mnt_surveillance

Implement a video surveillance and analysis system that harnesses the power of distributed computing, enabling multiple services to run seamlessly across various machines. The design will leverage ROS2 (Robot Operating System 2) for its robust communication framework, ensuring efficient data flow and service integration.

Video Source(Python / CPP)

- Web or USB or built in Camera
- File Player (for pre-recorded (**resulted from the recorder you implement**) raw data files)

Specifications

Video Acquisition

- The system should be equipped to access a live feed from a Web or USB camera or retrieve frames from a pre-recorded raw data file.

Frame Processing

- Each video frame should conform to a specified resolution of your choice.
- Frames can be assumed to be grayscale.
- The pixel depth of each frame should be adjusted to 10 bits. This might necessitate upscaling from 8 bits or downscaling from 16 bits, depending on the source.

Streaming

- Processed video frames should be streamed to a predefined ROS topic: /video/raw_data.
- The streamed data should exclusively consist of raw pixel values.
- Each message transmitted to the topic should not exceed a size of 100 bytes.

Frame Rate

- For both video sources, the system should maintain a consistent frame rate of 30 frames per second (FPS).

Recorder (CPP)

- The recorder process will be configured to a specific video stream and file path. It should store the video in a binary file. Store only the raw data and do it in a packed manner, meaning only the 10 active bits should be stored without any padding.

Analyzer (CPP)

- The analyzer framework will listen to the video stream and once a frame is ready it will apply various algorithms on it.

- The applied algorithms should be configurable from the outside and are provided as 3rd party either source or precompiled libraries. Please elaborate on how a new algorithm will be introduced to the system.
- A configuration yaml file is to be used to configure which algorithms will be applied and in what order
- For example one of the algorithms can be over-exposure detection to protect the system from burglars who will try to apply direct light into the camera to avoid detection. An alarm will be sounded if more than 20% of the image is fully saturated.

Alarm (Python / CPP)

- A listener node that will expose a ROS2 service, once an alarm is triggered a log will be displayed on the screen.

Display

- This node will open a window and show real live data upon request.