Video editing

Objectives

- Get to know ways of quick video editing to be able to provide accessible videos
- Learn how video editing can be distributed and done the same day.

Instructor note

- Teaching: 20 min
- Exercises: 20 min

Video recordings could be useful for people watching later, but also are (perhaps more) useful for **immediate review and catching up after missing a day in a workshop**. For this, they need to be released immediately, within a few hours of the workshop. CodeRefinery does this, and you can too.

Hypothesis: videos must be processed the same evening as they were recorded, otherwise (it may never happen) or (it's too late to be useful). To do that, we have to make processing *good enough* (not perfect) and *fast* and the work *distributeable*.

Primary articles

- Video editor role description: https://coderefinery.github.io/manuals/video-editor/
- ffmpeg-editlist: the primary tool: https://github.com/coderefinery/ffmpeg-editlist
  - Example YAML editlists: https://github.com/AaltoSciComp/video-editlists-asc

How this relates to streaming

- If you stream, then the audience *can not* appear in the recorded videos
- This allows you to release videos *very quickly* if you have the right tools.
- When you have a large audience, the videos start helping more (review a missed day, catch up later, review later)
- Thus
  - If you would never want videos, there may never be a benefit to streaming
  - If you want videos, it gives motivation to stream.

Summary

- Basic principle: privacy is more important than any other factor. If we can't guarantee privacy, we can't release videos at all.
    - Disclaimers such as "if you don't want to appear in a recording, leave your video off and don't say anything", since a) accidents happen especially when coming back from breakout rooms. b) it creates an incentive to not interact or participate in the course.
- Livestreaming is important here: by separating the instruction from the audience audio/video, there is no privacy risk in the raw recording. They could be released or shared unprocessed.
- Our overall priorities
    1. No learner (or anyone not staff) video, audio, names, etc. are present in the recordings.
    2. Good descriptions.
    3. Removing breaks and other dead time.
    4. Splitting videos into useful chunks (e.g. per-episode), perhaps equal with the next one:
    5. Good Table of Contents information so learners can jump to the right spots (this also helps with "good description".)
- ffmpeg-editlist allows us to define an edit in a text file (crowdsourceable on Github), and then generate videos very quickly.

How we do it

The full explanation is in the form of the exercises below. As a summary:

- Record raw video (if from a stream, audience can't possibly be in it)
- Run Whisper to get good-enough subtitles. Distribute to someone for checking and improving.
- Define the editing steps (which segments become which videos and their descriptions) in a YAML file.
- Run ffmpeg-editlist, which takes combines the previous three steps into final videos.

Exercises

Exercise A

These exercises will take you through the whole sequence.

✍ Editing-1: Get your sample video

Download a sample video:

- Video (raw): http://users.aalto.fi/~darstr1/sample-video/ffmpeg-editlist-demo-kickstart-2023.mkv
- Whisper subtitles (of raw video): http://users.aalto.fi/~darstr1/sample-video/ffmpeg-editlist-demo-kickstart-2023.srt
- Schedule of workshop (day 1, 11:35–12:25) - used for making the descriptions. :::::

✍️ Editing-2: Run Whisper to generate raw subtitles and test video.

First off, install Whisper and generate the base subtitles, based on the. Since this is probably too much to expect for a short lesson, they are provided for you (above), but if you want you can try using Whisper, or generating the subtitles some other way.

You can start generating subtitles now, while you do the next steps, so that they are ready by the time you are ready to apply the editlist. ffmpeg-editlist can also slice up the subtitles from the main video to make subtitles for each segment you cut out.

Whisper is left as an exercise to the reader.

✔ Solution

Example Whisper command:

```
$ whisper --device cuda --output_format srt --initial_prompt="Welcome to
CodeRefinery day four." --lang en --condition_on_previous_text False
INPUT.mkv
```

An initial prompt like this make Whisper more likely to output full sentences, instead of a stream of words with no punctuation.

✍️ Editing-3: Create the basic editlist.yaml file

Install ffmpeg-editlist and try to follow its instructions, to create an edit with these features:

- The input definition.
- Two output sections: the “Intro to the course”, and “From data storage to your science” talks (Remember it said the recording started at 11:35… look at the schedule for hints on when it might start!). This should have a start/end timestamp from the *original* video.

A basic example:

```
- input: day1-raw.mkv

# This is the output from one section.  Your result should have two of these
sections.
- output: part1.mkv
  title: something
  description: >-
    some long
    description of the
    segment
  editlist:
    - start: 10:00    # start timestamp of the section, in *original* video
    - end: 20:00      # end timestamp of the section, in the *original* video
```

✔ Solution

This is an excerpt from our actual editlist file of this course

```
- input: day1-obs.mkv

- output: day1-intro.mkv
  title: 1.2 Introduction
  description: >-
    General introduction to the workshop.

    https://scicomp.aalto.fi/training/kickstart/intro/

  editlist:
  - start: 00:24:10
  - end: 00:37:31


- output: day1-from-data-storage-to-your-science.mkv
  title: "1.3 From data storage to your science"
  description: >-
    Data is how most computational work starts, whether it is
    externally collected, simulation code, or generated. And these
    days, you can work on data even remotely, and these workflows
    aren't obvious. We discuss how data storage choices lead to
    computational workflows.

    https://hackmd.io/@AaltoSciComp/SciCompIntro

  editlist:
  - start: 00:37:43
  - end: 00:50:05
```

💬 Discussion: what makes a video easy to edit?

- Clear speaking and have high audio quality.
- For subtitle generation: Separate sentences cleanly, otherwise it gets in a “stream of words” instead of “punctuated sentences” mode.
- Clearly screen-sharing the place you are at, including section name.

- Clear transitions, “OK, now let’s move on to the next lesson, LESSON-NAME. Going back to the main page, we see it here.”
- Clearly indicate where the transitions are
- Hover mouse cursor over the area you are currently talking about.
- Scroll screen when you move on to a new topic.
- Accurate course webpage and sticking to the schedule

All of these are also good for learners. By editing videos, you become an advocate for good teaching overall.

✍ Editing-4: Run ffmpeg-editlist

Install ffmpeg-editlist: `pip install ffmpeg-editlist[srt]` (you may want to use a virtual environment, but these are very minimal dependencies).

The `ffmpeg` command line tool must be available in your `PATH`.

✔ Solution

It can be run with (where `.` is the directory containing the input files):

```
$ ffmpeg-editlist editlist.yaml .
```

Just running like this is quick and works, but the stream may be garbled in the first few seconds (because it’s missing a key frame). (A future exercise will go over fixing this. Basically, add the `--reencode` option, which re-encodes the video (this is **slow**). Don’t do it yet.

Look at the `.info.txt` files that come out.

✍ Editing-5: Add more features

- Several chapter definitions.(re-run and you should see a `.info.txt` file also generated). Video chapter definitions are timestamps of the *original* video, that get translated to timestamps of the *output* video.

```
- output: part1.mkv
  editlist:
  - start: 10:00
  - -: Introduction    #  <-- New, `-` means "at start time"
  - 10:45: Part 1      #  <-- New
  - 15:00: Part 2      #  <-- New
  - end: 20:00
```

Look at the `.info.txt` files that come out now. What is new in it?

- Add in “workshop title”, “workshop description”, and see the `.info.txt` files that come out now. This is ready for copy-pasting into a YouTube description (first line is the title, rest is the description).

  Look at the new `.info.txt` files. What is new?

✔ Solution

- This course actually didn’t have chapters for the first day sessions, but you can see chapters for day 2 here, for example.
- Example of the workshop description for this course
- Example info.txt file for the general introduction to the course. The part after the `-----` is the workshop description.

```
1.2 Introduction - HPC/SciComp Kickstart summer 2023

General introduction to the workshop.

https://scicomp.aalto.fi/training/kickstart/intro/

00:00 Begin introduction       <-- Invented for the exercise demo, not real
03:25 Ways to attend           <-- Invented for the exercise demo, not real
07:12 What if you get lost     <-- Invented for the exercise demo, not real

-----

This is part of the Aalto Scientific Computing "Getting
started with Scientific Computing and HPC Kickstart" 2023
workshop.  The videos are available to everyone, but may be
most useful to the people who attended the workshop and want
to review later.

Playlist:
https://www.youtube.com/playlist?list=PLZLVmS9rf3nMKR2jMglaN4su3ojWtWMVw

Workshop webpage:
https://scicomp.aalto.fi/training/scip/kickstart-2023/

Aalto Scientific Computing: https://scicomp.aalto.fi/
```

✍️ Editing-6: Subtitles

Re-run ffmpeg-editlist with the `--srt` option (you have to install it with `pip install ffmpeg-editlist[srt]` to pull in the necessary dependency). Notice how `.srt` files come out now.

Use some subtitle editor to edit the *original* subtitle file, to fix up any transcription mistakes you may find. You could edit directly, use `subtitle-editor` on Linux, or find some other tool.

What do you learn from editing the subtitles?

✔ Solution

```
$ ffmpeg-editlist --srt editlist.yaml
```

There should now be a `.srt` file also generated. It generated by finding the `.srt` of the original video, and cutting it the same way it cuts the video. Look and you see it aligns with the original.

This means that someone could have been working on fixing the Whisper subtitles while someone else was doing the yaml-editing.

✍ Editing-6: Subtitles

Re-run ffmpeg-editlist with the `--srt` option (you have to install it with `pip install ffmpeg-editlist[srt]` to pull in the necessary dependency). Notice how `.srt` files come out now.

Use some subtitle editor to edit the *original* subtitle file, to fix up any transcription mistakes you may find. You could edit directly, use `subtitle-editor` on Linux, or find some other tool.

What do you learn from editing the subtitles?

✍ Editing-7: Generate the final output file.

- Run ffmpeg-editlist with the `--reencode` option: this re-encodes the video and makes sure that there is no black point at the start.
- If you re-run with `--check`, it won't output a new video file, but it *will* re-output the `.info.txt` and `.srt` files. This is useful when you adjust descriptions or chapters.

💬 **Discussion: how to distribute this?**

Create a flowchat of all the parts that need to be done, and which parts can be done in parallel. Don’t forget things that you might need to do before the workshop starts.

How hard was this editing? Was it worth it?

Exercise B

This is a more limited (and older) version of the above exercise, using an synthetic example video.

✍️ **Use ffmpeg-editlist to edit this sample video**

Prerequisites: `ffmpeg` must be installed on your computer outside of Python. Be able to install ffmpeg-editlist. This is simple in a Python virtual environment, but if not the only dependency is `PyYAML`.

- Download the sample video: http://users.aalto.fi/~darstr1/sample-video/sample-video-to-edit.raw.mkv
- Copy a sample editlist YAML
- Modify it to cut out the dead time at the beginning and the end.
- If desired, add a description and table-of-contents to the video.
- Run ffmpeg-editlist to produce a processed video.

✔ **Solution**

```
- input: sample-video-to-edit.raw.mkv
- output: sample-video-to-edit.processed.mkv
  description: >
  editlist:
    - start: 00:16
    - 00:15: demonstration
    - 00:20: discussion
    - stop: 00:25
```

```
$ ffmpeg-editlist editlist.yaml video/ -o video/
```

Along with the processed video, we get `sample-video-to-edit.processed.mkv.info.txt` ::

```
This is a sample video


00:00 Demonstration
00:04 Discussion
```

See also

- ffmpeg-editlist demo: https://www.youtube.com/watch?v=thvMNTBJg2Y
- Full demo of producing videos (everything in these exercises): https://www.youtube.com/watch?v=_CoBNe-n2Ak
- Example YAML editlists: https://github.com/AaltoSciComp/video-editlists-asc

Keypoints

- Video editing is very useful for learning
- Set your time budget and make it good enough in that time
- Reviewing videos improves your teaching, too.