Parallel Video Editing Application

Group 23

Presentation Overview

- Research
- Implementation
- Parallelisation Concepts
- Demo
- Future Work

Research

Existing Video Editing Apps and features

- Adobe Premier Pro
- Lightworks
- Avidemux
- Natron

All of these apps have the following features

- Audio/Video Mapping
- Colour manipulation
- Titling and visual effects
- Applying Filters to videos

OpenCV

OpenCV (Open Source Computer Vision) is an open source library mainly aimed at real-time computer vision. Applications like CamScanner,Motion Detector Pro use OpenCv.

OpenCV's application areas include:

- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Object identification
- Motion tracking

FFmpeg

FFmpeg is an open source software that has libraries for handling multimedia data. Multimedia applications like VLC Player uses FFmpeg.

FFmpeg includes

- libavcodec - an audio/video codec library used by several multimedia apps
- libavformat - an audio/video container mux and demux library
- the ffmpeg command line program for transcoding multimedia files.

JavaCV

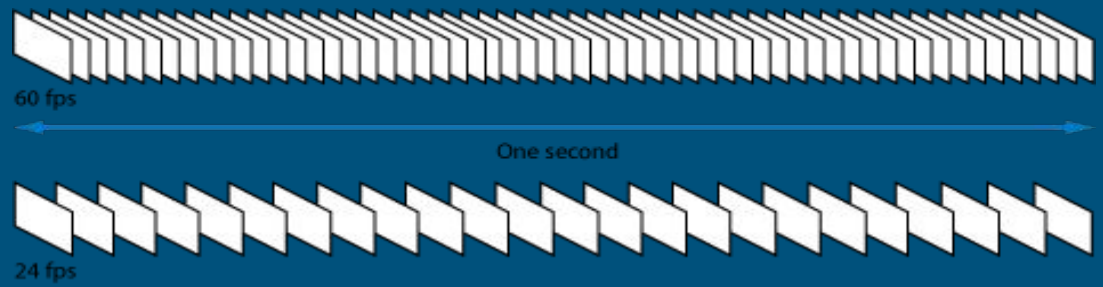
JavaCV is a Java wrapper which incorporate libraries such as OpenCV and FFmpeg.

This means that it takes the functionality of these API and implements them in classes so that they can be used in Java development.

It simplifies the use of the underlying libraries by simplifying interfaces.

Video Frames

- Every video is made up of a sequence of image frames.
- These frames are played at a certain speed in order for to create the motion picture.
- The frame rate is what determines how fast the frames are played.



Implementation

Implementation

In our implementation we have utilised JavaCV in order to edit videos.

We have used a few useful classes to filter videos.

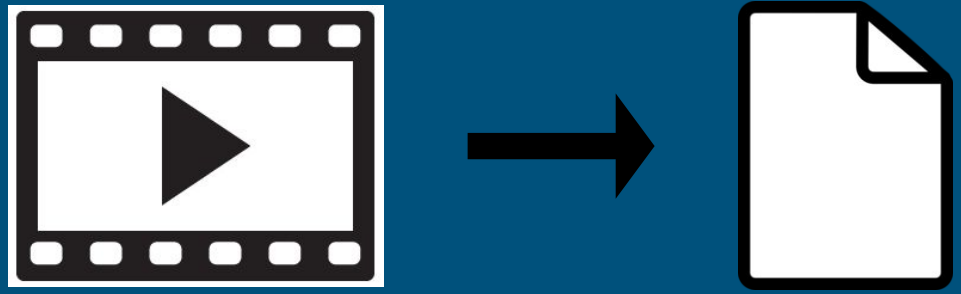
- Frame
- FFmpegFrameGrabber
- FFmpegFrameFilter
- FFmpegFrameRecorder

JavaCV - Frame

- This class is used to represent a frame of a video file.
- It holds the image and the audio samples for a particular frame of a video.
- These can be accessed and modified as required.

JavaCV - FFmpegFrameGrabber

- This component allows you to open a video file & allows access to frames of a video.
- Once the file is open you can call grabFrame() which grabs the next frame in the video.
- Used as an input stream.
- You can use this to modify frames within a video.



JavaCV - FFmpegFrameFilter

- This class is used to apply filters to a frame.
- Frames are pushed and a filter is applied, then they are pulled with the filter applied.
- It is initialized with a preset FFmpeg filter.
- The FFmpeg library contains many filter presets to choose from. A few examples are:
  - Greyscale
  - Sepia
  - Negative

Filter Examples

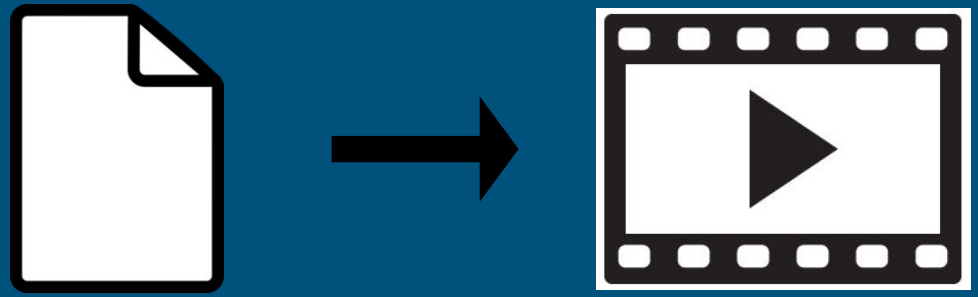
Original

Greyscale

Sepia

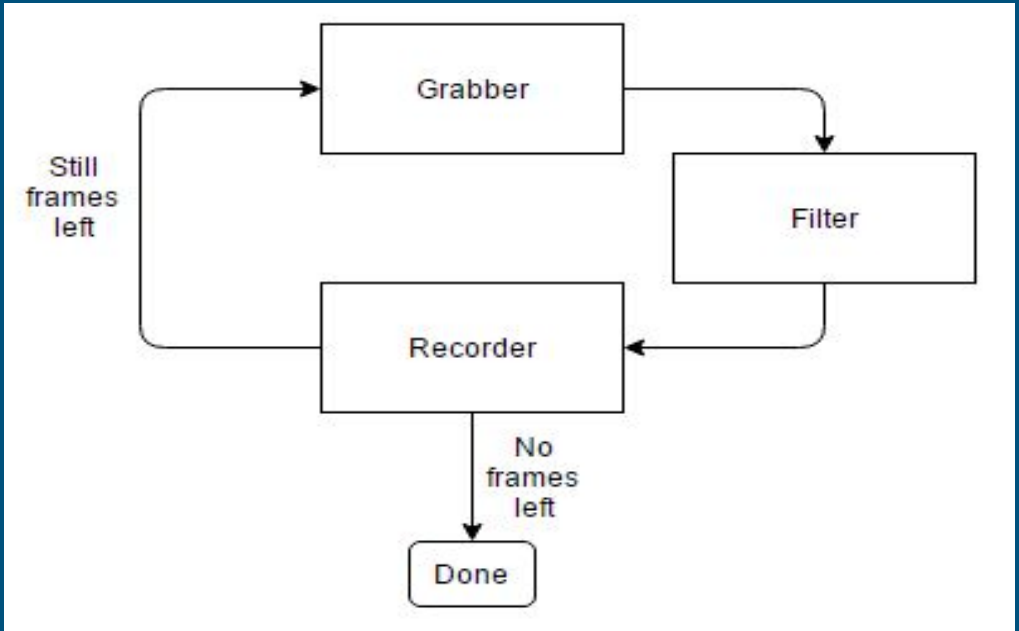
JavaCV - FFmpegFrameRecorder

- This component is used to record frames to an output file.
- An output file is declared on initialization.
- Used as an output stream.
- The timestamp of a frame can be retrieved and set to the same timestamp on the output file.
- This means that the video frames will be in order.



Frame Filtering Process

These components are all used together in order to apply filters to a video.



GUI

Currently we have used Swing to create our GUI.

We have used JavaFX for displaying video files.

Parallelisation Concepts

Parallelisation Process

Decomposition

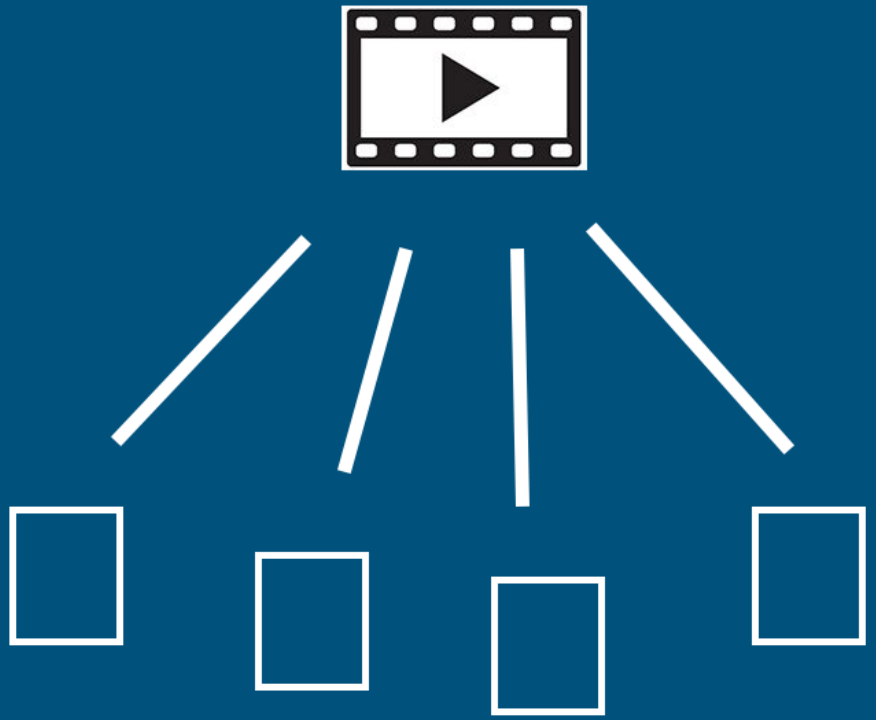
- Data decomposition

Dependency analysis

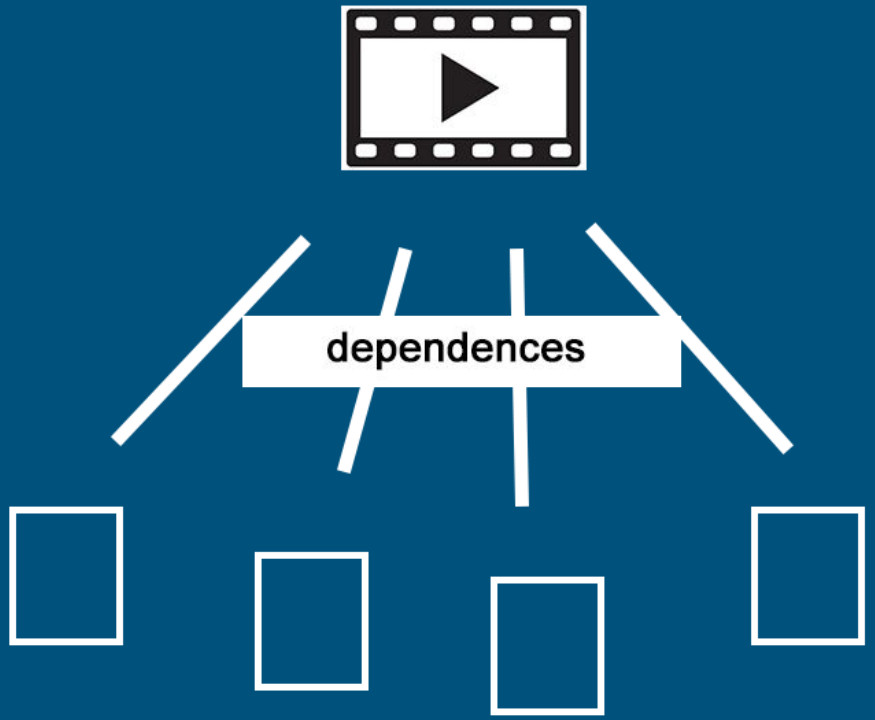
Scheduling

- Work Sharing

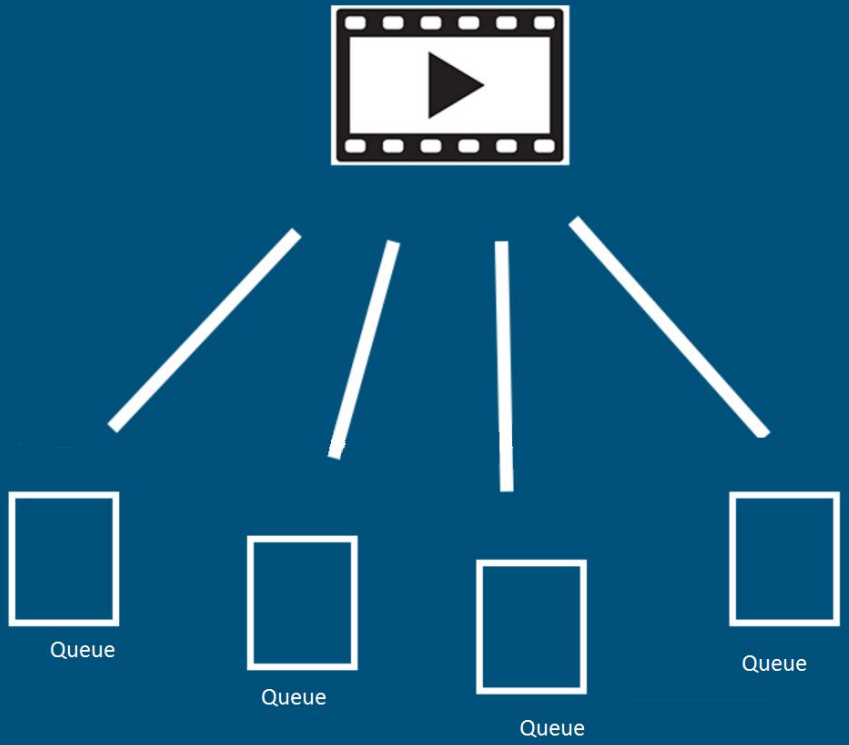
Decomposition



Dependency



Scheduling



Parallelisation Tasks

Computational tasks:

- Split video into subsections (One-off task)
- Add filters to frames in subsections (Multi task)
- Combine subsections together (One-off task)

I/O tasks

- GUI thread - EDT

ParaTask

Parallel Task (ParaTask)

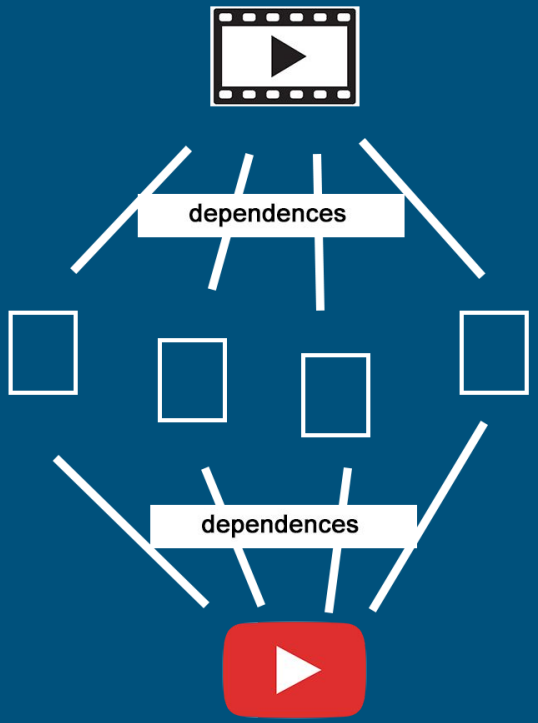
Main concepts

- TASK
- TASK(*)
- IO_TASK (was INTERACTIVE_TASK)
- dependsOn
- getResult( )
- notifyGUI

Parallel Task

- ~~Code restructuring~~
- ~~Thread management~~
- ~~Dependence management~~
- ~~Coupling between tasks~~
- ~~Performance hit~~
- ~~Not suitable for GUI applications~~
- ~~Model does not unify parallelisation concepts~~

Putting together



Putting together

```
public void actionPerformed()
{
        ... ...
        TaskID id = videoEditor.work() notify( finished());
}

TASK public void work()
{
        TaskID idFrames = split(videoPath);
        TaskID idFilteredFrame = addFilter() dependsOn(idFrames);
        TaskID idFilteredVideo = combine() dependsOn(idFilteredFrame);
        idFilteredVideo.waitTillFinished();
}
```

Demo

Demo

Main Functionalities

- Filters
- Audio/Video
- Saving filtered video to file

Future Work

Future Work

- Implement parallelised version of video filtering.
- Examine different length videos to gain an understanding of when parallelising is worthwhile and when it is not.
- Add in a filter preview for the video on the GUI.
- Add in a progress bar.
- Improve GUI.

References

https://github.com/bytedeco/javacv

http://parallel.auckland.ac.nz/ParallelIT/PT_About.html

https://computing.llnl.gov/tutorials/parallel_comp/

Questions?