AKADEMIA GÓRNICZO-HUTNICZA

Documentation for the project

Closed-Circuit TeleVision system

For the subject:

Design Laboratory

Elektronika i telekomunikacja 3 rok

Bartłomiej Kisielewski, Kacper Śliwa

Tuesday 8:00

Course leader Dr. Inż. Jacek Kołodziej

29.01.2025r.

Design analysis, models and prototype

1. **Prototype Version: camera.py**

- **The camera.py file served as an initial prototype for the CCTV system, implementin basic functionality:**
  - Connecting two cameras – one via RSTP, the other with usb camera
  - Motion detection and recording – using Haar classifiers for face and body detection
  - Video recording – saving footage in MP4 format with a defined codec.
  - Combining camera feeds – displaying merged frames in a single window.
  - Recording control – continuing to record for 5 seconds after motion detection stops
- However, the prototype has several limitations:
  - The code operates in a single main loop, handling cameras without modularity
  - No advanced recording management (e.g., automatic deletion of old videos).
  - No dynamic camera addition or real-time mode switching

2. **Final version: cctv_project.py and run_cameras.py**

The final version introduces modularity and scalability.

- cctv_project.py – The CameraMonitor Class
  - Object-oriented approach – each camera is an independent object
  - 2 recording modes – motion detection and continuous recording
  - Automatic deletion of old recordings – based on a retention period
  - Dynamic recording management – recordings are assigned per camera instance

- run_cameras.py – Main Control Module
  - Support for multiple cameras – users can define the number of cameras and sources (usb, rstp, htpp, etc.)
  - Flexible mode switching – users can switch between motion detection and continuous recording
  - User – friendly interface – displays a camera grind in a single window

Link to repository: https://github.com/Darned99/CCTV-System-Design-Lab-PY

- During testing, it was observed that the system performed differently depending on the number of active cameras. With two cameras, the transmission was fast and had minimal delay. However, when scaling up to four cameras, there was a noticeable increase in latency, affecting real-time performance. This suggests that the processing unit plays a crucial role in system efficiency, and upgrading to a more powerful unit could enhance performance.

- Additionally, the motion detection algorithm parameters should be further analyzed. Increasing detection accuracy in:

  - faces = self.face_cascade.detectMultiScale(gray, 1.3, 5)
  - bodies = self.body_cascade.detectMultiScale(gray, 1.3, 5)

  reduces the number of frames processed per second, as more computational resources are required.

- Further optimization could involve:
  - Choose better processing unit to maintain low latency with multiple cameras
  - Fine – tuning detection parameters to balance accuracy and frame rate.
  - Exploring alternative detection models

The final version of the system represents a significant step forward, but further testing and refinements are needed to optimize both scalability and detection performance.