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**

Note: Every instruction described in this file is also present in the User Guide Appendix of the original paper. The structure is different between both, so the best one for the user individual can be chosen to be read. Additionally, the amount of information present might vary slightly but it should not affect the successful installation or usage of the model.

Installation

Python 3 is required for this project. It can be downloaded from https://www.python.org/downloads/. The version used in this project is 3.6.8.

If **Python 2** is already present on the machine and set to default, then all commands might need to be run with **python3** and not just by **python**.

To install the project, the following steps can be followed:

1. **Python 3 virtual environment** - creating a virtual environment can help preserve the system from installing and uninstalling different requirements, as well as it will provide a clean installation process. To create that environment, the following command can be executed in a terminal:
    - **python3 -m venv verification-env**

    The name of the environment is specified after the "venv" command. In this case it is "verification-env". The environment will create a directory at the current location that contains a copy of the python interpreter. After successfully creating the environment, it needs to be activated.

    This can be done with the following commands

    - **Windows** - verification-env\Scripts\activate.bat
    - **Unix or MacOS** - source verification-env/bin/activate

    After running the command, according to the system, the environment should show in the terminal. Now, the requirements for the project must be installed.

2. **Installing requirements** - the requirements can be installed with the help of pip, or more specifically pip3.

If pip is not already installed with python 3, which it should be by default, the original guide can be followed.

It can be found on https://pip.pypa.io/en/stable/installing/ . The pip setup file must be downloaded and then executed with python.

After it is made sure that pip is installed on the system, the following command can be run to install the requirements necessary in the directory of the project:

- **pip install -r requirements.txt**

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 was installed. That is because different versions of Tensorflow and Gast 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. Additionally, the problems described above, should not occur if installed in a virtual environment.

*The code can be run in Google Colab as well. This can be done by simply providing any of the files to Colab and running the commands provided below.

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 -thr 0.679

Patch Matching Prediction - python verificationTool.py -p ./Test/images/image2Q.jpg ./Test/images/image2R.jpg -mm pm -thr 0.652

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

Different CNNs – The current model supports the use of different CNNs for feature extraction. The command is (**--model)** followed by the choice of CNN. The choices are – **['resnet50', 'resnet101', 'resnet152', 'vgg19', 'inception']**. This command can work in conjunction with the (**-mm)** one. An example would be - **python verificationTool.py -p ./Test/images/image2Q.jpg ./Test/images/image2R.jpg -mm pm --model resnet152**

Different Similarity Measures – The current model supports different similarity measures as well. The command to do that is (**--measure)**. And the choices are – **['cc', 'ncc']**. 'CC' is **Correlation Coefficient** and 'NCC' is **Normalized Cross-Correlation**. It is recommended that the paper is being followed when picking threshold or any other parameter. An example of the command would be - **python verificationTool.py -p ./Test/images/image1Q.jpg ./Test/images/image1R.jpg -mm tm --model vgg19 --measure ncc**

Note: There could be many different configurations. The paper could be followed in order to change the parameters so they match the ones used during testing. For instance, testing the **Normalized Cross-Correlation** would yield a command like - **python verificationTool.py -p ./Test/images/image1Q.jpg ./Test/images/image1R.jpg -mm tm --measure ncc**

However, it must be also stated that this is only for predicting on a single pair of images. The whole testing datasets must be downloaded in order to achive the same results as in the paper.

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!

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! Additionally, the (**--model)** and (**--measure)** commands can be used as well.

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 results 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

In the actual file it will look like:

0.50 1

The labels text file when **testing** should have the following layout:

Query,Reference,Label

Where in the actual file it will look like:

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. The model does not have the option to use such similarity measures. However, if it was to be modified, then the threshold needs to be adjusted accordingly. That is true for the **SURF** method as well. Moreover, two pairs of test files have been provided under **Test/threshold/** folder. The pairs are ("negbesttm.txt", "posbesttm.txt") and ("negbestpm.txt", "posbestpm.txt"). These files represent the values for the best configuration recorded for the Template and Patch Matching methods accordingly. The commands for getting the best threshold for both pairs are:

- python thresholdTool.py -a ./Test/threshold/negbesttm.txt ./Test/threshold/posbesttm.txt
- python thresholdTool.py -a ./Test/threshold/negbestpm.txt ./Test/threshold/posbestpm.txt

Get Best Threshold Default:

- python thresholdTool.py -a "path to negative values file" "path to positive values 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

Adjust the intensity of the threshold - this command can be used to change the intensity with which the threshold is being found by the script. For example, if the intensity is **0.0001**, then the script will loop through every value incrementally using that intensity. It would start from **0**, then **0.0001**, then **0.0002**, etc. It must be noted that lower values can find better threshold, however the computation time will increase as well. Additionally, if the intensity is too low for values like **SSD** or **SURF**, then a memory issue might occur. Therefore, this parameter needs to be adjusted with caution. The default value is **0.000001**. The command is (**-ints "someNumber")**. An example would be:

python thresholdTool.py -a "path to negative sample values file" "path to positive sample values file" -ints 0.0001

<u>Additional Information</u>

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

Or in the file it will look like:

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.

```
# Uncomment to write samples to file
# self.write_samples_to_file(neg, pos)
```

Additionally, the path files of the two files can be changed in the acutal function:

```
# This function writes the predicted values to two files:
# negative and positive.
# This is needed for the thresholdTool script
# to find the best threshold given these two files.
#
# @param neg The list of negative predictions
# @param pos The list of positive predictions
def write_samples_to_file(self, neg, pos):
    with open('./negative.txt', 'w') as f:
        for item in neg:
            f.write(str(item)+"\n")
        f.close()
    with open('./positive.txt', 'w') as f:
        for item in pos:
            f.write(str(item)+"\n")
        f.close()
```

Testing Datasets – Download & Evaluation

The following **URL** can be followed to download both datasets used immediately - https://emckclac-my.sharepoint.com/:f:/g/personal/k1763856_kcl_ac_uk/EiSS6CNIVuRFudp28yFeRfwBUgyjEnLCA_8EnWeGMwo94g?e=ctSrOB. It contains two folders called "**caltech-buildings**" and "**wiki_commons**" respectively. They contain the images used for testing this model. The whole folders must be downloaded, so the commands described above can be executed without errors. The naming of the folders is already set to the one used for the commands. A labels text file will be provided in the **Datasets** folder of the project for the Caltech dataset. The file is called **labels.txt**. To test on that dataset (for the best configuration) the following command can be executed:

python verificationTool.py --test ./caltech-buildings/ ./caltech-buildings/ ./Datasets/Caltech/labels.txt -mm tm -thr 0.679

Additionally, the **labels.txt** file for the Wiki_Commons dataset can be found under the **WikiCommons** in the **Datasets** folder. To run the best recorded configuration, the following command can be executed:

python verificationTool.py --test ./wiki_commons/queries/ ./wiki_commons/references/ ./Datasets/WikiCommons/labels.txt -mm pm -thr 0.652

Alternatively, there is another method that can be used to download both datasets:

To download the Caltech Buildings dataset, the following **URL** must be followed - http://www.mohamedaly.info/datasets/caltech-buildings

There, a download link could be found. The dataset is 195MB.

To download the **Wiki_Commons** dataset is trickier. The authors of the **BUPM** paper has provided the required files as well as the download guide for their dataset. It can be found at this **URL** https://gitlab.vista.isi.edu/chengjia/image-GPS

Some of the images might be corrupted or removed from the database, hence they need to be removed from the respective folders. The query images must be separated from the reference images as they have the same names. They must be put in two separate folders. Because the **Google API** is a paid service, the following tool can be used to download the reference images - https://svd360.istreetview.com/

The commands that execute testing for both datasets can be altered to match the actual file paths, as they are only given as an example. If the datasets are downloaded not in the main directory of the project, then the paths to them must be changed accordingly.

If the latter methods are chosen then it must be noted that not all images from the **Wiki_Commons** dataset have been used, as also stated in **Chapter 4** of the paper. Additionally, some of the images might not be able to download. Therefore, the former method is preferred, as all the images are provided at a single One Drive location.

Additional Information

The path files provided in both commands can vary depending on where the datasets have been downloaded and how have they been named. Moreover, the paper can be followed in order to

change the configurations to match the runs tested in the paper. Every figure in Chapter 4 (Evaluation) provides information about the configurations.

Arguments Limitation in Testing

There are certain limitations to the arguments that the model accepts. In order to fully replicate the test runs provided in the paper, some of the code must be changed. For instance, if the configuration is about the **Patch Matching** method, the following lines of code can be modified:

```
imgQ = cv2.resize(queryImg, (int(queryImg.shape[1] * float(0.15)),
int(queryImg.shape[0] * float(0.15))), interpolation=cv2.INTER_AREA)
query = self.process_image(imgQ)
# Get reference patches
patches = image.extract_patches_2d(referImg, (224, 224), max_patches=250)
```

All lines can be found under the **patch_matching** function. The first and second lines define the size by which the query image is resized. For instance, if the configuration says that the query image has been resized by 20% then the float 0.15 must be changed to 0.20. The last line defines the number of patches used (**max_patches**). If the configuration uses 150 patches then the 250 must be changed respectively.

If the Template Matching is being used, then several other things must be changed. The lines that must be changed can be found under the **template_matching** function:

```
if sum(queryImg.shape) > sum(referImg.shape):
    scales = [10, 12, 14, 16, 18, 20]
else: scales = [22, 24, 26, 28, 30]
```

For instance, if the configuration uses only one scale range then all lines must be removed and replaced with only - **scales = [13,14,15,16,17]** , depending on the configuration selected.

List of Used Libraries

Keras - 2.3.1

Tensorflow - 2.0.0

NumPy - 1.16.0

OpenCV, Open-Contrib - 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)

Gast – 0.3.3

Pillow - 7.1.1

***All the methods used in this README, as well as all the papers referenced, have been given credit to and have been cited accordingly in the original paper/report of this project!**