Batch Image Processing of Images in a Folder with Python

Hakan Bostan
Technical University of Łódź
Łódź, Poland
Email: hbostann@gmail.com

***Abstract*—In this report I will explain the python program developed to batch process images inside a folder. Program relies mostly on external *SciPy*, *Numpy* and *Scikit-image* packages. Program accepts the path to the images and an operation to perform on those images. Operations are not only limited to built-in functions as user created scripts can also be used to process the images.**

1. Introduction

Image processing is used to carry out many tasks, such as interpretation of medical images or inspection/sorting of products. This processing should be done in the fastest way possible to extract relevant and useful information out of images. Currently there are quite a few software (e.g. Adobe Photoshop, ImBatch) to batch process images. These solutions usually use built-in methods to process the images.

In this program I tried to create a batch image processor which gives the users the freedom of executing their own processes on images while keeping the general properties of a batch image processor. Program is created using Python2.7 and uses some external libraries.

2. Method

The program is written using Python2.7. It also requires *SciPy*, *Numpy* and *Scikit-image* packages alongside built-in ones to function properly. Program accepts the path to image files to be processed and the operation to be performed on them with options further specifying the operation. Users can also provide their own scripts as operations if needed. After the execution processed images are saved into a new folder called 'output' at the given path.

2.1. Implementation

Program can be run from command-line by providing appropriate options. It first uses `argparse` module to check if the provided data is valid and sufficient enough to carry out requested operation. There are 5 main and 3 optional arguments.

help	Prints the help text for the command and exits.
list	Prints the built-in operations that can be readily used and exits.
path	Used to pass in the path to the images to be processed.
operation	Used to define which operation should be performed. Cannot be used with `script`.
script	Used if user wants to execute their own script. Cannot be used with `operation`.
value	Optional. Used to pass in a integer value.
point	Optional. Used to pass in coordinates.
shape	Optional. Used to pass a structuring element.

For program to work properly, given path should exist. If a built-in operation is chosen program searches for it inside `scripts` folder, confirms it exists and assigns it to be the function to be used. If a custom script is to be used, it searches the function inside the given script file and assigns it to be the function to be used. Moreover if optional values are also passed it modifies the way the function is called accordingly.

After input validation it creates a new folder called `output` inside the given director, then it starts to process images. During processing program loads an image to memory, turns it into a 2D image if it is not already and passes it to the processing function determined by the user. Return value of the function is then saved inside the created output directory.

2.2. Scripts

2.2.1. Built-In Scripts

Each built-in operation is a different python script with only a single function with the same name inside. These functions needs to be the same name with the actual file because the program searches the namespace for the requested operation and uses `getattr` function call to access the function.

2.2.2. User Created Scripts

User created scripts should also be similar to built-in ones. There should only be a single function with the same name of the script file. Function must have an argument to receive the image (in numpy ndarray format) alongside optional arguments. Return value should also be a numpy ndarray which can be interpreted as an image.

3. Results

The program is tested with all the built-in functions. To test the user provided scripts, a script file from the script directory is simply removed and used as a user created script. Below are some results of the program run with different operations.

Figure 1: Result of gaussian blur on left, original on right

Figure 2: Result of sharpening on left, original on right

Figure 3: Result of region growing on left, original on right

4. Discussion

Program works as wanted but there is still a lot to improve. In the current situation program is not user friendly. This can be improved by implementing a GUI. Also algorithms used in some operations, for example `regionGrowing`, are really inefficient. These can also be implemented in a more efficient way. Overall this program can process images in a directory rapidly and present the output.

5. Project Materials

All of the material used in this project is available under https://github.com/hbostann/ImageProcessingProject. It includes the ReadMe file, source code for the main program, operation scripts and sample images used for testing.