Image Processing CSE2225

Delft University of Technology

November 2, 2020

The Image Processing course this year will provide project files along with a docker container (similar to a virtual machine, but more efficient). You can learn more about docker going through its extensive documentation. The setup of the docker container for MacOS, Linux or Windows is accomplished by running the corresponding shell/batch script provided:

- For Linux: **runDockerLinux.sh**
- For Windows: **runDockerWindows.bat**
- For MacOS: **runDockerMac.sh**

Once the right script is executed, the bash/command prompt should change to that of a root Linux (Ubuntu 18.04) user.

The project files are all mounted on the container, so you will find them in the 'imageprocessingcourse' folder (when the container runs, you should be able to see this directory with the `ls` command). Any changes made to the files inside the container will reflect also in your host machine. The docker container does **not** copy your files, there is only one copy that you work on shared among your native OS and the container.

Note: You are *NOT* allowed to move data, or scripts from this folder. Please develop all your code within this directory. You are allowed to add new scripts to this folder that you need to evaluate intermediate components in your code and solve the task. You are *NOT* allowed to change the scripts: 'evaluator.sh' and 'evaluation.py' (they have to be submitted identical to what was provided).

1 Setting up on Linux/MacOS

1. Install docker for Linux/MacOS (Installation page)
2. X11 forwarding is required. For Linux the 'xauth','xorg' and 'openbox' packages are required. Install them with the package manager for your Linux distribution. For instance on Ubuntu, the following command will be used - `sudo apt-get install xauth xorg openbox`

   For MacOS XQuartz can be downloaded from this link.
3. Run docker and X11/Xquartz. Check the option 'Allow connections from network clients' in the GUI of X11 application or you can also use the command `xhost +` from the terminal.
4. `cd` into the project files directory and run the appropriate shell script (based on your OS) - `./runDockerMac.sh` or `./runDockerLinux.sh`

1.1 Troubleshooting

- Docker daemon, docker container display related errors : Ensure that docker and x11 are running. Post installation of the applications, a reboot might be required. You can verify if the display works inside the container, first by running `cd imageprocessingcourse` and then `python show_plates.py`.
- If you still have display errors despite a properly running x11 and docker then you're running on an older MacOS or Linux version. You might have to edit the display forwarding to manually include your device IP. Get your IP with the command `ifconfig en0 | grep inet | awk '$1=="inet" print $2'`. In the 'runDockerMac.sh' or 'runDockerLinux.sh' change the -e flag input to `-e DISPLAY=YOUR_IP_HERE:0`

2 Setting up on Windows

1. Install docker for Linux/MacOS (Installation page)
2. Again, X11 forwarding is required. It is recommended to use VcXsrv. The easiest way to install it would be to use a package manager like chocolatey. Then you can use the command `choco install vcxsrv`.

   Note: Installing might require running the cmd.exe as an administrator.
3. Run Docker and Xlaunch from the start menu. Xlaunch should open a GUI, in the first window (display settings) change the display number to 0 and then click next. No changes needed in second window (client startup), so just hit next. In the third window (extra settings), remember to check the 'disable access control' box. You can save these configurations so that you don't need to setup for every launch.
4. `cd` into the project files directory and run the appropriate batch script - `runDockerWindows.bat`

2.1 Troubleshooting

- If you use Windows via bootcamp on a Macbook, then there is an unresolved virtualization issue. The workwround for this situation is to boot into your MacOS and 'soft' boot into your Windows. You can achieve this by going to System Preferences $->$ Startup disk and then choose the bootcamp partition.

3 Information about scripts and deliverables

- **evaluator.sh** : This script runs on startup of the docker container. It executes your pipeline by running the 'main.py' and also compares your output with the ground truth. It is important to note that the evaluation is set up in such a way that your final prediction must be in a csv file with the name containing the string "Output" in it.
- **runJupyterNotebookServer.sh** : A simple script to run jupyter notebooks.
- **show_plates.py** To verify that the display forwarding works inside the docker container.
- **requirements.txt** A list of libraries to install in the docker container. If you use a library not already on the list, please add it to the list in 'requirements.txt' and also highlight the usage of the new library in the report.

When submitting your code, you should make sure that you have added all new python packages needed to 'requirements.txt' and that your code correctly writes the predictions to a csv file containing the word "Output" in the name. You will need to upload a zip archive of your 'Docker_ImageProcessing' scripts to BrighSpace.

You can test your pipeline on a subset of the provided training videos or on self-recorded videos that you can rename as 'test' and for which you have defined the ground truth labels.

Note: When evaluating your code, we will extract the zip archive you submitted on BrighsSpace. We will upload in the 'imageprocessingcourse' directory a test video and its corresponding grount truth labels. We will run the 'evaluator.sh' script to evaluate your code.

This 'evaluator.sh' script loads the test video (containing in the name the word "test") and runs your code over this test video by calling the 'main.py' function, which should generate an output csv file (containing the word "Output" in it). This output csv file is then evaluated using the 'evaluation.py' script and the test ground truth (containing in its name the string "TruthTest"). The final accuracy we obtain represents your 'test-accuracy' grade.

IMPORTANT: Please test your code and make sure your code runs without problems. If we cannot run your code, you receive 0 points for the 'test-accuracy' grade.