Hochschule Hof

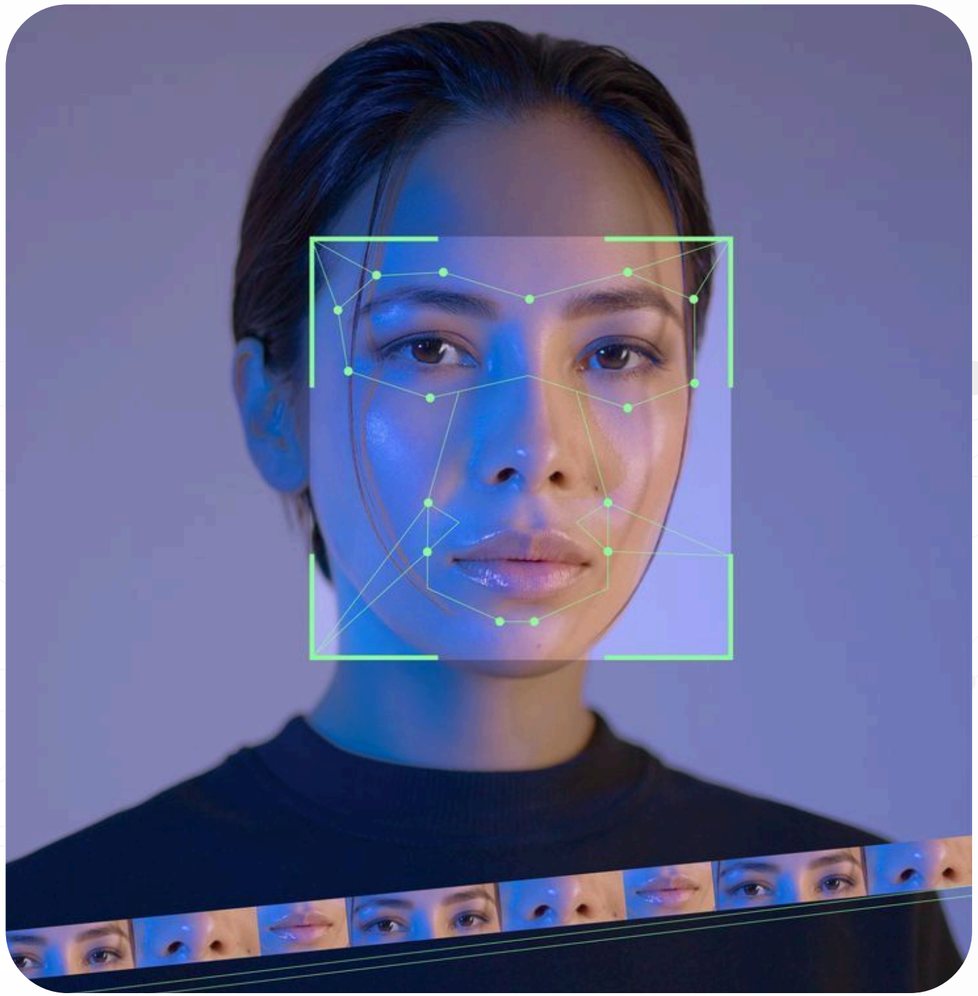
Presented by **Aruzhankalka Salimova, Aidin Kydyraliev, Malika Makkambai, Myktybek Abdykaiymov, Nurjamal Osmonalieva**

Facial Recognition Alert System

Facial Recognition Alert System
Home Security Robotics Project

Camera → Detection → Recognition → Alert (sound + email)

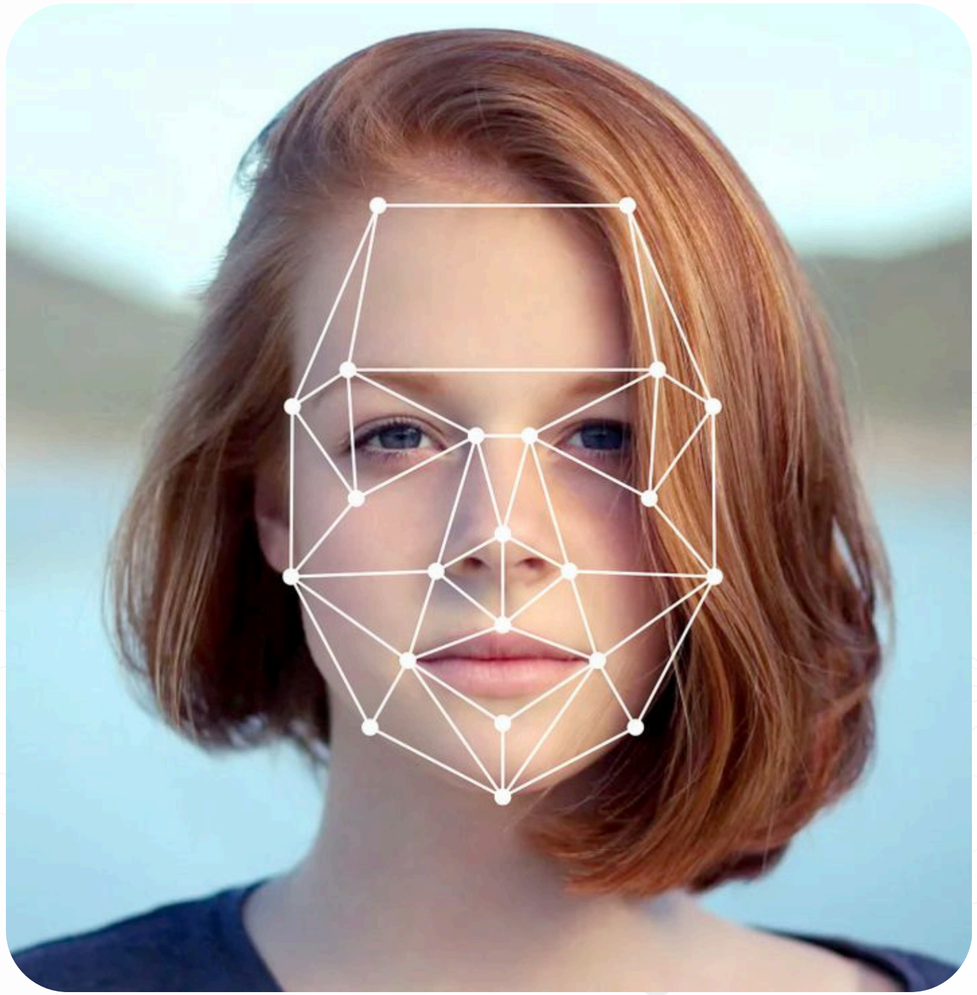
Group 20



Hochschule Hof

Outline

1.Project Goal
2.Essential Theory
    2.1 Face Detection
    2.2 Face Recognition
3.Practical Implementation
    3.1 SetUp
    3.2 System Architecture
    3.3 Alert Logic
4.Future Improvements
5.Conclusion



Project Goal

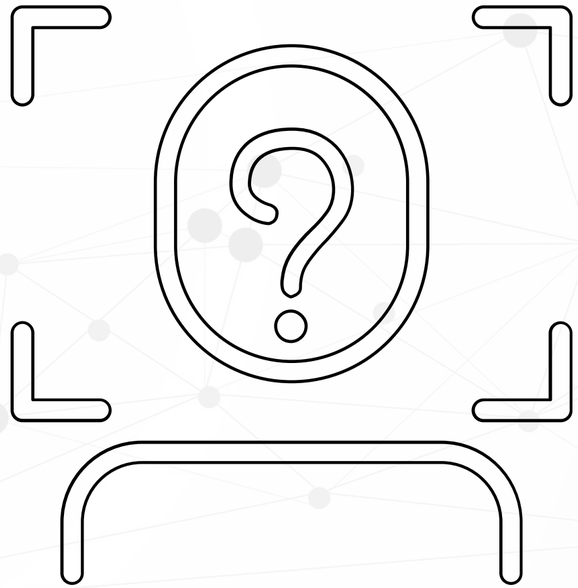
The goal of our project is to develop a prototype door security system that:

- captures live video from a camera
- detects faces in real time
- recognizes known individuals using a local database
- triggers a sound alert and email notification for unknown persons

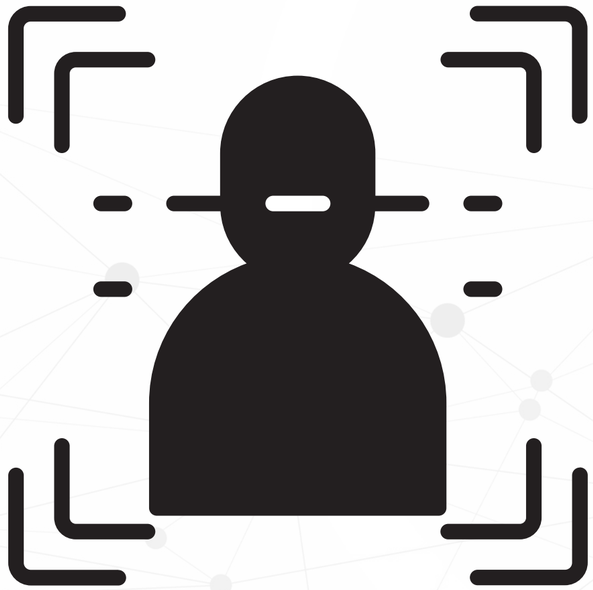
Hochschule Hof

Face Detection – Essential Theory

- Detect faces in each frame (bounding boxes)
- Crucial step before recognition
- Detection errors → recognition errors → false alerts



Hochschule Hof



Presented by **Group 20**

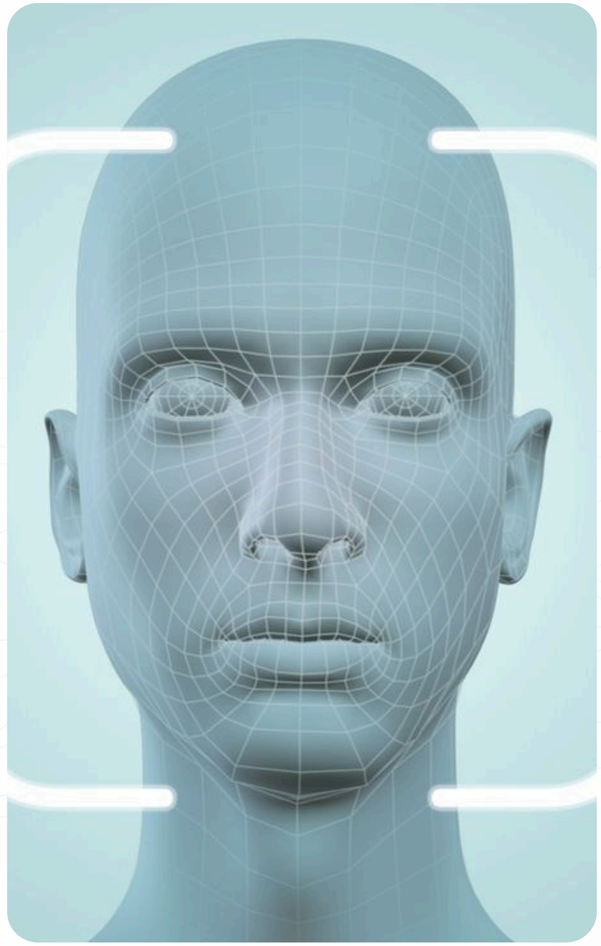
RetinaFace

- Modern deep learning–based detector
- Single-shot → fast, one pass per frame
- Multi-scale → detects faces of any size
- Robust in-the-wild → works with rotated faces, poor lighting, multiple faces
- Predicts key facial landmarks → increases accuracy

Hochschule Hof

RetinaFace Workscade

- Point regression predicts bounding box + key points (eyes, nose, mouth)
- Multi-task learning improves detection accuracy
- Feature Pyramid Network detects faces at multiple scales



Other Models We Tried

Criteria (door security):

- Haar Cascade: classical, brightness-based, fast, less accurate
- Dlib HOG: gradient-based, more robust to lighting than Haar, still less accurate
- Tested for experiments, but final system uses RetinaFace only

Final choice: RetinaFace

Face Detection Model Comparison

Model	Speed (CPU)	Accuracy & Robustness	Pros	Cons
OpenCV (Haar Cascade)	Very fast	Low–Medium	Fast, simple	Outdated, misses faces
dlib HOG	Slow	Medium	More robust than Haar	CPU-heavy
RetinaFace	Slowest	High	Finds almost all faces	Very slow on CPU

Final choice: RetinaFace

FACE RECOGNITION – Essential Theory

How it works?

- Detect a face in the image/video
- Crop the face (focus only on the face region)
- Extract a “faceprint” = a numeric vector (embedding)
- Compare that embedding with a database of known embeddings

Uses embeddings + similarity distance + threshold
If distance ≤ threshold → Known
If distance > threshold → Unknown

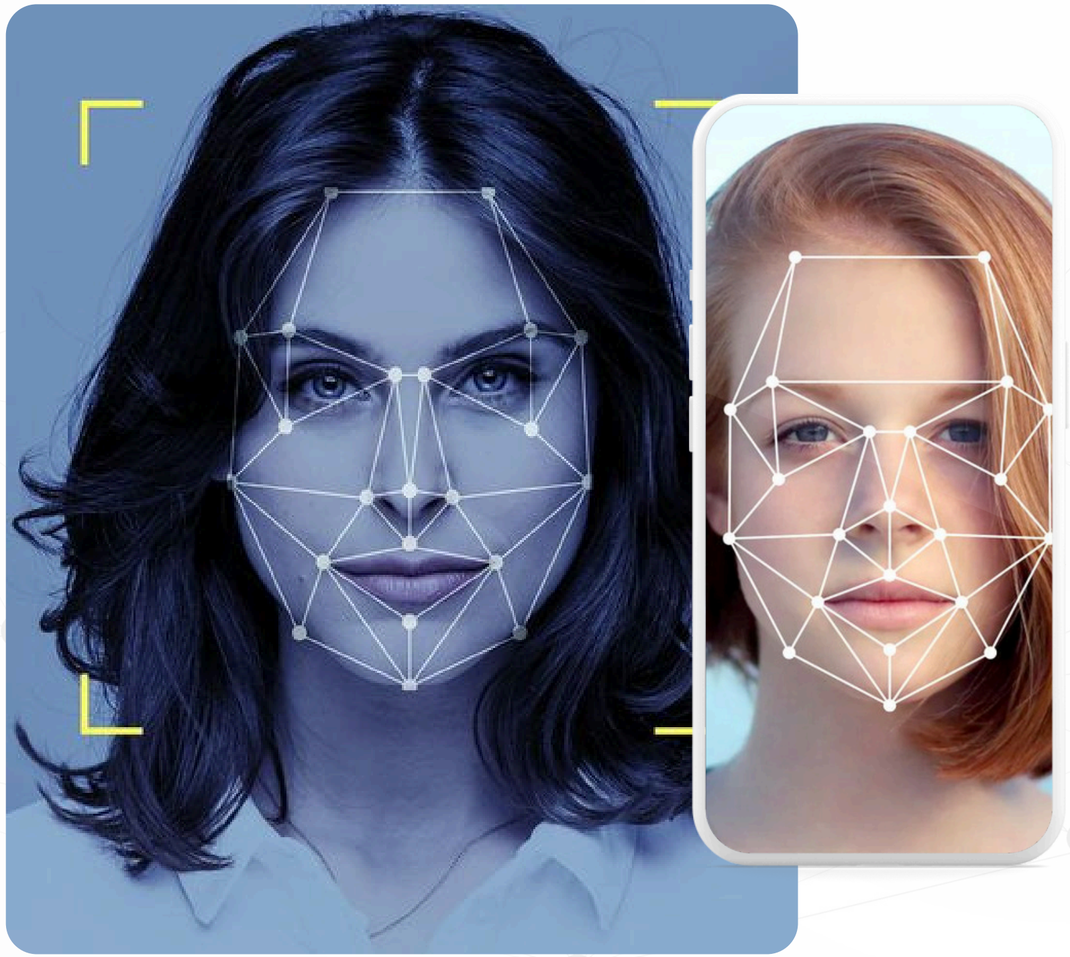
Known identities are stored in our local database (JSON/SQLite) as embeddings/templates.

How CNNs Create the “Faceprint” (Embedding)

CNN feature learning pipeline

- Convolution layers: learn patterns from simple to complex (edges → shapes → facial parts like eyes/nose/jawline)
- Pooling layers: shrink the feature maps (keeps strongest info, more robust to small shifts/lighting changes)
- Fully-connected layers: convert learned features into a compact vector

→ the embedding (“faceprint”) used for matching



ArcFace

ArcFace is a machine learning model that takes two face images as input and outputs the distance between them (threshold) to see how likely they are to be the same person.
It works as deep embedding model + Additive Angular Margin Loss to learn highly discriminative facial features.
The system compares embeddings using cosine similarity

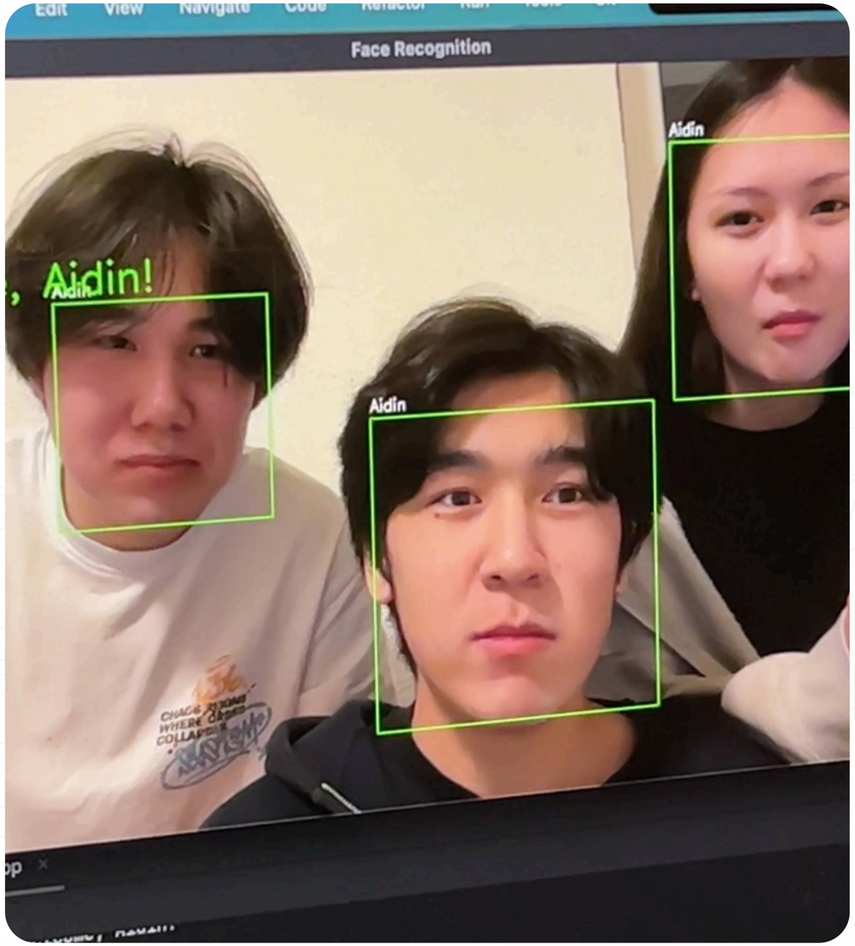
ArcFace performance depends strongly on detection quality → reason RetinaFace helps.

Presented by **Group 20**

Initial Problem

In the first implementation using a DeepFace CNN baseline,
the system sometimes recognized three different people as one identity.
This issue motivated us to:

- analyze face detection and recognition quality
- compare different models
- improve identity separation using more robust embeddings



Hochschule Hof

Face Detection Model Comparison

Presented by **Group 20**

Model	Accuracy (identity separation)	Speed on old CPU laptops	Robustness (light/pose)	Pros	Cons
Deepface (CNN model)	Medium	Faster	Medium	Simple baseline, easy to run in DeepFace framework	Can confuse similar faces, sensitive to crop quality / detector quality
ArcFace	High	Slow	High	Strong identity separation (margin-based training), works well with cosine distance	Heavier model → on old laptops FPS drops; on newer hardware or GPU it performs much better

Final choice: ArcFace

Practical Implementation

Setup

Software
Python 3.9-3.11
Libraries: opencv-python, numpy, ArcFace, retina-face, playsound, smtplib, email.message

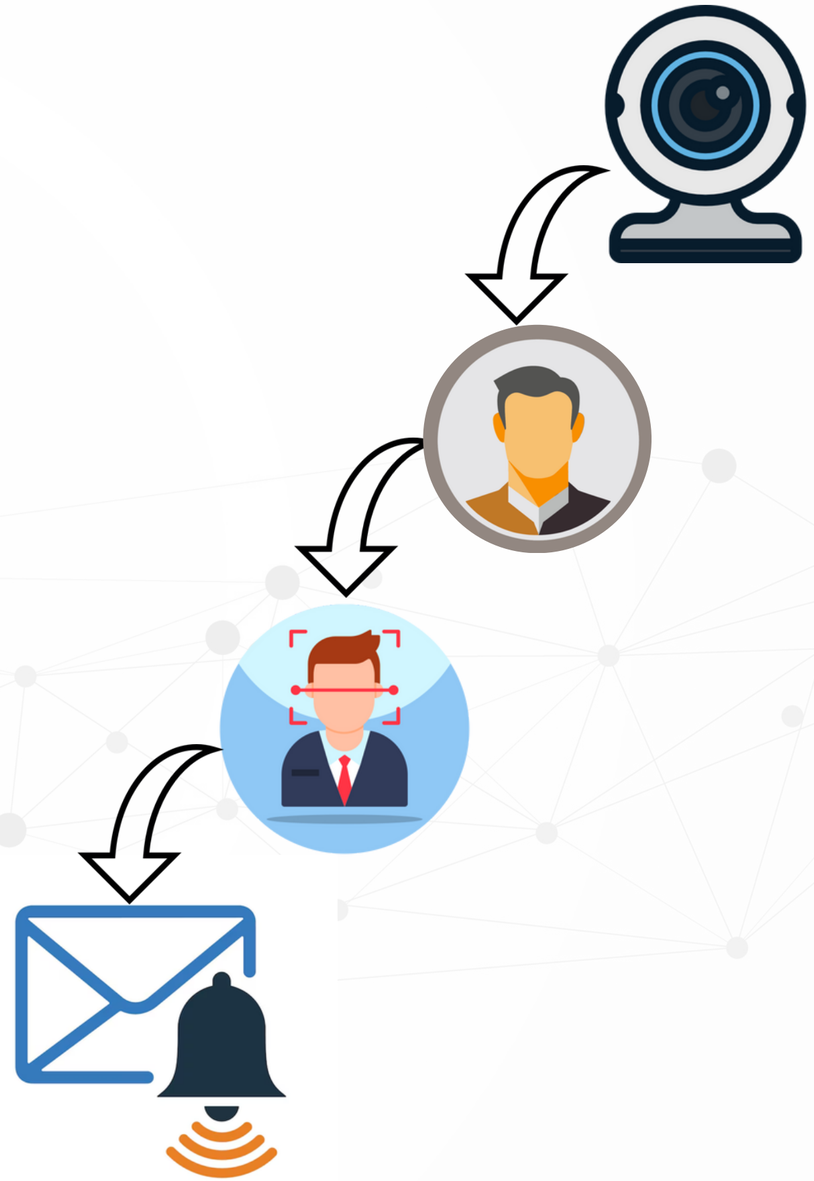
Hardware
Webcam
Speakers for alert

Database
JSON file with known face embeddings (gallery_db.json)
Format supports multiple embeddings per person

Email Setup
Environment variables:
ALERT_EMAIL – your Gmail
ALERT_EMAIL_PASS – app password
ALERT_TO_EMAIL – recipient

System Architecture

- Camera → Detection → Recognition → Alerts
- Live video from a web camera.
- Faces are detected, converted into embeddings, and classified as known or unknown.

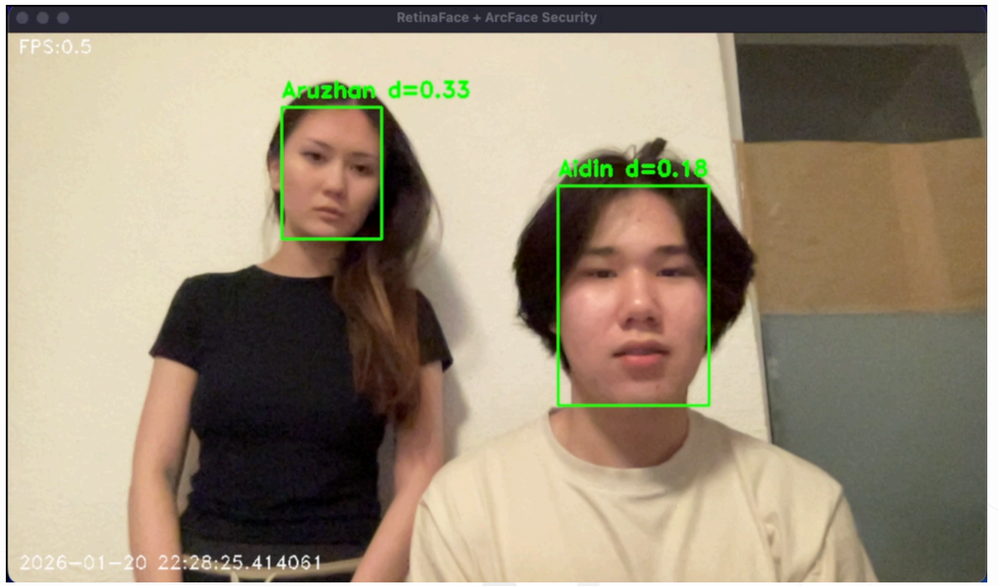


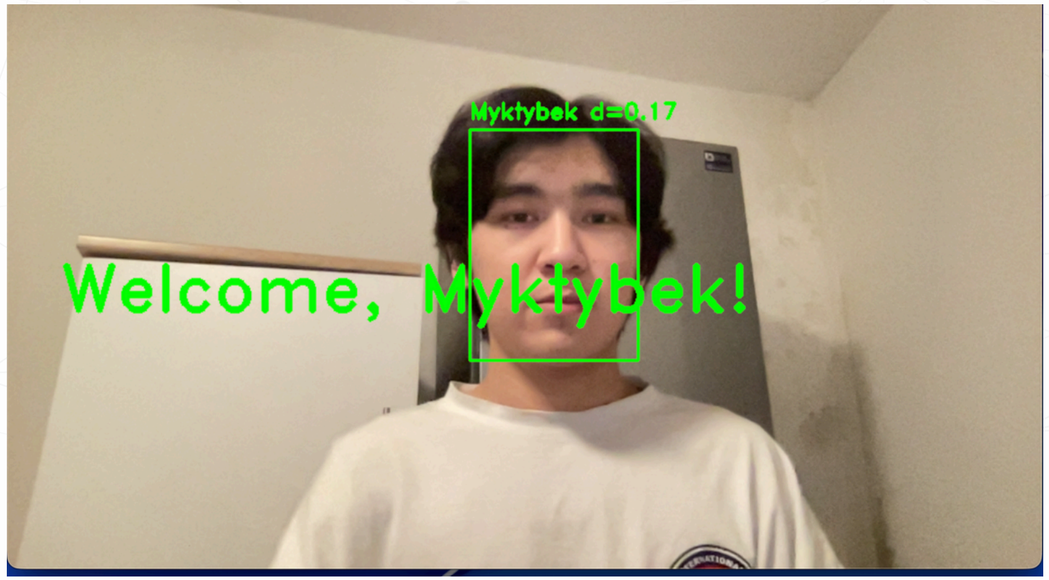
Hochschule Hof

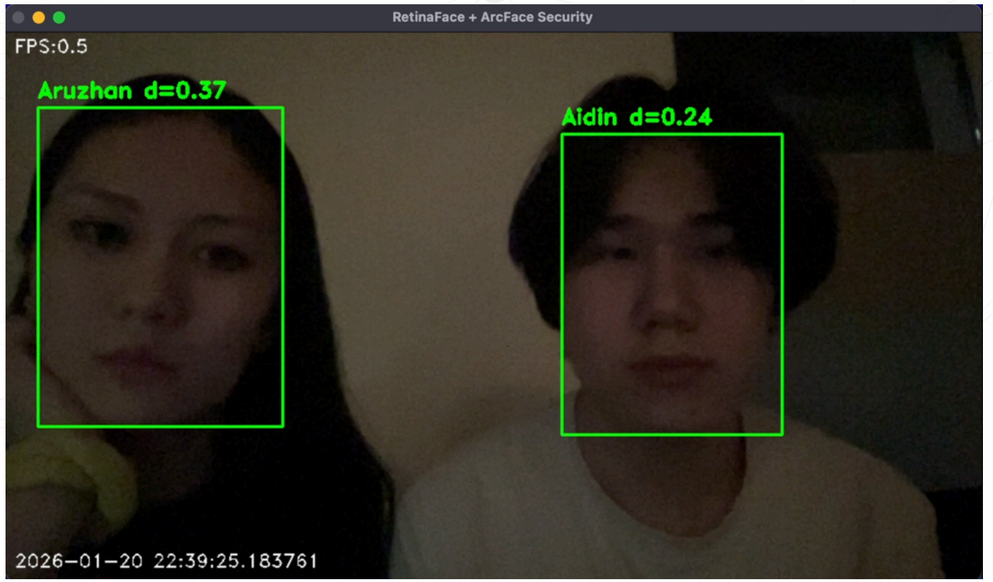
Alert Logic

For known person

- Matched person for the next 5 seconds
- Welcome + known_person_name





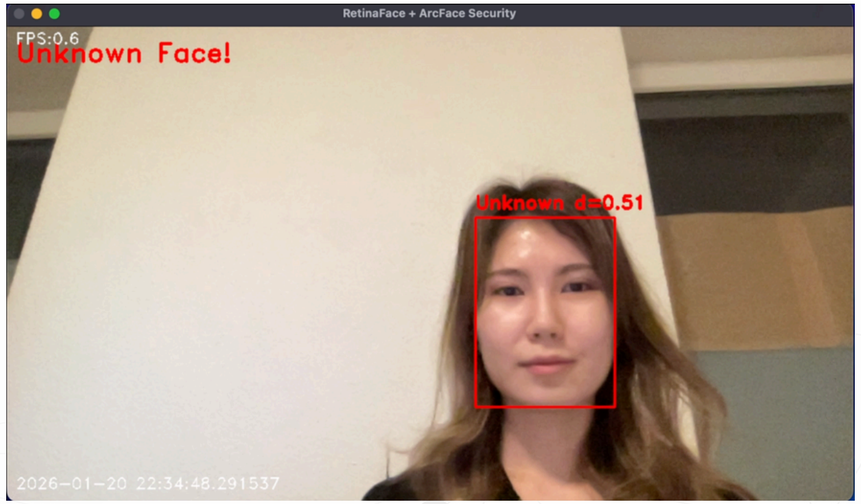


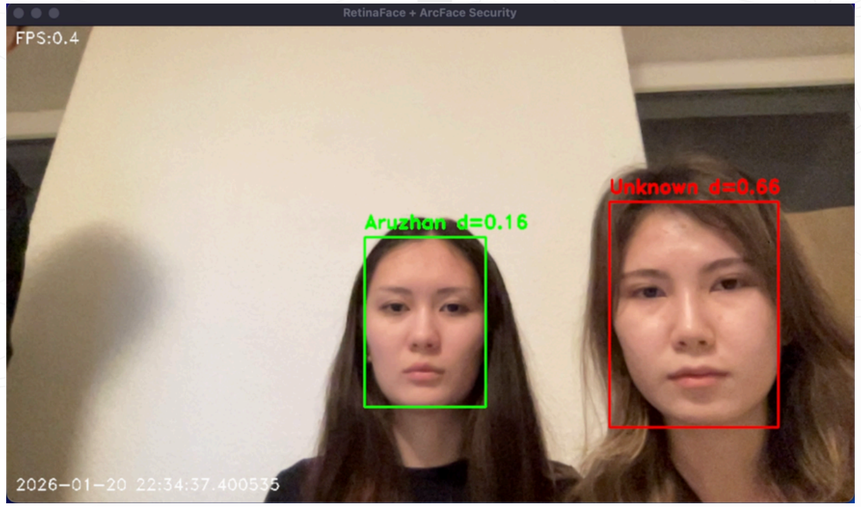
Hochschule
Hof

Alert Logic

For unknown person

- Unmatched person for the next 10 seconds
- Screenshot

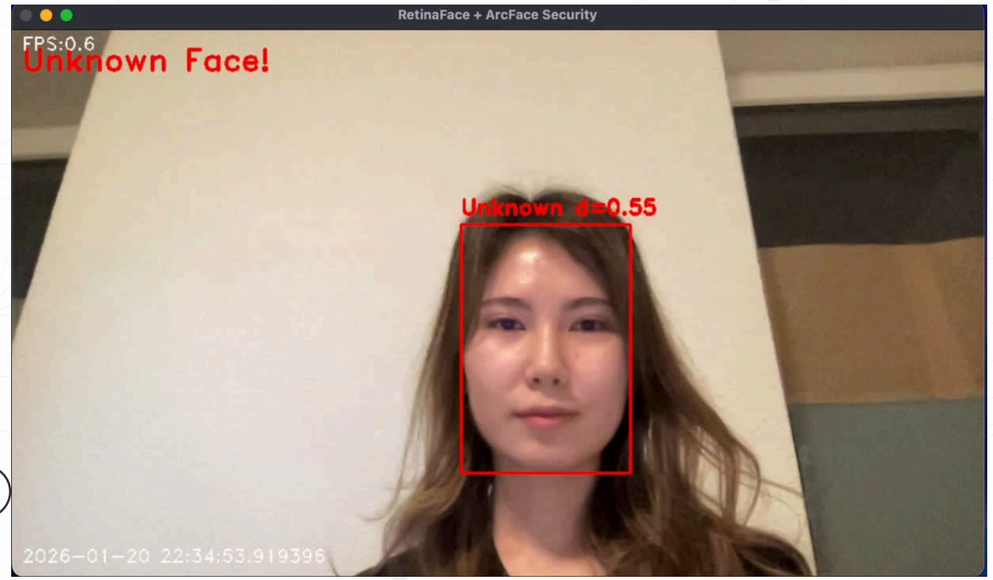


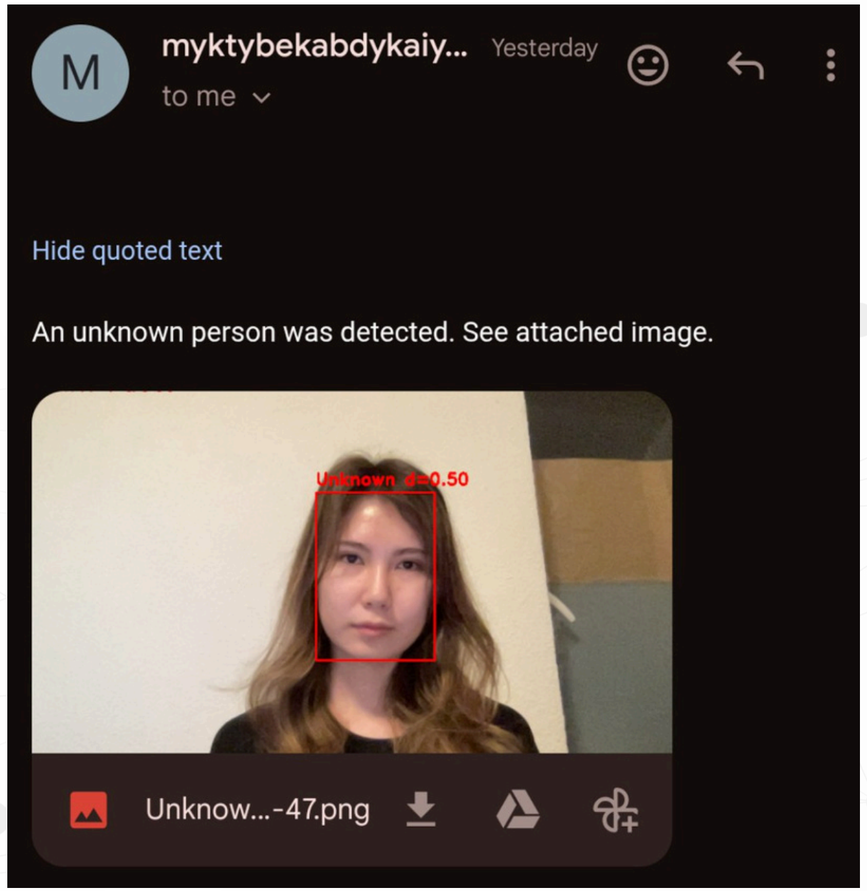


Alert Logic

For unknown person

- Sound Alert activate
- Send via email



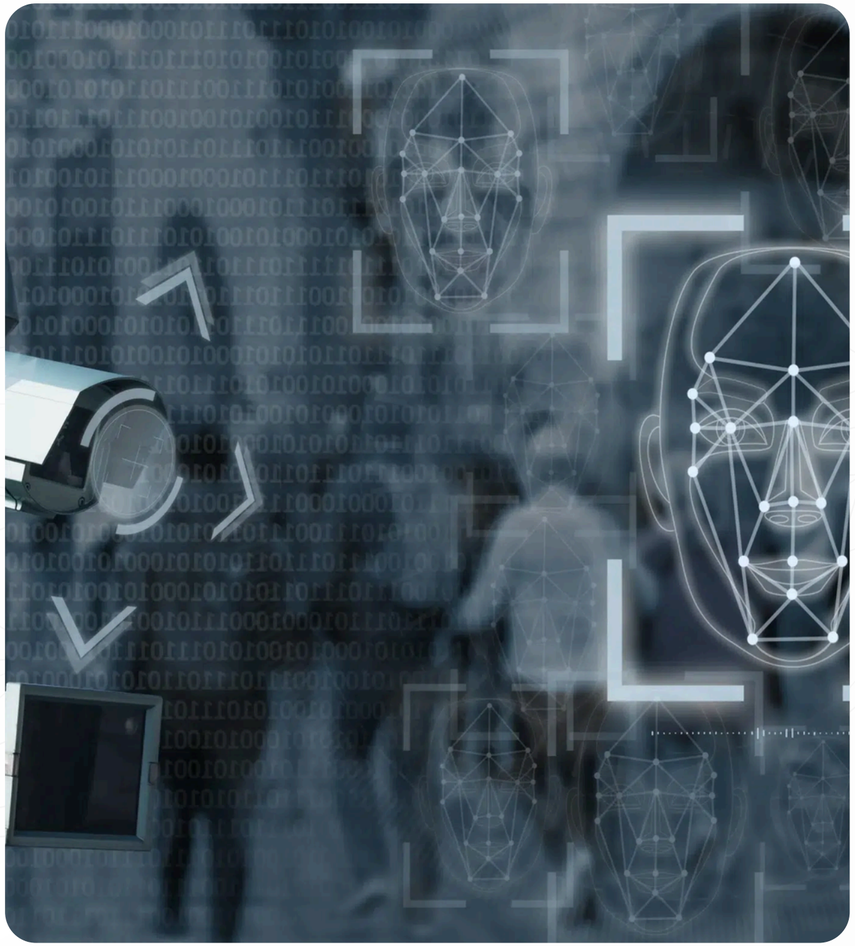


Hochschule Hof

Presented by **Group 20**

Outlook/Future improvements

- Replace laptop webcam with a real security camera
- Tune threshold with real camera samples (reduce false matches)
- Faster search in database (cache embeddings; optional SQLite later)
- Add anti-spoof / liveness (future security upgrade)



Conclusion

We developed a facial recognition–based home security prototype using a modular pipeline:
Camera → Detection → Recognition → Alert
Our comparison showed that:

- Detection quality directly impacts recognition accuracy
- RetinaFace provides the most reliable face detection
- ArcFace embeddings achieve better identity separation than simpler models

Despite lower CPU performance, the RetinaFace + ArcFace combination significantly improved system reliability, making it suitable for real-world security scenarios.

References

Books and Journals

Deng, J./Guo, J./Ververas, E./Kotsia, I./Zafeiriou, S.: RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020, pp. 5203–5212.

Electronic Resources

Cochard, D.: ArcFace: A Machine Learning Model for Face Recognition. Medium, 2021. Available at: [Accessed: January 20, 2026].

Khan, A.: What is Face Detection? The Ultimate Guide. LearnOpenCV, 2022. Available at: [Accessed: January 20, 2026].

King, D.: dlib C++ Library, 2009. Available at: [Accessed: January 20, 2026].

OpenCV: Cascade Classifier — Object Detection using Haar features, 2023. Available at: [Accessed: January 20, 2026].

Hochschule Hof

Thank you for your attention!