Basics of Digital Images and Tools for TSL Data Science Team

Written by Kayvon Khosrowpour

Practice Problems Due: 10/17/18 at 12:00pm

Intro

Our future work in TSL will require using scientific python tools to collect, analyze, clean, and manipulate data, implement machine learning algorithms, and build neural networks. We'll also have to understand how digital images work to intelligently design our algorithms.

Although everyone on the team has exposure to these tools, people on the team have different experiences with using these tools for image processing. Some of us are taking (or have taken) machine learning/image processing courses. Some team members have more experience with certain tools than others. My goal this week is to make sure we're all familiar with the most basic image processing concepts before we try to learn machine learning or more complex image processing.

It's not my intention to assign unnecessary tasks. We need to understand how to manipulate image data before we can use the data in machine learning. This reading and the problems provided are to help us understand image data and basic image processing algorithms.

It is not necessary for you to be able to implement these algorithms yourself, <u>but at a high level, you should understand what they do and how they work</u>. I will ask every member to explain one algorithm/concept mentioned within in this doc at the next meeting.

If you're comfortable with these concepts, let me know so we can determine how you can best spend your time. At least, you should have the tools mentioned below installed.

Tools

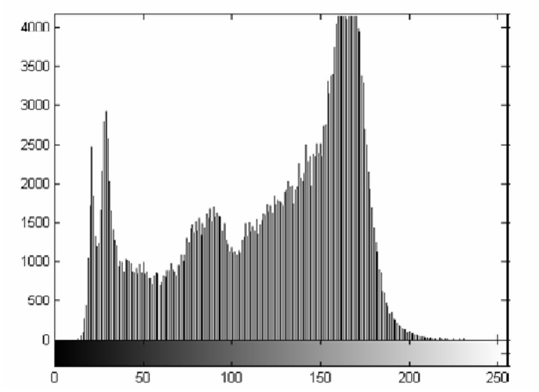
- Python 3 with pip
- Numpy
- Matplotlib
- OpenCV for Python
- Pandas
- Scikit (sklearn, skimage)

Once python is installed, the rest of the tools are installable using pip. If you have trouble installing these tools, let me know as soon as possible.

Digital Images

The two types of images we'll mostly use are grayscale, binary, and RGB. A grayscale image is a 2D matrix where each entry (pixel) in the matrix is generally a value from 0 to 255 (represented with 8 bits). A higher value/level means higher intensity (brightness) and a lower level means lower intensity (0 is black, 255 is white). A binary image is a 2D matrix where each pixel is a logical 0 or 1. We don't usually display these kinds of images without some manipulation, but they're useful for applying image processing algorithms and determining shape. An RGB image is a 3D matrix where, each pixel has a red, green, and blue component. In an RGB image, the 2D matrix of all the red intensities (i.e. the red “layer”) is the red channel. The 2D matrix of all the blue intensities is the blue channel, and so on.

A histogram of a grayscale image is a plot of the frequencies of levels that appear within the image. Some image processing algorithms (binary thresholding, morphological operators like erode/dilate/open/close) are best utilized through analysis of grayscale image histograms. The x-axis is the range of intensities. The y-axis is the number of pixels that have a given intensity. In this image, there’s very few white pixels and many medium-ranged-valued pixels.



Practice Problems
All these problems should be completed using python, numpy, opencv, and matplotlib.

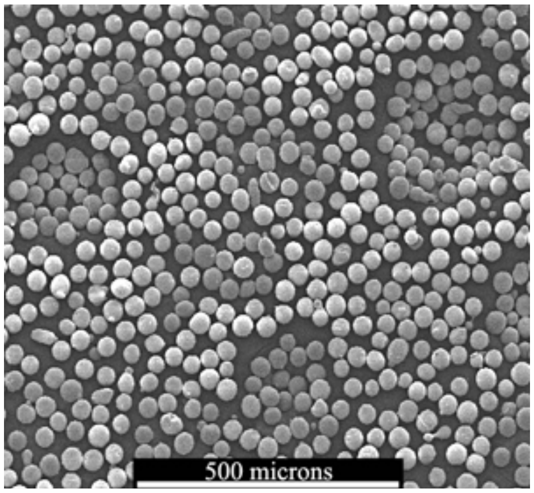
Digital Image Structure

1. Read an image as RGB and display it on the screen (use opencv, matplotlib)
2. Read an image as grayscale and display it
3. Read an image as RGB and convert it to grayscale, display it
4. Read an image as RGB and separate the image’s color channels (use split() or numpy indexing - Which method is computationally faster?).
    a. After separating the channels, make 3 new RGB images and display them
    b. The first new RGB image should have only the red channel (no blue/green)
    c. The second new RGB image should have only the green channel (no red/blue)
    d. The third new RGB image should have only the blue channel (no red/green)
5. Cut out a rectangular section of the image and display it on screen (use numpy indexing)

Digital Image Characteristics

1. Read an image as grayscale and plot the histogram of its intensities (use matplotlib)
2. Read two images as grayscale and plot both of their histograms on the same plot with different colors (use matplotlib)

Morphological Operations
These operations are used to modify an image’s shape. Often, grayscale images are converted to binary images using thresholding. Thresholded images are often used for contouring and extracting shape from a complex image. They’re also used for finding objects in the image.



1. Save the above image and perform binary thresholding on the image to separate the particles from the background. Play around with a binary threshold to find the best value or use a histogram to determine the best threshold value.
2. Read about morphological operations on OpenCV. On the binary image, apply erosion, dilation, opening, and closing to the image to create 4 new images.
   a. Play with different windows/structuring elements
   b. Display the original image and the 4 new images after applying these operations. What did each of the operations do to the image?
   c. After applying the operations to the image, determine which approach is best used to separate each particle. Hint: dilation is not useful for this goal.
3. After determining the best operation to separate the particles in the binary image, apply connected component labeling (CCL) to count the number of particles. Unfortunately, CCL isn't documented for python but it is implemented. Your estimate does not need to be exact.

Notes

- Images in OpenCV are not implemented as RGB. They are implemented as BGR. So, if you want the red value at pixel index [i, j] then the index of the red channel at [i, j] is [2], not [0].
- MacOS has known problems with OpenCV waitKey() functions. Hopefully OpenCV fixes it in the future. In the meantime, use matplotlib to display images if you're on mac.

Reference

1. OpenCV Python Tutorials

Submission Instructions

1. Create a public repository on GitHub and upload your code.
2. Save all of your image results in a word document and push it to your repo.
3. DM me on Slack the link to your solution before 10/17/18 at 12:00pm.