Parallel Frame Processing System

Overview

The Parallel Frame Processing System is designed to capture frames from an MP4 file and apply parallel processing. The system consists of the following classes:

- **SystemApplication**: Provides the system **API**. Coordinates the management of frame capture and processing modules.

- **FrameCapture**: Reads video frames from a video file.

- **ProcessingModule**: Abstract base class. Defines a common interface for all processing modules.

- **FastModule**: A processing module that calculates the frame rate every 5 seconds (a defined parameter).

- **SlowModule**: A processing module that saves frames from a queue with a maximum capacity of 5 frames (a defined parameter) and applies a sleep period (a defined parameter) after each file save.

- **Logger**: Provides logging of system events.

API – System Application Class

1. Constructors:

The class provides two Constructors:

- **SystemApplication(const std::string& filePath)**

Initializes the system with the provided video file and is ready for processing execution.

Convenient for single video file processing.

- SystemApplication()

Initializes the system without a specific video file. Requires the video file to be set before starting the processing.
Convenient when processing different files is required.

2. Methods

- void setFilePath(const std::string& filePath)

Sets or updates the video file path to be processed.

- void start()

Starts the frame processing pipeline.

Execution Flow

1. Verifies if the video file is valid and opened, if not, terminates the method.
2. Spawns separate threads for the FrameCapture – frames reading and each processing module.
3. Waits for the FrameCapture thread to complete frames reading.
4. Notifies all processing threads that frame reading is complete.
5. Waits for all processing modules to finish.

* Threads flow diagram provided in page 3.

3. Private Members

- Logger m_logger **-** Manages system logging.

- String m_file_path - The file path/name of the video file being processed.

- **bool** m_done - A flag indicating when frame capture is completed.

- vector<ProcessingModule*> m_processors - Enables smooth operation of all processing modules and convenient extensibility.

- FrameCapture m_frame_capture - Responsible for reading frames from the video file and passing them to the processing modules.

- FastModule m_fast_module - Responsible for calculating and logging the frame rate every 5 seconds.

- **Slow Module** m_slow_module - Responsible for saving frames from a queue. It drops the oldest frame when the queue is full.

Thread Flow Diagram