Count Duplicate images

Jaswanth Bandlamudi

1 Introduction

The challenge is to traverse through the dataset provided and find images which are visually similar. The dataset consists of sequnce of an image collected in a parking area with main differences being

1. External Illumination
2. Artificial Lights (infrastructre and vehicles)
3. Vehicle presence
4. Access door movement

In order to establish the similarity between the images. We considered the difference in the object presence and lighting changes, while excluding the change in external illumination.

2 Available resources

We are provided with a sample file of pre-processing and processing function needed to obtain a score which resembles the image similarity.

3 Methods

In order to find the similarity we used the method based on the functions available along with another method which applied hashing on images to find the similarity.

3.1 Absolute Difference based duplicate image extraction

In this method,

1. images are converted into grayscale to reduce the chromatic difference effecting the comparison.
2. We decided not to apply any gaussian blur since every feature is important in calculating the difference between images. Applying Gaussian Blur results in loss of such information.
3. Using absdiff function from opencv library we calculated the difference between two images.

4. The differential image is converted to a binary map using the threshold function. To improve the edge appearance dilation is applied to the differential image.

5. Contours are extracted from the dilated map to extract the area of the differences between the images.

6. A threshold is applied to the area calculated, so as to not consider the minor changes due to external illumination effects. The threshold is decided based on the area of smallest most differential object from the images.

Using this method we were able to report a total of **271** duplicate images. The processing time took by this method on an Intel i7 8600H processor powered laptop is 74 minutes

3.2 Average hash based duplicate extraction

Hashing is a function that applies to an arbitrary data and produces the data of a fixed size (mostly a very small size)

In this method

1. convert the input image to grayscale and discard any color information. This helps in easier hashing and reduced effect of colorspace.

2. Grayscale image is resized into a (8 x 8) or a (16 x 16). Since there are some finer details in the provided images, we chose to resize them to (16 x 16) aspect ratios

3. A 64 bit hash is created based on whether the pixel's value is greater than the average color of the image.

For similar images hash values match each other. Using this method we found that there are **234** duplicate images in 4.5 minutes on an Intel i7 8600H processor powered laptop.

4 Observations and Discussions

From the above performed experiments, we observed

1. The currently used function might be intuitive and mathematically strong. But, it needs alot of tuning for the dataset and also might not work in case of affine transformations on the images.

2. The current method is also very time and power consuming. Because of the need for dual traverse through image resulting in time complexity of $O(n^2)$, which is not ideal

3. On the other hand, hashing techniques are faster and computationally efficient with a time complexity of $O(n)$.

4. The current implemented hash based algorithm is simple and rudimentary. but I believe with usage of hamming distance and threshold application the performance can be further improved.

5 Question and Answers

1. What parameters you decided to use for the provided example dataset

Ans: I decided to modify minimum contour area to represent the smallest possible considerable difference in the images.

2. How you found these values ?

Ans: Careful human observation and contour area calculation for unique images.

3. What amount of duplicates script found with these parameters

Ans: using the functions provided I managed to find a total of 271 duplicates. While using Average-Hashing technique, we found 234 duplicate images

4. What you would suggest improving to make data collection of unique cases better

Refer Section 4

5. Any other comments about imaging interview.py or your solution ?

Refer Section 4