Internship Report on
Image segregation using Python In
Ameya Intelligent Machine Labs

Supervised by:

Lalit Vidyasagar
Santhiraj K
Raja Emmadi

Submitted by:

Bhargav Malyala

Intern

Bhargav4402@gmail.com

Problem Statement:

Three folders with images will be provided , all stored with unique names of which some images are similar, and some are not. For this reason , two new folders are to be created , one containing all the unique images in it and the other folder containing the duplicate images.

Scenarios

- In WhatsApp, we frequently get same images forwarded from various groups which is redundant data. Tackle this issue, we can use this code to separate the duplicates from group of images and can delete the duplicates if required.

- While doing backup of our images, we usually backup all the images and do not check for duplicates which may in turn eat up our storage, so this code can be used in such scenarios as well.

- In any events, generally the photographer clicks multiple images at a time to get the best picture from a given situation, and so as to segregate the similar images, we can use this code.

How does the code work?

The code takes a folder path containing large group of images then we loop the images such that, every image is compared with all other images present in the list from itself and we start the process on the two images by importing **cv2 library** (OpenCV) we make use of **"cv2.cvtColor( )"** method to convert the image from BGR (or RBG) to HSV (Hue saturation value) to obtain the image's colour (or hue/tint) in terms of their shade (saturation) and brightness value. We create a **histSize** of [50,60] and can be changed as per the requirement. (Note: histSize is the number of lines or commands that are stored in memory in a history while the bash session is

going on). We then convert the HSV image to histogram using **"cv2.calcHist( )"**and with channel[0,1] i.e., greyscale(black and white) and of size [50,60] i.e., the histSize that we initialized before. Using **"cv2.normaize"** we apply normalization on the entire image(i.e., change the intensity level of the pixel which is the histogram of the image) and normalize both the images. We the calculate the correlation of two histograms and the value ranges from 0 (0% similarity) to 1(100% similarity) for the images using **"cv2.compareHist".** We can change the similarity percentage as per our requirements.

Test Case:

Test Case ID	Test Case Description	Test Steps	Expected Results	Actual Results	Pass/Fail
1	User inputs image, unique and duplicate folders' paths	1. Enter path of folder containing images 2. Enter unique folder path 3. Enter duplicate folder path	Images will be segregated into unique and duplicate folders	As Expected.	Pass
2	User inputs image folder and unique folder paths	1.Enter path of folder containing images 2.Enter unique folder path	New folder for duplicate images will be created and images will be segregated.	As Expected.	Pass

3	User inputs image folder and duplicate folder paths	1.Enter path of folder containing images 2.Enter duplicate folder path	New folder for unique images will be created and images will be segregated	As Expected.	Pass
4	User inputs path of images folder only	1.Enter images folder path	New folders for storing unique and duplicate image will be created and the images will be segregated	As Expected.	Pass
5	Images contains cropped and rotated Versions of images	1. Enter images folder path 2. Enter unique folder path 3. Enter duplicate folder path	Even the cropped and rotated images are considered as duplicates based on the percentage similarity value(compareHist)	As Expected	Pass
6	Images of different resolutions are compared	1. Enter images folder path 2. Enter unique folder path 3. Enter duplicate folder path	The images with different resolutions are treated as duplicates and will be separated to unique and duplicate folders	As Expected	Pass

Limitations:

- Long execution time as each image is compared with all other images.
- As per the accepted similarity percentage given, it may treat a zoomed image as unique.
- New folders are being created to store unique and duplicate images leading to more space wastage. (If unique and duplicate images are copied instead of moving them)

Future Enhancement:

- The code can be developed to reduce number of comparisons to increase its execution time and efficiency.
- The code can be developed to automatically find images folder in the system and perform comparison and segregation of images periodically.