פרויקט מסכם לקורס

נא לקרוא בעיון את **כל** הנהלים להגשת הפתרונות:

1. נא להגיש את הפתרון בהתאם להוראות ההגשה המפורטות בהמשך (בתיבת ההגשה ב-**Moodle**). תאריך ההגשה מעודכן בתיבת ההגשה.
2. הפרויקט כתוב באנגלית למניעת אי-הבנות בדרישות.
   אתם יכולים לענות בעברית או באנגלית כרצונכם.

3. אם קיימות שאלות בנוגע לפרויקט נא לרשום בפורום המיועד לפרויקט במודל.

4. התרגיל יוגש בזוגות בלבד בדומה להגשה של תרגילי הבית.

5. ניקוד יופחת על חוסר סדר, פתרון לא מתועד, קוד שלא רץ וכיו"ב בהתאם לשיקול דעת צוות הבדיקה.
6. אין להשתמש בפרויקט ב-Deep-learning או ברשתות לומדות לא עמוקות.
7. אין להשתמש בפונקציונאליות מובנית בספריות פייתונית אשר מייתרת פתרון אחד מהבלוקים העיקריים בפרויקט. בפרט, עקיבה, חיסור רקע, וייצוב וידאו.

בהצלחה☺ !

Introduction:

This project is about stabilization, background subtraction, matting and tracking. We will give you a shaky video of a person walking. You will implement several algorithms throughout this project in which eventually, you will output a stabilized video of that person walking, with a different background and rectangle around the person tracking it through the entire video.

From:

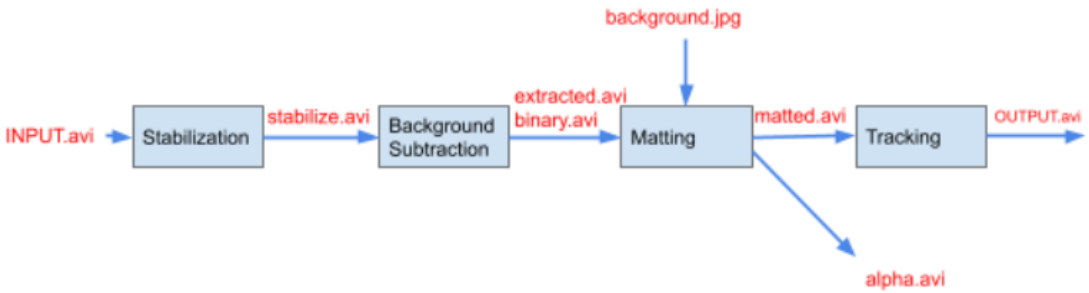
To:

At test time, we will test your code on two videos. The first is the input video given here. The second is a secret video. The second video contains one person walking on some background. The new background at test time is also held secret. To motivate this approach think of a scenario of application development. There is some train data you can access

while designing the app, but different (yet correlated) data is presented to you in production (test time).

Evaluation of the projects will be done with an automatic tester. Make sure your code and results are organized as explained in the following sections.

Generally, the flow diagram we will implement is the following:



Blocks you'll implement are shown in blue, the videos are in red. First we will stabilize the input video to get: **stabilize.avi**. Next, we will subtract the background from the stabilized video to achieve: **extracted.avi** and **binary.avi**. Then we take the new background image and perform matting to output a matted video: **matted.avi** along with an **alpha.avi** video. The final phase is Tracking. We will track the subject walking in the video and put a rectangle around it. We get an **OUTPUT.avi**.

Examples of frames from each one of the videos are shown in the table below (headlines first and the corresponding frames from the videos below them). Note that the frame examples below are just examples. Your solution can achieve better results.

The stabilized video (**stabilize.avi**) looks something like this: Note the black pixels on the top right of this frame. It's OK!	**INPUT.avi:**
	
a frame from the **extracted.avi** video:	The binary video (**binary.avi**) will look something like:
	
and an example from: **alpha.avi**	an example from the matted video: **matted.avi**
	
	OUTPUT.avi:

Code, environment and your solution:

We supplied a shaky video and a background image. The image will be the new background for your video. The background image can be of any size. You will output a video with frames with the same size as those of the input video. For that, you may change the background image size (there is no need to preserve the background image aspect-ratio).

The following outlines the general tasks you will have to implement. You may use your creativity in solving them as long as you don't use someone else's code. We've supplied a **requirements.txt** file for you (as done for the homework assignements). You can ask to use other libraries only in the **Moodle Forum**. We don't think other packages are necessary. But you can ask to use other libraries only in the **Moodle Forum**. When you ask for a package- tell explicitly how you're going to use it.

We will update the **requirements.txt** file when we approve new libraries. This update will occur once a week at most, without any new package requests in the last 3 weeks before the submission date.

Your code cannot expect user inputs, should not plot figures / videos to the screen. Your code may write prints to the command line. Your code should run on an Ubuntu 22.04 Machine with a conda environment installed using the **requirements.txt** and should end gracefully without any manual interactions/errors.

If you choose to use a function that is too specific (e.g. a function that includes the required steps to stabilize a given frame), add an explanation of its main functionality as well as its source code to your documentation (if your explanation won't be detailed enough you will lose points). Note that if the function makes the stage totally redundant points will be decreased.

If you implement a solution not studied in class, you *have* to add an explanation about the operation just so we know you understood what you did. No need to explain algorithms that we studied in this course (you can mention them without explaining), but if you made some changes to them, explain them.

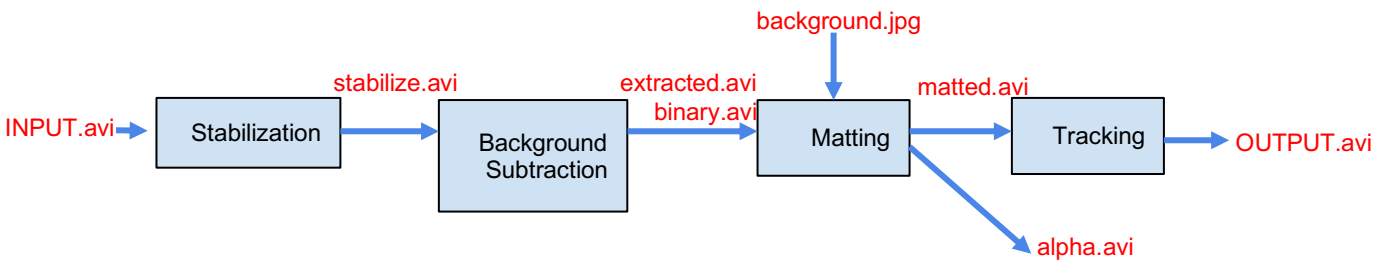
There is no one definite way to solve this project and you will be graded based on the quality and performance of your algorithm and the quality of your report.

Main steps:

ALL STEPS REQUIRE COLOR IMAGES PROCESSING AS INPUTS AND OUTPUTS – NO GRAYSCALE (the binary.avi video may be excluded from this demand)! WITHIN THE FUNCTIONS YOU MAY WORK ON ANY CHANNELS YOU WISH.

1. Stabilize the video in '**INPUT.avi**'. Save the stabilized video as '**stabilized_ID1_ID2.avi**'.
   a. Warning: Try not to damage the quality of the image (like blur) as it can cause you problems in the matting stage.
2. Extract the walking person from '**stabilized_ID1_ID2.avi**' using background subtraction to a video named '**extracted_ID1_ID2.avi**'.
   a. This video will only contain the walking person with a black background (0s background).
3. Save also a binary mask of the walking person as '**binary_ID1_ID2.avi**'. The person should be coded with 1's and not-person pixels as zero.
4. Using image matting, place the walking person from '**extracted_ID1_ID2.avi**' in '**background.jpg**'. Save this video as '**matted_ID1_ID2.avi**'.
   a. The new video file will have a walking person with a static background.
5. You will also save a video named '**alpha_ID1_ID2.avi**' which is the alpha (you will calculate) of each frame in the matted video (values in range of [0,1] where 0 means background and 1 means foreground and different values for pixels near the boundaries).
6. Automatically select the walking person in '**matted_ID1_ID2.avi**' and track it throughout the video showing a rectangle around that person in all frames. Save the video with the rectangle highlight as '**OUTPUT_ID1_ID2.avi**'.

Flow diagram:



How will we run your code and what do we expect to find?

First, as in the homework assignments, we will set up a virtual environment and install the packages in the **requirements.txt** file attached to the project.

We will run your code from a single python file: **main.py.** Specifically, we will run the following command in our shell: **python Code/main.py**. This, call should process **INPUT.avi** and **background.jpg** produce the following videos:

- stabilize_ID1_ID2.avi
- extracted_ID1_ID2.avi
- binary_ID1_ID2.avi
- alpha_ID1_ID2.avi
- matted_ID1_ID2.avi
- OUTPUT_ID1_ID2.avi

where: ID1, and ID2 will be replaced by the students IDs. Make sure there are no white spaces in the videos names.

All videos created by this script will be saved in a directory named **Outputs.** You need to submit the project with the result videos you calculated on your computer in this folder.

Your code cannot expect user inputs, should not plot figures / videos to the screen. Your code may write prints to the command line.

You should create a json called **timing.json** in which there is a single dictionary dumped into it. The dictionary have the following structure:

```
{
    "time_to_stabilize": 1,
    "time_to_binary": 2,
    "time_to_alpha": 3,
    "time_to_matted": 4,
    "time_to_output": 5
}
```

Instead of the values in the screenshot, you will put the time passed **in seconds** from the start of the main.py script to the time the corresponding video was created. The time a video is created is when you finish filling it up with all its frames.

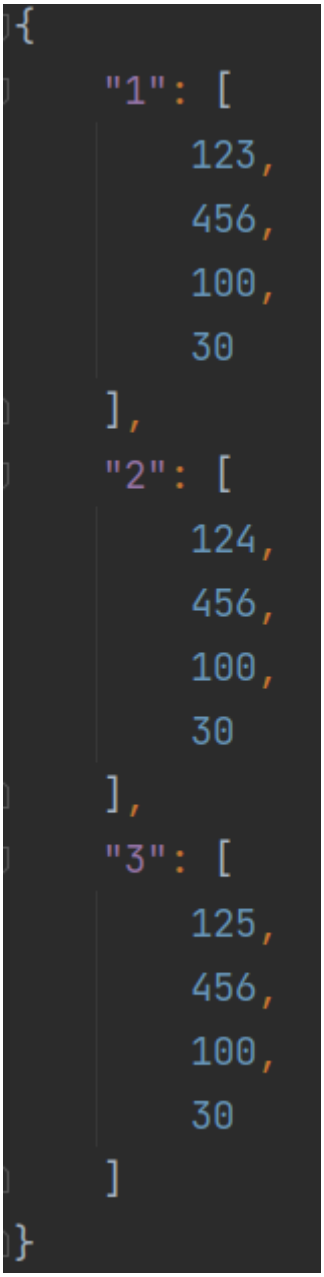
You can test your json with the following code:

```python
import json


d = {
    "time_to_stabilize": 1,
     "time_to_binary": 2,
     "time_to_alpha": 3,
     "time_to_matted": 4,
     "time_to_output": 5,
}

d_test = json.load(open('timing.json', 'r'))
for k in d:
    if k not in d_test:
        assert False, f"Your JSON does not include: {k}"
```

You also have to create a tracking results json called **tracking.json**. This json will contain a single dictionary. This dictionary will map integers to a list. The integer keys correspond to frame numbers. The list will contain four integer numbers corresponding to: [ROW, COL, HEIGHT, WIDTH]. See example in the next page:



Surely, the **INPUT.avi** video contains more than three frames. The length of the dictionary in this json file is the number of frames in INPUT.avi (which is the same number of frames as in **OUTPUT_ID1_ID2.avi**).

Both jsons will be created in the **Outputs** directory.

Project Benchmark and Grading:

There is a splendid result, one can obtain by running this paper's code on our video. Such a result is something which will be hard to reproduce with classical algorithms. We stress that it is forbidden to use Deep Learning methods.

Though it is not a perfect result, it will serve as a reference result for the background subtraction phase. We will hold secret how specifically we convert the video to something that this repo can manage. But we encourage you to be creative and think about ways to benchmark yourselves.

As for the tracking step, we will check the deviation of your result from the secret tracking results we hold.

The total runtime should not be more than 20 minutes on Tochna Computers (floor -1). After 20 minutes we will abort your project run.

The grade will be based on (not only) your runtime and quality of results. We keep the quality metric and the combination between runtime and quality secret. Your grade will also be affected by your report (PDF).

We may run your code on a different video of the same person walking (in different clothes). We keep a secret if the background is the same or not.

A way out:

If for any reason, you prefer to have a way out of the project, one can implement a solution which is based on a python package named **opencv-contrib**. Most of the steps in the project will take no more than ±10 lines of code using this package. Choosing this option will give you a maximal grade of 70 points for the project.

What you submit:

1. A pdf document containing a detailed explanation of your solution for each step. Do not underestimate the presentation; it is our way to understand your solution.
2. Complete Python files (containing your implementation) + input video file + new background image + output video files (6) + 2 json files.
3. Screen recording (video) of running your code (from start to finish, until the output is created). You can compress the video by sending it from one project submitter to another by Whatsapp. Add the compressed video to the submission in the **ScreenRec** folder you submit.
4. Use all the files in relative folders so the project runs perfectly using **python Code/main.py** as explained above.
   Do not use any direct path folders in your code ('cd c:\video_project\' etc. is unacceptable). Use relative paths. Use python's os library to define paths and relative paths.
5. The root folder - FinalProject_ID1_ID2.zip (which would be submitted via the Moodle submission box) will contain folders (in **bold**), each containing the files[1] written in the brackets:
   - **Document** (containing a PDF document of your report)
   - **CODE** (containing **main.py** + all Python files)
   - **Input** (containing INPUT.avi + background.jpg)
   - **Outputs** (containing stabilize_ID1_ID2.avi, extracted_ID1_ID2.avi, binary_ID1_ID2.avi, alpha_ID1_ID2.avi, matted_ID1_ID2.avi, OUTPUT_ID1_ID2.avi + timing.json + tracking.json)
     - ID1, and ID2 should be replaced with your IDs.
     - **Outputs** folder should not contain more videos / json files than what's specified here.
   - **ScreenRec** (containing the screen recording of you running your **main.py** creating the output videos and jsons).
   - **Temp** (If needed to save temporary artifacts – you can choose if you want to submit these files).

Zip the above folders and upload to Moodle.

Note that running the main function (main.py) should create all required outputs.

[1] Note that the case sensitivity here is important, and that no spaces are needed in any file or folder.

Frequently Asked Questions:

1. Stabilizing the video and resizing each frame causes black regions along image edges - is this OK? **Yes.**
2. What values are expected to be in the extracted.avi video? **The result video should extract the person from the stabilized video (this means that the person will have the same values as in the input video) and 0's everywhere else.**
3. What is the color code in the binary video? **0 = no person; 1 = person.**
4. What is the value range we allow for alpha.avi? **[0, 1].**
5. What happens if we cannot save a video with pixels in range [0, 1]? **Convert the range to [0, 255] by multiplying values by 255 and then quantize these values by converting them to np.unit8 save and submit the scaled & quantized version.**
6. Will there always be a single person to track in input videos for this project? **Yes.**
7. What are the inputs to the matting block? **background image, stabilize.avi and binary.avi.**
8. Can we analyze color spaces in the video in an offline manner and put them as constants in our code? **Yes, but note that overfitting too much might affect your grade when your code is presented with another (secret) video. Note that the subject of the second video is the same but we tell you that he changed clothes.**
9. Can we use scribbles in the first frame? **Yes, you can put scribbles. Use them as constants in your code. Note that these scribbles might be incorrect for a new video input.**
10. Can we use opencv's createBackgroundSubtractorKNN / createBackgroundSubtractorMOG2? **Yes. You can use every function of opencv. But, you need to explain how they work in your report PDF.**