2D Computer Graphics and Image Processing
Assignment 2: Drawing with OpenCV

Due: October 9th, 2018

Abstract

This assignment is an extension from assignment 1. We will modify the `print_info` function we implemented to also draw the parsed text on screen. Unfortunately, the submission for assignment 1 is still open, thus the solution for the assignment 1 will be released only a week after the initial announcement.

Instructions are provided first for a reason.
Please read the instructions carefully.

1 Instructions

1.1 Submission package

Within the homework assignment package, there should be a `submission-package.zip`, which contains the directory structure and empty files for you to get started. Please edit the contents within the file for code, and then create a `zip` archive with the file name `submission-package.zip`, and submit it. **Do not use other archive formats such as `rar` or `tar.gz`.**

All assignments should be submitted electronically. Hand written reports are **not** accepted. You can, however, include scanned pages in your report. For example, if you are not comfortable with writing equations, you can include a scanned copy.

1.2 Code

All assignments should be in `Python` 3. Codes that fail to run on `Python` 3 will receive 20% deduction on the final score. In other words, do **not** use `Python` 2.7.

For this assignment, you should **not** need to create additional files. Fill in the skeleton files in the submission package. Do **not** change the name of these scripts. We will run, for example,

```
python solution.py
```

to test if your code runs as the assignment specifications.

It is **strongly encouraged** to follow `PEP8`. It makes your code much more readable, and less room for mistakes. There are many open source tools available to automatically do this for you.

1.3 Delayed submission

In case you think you will not meet the deadline due to network speed or any other reasons, you can send an email with the `SHA-256` hash of your `.zip` archive first, and then submit your assignment through email later on. This will **not** be considered as a delay.

Delayed submissions are subject to 20% degradation per day. For example, an assignment submitted 1 minute after the deadline will receive 80% of the entire mark, even if it was perfect. Likewise, an assignment that was submitted one day and 1 minute after the deadline will receive 60%.

1.4 Use of open source code

Any library under any type of open source license is allowed for use, given full attribution. This attribution should include the name of the original author, the source from which the code was obtained, and indicate terms of the license. Note that using copyrighted material without an appropriate license is not permitted. Short snippets of code on public websites such as StackOverflow may be used without an explicit license, but proper attribution should be given even in such case. This means that if you embed a snippet into your own code, you should properly cite it through the comments, and also embed the full citation in a LICENSES file. However, if you include a full, unmodified source, which already contains the license within the source file, this is unnecessary. Please note that without proper attribution, *it will be considered plagiarism.*

In addition, as the assignments are intended for you to learn, (1) if the external code implements the core objective of the task, no points will be given; (2) code from other CSC205 students will count as plagiarism.

2 Implement **`parse_and_draw`** (5 marks)

Modify your implementation for `print_info` in assignment 1 to also draw the object you parsed. We did not have a field for colour, thus we will simply use black (0.0) to draw. The canvas we create will be white (1.0). Note that, unlike in the previous `print_info` case, our function now returns the updated canvas.

3 Display figure (2 marks)

With the final updated canvas from `parse_and_draw`, use `cv2.imshow` to show the drawing result in a window named canvas. Note that `input.txt` for assignment 2 is different from assignment 1. Also use `cv2.waitKey` to wait for a keyboard input before closing.

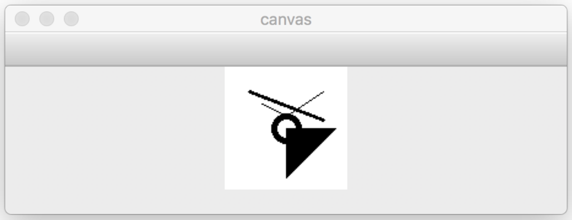

Figure 1: Example output.