*GUI*	*Graphical User Interface*
7	*STL*	*STereoLithography/ Standard Triangle Language*

Word Count: 5980

Plagiarism Percentage 12%

Matches

1 **World Wide Web Match**
View Link

2 **World Wide Web Match**
View Link

3 **World Wide Web Match**
View Link

4 **World Wide Web Match**
View Link

5 **World Wide Web Match**
View Link

6 **World Wide Web Match**
View Link

7 **World Wide Web Match**
View Link

8 **World Wide Web Match**
View Link

9 **World Wide Web Match**
View Link

10 **World Wide Web Match**
View Link

11 **World Wide Web Match**

View Link

An Entity to 3D Model Prototyping from a Photo

World Wide Web Match

View Link

13 **World Wide Web Match**

View Link

14 **World Wide Web Match**

View Link

15 **World Wide Web Match**

View Link

16 **World Wide Web Match**

View Link

17 **World Wide Web Match**

View Link

18 **World Wide Web Match**

View Link

19 **World Wide Web Match**

View Link

20 **World Wide Web Match**

View Link

21 **World Wide Web Match**

View Link

22 **World Wide Web Match**

View Link

23 **World Wide Web Match**

View Link

24 **World Wide Web Match**

View Link


PES
UNIVERSITY