Image-Based GPS Verification README

This project utilises different AI and Computer Vision methods

to solve the long-pressing issue of Image Verification. The approach

taken is to compare a query and a reference image taken from some coordinates

and extract a similarity value. Additionally, verification is performed by comparing

that value against a threshold.

- The best threshold is picked by running the **thresholdTool.py**

- The image verification is performed by running the **verificationTool.py**

Installation

Python 3 is required in conjunction with **'pip'**.

Python 3 can be downloaded from **https://www.python.org/downloads/**.

The version used in this project is 3.6.8.

The guide to installing **pip** can be found on the following website - **https://pip.pypa.io/en/stable/installing/**.

Moreover, the **pip** file is provided in the main directory of the project. It is called **get-pip.py**.

The command to install that file is as follows:

- **python get-pip.py**

For more installation options, the provided guide above can be referenced.

To install all libraries run - **pip install -r requirements.txt**.

This will get all the requirements needed and install them recursively.

Additionally, the requirements can be installed manually. An issue could occur on some systems after installing the **Tensorflow** library, especially if it was already present on the machine. The error would say that the **gast** library is missing, although it could be installed. That is because different versions are not compatible together. The fix is rather simple. The **gast** library must be removed and then installed again:

- **pip uninstall gast**

- **pip install gast**

Note: For some systems, administrator privileges might be required.

verificationTool User Guide

The model implements multiple arguments and commands that can be invoked by running them in the terminal. The commands shown here are for the test data provided in the 'Test' folder. However, these commands can be used on custom data as well. Additionally, there could be multiple configurations of the available parameters and they can be used in conjunction with other parameters. All the following commands have been run from the current directory.

Predict

Template Matching Prediction – python verificationTool.py -p ./Test/images/image1Q.jpg ./Test/images/image1R.jpg -mm tm

Patch Matching Prediction - python verificationTool.py -p ./Test/images/image2Q.jpg ./Test/images/image2R.jpg -mm pm

To print the likelihood maps, add the **-pm** command to either of the commands above. A figure should appear on the screen if and only if the prediction is positive, otherwise it would not print anything!

To change the threshold used, add the **-thr someNumber** command to either of the commands above!

Test

Template Matching Test - python verificationTool.py -test "path to query images" "path to reference images" "path to labels.txt file" -mm tm

Patch Matching Test - python verificationTool.py -test "path to query images" "path to reference images" "path to labels.txt file" -mm pm

Again, the user can change the threshold by adding the **-thr someNumber** command to either of the commands above!

Extract Similarity Values to File and Plot

Extract Values - python verificationTool.py -e -ed "path to query images" "path to reference images" "path to labels" -ep "path to the new text file"

The **-e** command invokes the extraction mode, **-ed** command provides the query and reference images, as well as the labels text file, and **-ep** provides the path where the new file will be saved to.

Plot Values - python verificationTool.py --plot -ep "path to a text file with similarity values"

The **--plot** command is used with the **-ep** command. Here the **-ep** command provides the text file and it is not used to create a new one. The result of plotting is shown in the paper of this project and should produce a graph of the values.

SURF

To use the SURF method, simply invoke the **-s** command after either any of the prediction or testing commands. It would invoke the SURF method that is also described and referenced in the paper of this project.

Note: It does not use any of the methods invoked from the -mm command. Thus, it must be omitted.

python verificationTool.py -test "path to query images" "path to reference images" "path to labels.txt file" -s

Additional Information

The labels text file when extracting or plotting should have the following layout:

SimilarityValue Label

0.50 1

0.30 0

The labels text file when testing should have the following layout

Query,Reference,Label

queryImage.jpg,referenceImage.jpg,0

Note: The full paths are received by the other two arguments when invoking the -test command (query images path, reference images path).

thresholdTool User Guide

This is the script that iterates over multiple thresholds and picks the best one for the data provided. It requires two files. One with negative sample similarity values and one with positive ones. Additionally, the **-r** command can be invoked to reverse the order of importance. Meaning, lower values will be better. This mode can be invoked depending on the similarity measure used for extracting the similarity values.

Get Best Threshold Default – python thresholdTool.py -a "path to negative text file" "path to positive text file"

An extra argument can be added that specifies the location of where the produced plot graph will be saved. If not specified, it will be saved to a **default.jpg** image!

Get Best Threshold Reverse Order - python thresholdTool.py -a "path to negative text file" "path to positive text file" "path to a new jpg file" -r

Additional Information

Both files must be passed through in that exact order. The negative file comes before the positive one, as shown in the examples above. Both files should have the following layout:

SimilarityValue

0.50

0.60

To extract those values, there is a commented-out method in the **verificationTool.py**, that saves two files given the values predicted by the model. It can be found in the **test** function.

List of Used Libraries

Keras - 2.3.1

Tensorflow - 2.0.0

NumPy - 1.16.0

OpenCV - 3.4.2.16

Sklearn - 0.22.1

Argparse - 1.1

Matplotlib - 3.1.0

Time - built into python's interpreter (python 3.6.8)

Warnings - built into python's interpreter (python 3.6.8)

OS - built into python's interpreter (python 3.6.8)