COMPUTER VISION

Assignment 1: Color Quantization and Connected Component Analysis

Opened: Friday, 14 October 2022, 12:00 AM
Due: Thursday, 27 October 2022, 11:59 PM

In this assignment, you will be implementing several tasks related to color quantization and connected component analysis.

Task 1: Color Quantization

Color quantization is the process of reducing the number of colors used in an image while trying to maintain the visual appearance of the original image. In general, it is a form of cluster analysis where each RGB color value is considered as a coordinate triple in the *3D* colorspace. Example below shows an RGB image (~16 Million possible colors) quantized using *2*, *4*, *8*, *16*, and *32* colors.

As for your first task, you will be working on a ***Python*** implementation of basic image processing operations and color quantization techniques.

You are going to write a function called `quantize(img, k)` where `img` is an RGB formatted image and `k` is the number of colors in the output image after quantization. The function will calculate and return the quantized version of the image using the following approach;

1. Convert input image to a matrix of pixels. Depending on your image size, this matrix should be of size ***height x width x 3***
2. Pick ***k*** initial colors to begin the quantization process. Use the following steps to initialize your algorithm;
   - Choose initial ***k*** color centers *manually*, by clicking on the image; you can achieve this with ***pyplot.ginput*** function of *matplotlib* library.

     ```
     from PIL import Image
     import matplotlib.pyplot as plt

     im = Image.open("frog2.jpg")
     plt.imshow(im)
     points = plt.ginput(3, show_clicks=True)
     print(points)
     ```

   - Choose initial ***k*** color centers *randomly*. Pick color centers using ***numpy.random.uniform*** function.
3. Using your own ***k-means*** implementation, cluster your color matrix and find ***k*** clusters and their centers. You may limit the maximum number of iterations to a suitable number depending of the size of your images.
4. Generate the output image in which each pixel should have the color of the cluster-center that pixel has been assigned to.

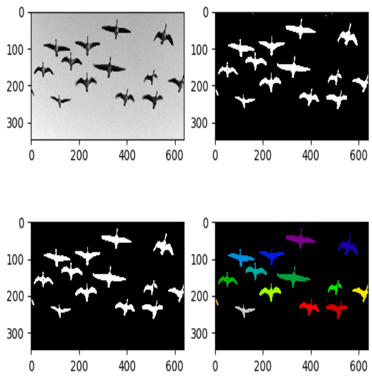
Once you are done with your quantization pipeline, you are going to experiment on the following images and present color quantized versions with **k = [2, 4, 8, 16, 32]** of these images using both initialization methods in your report.

Bonus

Try using k-means clustering on an uniform color coordinate system (***Lab*** colorspace)

Task 2: Connected Component Analysis

In this part of your assignment, you will be implementing a connected component analysis algorithm. In the approach, you will begin with color-based thresholding of the image. Following noise removal with morphological operations, you will have an image containing connected and disconnected regions. Since these regions are countable, you can determine the number of distinct objects on an image such as the number of birds in the example below.



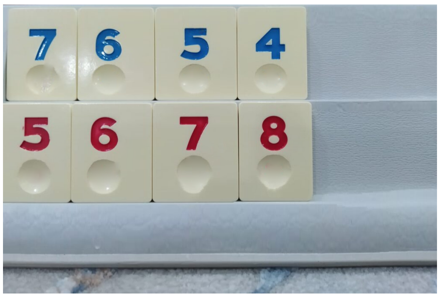
To perform connected component analysis and count the number of object in the images given below, you will implement a function called count_connected_components(img) where img is an RGB formatted image. The function will calculate and return the number of objects in the image as output. It is highly recommended that you preprocess the image either prior to or inside the function using techniques such as thresholding, and color manipulations (normalization, color model change etc.) as needed.

The steps you should implement can be described as follows;

1. Read the given input image and apply preprocessing and thresholding. The output of this process should give you a binary image where pixels have the value of **[0, 1]**. You may choose to reduce the image size or perform consecutive thresholding operations.
2. If needed, apply morphological operations such as erosion or dilation to clean salt&pepper noise.
3. Implement connected components analysis with **8 connected neighborhoods** for each pixel. Traverse the image, going over all pixels to find foreground pixels. Once the first pixel of a connected component is found, all the connected pixels of that connected component are labeled before going onto the pixel in the image. Finally, return the number of desired connected components.

You are not allowed to use functions already implemented in the **OpenCV (cv) library or similar other libraries for this task other than **numpy**.**

Once you are done with your connected components analysis pipeline, optimize the usage of binarization thresholds, morphological operations and other necessary thresholding techniques for each of the six images below. For each image, return ***the binary image*** (the input of the connected components analysis), and ***output of the function*** (i.e number of birds, the number of dots on the dice and the number of pieces on the okey board).



Development Environment

Installing Python

Throughout the semester, we will be using **Python 3** as our programming language. We recommend that you use *Conda* and suitable *IDE* (i.e PyCharm, VSCode) to setup your development environment and install the necessary libraries.

You can obtain lightweight version of *Conda* from **Miniconda**. Follow the installation instruction for your operating system. Once the installation is complete, create a new virtual environment and install the following libraries.

- Numpy
- OpenCV
- Pillow
- Matplotlib

Installing an IDE

We recommend that you use an IDE (PyCharm or VSCode are suitable choices) for ease of debugging your implementations. You are free to use any IDE of your choice. For PyCharm, you can follow the instructions below;

Obtain PyCharm Professional Edition for free by first signing up as a student at https://www.jetbrains.com/shop/eform/students with your university email and then downloading the professional version of the software.

Login to your new PyCharm account to obtain your username and password, and use them to register your IDE.

Run PyCharm, open a new project. While creating the project, select the python environment you created for this project.

Deliverables

- **Project Report (pdf & tex):** A maximum two page **pdf** report file containing small versions (resize image so that 4-5 of them should fit a single line) of all the requested figures, image and descriptions of the methods that you have implemented. **Comment on the results and elaborate on your conclusions. Do not print any source code in your report.**
- **Source code (.py files):** Submit all the functions you have implemented for this assignment in two separate files named as *task1.py* and *task2.py.* Your submitted codes should have a "*__main__*" methods for us to test your code without typing in any external arguments. If you use JupyterNotebook in your assignment, please include a main **.py** file along with it in your submission as well (containing the functionality wanted above).

WARNING! Submit all files as one zip file. Please send files in correct format. Use zip for packaging, do not use rar, 7z etc. If you are not be able to upload your submission files to Moodle, submit a **Dropbox** or **Google Drive** link of the compressed package with the name; "**[cmpe537-fall22-01][Your Student Id][Your Name].zip**".

For questions & comments please email ogulcan.ozdemir@boun.edu.tr

Important Note: If you are using PyCharm, you may need to do the following fix for ginput. Under Settings => Tools => Python Scientific *Uncheck the (only) box "Show plots in toolwindow"*. Future plots should appear "normally" and not in **SciView**.

images.zip 14 October 2022, 2:22 PM

Submission status

Submission status	No attempt
Grading status	Not graded
Time remaining	4 hours 31 mins
Last modified	-
Submission comments	Comments (0)

Add submission

You have not made a submission yet.

◄ color sensing

Jump to...

Please upload term project proposal here ►