National College of Ireland

TR-SONIC
Configuration Manual

MSc Research Project
MSc Artificial Intelligence for Business

Roque Ivan Lopez Lopez
Student ID: x23380501

School of Computing
National College of Ireland

Supervisor: Rajwanul Haque

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Roque Ivan Lopez Lopez

Student ID: x23380501

Programme: MSc AI for Business (MSCAIBUS) **Year:** 2025

Module: Practicum 2

Lecturer: Rajwanul Haque

Submission Due Date: 15 - September - 2025

Project Title: TR-SONIC: Configuration Manual

Word Count: 3,441 words Page Count: 20 Pages

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Roque I. Lopez

Date: 11 - September - 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☑
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☑
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☑

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

TR-SONIC
Configuration Manual

Roque Ivan Lopez Lopez
Student ID: x23380501

1 File Structure and Requirements

The artifact attached to this research follows the following structure:

```
Thesis_Final_Artifact/
├── main.py                                  # Main application entry point
├── requirements.txt                         # Python package dependencies list
├── README.md                                # Project documentation and setup guide
│
├── configs/                                 # Configuration of the different components
│   ├── bias_mitigation_config.yaml          # Settings for bias reduction algorithms
│   ├── expression_tolerance_config.yaml # Facial expression variation handling
│   ├── improved_training_config.json        # Enhanced model training parameters
│   └── paths.yaml                           # File and directory path configurations
│
├── data/                                    # Data storage and datasets
│   ├── landmark_data/                       # Facial landmark coordinate datasets
│   ├── sample_models/                       # Sample 3D face models for testing
│   │   ├── sample_attacker_1.obj            # Test attacker model 1 (3D mesh)
│   │   ├── sample_attacker_2.obj            # Test attacker model 2 (3D mesh)
│   │   └── sample_user.obj                  # Legitimate user model (3D mesh)
│   └── sonar_data/                          # Acoustic signature simulation data
│
├── models/                                  #Trained models and metadata
│   ├── backups/                             # Model backup storage for recovery
│   ├── expression_tolerance_config.json     # Expression handling configuration
│   ├── facial_transformer.pth               # Primary facial recognition transformer model
│   ├── facial_transformer_improved.pth      # Enhanced facial recognition model
│   ├── landmark_transformer_latest.pth      # Latest landmark detection model
│   ├── latest_training_history_backup.json # Training progress backup
│   ├── model_update_status.json             # Model versioning and update tracking
│   ├── sonar_similarity_transformer.pth     # Sonar-based similarity model
│   └── sonar_transformer_latest.pth         # Latest sonar processing model
│
├── output/                                  # Generated results and analysis outputs
│   ├── logs/                                # Application execution logs and debugging info
│   ├── results/                             # Training results and performance metrics
│   └── visualizations/                      # Generated charts, graphs, and visual outputs
│
├── scripts/                                 # Utility and analysis scripts
```

```
    ├── __init__.py                          # Python package initialization
    ├── generate_anova_analysis.py           # ANOVA statistical analysis generator
    ├── generate_confusion_matrix_analysis.py # Confusion matrix analysis generator
    ├── generate_distance_analysis.py        # Distance-based similarity analysis
    ├── generate_statistical_analysis.py     # General statistical reporting
    ├── generate_system_analysis.py          # Overall system performance analysis
    ├── improved_training_integration.py     # Enhanced training workflow integration
    ├── improve_expression_tolerance.py      # Expression tolerance enhancement
    ├── model_backup_system.py               # Automated model backup utilities
    ├── model_management_analyzer.py         # Model performance and management analysis
    ├── system_status_summary.py             # System health and status reporting
    ├── test_expression_improvements.py      # Expression tolerance testing
    └── update_system_metrics_with_stats.py # Metrics updating with statistics

├── src/                                     # Core source code modules
    ├── __init__.py                          # Package initialization
    ├── authentication/                      # User authentication and identity verification
    ├── comparison/                          # Biometric comparison/matching algorithms
    ├── detection/                           # Facial landmark and feature detection
    ├── sonar_simulation/                    # Acoustic signature simulation and processing
    ├── training/                            # Machine learning model training workflows
    ├── utils/                               # Utility functions (config loading, logging)
    └── visualization/                       # 3D rendering, mesh processing, and tools

└── ui/                                      # Graphical user interface (GUI) components
    ├── __init__.py                          # Package initialization
    ├── authentication_screen.py             # User login and authentication interface
    ├── model_training_screen.py             # Model training control panel
    ├── system_analysis_screen.py            # System performance monitoring interface
    └── welcome_screen.py                    # Main application launcher and navigation
```

To reproduce this implementation, you should have Python 3.8 or higher installed, along with an environment that includes the necessary scientific computing libraries, deep learning frameworks, 3D visualization tools, and essential utility packages to ensure full compatibility with the project's requirements. These include:

- Scientific computing: numpy, scipy, matplotlib, scikit-learn, pandas
- Machine learning & 3D visualization: torch, trimesh, plotly, opencv
- Utilities & configuration: pyyaml, tqdm, pillow, psutil
- GUI framework: tkinter

Note: All these packages are also listed in the provided requirements.txt file to help ensure your environment is properly configured before running the model for the first time.

2 Dataset Information.

Table 1 Facescape dataset specifications **(Yang *et al.*, 2020; Zhu *et al.*, 2023)**

Source:	CITE LAB, Nanjing University
Content:	847 subjects × 20 expressions high-resolution 3D facial meshes with detailed geometric features, in a total of 16940 models
Diversity:	Ethnically diverse population coverage for comprehensive bias testing
Original Format:	Wavefront OBJ files (.obj format)
Processed Format:	Wavefront OBJ files (.obj format – used for implementation)
Resolution:	High-fidelity meshes with around 50,000 to 100,000 vertices per model
Location:	/facescape_trainset_001_100/

Dataset License of usage:

The FaceScape dataset is licensed for non-commercial research use only. The dataset is provided by the CITE LAB, Nanjing University, and comes with the following conditions:

- *Permitted Use: Academic research and experiments only.*
- *Prohibited Use: Any commercial use, sublicensing, redistribution, or sharing of the dataset.*
- *Publication Restrictions: Direct portraits (images and rendered models) cannot be published. Only portrait-authorized data may appear in publications (refer to the dataset website for the index of authorized data).*
- *Attribution: Any research output using the dataset must properly cite the following papers:*
  - *Zhu et al., 2023 (IEEE TPAMI)*
  - *Yang et al., 2020 (CVPR)*
- *Liability: The dataset is provided "as is," without warranties or guarantees of accuracy or completeness.*

This research strictly adheres to the FaceScape license terms. All experimental procedures, data processing, and bias mitigation evaluations are conducted within the scope of academic research. No direct facial portraits or unauthorized data representations are included in this publication.

2.1 Dataset Preparation and Bias mitigation parameters

The Data preparation, Bias mitigation and train/validation split procedure is implemented in src/ui/model_training_screen.py and performs subject-wise stratification to ensure balanced representation. The pipeline proceeds through the following steps:

1. Dataset loading and file scan:

The FaceScape dataset is loaded and split using _load_facescape_dataset() function:

```
def _load_facescape_dataset(self):
    """Load and split the FaceScape dataset with 80/20 train/validation split"""
    facescape_path = self.facescape_path_var.get()

    if not os.path.exists(facescape_path):
        messagebox.showerror("Error", f"FaceScape dataset path not found: {facescape_path}")
        return None, None
```

2. Dataset splitting:

An 80/20 split is performed such that 80% of each subject's models are allocated to the training set and the remaining 20% to the validation set. This ensures that all subjects are represented in both subsets without leakage.

```
    # Split each subject's models 80/20
    for subject in unique_subjects:
        indices = subject_indices[subject]
        np.random.shuffle(indices)

        # Calculate split point (80/20)
        split_idx = int(len(indices) * 0.8)

        # Add to training set
        train_indices = indices[:split_idx]
        X_train_paths.extend(X_paths[train_indices])
        y_train_ids.extend([subject] * len(train_indices))
```

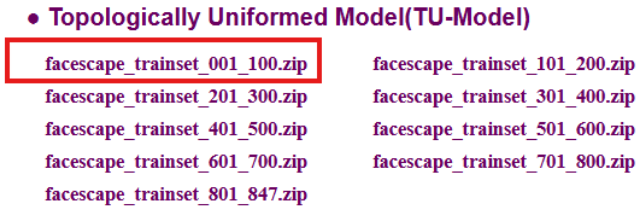
3. Data bias mitigation:

If bias mitigation is enabled by the user (self.enable_bias_mitigation.get()), the system invokes the _apply_bias_mitigation_to_dataset() function:

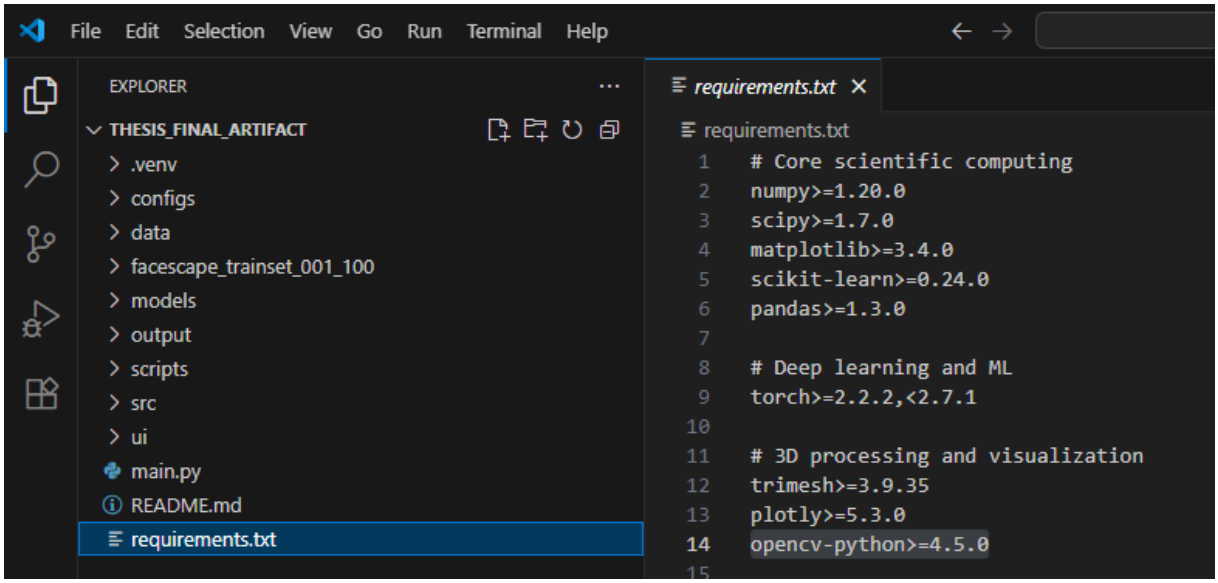
```
def _apply_bias_mitigation_to_dataset(self):
    """Apply bias mitigation analysis to the prepared dataset"""
    try:
        # Simulate bias analysis on the dataset
        self.logger.info("Applying bias mitigation analysis to dataset...")

        # Create synthetic protected groups for demonstration
        # In a real implementation, this would extract demographic information
        train_data, train_labels = self.training_data
        val_data, val_labels = self.validation_data

        # Simulate protected group assignment (this would be real demographic data)
        np.random.seed(42)  # For reproducibility
        train_protected_groups = np.random.choice([0, 1], size=len(train_data), p=[0.7, 0.3])
        val_protected_groups = np.random.choice([0, 1], size=len(val_data), p=[0.7, 0.3])
```

3 System .zip Compressed folder information

The system is provided in a compressed ZIP folder named Thesis_Final_Artifact_v10 (Final with trimmed dataset). The dataset was trimmed solely for submission purposes, as the original dataset is composed of 3D objects that exceed 20 GB due to the variety of facial models and detailed features of everyone. Nevertheless, the system was trained on the complete dataset to ensure proper execution. If it is necessary to fully verify the results, it is recommended to download the complete dataset from the following link (subject to obtaining the required permissions): **"https://facescape.nju.edu.cn/Page_Download/"**

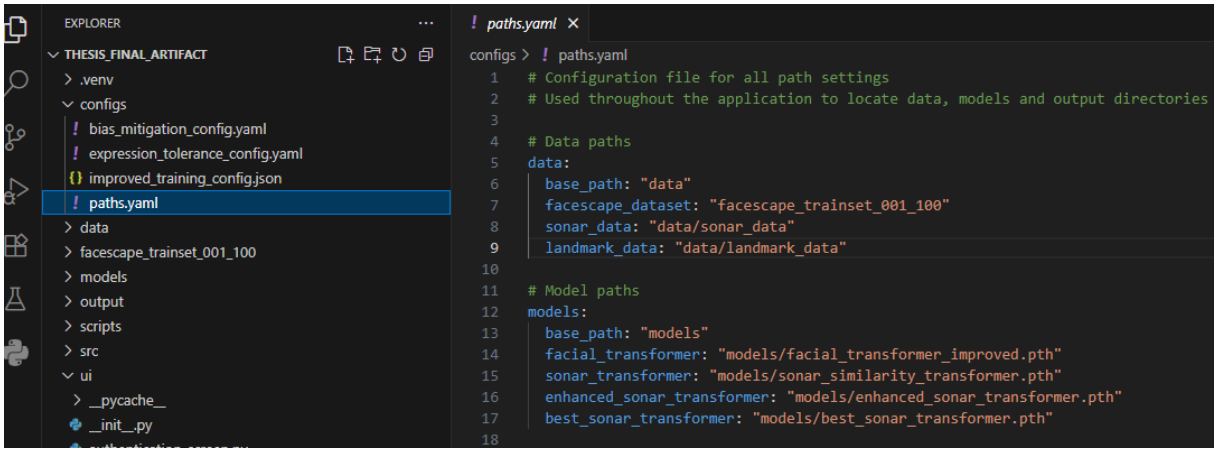


To run the program in a production environment, extract the folder, open it in Python, and install the required dependencies either manually from the list provided earlier or by loading the requirements.txt file to set up the environment without issues.

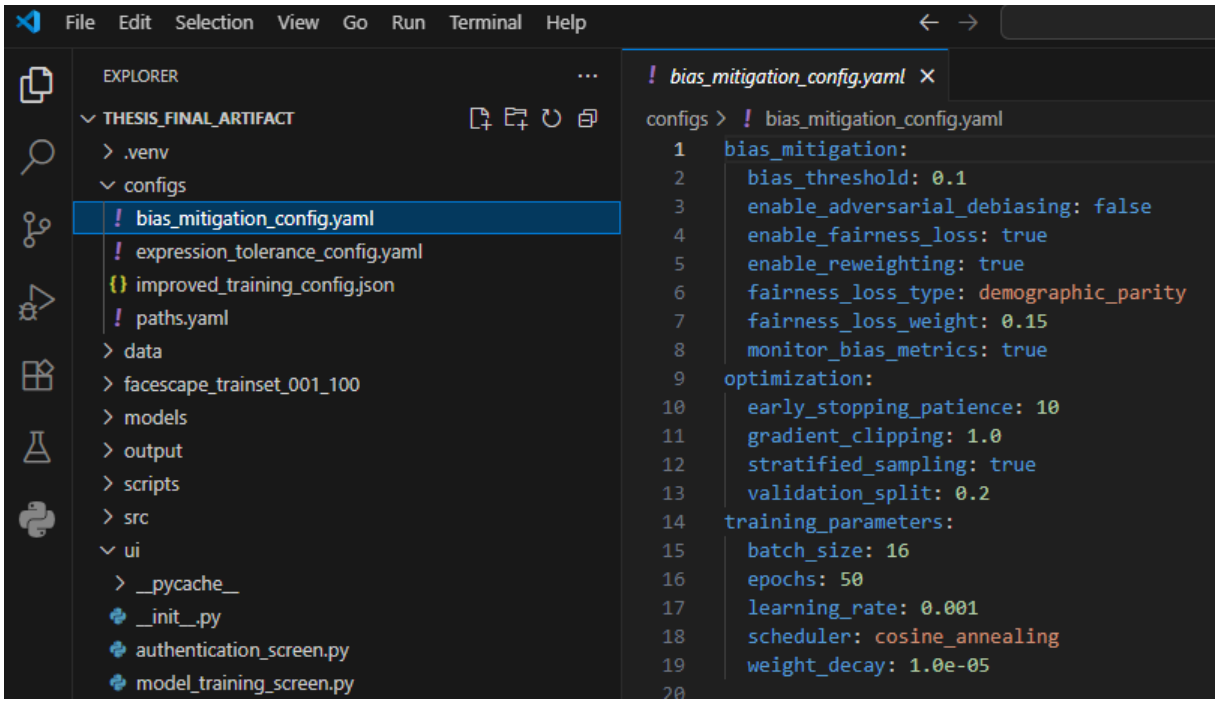


4 Paths Management and Parameters Configuration

To avoid the confusion caused by the large number of code files, a section dedicated to managing paths and configuring model parameters was added. To this end, the /configs folder includes the files necessary for parameterization, as well as a paths file that defines the location of models, results, outputs, etc.



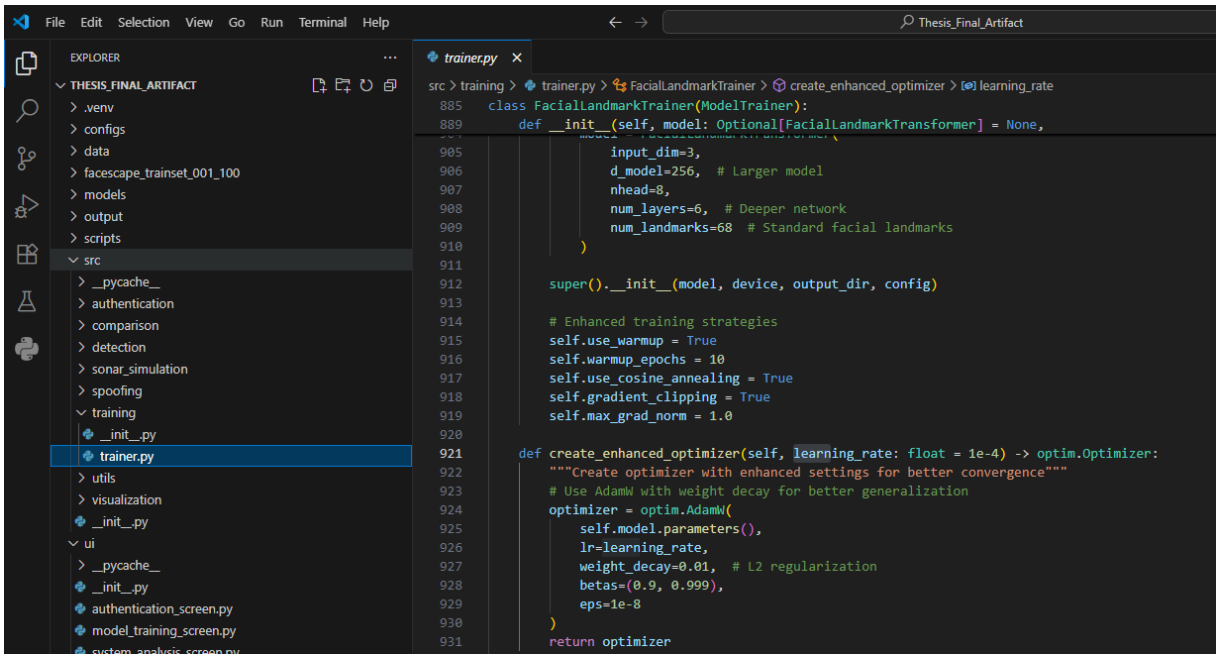
This organization allows users to easily access and modify specific program parameters, such as bias mitigation and facial expression tolerance settings, without having to explore the entire codebase.



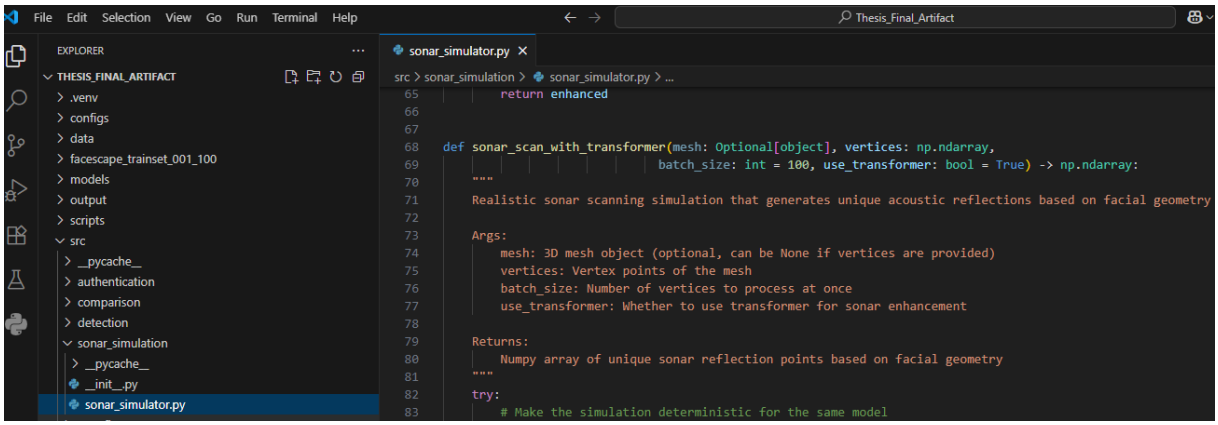
However, it is important to note that the system's graphical interface is designed to allow parameter modification directly through the GUI, minimizing the need to interact with the code.

5 Trainer and utilities configuration

To manage training parameters such as the number of epochs, learning rate, and transformer model architectures, along with other functionalities such as authentication, sonar simulation, landmark detection, and spoofing simulation, the /src folder contains the code for each of these functions. If necessary, changes to the system architecture can be made here.

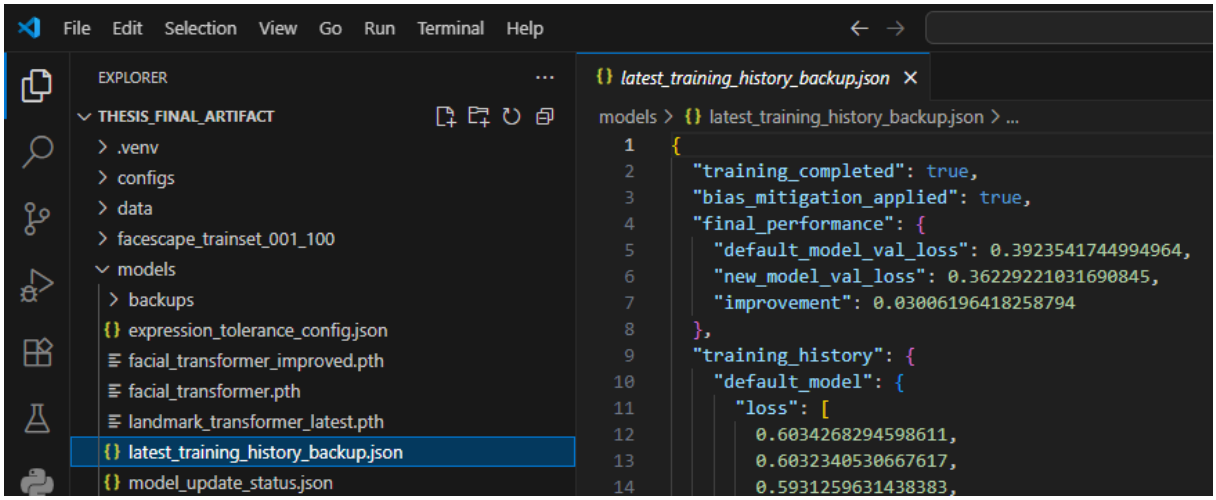


For example, if you want to modify how the transformer-controlled sonar scan is generated, you can access the src/sonar_simulation/sonar_simulator.py file and modify the sonar_scan_with_transformer function, which is responsible for generating the scan, based on the 3D mesh and the geometric structure of the 3D face.

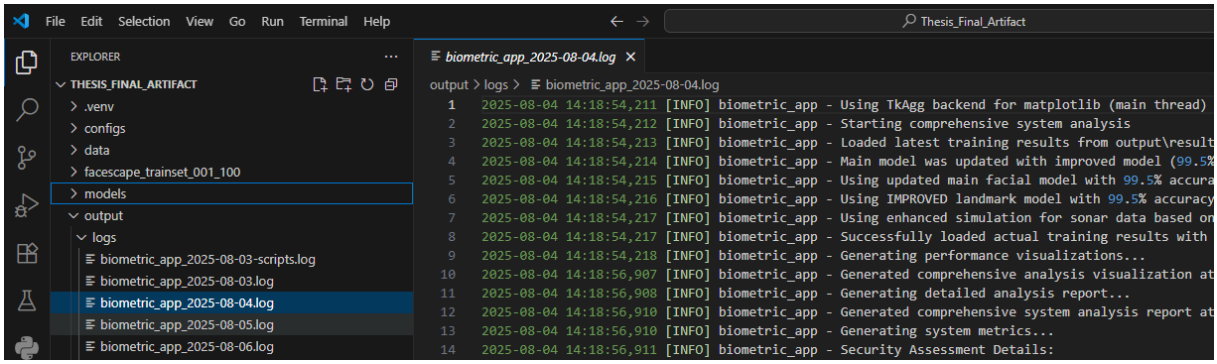


6 Trained Models, Outputs, and Analysis tools.

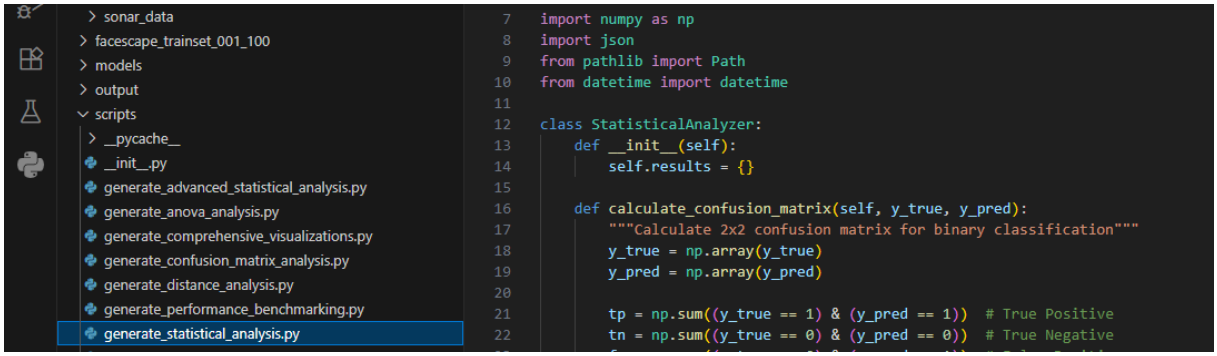
The program includes a dedicated directory where all trained models, their backups, and the results from the training processes are stored. This folder, named \models, serves as a centralized location to organize and manage the outputs of the model training, ensuring easy access and maintenance. By keeping the trained models and related files in one place, the system facilitates efficient version control and helps streamline any future updates or evaluations.



Similarly, there is a folder called \output, which stores the results, logs, and visualizations generated by the program. This separation helps maintain a clear organization between the models themselves and the various output data produced during execution.

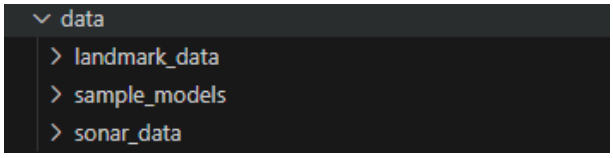


Similarly, there is a folder called \scripts where the code files used for system evaluation, calibration, improvement, and backup are stored.



7 Temporary used data storage.

During training and authentication, the system generates temporary data such as sample models, sonar data, and landmark information. To manage this, a \data folder was created where all this temporary information is stored only for the duration it is needed and is deleted once its use is complete.



8 System Execution Guide

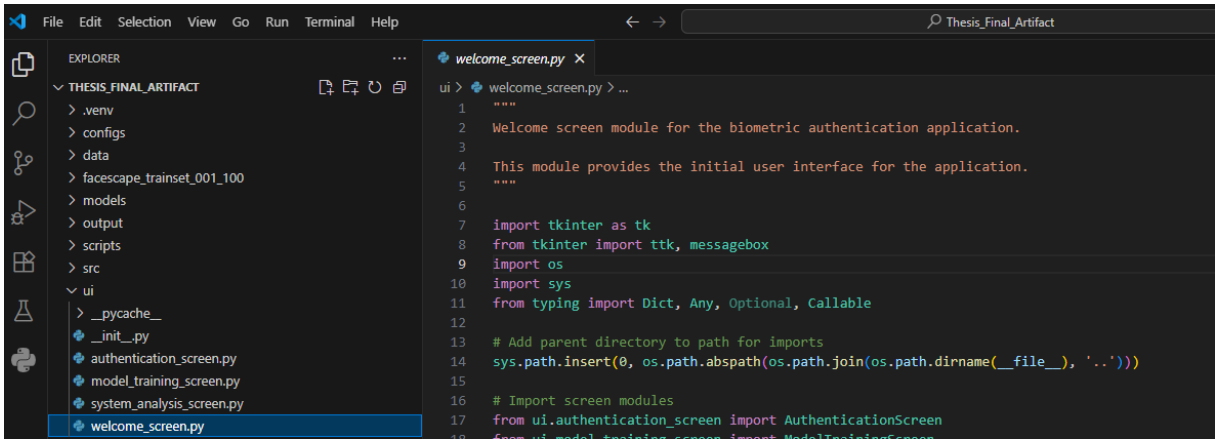
Run the main.py file with the --gui flag. Without the --gui flag, the program will not operate in its interactive mode and may fail to run properly. However, if the program is run without the --gui flag, the program will notify the user to restart it using the proper flag.

```
PS C:\Users\AI4Business\Documents\PythonExercises\Thesis_Final_Artifact> python main.py
2025-08-09 18:32:44,840 [INFO] biometric_app - Using device: cpu
2025-08-09 18:32:44,841 [INFO] biometric_app - Starting Advanced Biometric Facial Recognition System
2025-08-09 18:32:44,841 [INFO] biometric_app - Running in command line mode
Advanced Biometric Facial Recognition System
----------------------------------------
Use --gui flag to launch graphical interface
PS C:\Users\AI4Business\Documents\PythonExercises\Thesis_Final_Artifact> python main.py --gui
```

Note: The program includes a parse_arguments() function, which processes command-line options for customizing how the program runs. One of these options is the --gui flag, which is essential for launching the GUI. Running the system with this flag ensures that users can interact visually, as certain features and workflows are only accessible through the GUI.

```python
def parse_arguments():
    """Parse command line arguments"""
    parser = argparse.ArgumentParser(description='Biometric Authentication System')
    parser.add_argument('--config', type=str, default=None, help='Path to config file')
    parser.add_argument('--model', type=str, default=None, help='Path to model file')
    parser.add_argument('--gui', action='store_true', help='Launch with GUI')
    parser.add_argument('--debug', action='store_true', help='Enable debug logging')
    return parser.parse_args()
```

After running the program, the graphical interface called the welcome screen will appear, this welcome screen code located in the welcome_screen.py file code inside the \ui (user interface) folder. This welcome screen serves as the responsible for allowing access to, and starting, the three main modules of the program, model training, system analisys and user authentication.



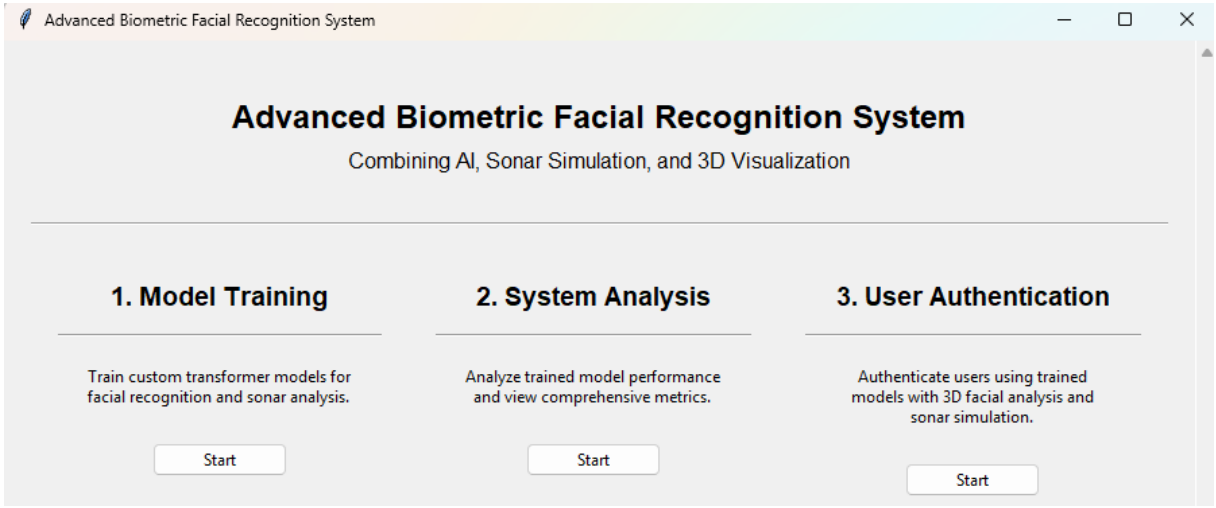
Each one of these screens was implemented in its own dedicated Python module located in the same \ui folder: model_training_screen.py, system_analysis_screen.py, ui\authentication_screen.py

9 GUI: Design and User Flow

The program is developed in a modular structure, with separate files dedicated to each GUI screen. This design ensures that users can interact with the system in a clear and convenient way. Each GUI screen is tailored to allow users to easily view, modify, and adjust the system's parameters and configurations, providing an intuitive experience for managing and customizing the system's behaviour.

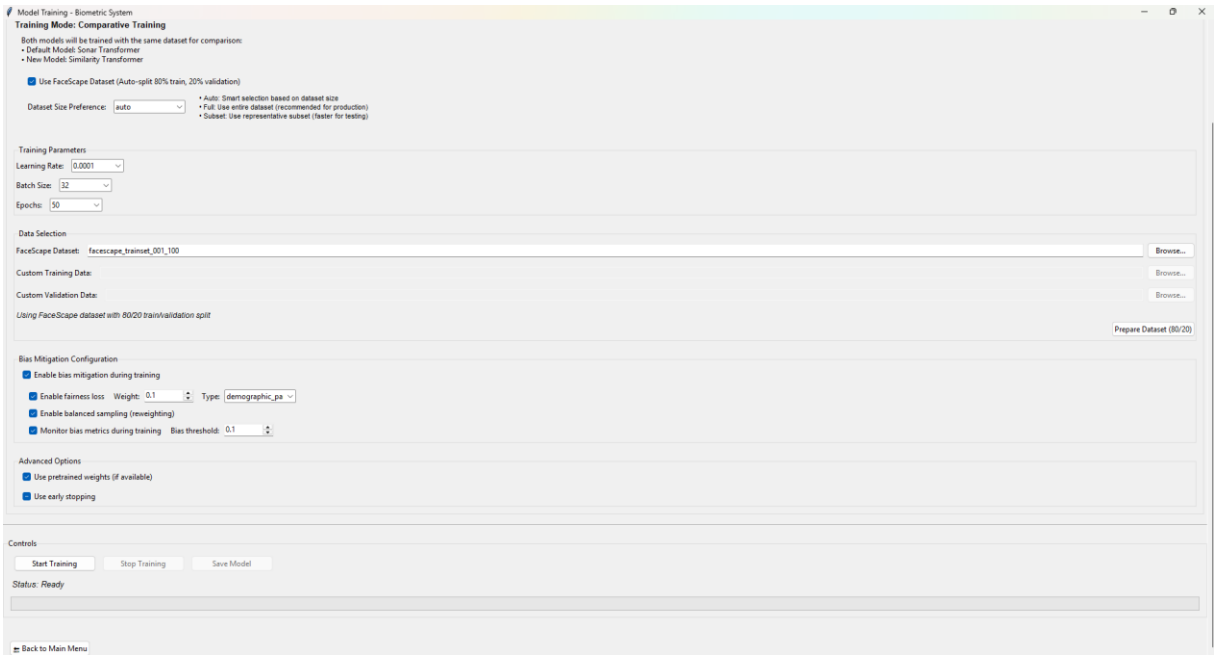
9.1 GUI – Welcome Screen

The welcome interface is the user's first contact with the system. It provides a brief description of what the system can do, allowing the user to select the desired application mode: train a model, analyse the system, or authenticate a user. This interface displays the three functions in numerical order. However, it is not mandatory to follow this sequence. For example, if the user has already trained a model and wants to directly access authentication, they can simply click the Launch button in the user authentication section. The same applies if they only want to view reports or generate a new analysis. This design gives the user full control over the workflow, while also offering a suggested order for first-time use when the application has no data.

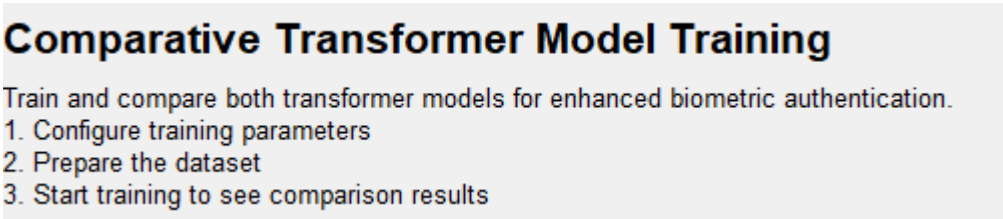


9.2 GUI – Model Training Screen

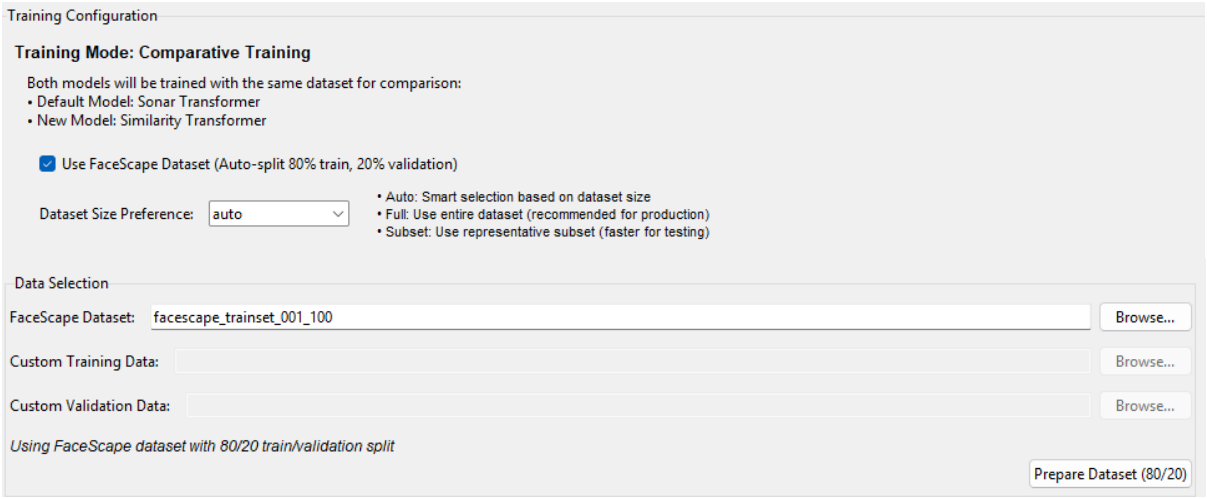
The model training interface is the recommended first interface when running the system for the first time. This allows the user to train models with the Transformer architecture, which will later be used for user authentication. If no trained model is available and the user wishes to authenticate a user in the user authentication interface, the system will automatically train a simple model for the process. However, this is not the intended or optimal use of the system.



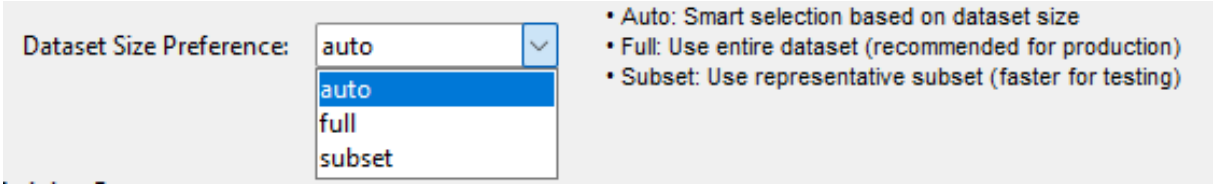
At the top of the interface, the system displays usage instructions. Below this, the user can configure the training process.



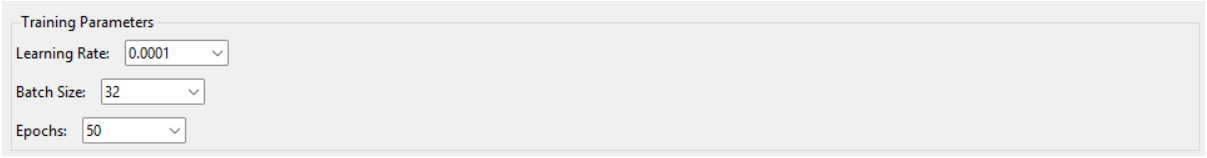
Due to the special structure of the FaceScape dataset, a checkbox is presented to the user. By default, this checkbox is selected. When it is active, the system allows the user to select the folder containing the FaceScape dataset to automatically split, prepare, and preprocess it into the optimal format for training. If the user does not wish to use the FaceScape dataset, they can uncheck the box and instead provide the path to the location of their own training and validation data, continuing the training process with that dataset.



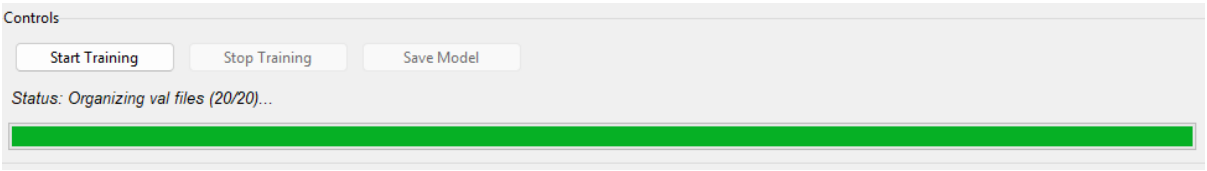
When using the FaceScape dataset, the user can also choose whether to train the model with the entire dataset or a subset. The default option is Auto, which, after the initial analysis of the dataset, will prompt the user to decide whether to use the full dataset or a subset.



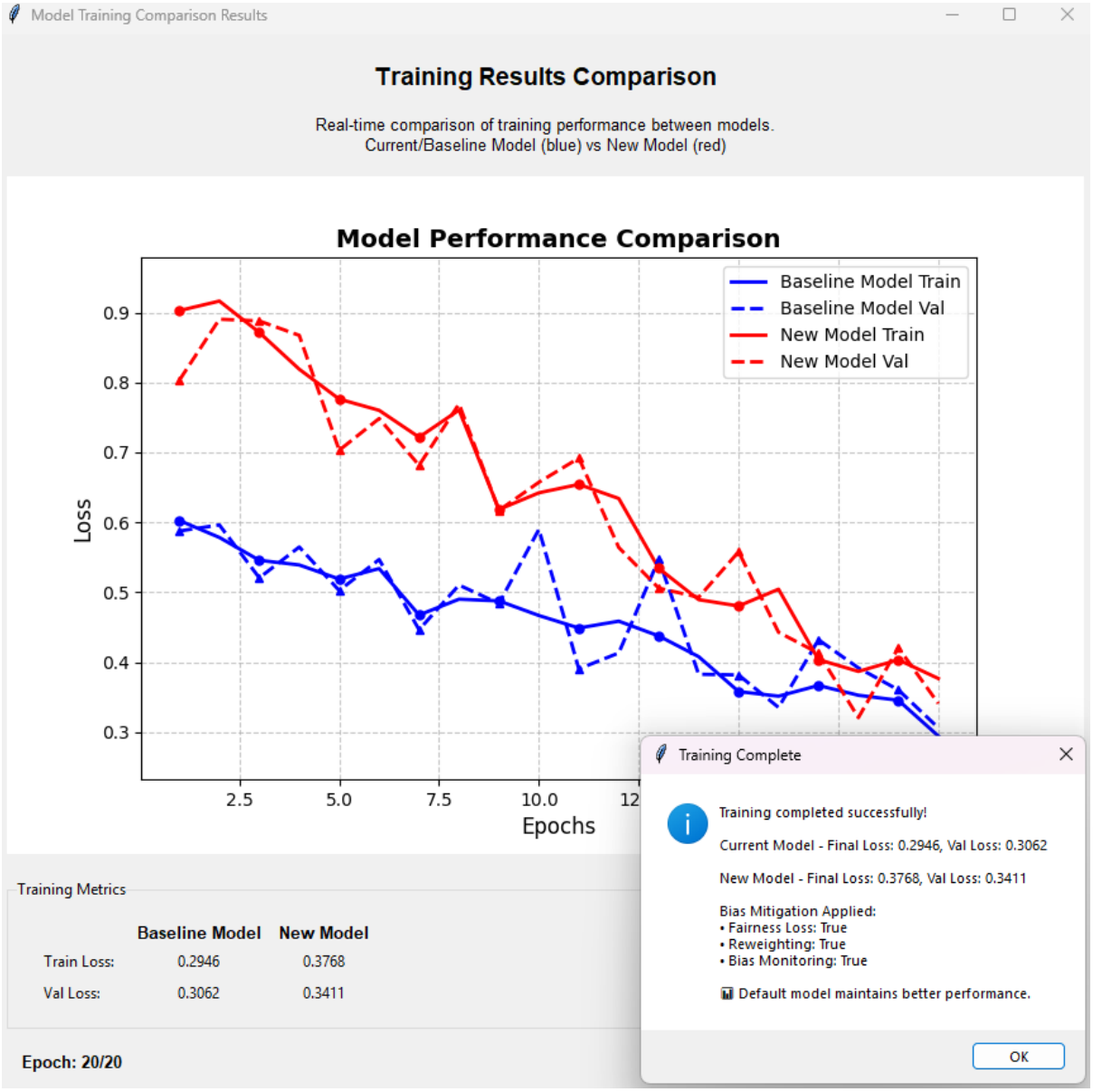
Further down the same screen, the program allows the user to modify or configure training parameters such as the learning rate, batch size, and number of epochs. Similarly, after configuring the dataset path, there is an option to configure bias mitigation, allowing the user to modify the mitigation weights or the type of mitigation applied. This feature is important because it allows the user not only to run the training with or without mitigation, but also to adjust the type of mitigation and its parameters.



Finally, at the bottom of the screen, the system offers a "Start Training" button. Clicking this button will initiate data preprocessing, apply bias mitigation, split the dataset if necessary, and continue training. During training, the system will display multiple notifications regarding progress, requesting user authorization to continue. Progress will also be displayed in a status bar that will fill up as the process progresses.

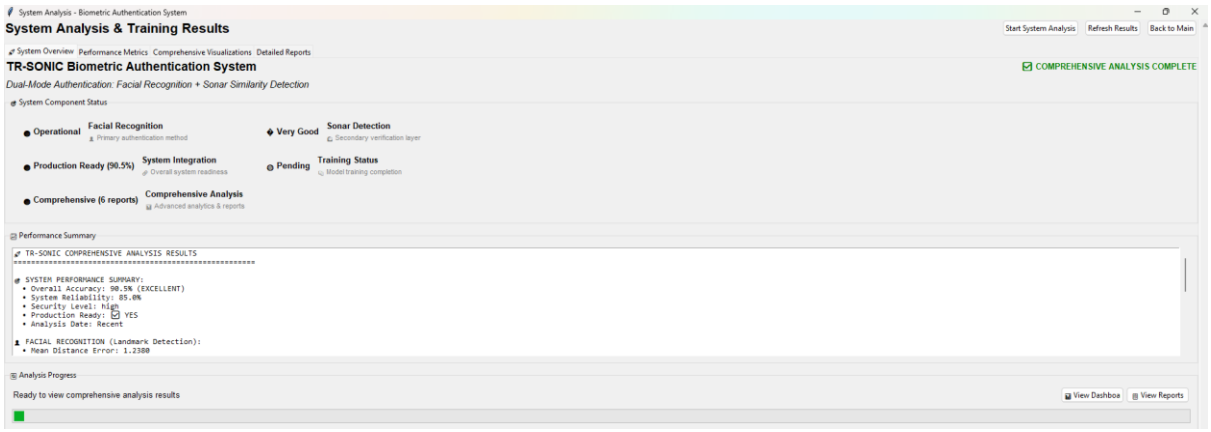


Once completed, a model loss comparison will be generated, allowing the user to decide whether to replace the existing model if the new one shows better performance.

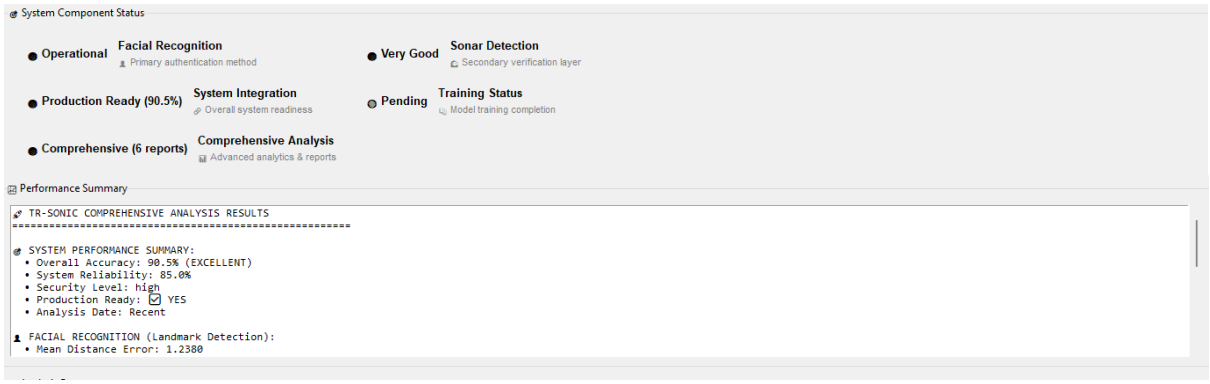


9.3 GUI – System Analysis Screen

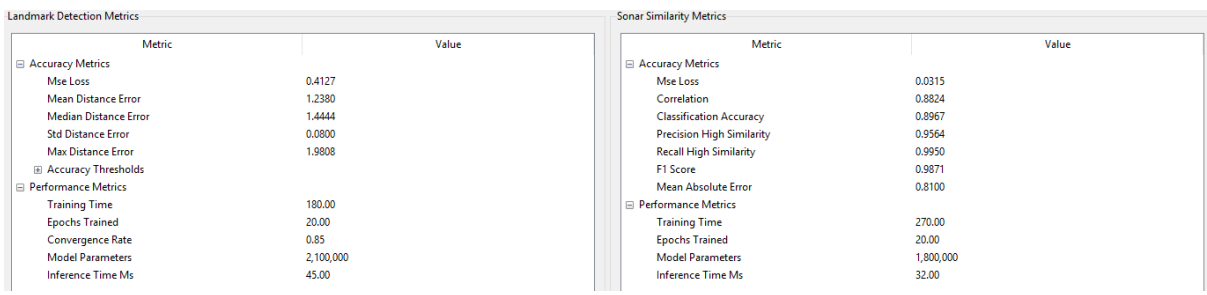
The system analysis interface allows the user to analyze the loaded models, evaluate their metrics, and view both the training results and the overall status of the deployment.



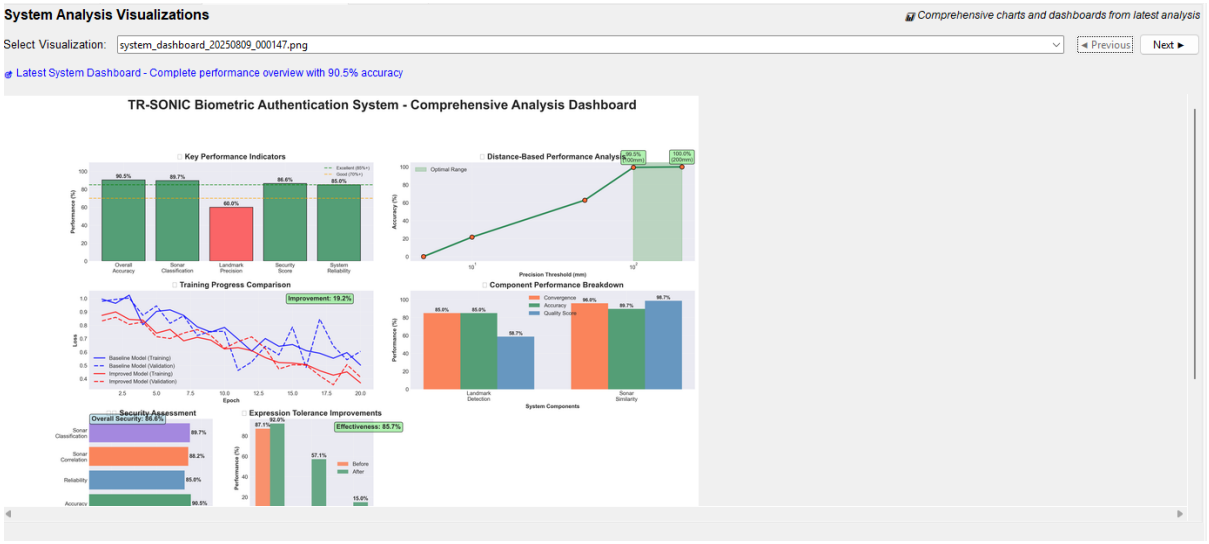
System Overview: allows the user to view the overall status of the system deployment and quickly identify if any model or feature is not ready to go. Each time this tab is opened, a system analysis is launched, which is presented as a performance summary report. This same tab includes quick access buttons: View Dashboard, which takes the user to the tab Comprehensive visualizations, and View Reports, which directs the user to the detailed reports tab.



Performance Metrics: presents the system analysis results of the trained models. These are displayed in comparison mode and are divided primarily into two sections: Landmark Detection Metrics on the left and Sonar Similarity Metrics on the right. Each section displays accuracy and performance metrics, including values such as MSE loss, distance errors, correlation, classification accuracy, precision, recall, F1 score, mean absolute error, total training time, number of epochs, model parameter count, and inference time in milliseconds.



Comprehensive Visualizations: displays visualizations generated from the latest system analysis, allowing the user to evaluate and interpret its performance through a series of detailed charts and dashboards. These visual aids present key performance indicators, component-specific metrics, progress comparisons, and security assessments in an intuitive format, making it easy to identify strengths, weaknesses, and areas for improvement.



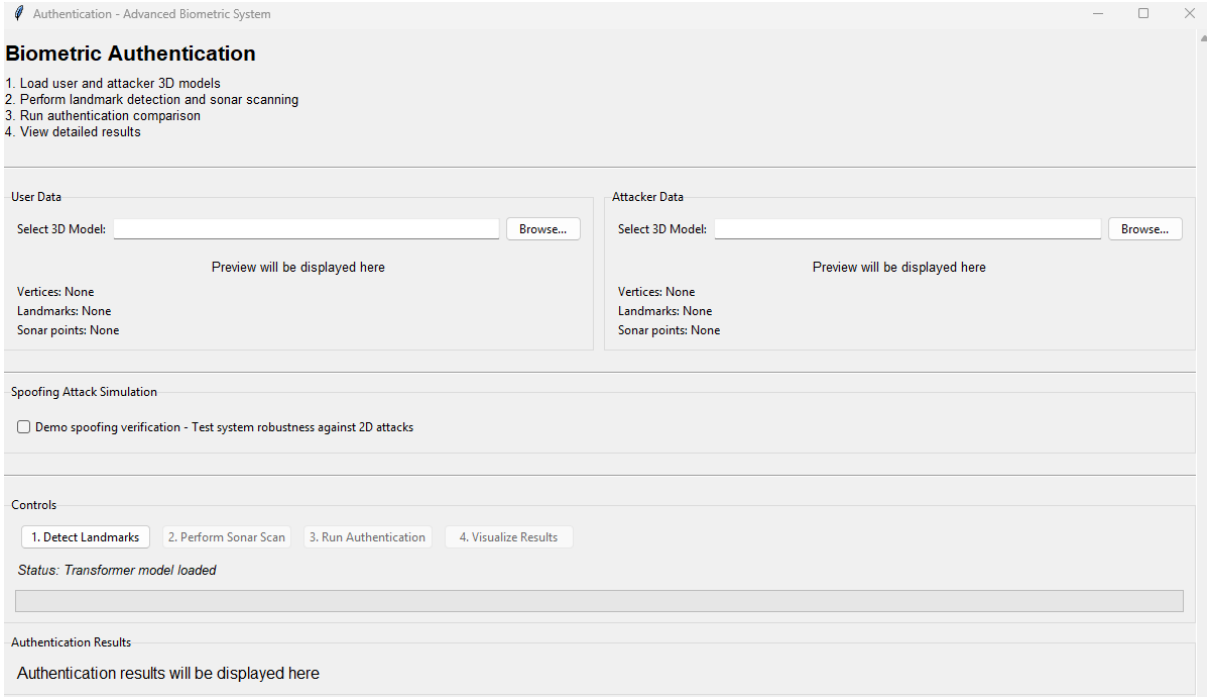
Note: By default, the most recent analysis is shown in the foreground, ensuring that the user has immediate access to the latest performance insights, however, this section is configured to display all generated visualizations, including those from previous models, as long as they are still present in the folders and have not been deleted.

Detailed Reports: presents the Executive Summary of the Comprehensive Analysis of the TR-SONIC Biometric Authentication System, which consolidates the results of a complete evaluation and benchmarking of the system. In short, this tab provides a detailed breakdown of the analysis in text format.

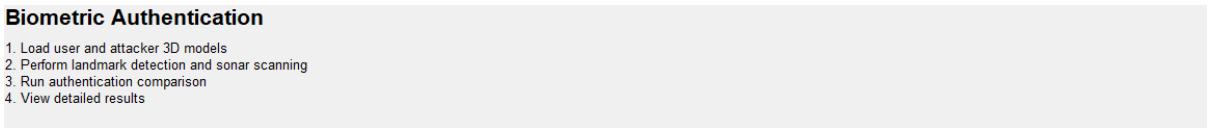
Some of the reports presented include an overview with performance ratings, deployment readiness, competitive position and security, key performance indicators such as accuracy, processing speed, reliability, convergence rates and security metrics, and the results of statistical analysis such as the impact of distance and lighting conditions, user patterns and predictive model performance.

9.4 GUI – User Authentication Screen

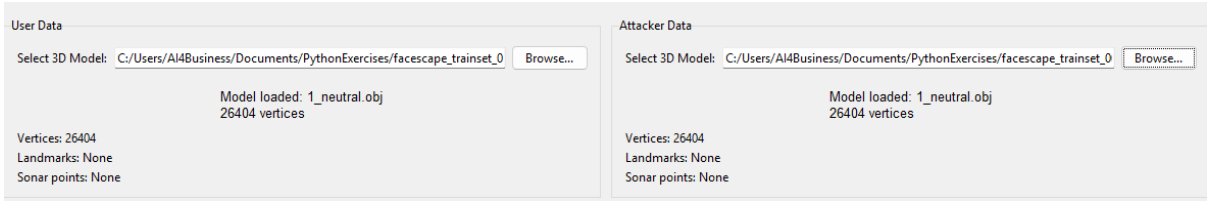
The user authentication interface allows the user to authenticate two three-dimensional facial models using models trained with the Transformer architecture, generates precisely to be used for user authentication.



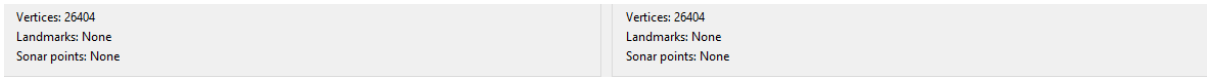
To perform this authentication, two 3D facial models are analysed, one being identified as the user and the other as the attacker. Thus, the system considers that for user authentication to be successful, all the parameters for comparing the user model and the attacker model must be met according to the configured parameters.



To perform this authentication, the system requests the user to select a 3D facial object from the database (user data) and similarly requests a different or identical 3D facial object from the database (attacker data).

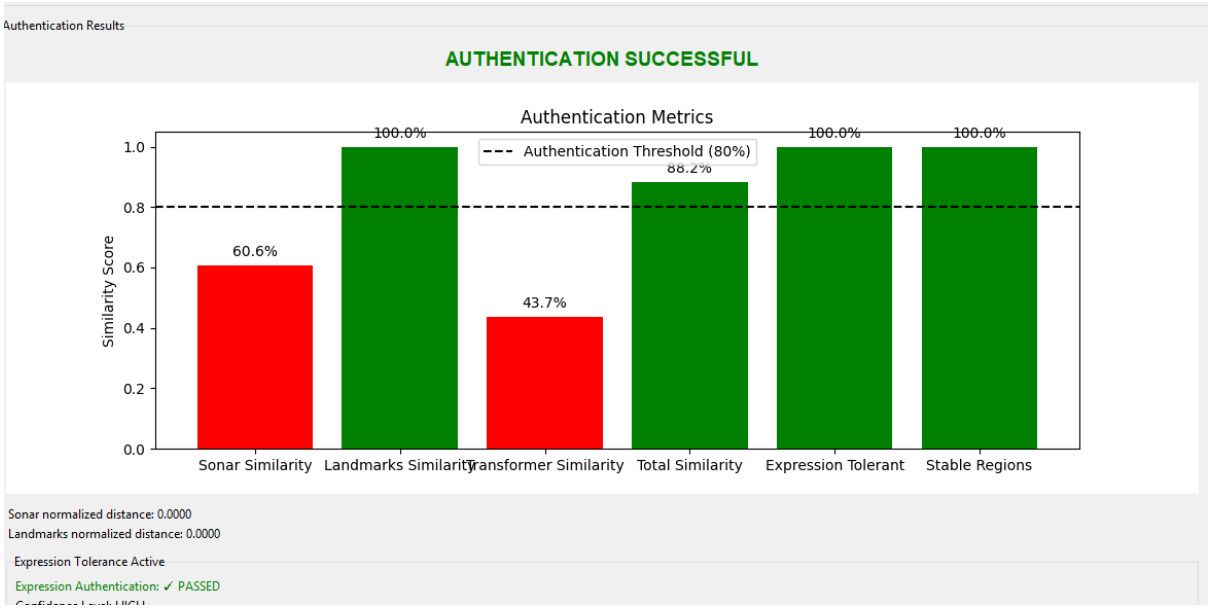


Once the paths to the 3D facial models have been selected and loaded, the first step consists of clicking the button Detect Landmarks. This will use the model trained to detect landmarks to determine the most representative landmarks of both 3D models (user/attacker).

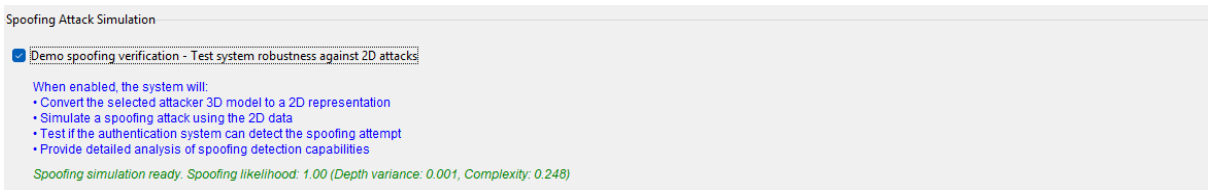


Once the landmarks are detected, the system enables the second step, which consists of performing a sonar scan of both models (user/attacker) using the previously trained sonar transformer model.

Once the sonar signature of each user has been generated, the system will activate step 3 (Run Authentication). This consists of comparing the information obtained and verifying whether the landmarks, the sonar signature, and other elements are similar enough for the system to distinguish them as the same individual. If they are similar enough, the system will reject the authentication. Otherwise, the system will reject the authentication.

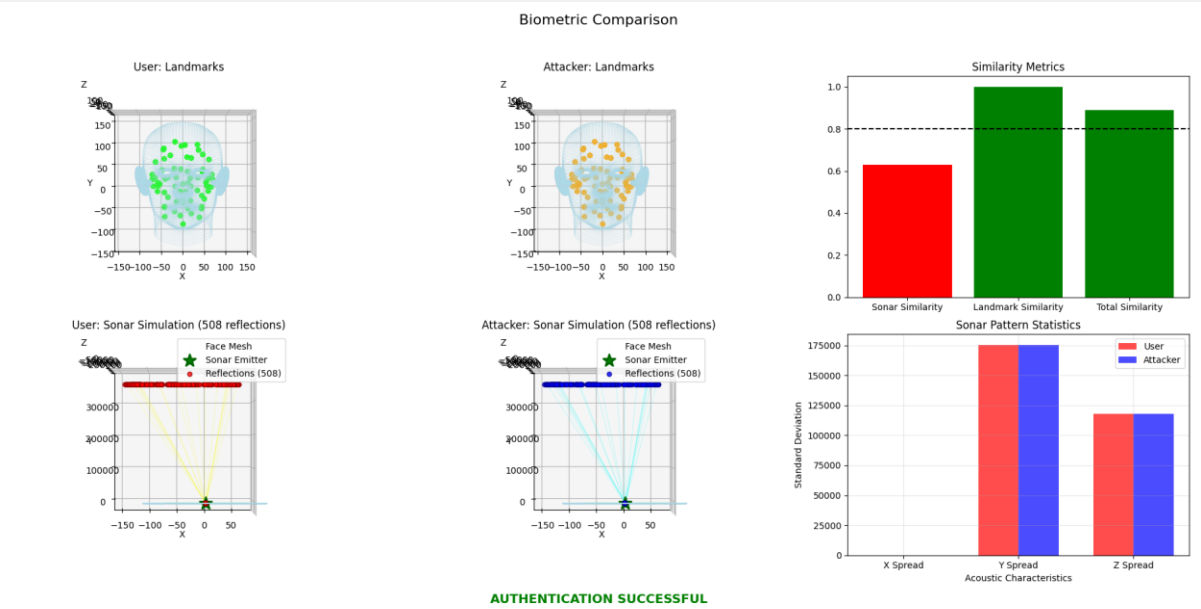


As an added feature and considering the use of this system to identify identity theft by AI or manipulated images, an additional feature called Spoofing Attack Simulator was added. When enabled, instead of performing normal authentication, it adds an extra step by taking the attacker's 3D facial object and converting it into a 2D object that simulates a 3D object. This is intended to demonstrate that the system has the potential to detect identity theft thanks to its sonar system.

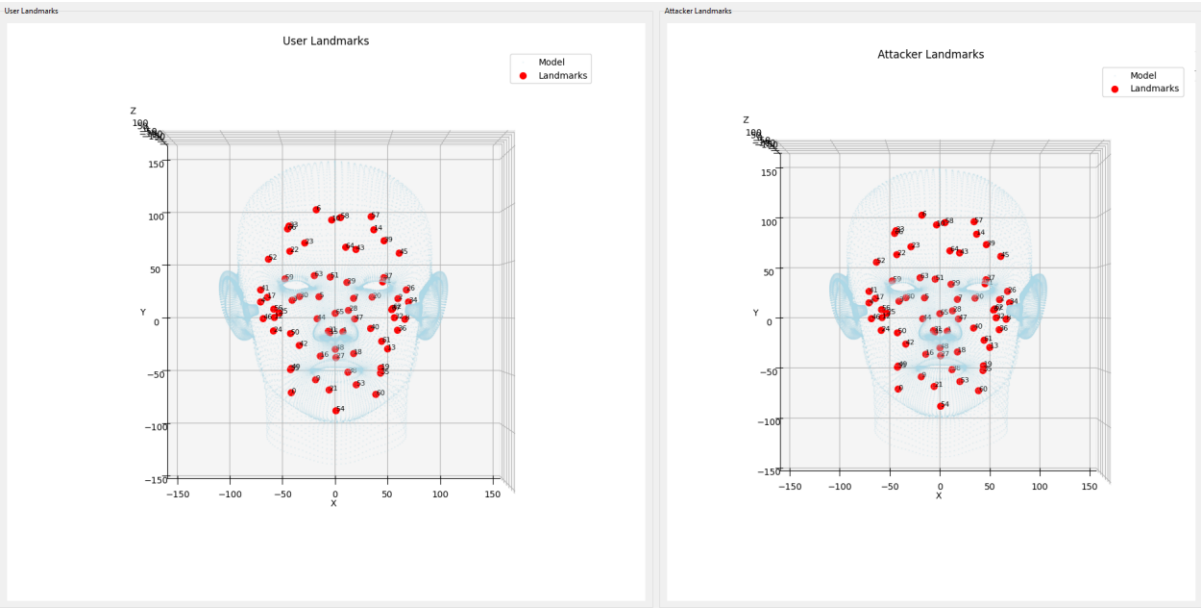


Finally, step 4, "Visualize results," allows the system user to see the various authentication processes in detail and graphically. It also provides more information about why the authentication was rejected or approved. For example, regarding the detection of landmarks,

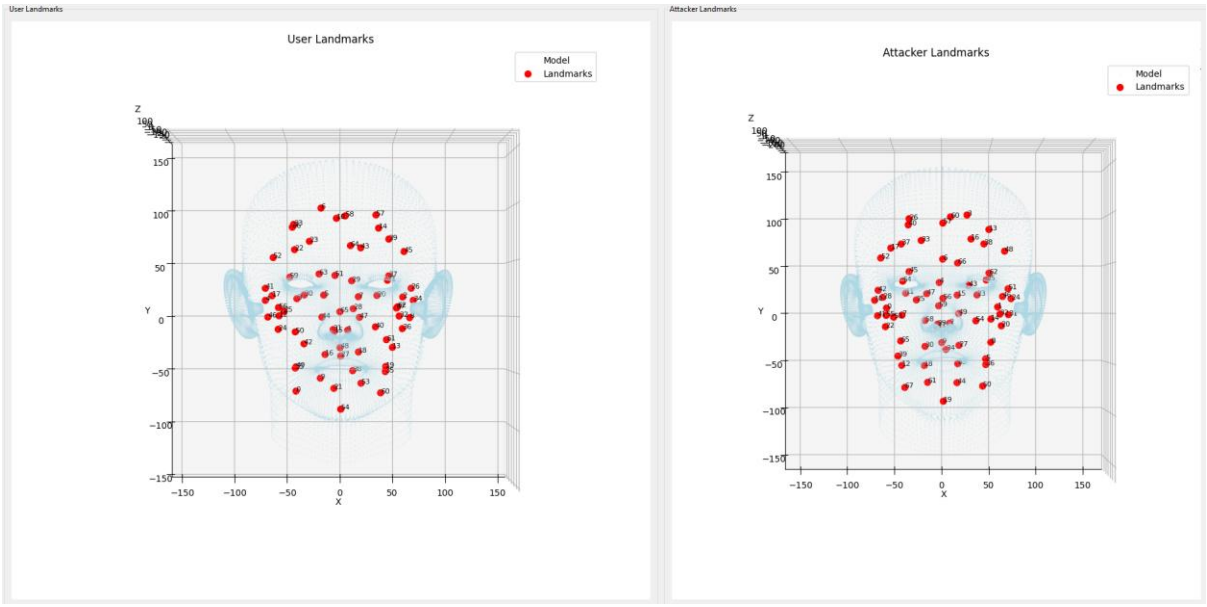
the system displays a 3D visualization of both the user and the attacker, reflecting the landmarks identified for each, thus visually demonstrating to the user the differences.



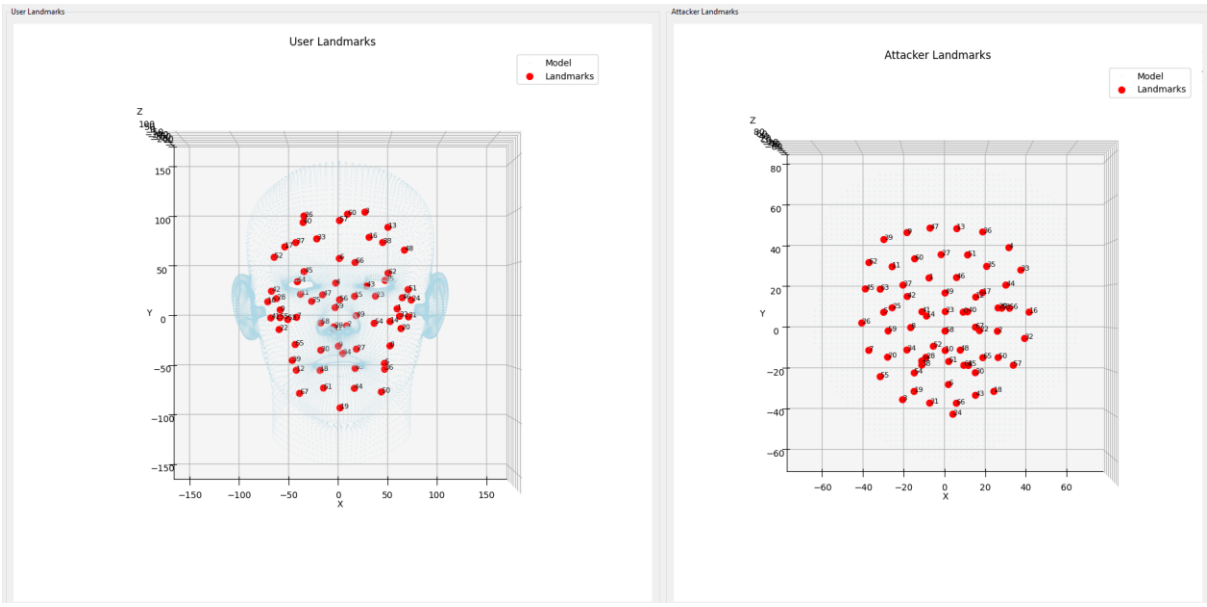
10 GUI: Visualization Results in 3 Different Scenarios



Scenario 1 Comparative of the Same Attacker and User 3D Facial Models



Scenario 2 Comparative of different Attacker and User 3D Facial Models



Scenario 3 Spoofing demo comparative of the same Attacker and User 3D Facial Models

References

Yang, H. *et al.* (2020) ‘FaceScape: a Large-scale High Quality 3D Face Dataset and Detailed Riggable 3D Face Prediction’. arXiv. Available at: https://doi.org/10.48550/arXiv.2003.13989.

Zhu, H. *et al.* (2023) ‘FaceScape: 3D Facial Dataset and Benchmark for Single-View 3D Face Reconstruction’. arXiv. Available at: https://doi.org/10.48550/arXiv.2111.01082.