CMPSC 102
Discrete Structures
Fall 2019

Practical 6: Creating an Analysis of Snow From Images, Determining Color in images and Hacking Python Code

Refer to your notes, slides and sample Python code from this week and other weeks.

Summary

Image analysis, using algorithms, has become an important area of computer science for its ability to scan countless pieces of graphical data (files) to search for specific types of information (parsing). In environmental research, for example, image analysis can be done by automated means and is used to determine erosion, changes in the landscape and other types of environmental change over time.

For instance, by studying a series of satellite images of mountain ranges (taken at the same time of year and spanning across several years), snow coverage can be determined using image analysis software. The software works by counting and comparing the number of white pixels (representing snow) in each photos.

Ascertaining the the differences of snow fall for a particular time of year enables scientists to monitor types of climate change. This data analysis also helps to drive further research to learn of other types of change in terrains (such as plant coverage and desert expanse) and could motivate the creation of international policies of environmental and climate protection.

GitHub Starter Link

https://classroom.github.com/a/3obBAEdU

To use this link, please follow the steps below.

- Click on the link and accept the assignment.
- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab.
- Clone this repository (bearing your name) and work on the practical locally.
- As you are working on your practical, you are to commit and push regularly. You can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):
  - `git commit <nameOfFile> -m ‘‘Your notes about commit here’’`
  - `git push`

  Alternatively, you can use the following commands to add multiple files from your repository:

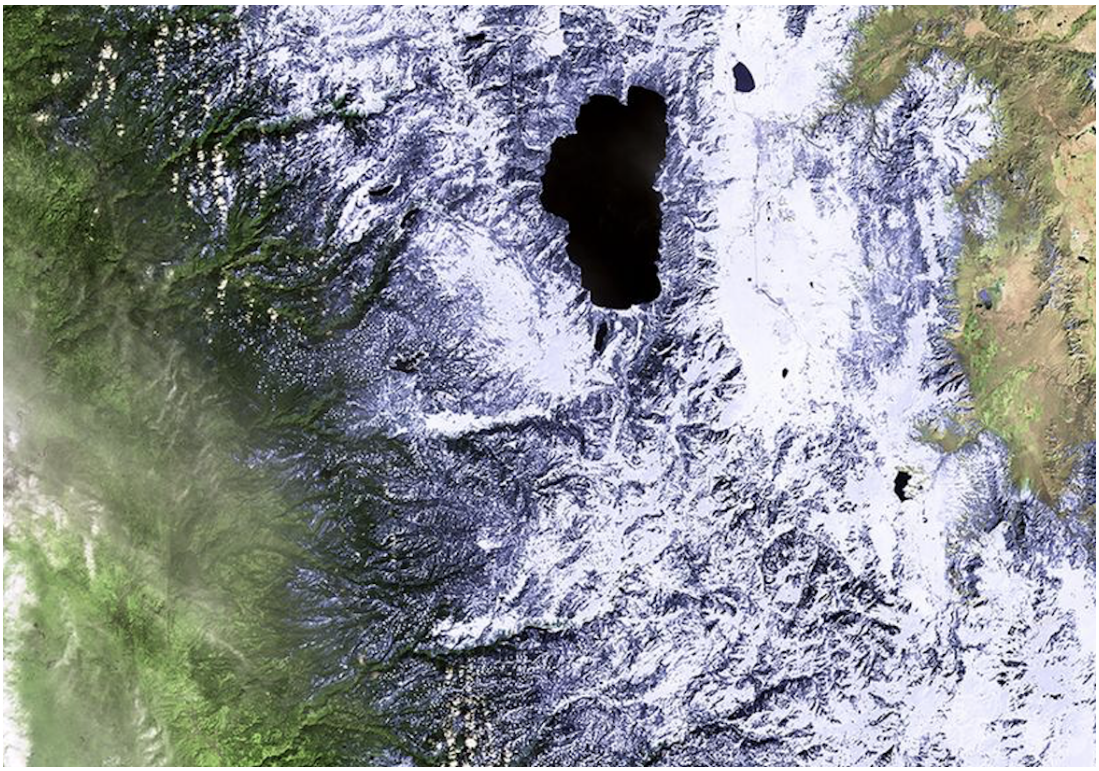

Figure 1: A satellite image of a California mountain range showing lots of white snow on the range. If we look at more recent photos, is there less white snow to see?

- `git add -A`
- `git commit -m ‘‘Your notes about commit here’’`
- `git push`

What to do for this practical

In this practical, you will be experimenting with image analysis to determine the snow coverage (featured in the images). Your code will be supplied with a Python3 program to perform the analysis. You will then *hack* the program's function, `computeFoliage()`, to perform an analysis of foliage content in the images. Note, the program's functions, `computeSnow()` and `computeFoliage()` are nearly identical except for the way that each function determines whether or not to count pixels of a particular type of color in an image file. The decision to count a particular pixel will be decided using `True` and `False` statements in the code.

You will use the output of the function `getAvgRGB()` to determine working thresholds for the red, green and blue colors that are used to count the pixels that relate to foliage cover (plants and vegetation) in the images. These thresholds are to be used in the function `computeFoliage()` for the foliage analysis.

Study and experiment with the given program

Locate the program source code (file: `src/imageScanner_i.py`) with which you are to work. The program works by looking through each pixel of an image file. If the pixel matches some predetermined definition of a particular color (defined by the threshold values) then the pixel is counted to return a score of coverage. Take some time to study the code and to learn how it works. Then, begin your work. Please be sure to add your name to your code.

Locate the graphics files in the `graphics/` directory, such as the one shown in Figure 1. The files, `Feb2011.png`, `Feb2013.png` and `Feb2014.png` are satellite images of a mountain range in California from the years; 2011, 2013 and 2014, respectively. Launch your image scanner program to check that it works to count snow coverage. Ask yourself whether the snow coverage assessment from the code makes sense when you view each image.

Libraries to install: MatPlotLib, NumPy

You may need to install two libraries for the code to run.

- **MatPlotLib**, `https://matplotlib.org`
- *Installation (one of the below lines)*
  - `pip3 install --user matplotlib`
  - `pip install --user matplotlib`
- **NumPy**, `https://docs.scipy.org/doc/numpy-1.10.1/user/install.html`
- *Installation (one of the below lines)*
  - `pip install --user numpy`
  - `pip3 install --user numpy`

Run your code

To run the program, type the following commands.

```
chmod +x imageScanner_i.py #used to make the file executable in the unix terminal
# then,
./imageScanner_i.py ../graphics/Feb2011.png ../graphics/Feb2014.png

# windows users:
python3 imageScanner_i.py ../graphics/Feb2011.png ../graphics/Feb2014.png
```

Determine magnitudes of red, green and blue color values

You can use the `getAvgRGB()` function to your source file to study the full colors of blue, red, yellow and green. Find your experimental graphics in your `graphics/` directory and determine

what the threshold values are to guide your work when you are scanning images using the function `computeFoliage()`.

For example, run `imageScanner_i.py` using the file `green.png` as a parameter. Type the following command.

```
./imageScanner_i.py ../graphics/green.png
```

The output is the following (from this part of the program).

```
+ ../graphics/green.png | getAvgRGB(), the average RGB colors:
      [0.0, 0.65, 0.36]
```

This output represents the average magnitudes of **red**, **green** and **blue** pixels, respectively, that were found in the `green.png` file. There are other color image files with which you may experiment. Since each of these images contains only one color, the *average* of the magnitudes may be understood to be the measurement of the only hue (i.e., red, green or blue) in the entire image.

So, did you say that there were True and False parts of this practical ...?

In the function, `computeSnow()`, (shown below) there is a line that computes `True` or `False` depending on whether the color values from the images are greater than a set threshold or not. This is the part of the program that, if the line is equivalent to `True` then a pixel count is incremented for a final score of color in the image. Note, the i and j values make-up the Cartesian address of the pixels in the image. Your code to check for foliage will work the same as the function that checks for snow, however, the color values of red, green and blue will be different. Can you determine what the best thresholds should be for foliage colors?

```
redVal_flt   = 0.75
greenVal_flt = 0.75
blueVal_flt   = 0.75
...
# the Red Green Blue values (channels of color)
if (cam[i,j,0] > redVal_flt) and (cam[i,j,1] > greenVal_flt) and (cam[i,j,2] > blueVal_flt)
```

Deliverables

1. Your completed Python3 code (`src/imageScanner_i.py`) containing working threshold values in the function, `computeFoliage()`.
2. Your mini-reflection Markdown document `writing/miniReflection.md` containing your responses to the questions posed in the document (please contribute one or two **meaningful** sentences for each question).