Working Guide for Unauthorized Person Detection System

July 2025

A guide for setting up and running the Unauthorized Person Detection System

1 Introduction

This guide provides detailed instructions for setting up and operating the Unauthorized Person Detection System, which includes two Python-based models:

- **Face Recognition Model** (`final.py`): Uses DeepFace with FaceNet to identify authorized and unauthorized individuals.
- **Restricted Area Detection Model** (`final1.py`): Detects faces in a user-defined restricted area using Haar Cascade.

Both models rely on OpenCV for video capture and processing, and the face recognition model requires pre-trained face encodings.

2 System Requirements

- **Hardware**: A computer with a webcam.
- **Operating System**: Windows, macOS, or Linux.
- **Software**:
  - Python 3.8 or higher
  - Python packages: `opencv-python`, `numpy`, `deepface`
  - OpenCV's Haar Cascade file (`haarcascade_frontalface_default.xml`)

3 Installation

3.1 Cloning the Repository

Clone the GitHub repository to your local machine:

```
git clone https://github.com/mohamednizar17/Unauthorized-person-detection
cd Unauthorized-person-detection
```

3.2 Installing Dependencies

Install the required Python packages:

```
pip install opencv-python numpy deepface
```

Ensure OpenCV's Haar Cascade file is available (included with `opencv-python`).

3.3 Preparing Face Encodings (Face Recognition Model)

The face recognition model requires a pre-trained `authorized_encodings.pkl` file containing face embeddings and names. To generate this:

1. Collect images of authorized individuals.
2. Use a training script (not provided in the repository) to generate embeddings using DeepFace's FaceNet model.
3. Save the embeddings and names in `authorized_encodings.pkl` using Python's `pickle` module.

Place the `authorized_encodings.pkl` file in the repository's root directory.

4 Running the Models

4.1 Face Recognition Model (`final.py`)

1. Ensure `authorized_encodings.pkl` is present.
2. Run the script:

   ```
   python final.py
   ```

3. The webcam will start, and the system will:
   - Detect faces using MTCNN.
   - Compare detected faces against known embeddings.
   - Display green rectangles for authorized faces and red for unauthorized.
   - Log unauthorized detections to `security_log.txt` and save frames to the `logs/` directory.
   - Show recent detections on the console.
4. Press q to exit.

4.2 Restricted Area Detection Model (`final1.py`)

1. Run the script:

   ```
   python final1.py
   ```

2. The webcam will start, displaying a green square (restricted area).

3. Drag the square with the mouse to reposition it.
4. If a face is detected in the square, it turns red, and ”INTRUDER!” is displayed.
5. Press q to exit.

5 Troubleshooting

- **Webcam not detected**: Ensure the webcam is connected and accessible. Check the device index in `cv2.VideoCapture(0)`.
- **Encodings file not found**: Generate `authorized_encodings.pkl` using a training script for `final.py`.
- **Face detection issues**: Ensure good lighting and clear webcam input. Adjust the `FACENET_THRESHOLD` in `final.py` if needed.
- **Dependency errors**: Verify all packages are installed correctly using `pip list`.

6 Notes

- The restricted area in `final1.py` is draggable but not resizable. Future updates may add resizing functionality.
- Logs and frames are saved for unauthorized detections to aid in security monitoring.
- The system assumes a single webcam. Modify the code if using multiple cameras.

7 Contact

For issues or contributions, visit the repository: `https://github.com/mohamednizar17/Unauthorized-person-detection`