Imdedu project

Why image hash 9\*8

<https://pyimagesearch.com/2017/11/27/image-hashing-opencv-python/>

image rotation

<https://stackoverflow.com/users/885287/aaronsnoswell> - [Rotate image and crop out black borders](https://stackoverflow.com/questions/16702966/rotate-image-and-crop-out-black-borders)

imagehash

<https://pypi.org/project/ImageHash/>

<https://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html>

Generally in this project, we want to solve an image processing problem. Imagine we have a hard drive that consists of a lot of images in different formats. some of them are unique images and the rest have one or more other versions. By version, we mean a exact copy of the image, cropped or rotated, or color filtered, etc. In this project, we tried to tackle this problem and we believe we reached a baseline that can be used for further studies and future works in this specific area.

Find\_Save\_Duplicates.py

For the first part, we created a module called Find\_Save\_Duplicates. In this module, we will find all the images in the current hard drive and save their SHA256 hashes with their path in which we can find them in the hard drive. We save all the mentioned information in an SQL file for the next steps.

This module has these functions.

add\_values\_to\_database(Hash, File\_path):

This function gets the calculated hash and the path of the image which been found in the memory and will add and save it to the database that we created.

find\_image\_files(directory):

this function checks the format of the files in the given repository and at the end return the path of all the image files in that repository.

def get\_hash(file):

will calculate the sha256 hash of the given file.

Notes:

Line 43 is the part of the memory that we want to loop through it and find all the image files.

Line number 44 is the given name for the final database file and its path in which we want to save it.

Get\_Save\_Visualhashes.py

With the help of visual hashing algorithms, in this file, we are trying to find the images that belong to each other. Now that we found all the images in the memory, just by a simple query we can get all the unique images based on their content. In the previous step, we calculated sha256 for each image. We know that with the properties of security hashes like sha256 now we know which images are unique based on their content hash. So in this program, we will send a query to the the database and get all the unique images, and then we calculate the visual hash of each image by using the Imagehash library.

Because the dhashing is not robust against the rotation of the image in large angles, we provided a function that calculates the rotation of each image in certain angels and steps to compare the possible outcome of each image with others and tackle this problem.

The final output of this program is a database with all the calculated visual hashes of all the images found in the memory. The data base has path, Visual hash, and the degree of rotation.

def rotate\_image(image, angle):

def largest\_rotated\_rect(w, h, angle):

def crop\_around\_center(image, width, height):

These three functions are from user aaronsnoswell in Stackoverflow.

<https://stackoverflow.com/users/885287/aaronsnoswell> - [Rotate image and crop out black borders](https://stackoverflow.com/questions/16702966/rotate-image-and-crop-out-black-borders)

These three functions provide an image rotation functionality for our project. You may wonder why we can not use a simple rotation function in the OpenCV library which can be run in a single line. The problem is with those algorithms depending on the degree of rotation there will be a black background in the newly generated pixels for the image. This method is not the feature that photographers or users use to rotate an image. To rotate the image in the proper way we have to calculate the exact borders of the rotated image to prevent saving those black pixels and these three functions are for this job.

def Get\_Posible\_Rotation\_Hash(image, teta, n,source\_path):

in this function, we rotate an image teta degree n times and then calculate the visual hash of each of them, and then save the hashes in the database.

def get\_unique\_paths(database\_path,query):

this function will read all the unique images we found in the previous step with a simple SQL query.

def add\_values\_to\_database(Hash, File\_path, Rotation):

Again will add the values to the database.

def get\_hash(image):

This function will calculate the visual hash of the given image.

Notes:

In line 185 you have to provide the database name generated in the last step.

In line 190 you have to provide the path in which you want to save the database.