CS 490/548: Assignment 01

Programming Assignments (95%)

Your program MUST be written in **Python**.

Your code file should be named **A01.py**. The goal of this assignment is to open a video file, play it, and resave it as individual frames.

Graduate students will also have to be able to change the frame rate of the video.

It will also generally verify that your environment is ready.

A01.py should contain the following functions:

- **def load_video_as_frames(video_filepath)**
  - Use cv2.VideoCapture to load a video from video_filepath
  - If the video is not opened, print an error and return None
  - Loop through the video ONCE and add each frame to a list
  - Return list of frames
- **def compute_wait(fps):**
  - Compute the wait in milliseconds as int(1000.0/fps)
  - Return computed delay
- **def display_frames(all_frames, title, fps=30)**
  - Call compute_wait(fps) to get the wait time.
  - Loop through the frames ONCE and display using cv2.imshow(title, frame) and use cv2.waitKey to wait the appropriate amount of time.
  - Destroy all windows
- **def save_frames(all_frames, output_dir, basename, fps=30)**
  - Make the video folder name as basename + "_" + str(fps)
  - Make the full output path using os.path.join(output_dir, video_folder)
  - If the path already exists (os.path.exists()), remove it with shutil.rmtree()
  - Use os.makedirs() to (re)create the output path
  - For each frame
    - Use the pattern "image_%07d.png" % index to get the zero-padded filename for each image
    - Get the full path using os.path.join(output_path, filename)
    - Save the frame using cv2.imwrite

- ***CS 548 ONLY:***
    - **def change_fps(input_frames, old_fps, new_fps)**
        - Get the total number of frames (old_frame_cnt)
        - Compute the new frame count as int(new_fps*old_frame_cnt/old_fps)
        - For each new OUTPUT frame
            - Compute the index into the old array as int(old_fps*i/new_fps)
            - Append that frame to your list of output frames
        - Return the list of output frames
- **def main():**
    - If the number of command line arguments (len(sys.argv)) is less than 3, print an error and exit(1)
    - Read the input video path from sys.argv[1]
    - Read the output directory from sys.argv[2]
    - Get the core filename from the video path using Path(video_filepath).stem
    - ***CS 548 only:***
        - If the number of command line arguments is GREATER than 3, grab int(sys.argv[3]) as the new fps; otherwise, default to a new fps of 30.
    - Load all frames; if the function returns None, print an error and exit(1)
    - Display all frames with title "Input Video" and fps of 30
    - ***CS 548 only:***
        - Compute output frames with new fps
        - Display output frames with title "Output Video" and DISPLAY fps of 30
    - Save the (output) frames to the output folder
        - ***CS 548 only:*** Use OUTPUT fps

In addition, have the customary main function call on the bottom of your program:

if __name__ == "__main__": main()

Testing Screenshot (5%)

I have provided several files for testing:

- Test_A01.py – the test program for CS 490
- Test_A01_Grad.py – the test program for CS 548
- General_Testing.py – basic testing functionality
- assign01/
    - input/
        - Input video for testing
    - ground/
        - Ground-truth data

Copy these files/folders into the SAME directory as your python program.

You can either run the testing programs directly OR you can use the testing section of Visual Code.

You MUST run the tests and send a screenshot of the test results! Even if your program(s) do not pass all the tests, you MUST send this screenshot!

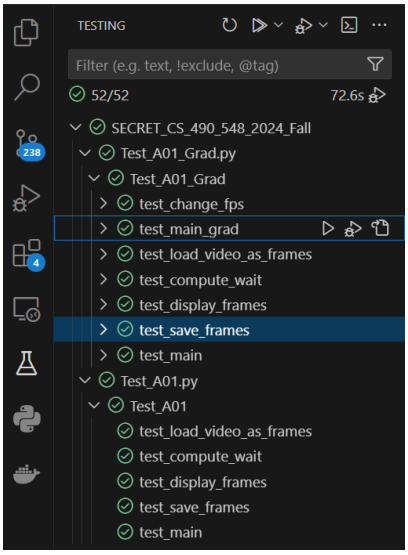
Graduate students (CS 548) MUST run Test_A01_Grad! (Please note also that you do need the Test_A01 file as well, since Test_A01_Grad relies on it.)

These tests do not DIRECTLY test the timing on display_frames(), so you (and I) will have to check that manually.

You may have to do "Command Palette" → "Python: Configure Tests" → unitttest → root directory → test_*py

This screenshot should show clearly:

- The final result of the test run on the command line ("OK" for all passing, "FAILED (failures=N)" for some or all failing).
- OR
- The testing view in Visual Code (see image on right)



Grading

Your OVERALL assignment grade is weighted as follows:

- 5% - Testing results screenshot
- 95% - Programming assignments

I reserve the right to take points off for not meeting the specifications in this assignment description. In general, these are things that will be penalized:

- **Code that is not syntactically correct (up to 60 points off!)**
- Sloppy or poor coding style
- Bad coding design principles
- Code that crashes, does not run, or takes a VERY long time to complete
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT
- Sharing code with other people in this class or using code from this or any other related class
- Output that is incorrect
- Algorithms/implementations that are incorrect
- Submitting improper files
- Failing to submit ALL required files