Image Processing – 67829 – Exercise 1

Due Date: 18.01 at 23:59

Version 1.0 - Last Update 03.01.2024

1 Task

In this exercise, you will need to detect scene cuts using the material taught in class this week (i.e. histograms). In the Moodel, under "Exersice 1" -> "Exercise Inputs" (found here) you are given 4 videos, each of the videos contains footage from two different scenes. Your task is to detect the frames for which a change of scene has taken place. The 4 videos are divided into 2 categories (named *category*1 and *category*2), each category may require a slightly different solution.

Toy Example: In Fig. 1 is a toy example of a scene cut, the cut takes place between frame 3 and 4.

frame 0 frame 1 frame 2 frame 3 frame 4 frame 5

Figure 1

Note1: When trying to detect scene cuts, you should transform the frames into grayscale ones and use the methods as they were taught in class (i.e. on grayscale images).

Note2: Checkout the coding bootcamp on the Moodle (or here) for a quick introduction to some libraries and methods that can help you throughout the exercise.

Note3: We recommend you start with solving the task for category 1 and only after having a working algorithm moving to category 2. Some of you may solve category 1 with an algorithm that also solves category 2. This is OK, however, if this is your case, you are still expected to discuss in your report the differences between the two categories and explain why your solution works for both (see Sec. 2.1 for more

details). This does not mean you need to create an additional algorithm to solve category 2, instead try to understand why you solution was able to solve both categories and what change in the algorithm will make it such that it only solves category 1.

2 Submission

Submission instructions are given in the "Submissions Guidelines" document published on the course web page (here). Please read and follow them carefully. Any updates to those guidelines will be posted in the news forum, so be sure you follow the forum.

Your code should follow the below API (you are welcome to extend it however much you like, but you must follow the API for the entry and exit points for your code). In the example from Fig 1 the code should return $(3, 4)$. A skeleton file can be found in the Moodle (here).

```
def main(video_path, video_type):
    """
    Main entry point for ex1
    :param video_path: path to video file
    :param video_type: category of the video (either 1 or 2)
    :return: a tuple of integers representing the frame number
                for which the scene cut was detected (i.e. the last frame
                index of the first  scene and the first frame index of the second scene)
    """
    pass
```

When submitting your solution to the Moodle, a **basic** presubmission script will check that your code executes and finishes with no errors. Additionally, the presubmission will run your code on an additional, hidden video and will indicate whether your code returned the correct solution for the video (you will not be able to see the video or the correct solution).

In addition, we ask that you submit a "requirements.txt" file with all the pip dependencies you've used in your solution. The presubmission script will create a pip environment with the dependencies listed in your requirements.txt file.

Note: A final submission that fails to pass the presubmission test will start its grade from 80**, we will not accept appeals related to failing the presubmission test.**

2.1 Report Guidelines

In addition to the code, you should submit a report describing your solution. The report must follow the following structure and address the topics below. We provide an English and Hebrew Google Docs template (you need to copy it to use it). In case you choose not to use it, please maintain a similar structure (font size, same sections, same number of figures, same number of pages, etc.), in particular, the report should be no longer than 4 pages and include the following sections and topics:

1. Introduction
   (a) In your own words, state the goal of the exercise and what was the main technique (i.e. an idea or concept you've learned in class, not a technical tool like numpy) you've used to solve it.
   (b) Briefly specify the differences between the two categories of videos (category1 and category2) and how these differences may affect the approach.
2. Algorithm
   (a) Clearly describe the steps involved in your scene cut detection algorithm (i.e. describe the conceptual steps and building blocks, e.g., if the algorithm is how to make coffee, the steps could be 1) Heat water, 2) Put coffee in glass, 3) Pour water, 4) Add sugar, 5) Add milk).
   (b) Clearly describe any modifications or adjustments made between the first video category and the second one.
3. Implementation Details
   (a) Describe your implementation of the algorithm (i.e. describe the actual implementation of the steps from section 2.a, using the same coffee example, here you would describe how much coffee, how much sugar, water to milk ratio, etc.).
   (b) Specify the parts that you implemented from scratch and those that you've used functionality from an existing library. What libraries did you use and why did you choose these?
   (c) Describe any necessary hyper-parameters, thresholds, or other choices used in your algorithm.
   (d) Discuss any challenges faced during implementation and how they were addressed.
4. Category 1 Results
   (a) Present the results of scene cut detection for each video.
   (b) Include figures and visualizations that support the choice for determining where a scene cut has occurred.
5. Category 2 Results
   (a) Present the results of scene cut detection for each video.

(b) Explain how are the videos in category 2 different from those in category 1 and what made you notice these differences.

(c) Include figures and visualizations that support the choice for determining where a scene cut has occurred.

(d) Include figures and visualizations that demonstrate why the algorithm used for category 1 didn't work for category 2.

6. Conclusion

(a) Summarize your key findings and insights.

Your final submission should be a tar file containing a PDF named "ex1.pdf", a python file named "ex1.py", and a requirements.txt file with your dependencies. To create a tar file you can run the following command:

```
tar -cvf ex1.tar ex1.py ex1.pdf requirements.txt
```

3 Grading

Your exercise will be graded based on a manual inspection of your report (and code) and a short automatic test that will run your code and check for the correct output. The grade breakdown is 80% for the report and 20% for the automatic test. As mentioned above, a final submission that fails to pass the presubmission test will start its grade from 80, we will not accept appeals related to failing the presubmission test.

Good luck and enjoy!