The Livius Documentation

Author: Parnia Bahar

May 21, 2015

Abstract

This Documentation serves as a short description for all required knowledge and previous works regarding the LiVius project in Max-Planck institute of intelligent system, Empirical Inference department. The main aim of the work is to automatically create an appropriate layout for online courses and lectures. In this case, a proper framework should be composed of both the lecture slides and the speaker/teacher. To this aim, computer vision and machine learning based algorithms are applied to detect the slides and track the person of interest.

1 Introduction

There are many advantages to online and computer-based learning when compared to traditional face-to-face courses and lectures, however, there are a few disadvantages as well. Online courses are great for self-motivated individuals who want to learn new skills and advance their careers and their knowledge without attending in the college or university. Getting education online will also save tons of money. Although some would argue that online education is only an awesome alternative to traditional education because of the savings and convenience, there are actually many other advantages. Only online education fully integrates itself into todays educational technology. It is also more efficient for fast and especially motivated learners and offers skills that lack resources in traditional education despite high demand from employers.

Class work can be scheduled around work and family. it reduces travel time and travel costs for off-campus students. Students may have the option to select learning materials that meets their level of knowledge and interest. Students can study anywhere they have access to a computer and Internet connection. Self-paced learning modules allow students to work at their own pace. Flexibility to join discussions in the bulletin board threaded discussion areas at any hour, or visit with classmates and instructors remotely in chat rooms. Instructors and students both report e-Learning fosters more interaction among students and instructors than in large lecture courses. e-Learning can accommodate different learning styles and facilitate learning through a variety of activities. Develops knowledge of the Internet and computers skills that will help learners throughout their lives and careers. Successfully completing online or computer-based courses builds self-knowledge and self-confidence and encourages students to take responsibility for their learning. Learners can test out of or skim over materials already mastered and concentrate efforts in mastering areas containing new. information and/or skills [**?**]. In this case, the demand of taking online classes is increasing and there are many universities and educational organizations which offer free online courses. Therefore, there are some attempts to provide students and online-audiences with a high quality and interesting videos so that they can follow all of the course material in a desire manner. For this purpose, besides the slides/board including the taught sections, it is worthy to watch the speaker/teacher's reactions during the class.

1.1 The Framework

As it was mentioned, we try to have both speaker and slide parts. In summer school 2013, there were some attempts to do it manually or to make it by helping of two cameras. one camera has been concentrated on the slides to record the video and another one has been manually controlled by one person to track the speaker during the lecture. But the synchronization of two cameras is challenging and spending a lot of time to record the videos for speakers' movements are time consuming. Thus, a high resolusion 4k camera is used in Livius project to record the lectures and by means of image analysis, computer vision and machine learning approaches, one detects not only the slides, but also tracks the speaker in an intelligent manner as automatic as possible.

One can see the whole framework of the project in the figure **??**. The whole project can be divided into three major tasks. The first one is video processing and it is related to all algorithms, pre- and post-processing steps for slide detection and speaker tracking. Here, the processing tasks can be done

either on the video file itself or on the separate frames. The second one can be called video editing and it includes all necessary works for the codecs of the video, the final layout, the background image, the logo and concatenating all together to form the final version of the video. The third part is audio processing and it refers to all works on audio signal either stereo or mono. Finally, the improved version of Audio signal will be concatenated and synchronized with video file to create the final video with desired layout **??**.

1.2 Output video layout

In order to broadcast the final video, the full HD size is used. The final video size to stream is 1920x1320. Figure **??** depicts the final layout and the the title and background images.

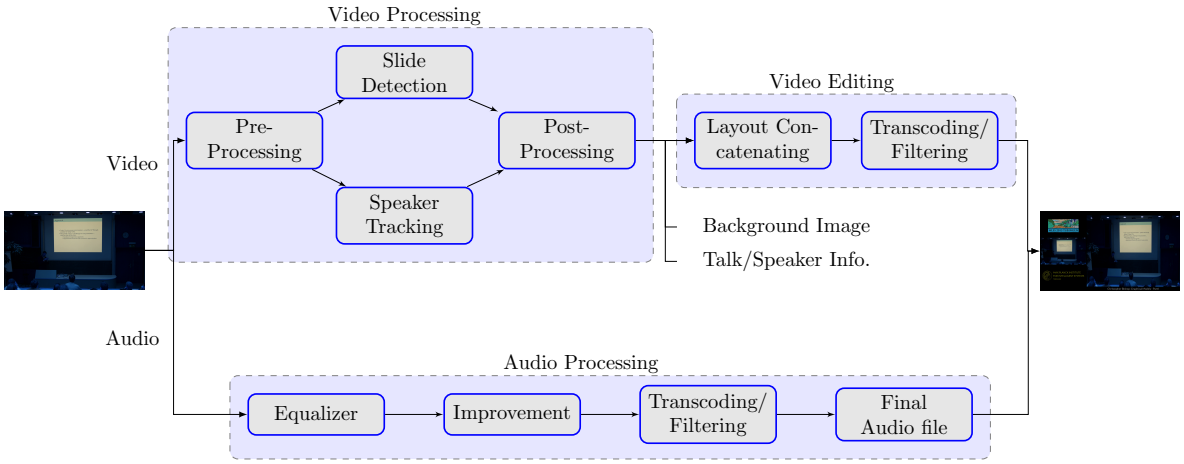
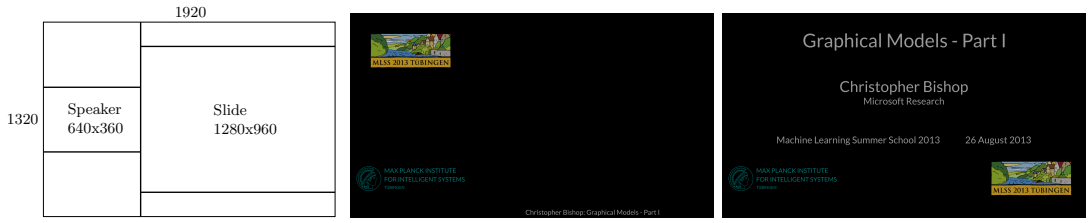


Figure 1: The whole framework



(a) Video layout (b) Background image (c) First Info. slide

Figure 2: The final video layout

2 Requirements

2.1 Hardware

The current hardware and tools are shown in figure **??**. Figure **??** indicates the 4k Blackmagic Camera, its power supply, audio cables and the ultrasonic 24mm lens. Figure **??** is the 2-TB USB3.0 external hard disk for any mobility. Figure **??** shows the 1-TB Solid State Drive (SSD) which suits with Blackmagic camera and **??** includes Sata HDD docking station and reader along with its secondary power supply and USB3.0 port. Besides the shown laptop, there is another desktop PC as an implementation platform with following properties.

- Processor: Intel Core i7-5930K CPU @ 3.50GHz 12
- Memory: 32 GiB
- Graphic: NVIDIA GeForce GTX 750/PCIe/SSE2
- Disk: 3 Data raid for 12 TB
- Operating system: Ubuntu 14.04 and windows 7

(a) (b) (c) (d) (e)

Figure 3: The hardwares

2.2 Software

There are two major aspects which should be taken into account in this work. As it was already mentioned the framework needs both video processing and video editing. Since, there is no open source library to handle both tasks, It is better to stay with C++ or Python. To this end, we checked several libraries and tools listed as follow.

1. Nuke production:

   A software for composing multiple images and effects, motion graphics, not open source, no good documentations. It seems to be for design, film editing, visualization, etc. `https://www.thefoundry.co.uk/products/nuke/`

2. MLT framework:

   It is an open source tool based on C++ for audio/video editing, media trasncoding, web stream and TV broadcasting. `http://www.mltframework.org/`

3. MoviePy:

   MoviePy is a Python module for video editing, which can be used for basic operations (like cuts, concatenations, title insertions), video compositing, or to create advanced effects. It can read and write the most common video formats. `http://zulko.github.io/moviepy/`

In order to develop new algorithms as convenient as possible, it was decided to take advantages from pre-built libraries and use Python language (Python 2.7.6 where /usr/bin/python). The major required libraries and packages have been listed as below.

- Moviepy
- ffmpeg
- OpenCV-2.4.8

2.2.1 Software Installation

In order to get Livius framework running on your local machine/laptop, one can easily follow the guided installation depicted in this section. The instruction is under linux operating system and it has tested on Ubuntu 14.04.

Python The first assumption to run the system is to have python installed on your local machine. You may find different Python versions released in the `https://www.python.org/downloads/` [**?**]. To check whether you have already installed the python or not and find the version, you can use the command `which python` and `python --version` in terminal. If you do not have it, then firstly you need to install some dependencies, therefore:

```
(sudo) apt-get install build-essential
(sudo) apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev
libgdbm-dev libc6-dev libbz2-dev
```

Then download using the following command, extract the files and go to the directory

```
cd ~/Downloads/
wget http://python.org/ftp/python/2.7.6/Python-2.7.6.tgz
tar -xvf Python-2.7.6.tgz
cd Python-2.7.6
```

Now, install using the command as below:

```
./configure
make
sudo make install
```

After installing the python, you can simply type `python` for python prompt. The next step is to install your required python packages. It is strongly recommended to install "pip"which is a package management system used to install and manage packages written in Python. "pip"should be included for Python 2.7.9 and later (on the python2 series), and Python 3.4 and later by default. (You may us get-apt package manager as well.)

```
sudo apt-get update
sudo apt-get install python-pip
```

you have to install the listed python packages.

```
sudo apt-get install python-pygame
sudo apt-get install python-matplotlib
sudo apt-get install python-scipy
sudo apt-get install python-numpy
```

Moviepy As it was mentioned, "moviepy"is one of the requirements to get the framework running as we use many methods from that [**?**] . MoviePy depends on the Python modules Numpy, imageio, Decorator, and tqdm, which will be automatically installed during MoviePy's installation.

```
(sudo) pip install moviepy
```

ffmpeg In order to install "ffmpeg"on your local computer, please follow the instruction on the website `https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu`.

OpenCV To run the OpenCV's libraries, one needs the cmake 2.8.12.2. Therefore, it does not exist on your O.S., use the following command.

```
sudo apt-get install cmake
```

If you are running 13.10 and you don't have a nvidia card then ensure you install "ocl-icd-libopencl1". Ubuntu 13.10 will install nvidia-319-updates as a dependency for libopencv-dev by default if ocl-icd-libopencl1 is not installed.

```
sudo apt-get install ocl-icd-libopencl1
```

To Install OpenCV on Ubuntu or Debian, you can do it from the repository or manually. a very good tutorial for this purpose is addressed in `http://milq.github.io/install-opencv-ubuntu-debian/`. You can install OpenCV from the repository.

```
sudo apt-get install libopencv-dev python-opencv
```

However, you will probably not have installed the latest version of OpenCV and you may miss some features (for example: Python 3 bindings do not exist in the repository). Therefore, install it manually with the help of mentioned link.

pykalman pykalman depends on the following modules,

- numpy (for core functionality)
- scipy (for core functionality)
- Sphinx (for generating documentation)
- numpydoc (for generating documentation)
- nose (for running tests)

All of these and pykalman can be installed using `easy_install` as:

```
easy_install numpy scipy Sphinx numpydoc nose pykalman
```

Alternatively, you can get the latest and greatest from github like

```
git clone git@github.com:pykalman/pykalman.git pykalman
cd pykalman
sudo python setup.py install
```

The installation instruction can be found in `https://github.com/pykalman/pykalman`.

3 Initial Attempts

3.1 Camera Setting

4k Blackmagic camera has different coding formats to store the recorded data and it supports two full HD sizes such as 4k and 2k video . The 4k has 1920x1080 pixels while the size of the 4f video recording is 3840x2160. For each one, there are four different codecs. The codecs are ProRes Proxy, ProRes LT, ProRes 422 and ProRes HQ so that the first one has the lowest quality and the last one takes the highest. Figure **??** shows the main menu and the sub-menus of the 4k blackmagic camera. This camera has two disk formats like HFS+ and exFAT (Figure **??**). The former supports Mac OS and the latter works for windows. There in no format for Linux. The most important part of the menu is setting by which one is able to specify various adjustments. The setting part has 4 different sections.

1. Camera (figure **??**)

   Here, it is possible to adjust camera Id, date, time, ISO, white balance and shutter angle. In our records, the ISO, white balance and shutter angle have been adjusted into 400, 5200k and 180° respectively.

2. Audio (figure **??**)

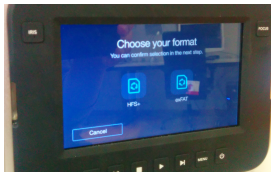
   The second section is related to audio adjustments and two microphones inputs.

3. Recording (figure **??**)

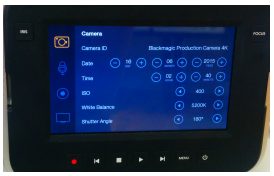
   The third part supports the recording task and provides different options. In recording format, one can choose either the raw data or one of the 8 codecs. Here, one might determine the fps and time lapse. In our experiments, we use 30 fps.

4. Display It includes some adjustments corresponding to the screen display of camera like brightness and so on.

(a) The main menu

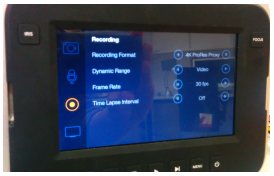


(b)



(c)

(d)



(e)

Figure 4: The setting menu of Blackmagic camera

3.2 Camera Codec Formats

The camera has 8 various recording formats as listed in table **??**. After trying them, we reached the conclusion that base on the fact that the bit-depth of the video for all formats is 10 bits and according to the size, $PSNR$ values and video quality, the **4k ProRes Proxy** was chosen to record the videos with 30 frame per second (fps). Moreover, different codings with the appropriate container have been examined as shown in table **??**. One can compare the quality of coded videos. The **.mp4** container and **x264** coder might be selected as the most promising ones for the final video.

Table 1: The different codecs of 4k Blackmagic camera

Camera codec formats	Size (MB/sec)	Size (GB/90 min)	$PSNR$
4k ProRes Proxy	23.1	125	20 Db
4k ProRes LT	51.6	279	33 Db
4k ProRes 422	74.0	400	22 Db
4k ProRes HQ	108.7	587	99 Db
2k ProRes Proxy	6.3	34	32 Db
2k ProRes LT	13.7	73	37 Db
2k ProRes 422	19.2	103	33 Db
2k ProRes HQ	28.0	151	99 Db

Table 2: The different codecs of 4k Blackmagic camera versus different kind of containers and transcoders. ✘ : bad quality , ✔ : good quality , ⊷ : medium quality

Camera codec formats	.webm (vpx)	.mp4 (x264)	.mp4 (mpeg4)	.avi (raw video)	.agv (theora)
4k ProRes Proxy	⊷	✔	✘	✘	✘
4k ProRes LT	⊷	✔	✘	✘	✘
4k ProRes 422	⊷	✔	✘	✘	✘
4k ProRes HQ	⊷	✔	✘	✘	✘
2k ProRes Proxy	✘	⊷	✘	✘	✘
2k ProRes LT	✘	✔	✘	✘	✘
2k ProRes 422	⊷	✔	✘	✘	✘
2k ProRes HQ	⊷	✔	✘	✘	✘

4 Source Code

4.1 Code Hierarchy

In this work, the Python language has been used to develop the methods. To access,follow, add and develop the source code, we have used the package and module hierarchy as follow.

```
SourceCode
├── demo.py
├── audio
│   └── audioProcessing.py
├── video
│   ├── editing
│   │   └── layout.py
│   └── processing
│       ├── CMT package
│       │   ├── CMT.py
│       │   └── util.py
│       ├── preProcessing.py
│       ├── postProcessing.py
│       ├── slideDetection.py
│       └── speakerTracking.py
└── util
    ├── templateMatching.py
    └── tools.py
```

The main folder is the "SourceCode"which includes a file `demo.py`. This file is to run the whole framework. One can simply specify the path to the input video, the path to the input background image, add some comments as the information of talk as the input arguments and get the final video with the corresponding sound.

According to figure **??**, the input file will be extracted into audio and video parts. For each part, one package is defined, i.e. two packages ("audio"and "video") are defined. Furthermore, it has a package named "util"composed of all methods for the utilities. If you intend to add any methods for secondary implementations, please ad it into `tools.py` or simply write a module and put it inside the "util"package like `templateMatching.py`.

The "audio"package is composed of the `audioProcessing.py` module in which one may add some methods/classes for audio analyses (For now, it is empty).

The "video"package is composed two sub-packages (editing and processing) as shown in figure **??**. "editing"has one module named `layout.py` to form the final layout, concatenate the audio file and write the final video. "processing"is composed of four modules such as `preProcessing.py` , `postProcessing.py`, `slideDetection.py` and `speakerTracking.py` and different number of sub-packages like "CMT"that contains all modules for the corresponding algorithm. If you need to use a ready written package for video processing, easily put it in the folder "processing".

- preProcessing: it can be considered as the module to have all pre -processing functions such as segmentation, contrast enhancement and so on.
- slideDetection: This module implements various algorithms for slide detection. For each algorithm, one class has been defined in which there is a method named *slideDetector* to call. Different classes have different attributes, but in order to call the corresponding function *slideDetector*, all classes are the same. It includes the following classes.

  1. templateMatch
  2. connectedComponent
  3. polygonDetection
  4. harrisCornerDetection
  5. getUserCropping

- speakerTracking: This module implements various algorithms based on CMT for speaker tracking. This algorithm has been combined with Kalman filter, and different approaches have been used to ease the computation time. For every algorithm, one class has been defined in which there is a method named *speakerTracker* to call. Different classes have different attributes, but in order to call the corresponding function *speakerTracker*, all classes are the same. It includes the following classes.

  1. `CMT_algorithm`
  2. `CMT_algorithm_kalman_filter`

3. `CMT_algorithm_kalman_filter_stripe`
4. `CMT_algorithm_kalman_filter_downsample`
5. `CMT_algorithm_kalman_filter_no_vertical`
6. `CMT_algorithm_kalman_filter_downsample_no_vertical`
7. `FOV_specification`

- postProcessing: all functions for post-processing like *perspective_transformation*

In the next section, a short overview of main tasks will be discussed.

4.2 Slide Detection

4.3 Speaker Tracking

4.3.1 CMT

Speaker tracking is still an unsolved problem due to various type of changes, poses, appearances and local and global illumination changes. In our work, we use one of the most promising algorithms named "Consensus-based tracking and matching of the keypoints "or so called CMT. As a short introduction of CMT, one has to mention that it is a free-model object tracking in a combination of both matching and tracking framework. It does not need a specific knowledge and training samples.

In CMT, the object of interest (ex. the speaker) is selected in the first frame and some keypoints are extracted. It is possible to use different detectors, descriptors such as SIFT/SURF, ORB, Brisk, etc to obtain the keypoints. In the downloaded version of CMT, the Brisk detector and descriptor are used. After obtaining all of the desired keypoints corresponding to the object of interest, the main idea is to again find the kepoints which were already included in the initial selection. This progress is done by employing two different steps.

1. It tracks the keypoints from previous frame to the current frame by estimating optical flow.
2. They match the keypoints by comparing their descriptors.
3. In each frame, in order to localize the object of interest, a voting factor is used to vote and compute the center of the object.
4. Finally, a consensus scheme is applied to remove the outliers and keep the related keypoints.

Let's assume, we have a sequence of the frames (from I_1 to I_n) and the desire object of interest inside the first frame. As you can see in the table, it is called b_1. So, the output is to find the selected object inside all of the next frames (b_2 to b_n). Then, by means of Brisk detector, the keypoints will be computed as well as the corresponding descriptors. We assign all of the desired keypoints which we want to look for them through the whole frames into O set. It means, O includes all of the keypoints of the object of interest. After that, for all of the frames (from 2nd to n-th), the candidate keypoints are computed so that they can be inside or outside the region of interest. Hence, P is a set composing of all candidate keypoints. For each of the P's member, the hamming distance between their descriptor and the descriptor of desired keypoints are calculated and matched by meand of nearest-neighbor matcher. The results will be put into M. In order to track the keypoints, we compute the displacement of them from frame I_{t-1} and I_t by Lucas and Kanade method for estimating the optical flow. Afterwards, to discard all tracked keypoints which are matched as well, we fuse T and M keypoints and form the set of keypoints names K. Due to some failure in the tracking and matching progress, K can be included the outliers. Therefore, one voting factor is used to estimate the center of object and a consensus-based method is applied to eliminate the outliers. In this case, the votes are clustered based on euclidean distance and a threshold value and then the consensus voting will be identified based on the highest number of votes. The votes inside the consensus cluster are kept and the remaining will be prone out. As an output of CMT for each frame, the mean value of the center of the object calculated by voted keypoints are extracted. Then, this loop will be continued till the last frame [**?**].

1. Initialise detector, descriptor, matcher
2. Get initial keypoints in whole image (first frame)
3. In each frame, in order to localize the object of interest, a voting factor is used to vote and compute the center of the object.
4. Finally, a consensus scheme is applied to remove the outliers and keep the related keypoints.

- Getting user's object of interest: (function: `get_rect` in `util` module in `CMT` package)
  1. It reads the first frame and get the rectangle input from the user.
- Initialization: (function: `initialise` in `CMT` module in `CMT` package)
  1. Initialise detector, descriptor, matcher
  2. Get initial keypoints in whole image
  3. Remember keypoints that are in the rectangle as selected keypoints and those which are not in the rectangle as background keypoints
  4. Assign each keypoint a class starting from 1, background is 0
  5. Get all distances between selected keypoints by calculating square root and compute all angles between selected keypoints
  6. Compute the vector from the first keypoint (k1) to the 2nd keypoint (k2)
  7. Compute angle of this vector with respect to x axis and store it
  8. Find the center of selected keypoints as the mean of all selected keypoints
- Tacking and Matching: (function: `process_frame` in `CMT` module in `CMT` package), These steps are sequentially repeated for all frames
  1. Detect keypoints, compute descriptors and get all matches for selected features
  2. Track the keypoints between the previous and current frame by means of optical flow method. (function: `track` in `util` module in `CMT` package)
     (a) Calculate forward optical flow and backward optical flow for previous location
     (b) Calculate forward-backward error
     (c) Set status depending on the calculated error
  3. Estimate the parameters for matching and voting. (function: `estimate` in `CMT` module in `CMT` package)
     (a) Get all combinations of keypoints and measure distance between all
     (b) Estimate the votes regarding one formula and remember all votes including outliers
     (c) Compute pairwise distance between votes
     (d) Compute linkage between pairwise distances
     (e) Perform hierarchical distance-based clustering
     (f) Count votes for each cluster and get largest cluster
     (g) Identify members of largest class and compute the object center as a mean of all members
     (h) Remember outliers and stop to track them
  4. For each keypoint and its descriptor
     (a) Retrieve keypoint location
     (b) First: Match over whole image
     (c) Compute distances to all descriptors
     (d) Extract class of best match and add keypoint to active keypoints
     (e) Add all tracked keypoints that have not been matched
     (f) Update object state estimate

5 Acknowledgments

I would like to express my very great appreciation to all of team members, Michael Hirsch, Senya Polikovsky, Edgar Klenske, Michael Schober and Parnia Bahar for their patient, valuable and constructive suggestions during this research work.

References

[1] I. S. University, "elearners."

[2] G. Nebehay and R. Pflugfelder, "Consensus-based matching and tracking of keypoints for object tracking," in *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pp. 862–869, IEEE, 2014.