TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Internet Network Architectures

IP Multimedia Lab WS 2018/19
Homework 3
Due: 15.01.2019, 11.55 PM

You can upload your submission to the ISIS online course (`https://isis.tu-berlin.de/mod/folder/view.php?id=628432`). Please submit your results as an archive, including your code and any scripts that you wrote to solve the assignments. Prepare a PDF file, in which you write down your results and answers to the questions. For your submission, please consider the following points:

- Please name your archive *worksheet_⟨ws_number⟩_group_⟨group_number⟩*
- Add a cover page to the PDF including your group number and all group members
- Describe all relevant steps you performed
- Make your plots easily understandable, this is supported by
  - labeling your x-axis and y-axis
  - using informative captions
  - setting an appropriate font size
  - adding legends when needed
- Write down which libraries/tools you used
- Describe your results and denote to which (sub-)task they belong
- All code must be properly documented using inline comments
- All files that belong to a specific question must have the question/subquestion in their filenames
- When you are asked to visualize results, you are expected to provide a plot, i.e. visualizing using a table is not sufficient!
- When using ffmpeg, please shortly refer to all flags used, what they do, and why you use them (it is sufficient to do so once for a flag, i.e. when you apply a certain paramter a second time, you do not again need to describe it)

For this tasks, you need ffmpeg [1]. Please use, whenever you apply an ffmpeg command, the flag *-threads 1*. You will need the following videos:

- Big Buck Bunny clip: `https://service.inet.tu-berlin.de/owncloud/index.php/s/96yk7CGaactEpDs`
- Big Buck Bunny full movie: `https://service.inet.tu-berlin.de/owncloud/index.php/s/h6DvYK8tmBTBQCi`
- Sintel trailer: `https://service.inet.tu-berlin.de/owncloud/index.php/s/PR2cs2JLgiiNplW`

[1] https://www.ffmpeg.org/

Question 1: (30 Points) *Spatial and temporal information of videos*

In this task, you will analyze spatial and temporal information of videos. First of all, you will get in touch with the Sobel filter[2]. For the spatial perceptual information (SI), this filter is applied to a video frame. Then, the standard deviation of each pixel in the filtered image is computed. To compute the SI of a video scene, this procedure is repeated for all frames and the maximum is chosen as SI for the scene.

The temporal perceptual information (TI) takes the motion difference between consecutive frames into account. The TI value between two frames is the standard deviation of its difference. If you want to compute the TI for a complete scene, the TI is computed for all consecutive video frames and the maximum is chosen as TI for the scene. More information can be found here: `https://tinyurl.com/ya4nxmu2` (Subsections 5.3.1 and 5.3.2).

(a) Download the Sintel trailer and the Big Buck Bunny clip. Convert both videos from `.y4m` to `.yuv`. What is the reason for the slightly decreased file size after conversion?

(b) Choose any frame from the Sintel or Big Buck Bunny clip and apply the Sobel filter on its Y-component (scipy.ndimage might help).

(c) Display the original frame, the Y-channel of the frame, and the frame with applied Sobel filter. What do you notice?

(d) Write your own function for spatial perceptual information measurement, called *computeSI*. The input is, upon others, the raw .yuv video. It returns SI for all frames.

(e) Write your own function for temporal perceptual information measurement, called *computeTI*. The input is, upon others, the raw .yuv video. It returns TI for all frames.

(f) Visualize the SI and TI values for all frames of Sintel and Big Buck Bunny.

Question 2: (45 Points) *Preparing videos for adaptive streaming*

HTTP Adaptive Streaming (HAS) allows to dynamically adjust the video quality to current network conditions. To do so, the video is split into several segments (video chunks) of equal duration, typically between two and ten seconds. Each video chunk is made available in different resolutions/bitrates, so to obtain a set of different qualities. After each video segment download, the client decides via a heuristic about the next segment's quality. In this task, you are asked to evaluate the impact of different segment durations.

(a) Use the Big Buck Bunny full movie and split the video into chunks of the following durations: $seg\_dur = 0.5, 2, 4, 8, 12$ seconds. Thereby, consider VBR encoding, with the following quality settings: $crf = 18, 23, 28$. Segment the videos so to obtain files of the type **.ts** and a playlist of the type **.m3u8**. Write down the ffmpeg command and all necessary flags. Also explain why the flags you used are needed and what they do.

(b) Give some basic statistics on the resulting videos (a table is sufficient):

1. Number of video chunks
2. Total (fragmented) video size
3. Average segment size
4. Standard deviation of segment size

(c) Take a look on the playlist (you can easily open it with any text editor). Which information does it provide to you?

(d) Have a closer on the frames' characteristics. For this task, only consider the fragmented videos resulting from the encoding with $crf = 23$.

1. How many I-frames in total are required for the different segmentations, i.e. $seg\_dur = 0.5, 2, 4, 8, 12$?
2. How much do the I-frames contribute to the overall filesize in the different segmentations ?

(e) Have a look on the average SSIM of the videos. Does it notably change among the different segmentations?

[2] https://en.wikipedia.org/wiki/Sobel_operator

(f) Considering the results you obtained in this task, would you rather take longer or shorter video segments?

(g) In the context of adaptive video streaming, what are the benefits and drawbacks of long video segments and small video segments?

Question 3: (10 Points) *Introduction to video streaming*

Download the file *ConventionalController.py* (`https://service.inet.tu-berlin.de/owncloud/index.php/s/jrOHQulSwhxGtQs`) and the file *main_playlist.m3u8* (`https://service.inet.tu-berlin.de/owncloud/index.php/s/by3TGttGQxQ6xqd`).

(a) Have a look on the file called *ConventionalController.py*. It is an implementation of an HAS heuristic. Find out which information is taken into account when deciding about the next segment's quality. In this context, also describe the purpose of the following functions:

1. calcControlAction
2. quantizeRate
3. ewma_filter

(b) Now take a look on the file *main_playlist.m3u8*. It is typically located on the video server and downloaded by the video client. Which information does it provide to the video client?

Question 4: (50 Points) *Setting up the test environment and first measurements*

For this task, you need Vagrant[3]. Download the archive from the owncloud called *vm_INET_ML.tar.gz*: `https://service.inet.tu-berlin.de/owncloud/index.php/s/VYQA4Q74FuEH081` It contains a Vagrant file and a provisioning script which sets up a virtual machine for you, running Ubuntu and an Apache web server. The virtual machine has two namespaces[4], one for the client and one for the server. They are connected via virtual interfaces. Shaping the traffic on an interface allows us to control the available bandwidth between the server and client. Therefor, we use the following commands which can be found in the provisioning script:

```
sudo ip netns exec ns_srv0 tc qdisc add dev ns_srv0_veth0 root handle 1: htb default 1
sudo ip netns exec ns_srv0 tc class add dev ns_srv0_veth0 parent 1:
classid 1:1 htb rate 10000Kbit
```

The streaming client, which uses the python-based Tapas-Player[5], is located in the namespace *ns_clt0*, the server is located in the namespace *ns_srv0*.

Perform the following steps after installing Vagrant and downloading the *vm_INET_ML.tar.gz* archive.

- extract the archive *vm_INET_ML.tar.gz*
- dir into the *vm_INET_ML* folder
- run `vagrant up`
- wait until the VM is completely provisioned
- ssh to the provisioned VM: `ssh vagrant@127.0.0.1 -p <port>`
- the default password is `vagrant`
- you should now be able to see two folders: tapas and test-content
- type the following command.
  ```
  sudo ip netns exec ns_clt0 python tapas/play.py -m fake -u
  http://192.168.10.2/test-content/tos.m3u8
  ```
- if you see an output like this:
  ```
  2018-12-11 18:37:23+0000 [HTTPPageGetter,client] 154455344341
  <TapasPlayer-140093542411152> fetchNextSegment level:  0 (12.21 KB/s) 3/10 :
  http://192.168.10.2/test-content/1000/out_003.ts
  ```
  everything works fine!

[3]https://www.vagrantup.com/
[4]https://en.wikipedia.org/wiki/Linux_namespaces
[5]https://github.com/ldecicco/tapas

Feel free to adapt the provisioning script and the Vagrant file according to your needs! The command `play.py -m fake -u http://192.168.10.2/test-content/tos.m3u8` starts video streaming using the Tapas player. Thereby, `-u` specifies the playlist, `-m fake` starts the fake media engine, which means that the videos are not rendered on your machine and playback is emulated.

(a) Instead of using the provided test-content, add your own content! Consider a segment duration of 4 seconds and the crf values $crf = 18, 23, 28$ (you can use your content from previous tasks). The following steps will be necessary:

1. Create your own playlist, which indicates the available video representations (see question 2b and the file tos.m3u8 in the test-content folder).
2. Make the three video representations available at the server.
3. Re-provision your VM.
4. Once again run the following command, but specify your own playlist instead of *tos.m3u8*.
   `sudo ip netns exec ns_clt0 python tapasplay.py -m fake -u`
   `http://192.168.10.2/test-content/tos.m3u8`
5. Check if everything is working so far.

(b) Have a look on one of the log files, which are located here: */tapas/logs/conventional*. Which information can you gather from the log file? (Only consider the first eight columns).

(c) Do measurements with different settings of the available bandwidth, i.e. change `htb rate 10000Kbit` in the provisioning script. For this task, it is sufficient to perform only one measurement run per configuration (in practice, you would perform several runs for statistical significance!) As available bandwidth, use:

1. the double of the average bitrate of your video representation with crf=18
2. the average bitrate of your video representation with crf=18
3. the average bitrate of your video representation with crf=23
4. the average bitrate of your video representation with crf=28

(d) Visualize how buffered play time over time for the four bandwidth configurations.

(e) Visualize the quality level over time for the four bandwidth configurations.

(f) Visualize the average buffer level and the average quality for all bandwidth configurations.