PHASE 1

Report By Shivam Malviya
Group 17

Group Members :

1. Aaryan Gupta
2. Fenil Madlani
3. Krisha Gala
4. Pranav Katariya
5. Radhika Ganapathy
6. Shivam Malviya

Abstract :

This report is a study and analysis on image features, vector models, and similarity/distance measures considering the different variations, advantages, disadvantages, and accuracy of the methods used. Recognizing similar images is a simple task for a human being even seeing a person after several years. But doing the same by a computer is not easy because the computer will have problems if there is a change in the lighting conditions, complex background, pose, and occlusion. The dataset considered for the study is the Olivetti Faces Dataset. This dataset contains a set of grayscale images of size (64*64) pixels. Three different types of features (Color moments, Extended Local Binary Patterns, Histogram of Oriented Gradients) are extracted from the grayscale images. Different distance metrics are used to compare the feature vectors and find a matching score Finally, we propose a way to combine the results from comparisons of all three feature vectors and find an overall matching score. The results from experiments show that the extraction of simple feature vectors from complex images can be beneficial in finding similar images.

Keywords :

1. Image features.
2. Color Moments.
3. Extended Local Binary Patterns.
4. Histogram of Oriented Gradients.
5. Distance metrics.
6. Feature Extraction.

Introduction:

The Olivetti faces dataset :

There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open/closed eyes, smiling / not smiling), and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). The image is quantized to 256 grey levels and stored as unsigned 8-bit integers; the loader will convert these to floating-point values on the interval [0, 1], which are easier to work with for many algorithms. [1]

Terminology :

1. ***Color Moments**:*

Color moments characterize color distribution in an image similarly as central moments uniquely describe a probability distribution. Color moments are mainly used to compare how similar two images are based on color. Color moments are scaling and rotation invariant. Color moments cannot handle occlusion very successfully. But they encode both shape and color information and so they are a good feature to use under changing lighting conditions. For a grayscale model there are 3 color moments for each pixel. [2]

a. Mean: Average color in the image.
b. Standard Deviation: Square root of the variance of color distribution
c. Skewness: Measure of how asymmetric the color distribution is.

2. ***Extended Local Binary Patterns:***

LBP by definition are invariant with respect to any monotonic transformation of the gray scale. This is achieved by considering just the signs of differences instead of the exact values of the gray scale. [3]

Let T ≈ t(s(g0 − gc), s(g1 − gc), . . . , s(gP −1 − gc))

In a local neighbourhood with gray levels of P(P > 1) image pixels. Where gP (p = 0, . . . , P −1) gray values, gc being the centre gray value and

s(x) = 1, x ≥ 0 0, x < 0 (2)

The sign is 1 if positive and 0 if negative.

The above is converted into a unique P-bit pattern code by assigning binomial coefficient 2^P to each sign s(gP − gc):

LBP(P,R) = Summation over P in (s(gP − gc)2^P) from 0 to (P-1)

ELBP features are calculated using the rotational invariant with “uniform patterns”. [3]

3. ***HIstogram Of Oriented Gradients:***

The technique counts occurrences of gradient orientation in localized portions of an image. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled.[4]

Algorithm : Gradient Computation -> Orientation Binning -> Descriptor Blocks -> Block Normalization

Goals:

Task 0 : Familiarize ourselves with the data set
- – Face images
- – Associated metadata

Task 1 : Implement a program which, given an image ID and one of the following models, extracts and prints (in a human readable form) the corresponding feature descriptors:
- – Color Moments
- – Extended Local binary patterns, ELBP
- – Histograms of oriented gradients, HOG

Task 2 : Implement a program which, given a folder with images, extracts and stores feature descriptors for all the images in the folder.

Task 3 : Implement a program which, given a folder with images and an image ID, a model, and a value “k”, returns and visualizes the most similar k images based on the corresponding visual descriptors. For each match, also list the overall matching score.

• Task 4: Implement a program which, given a folder with images and an image ID and a value “k”, returns and visualizes the most similar k images based on all corresponding visual descriptors. For each match, also list the overall matching score and the contributions of the individual visual models.

Proposed Solution:

The project was worked on an Ubuntu Machine in an Anaconda Environment.

Libraries Used and Installation Instructions:

1. “cv2”

   Available in the open-cv library.
   Run “*conda install -c conda-forge opencv*”

2. “numpy“

   Run “*pip install numpy”*

3. “glob”

   Run *“pip3 install glob3*”

4. “PIL”

   Run
   “*python3 -m pip install --upgrade pip”*
   “python3 -m pip install --upgrade Pillow”

5. “skimage”

   Run *“pip install scikit-image*”
6.

7. “scipy”

   Run “*git clone https://github.com/scipy/scipy.git scipy”*

8. “sys”

   Installed with python as part of standard library.

9. “natsort”

   Run “*pip install natsort”*

The code requires Python 3.9 to be installed in the system. Please check out https://www.python.org/downloads/ to install python for your specified system.

Task 1 and Task 2:

1. ***Color moments:***

   First, I created a function to split images into 8*8 windows. As the image size is 64*64, we get 64 windows. If arr is a 2D array, the returned array looks like n subblocks with each subblock preserving the "physical" layout of arr.

   Next I defined a function for calculating the standard deviation and skewness.
   The formula for standard deviation is:

$$\sigma_i = \sqrt{(\frac{1}{N}\sum_{j=1}^{N}(p_{ij} - E_i)^2)}$$

   where N is the number of pixels in the image and pij is the value of the j-th pixel of the image at the i-th color channel and Ei is the mean value, or first color moment, for the i-th color channel of the image.

   The formula for skewness is:

$$s_i = \sqrt[3]{(\frac{1}{N}\sum_{j=1}^{N}(p_{ij} - E_i)^3)}$$

   where N is the number of pixels in the image and pij is the value of the j-th pixel of the image at the i-th color channel and Ei is the mean value, or first color moment, for the i-th color channel of the image.

   THe formula being the same with difference just in the power, I created the function for the same any power n.

   Then I created a function for creating the color moment vector for all windows of the image. The image array is passed to the function as an argument. This first splits the image into 8*8 windows using the split function. Mean vector is then calculated using “numpy.mean”. The standard deviation and skewness vector is calculated using the function mentioned above. All three vectors are combined into a single vector and returned from the function.

2. *Extended Local BInary Patterns:*

I used the “local_binary_pattern” function available in the skimage python library to compute this feature vector.

Parameters:

1. image: grayscale image (64*64)
2. P (int) : Number of circularly symmetric neighbour set points .
3. R (float) : Radius of circle.
4. method :Method to determine the pattern.

I set the value of R as 1 and value of P as 8. This can be visualized easily by considering a 3*3 square which is further divided into 9 “1*1 squares”. The centre square is the pixel in consideration and this has 8 other squares as its neighbours.

Output:

The output is a feature vector of size (number of images, 64, 64)

3. *Histograms of oriented gradients:*

I used the “hog” function available in the skimage python library to compute this feature vector.

Parameters:

1. number of orientation bins = 9,
2. cell size = 8,
3. block size = 2,
4. L2-norm clipping threshold = 0.2 (L2-hys)

The image is first resized to 128*64 using numpy.resize and then passed to the function.

Output:

The output is a feature vector of size (number of images, 3780)For Task 1 the feature vector of an image is displayed using Image ID.
For Task 2 the output vectors from the above functions for the whole given folder are dumped into a .csv file using “csv” python library.

Task 3:

For Task 3 I tried out a lot of distance metrics to find out similar images using the feature vectors.

For the Color moments model, I finally chose to use the Manhattan distance (the sum of the absolute differences between the two vectors).

For the ELBP model I first tried converting the ELBP vector into histograms which is basically a probability distribution. Then I used K-L divergence to find out the matching score.
Next I tried to compare the same using Euclidean distance too. The results from K-L divergence were not able to perform as good as the results from Euclidean distances and so I finalised using the Euclidean distance.

For the HOG model, again I tried using K-L divergence and Euclidean distance. The results were much more explainable and visualizable when using Euclidean distance.

I went through some research papers for the same and found that Euclidean distance is mostly used for ELBP and HOG feature vector comparison and has been able to give good results in different applications.

Task 4:

For this task I first converted the distances calculated for each image into a standardised score using the z score function. A Z-Score is a statistical measurement of a score's relationship to the mean in a group of scores.

$$Z = \frac{x - \mu}{\sigma}$$

where,

Z = standard score.
x = observed value.
u = mean of sample
σ = standard deviation of the sample.

This is calculated using the zscore function present in the python stats library.

After getting this score for all features of each image, I combined them to get a single score using different weights for different features.

I used 0.2 weight for the color moment vector, and 0.4 weight each for the HOG vector and ELBP vector . The color moment vector is given less weight so as to lessen the effect of lighting conditions when calculating the results. The ELBP feature can represent local features in the images and is monotonic against grayscale transformations, thus I decided to give it a weight of 0.4. The HOG descriptor focuses on the structure or the shape of an object. It is better than any edge descriptor as it uses magnitude as well as angle of the gradient to compute the features. For this reason, I gave it a weight of 0.4 too.

Results:

The results are saved in the Results folder in the zip file.

TASK 2:

For Task 2, I have saved the output in the output_task2.csv file. This contains the image ID, image path, and the feature vectors.

For Task 3 and Task 4 the outputs are saved in outputs_Task3.txt and outputs_Task4.txt respectively.

TASK 3:

Query 1: set1 image0.png cm8x8 4

The output shows the image-2 , image-6, image-7, image-8 to be the most similar to image-0. This is expected as the images have all the faces front-facing and so the color and shape of the image matches for all the windows of the image.

Query 2: set1 image0.png elbp 4

The output shows the image-7 , image-8, image-2, image-6 to be the most similar to image-0. This is expected as the images have all the faces front-facing and so the texture of the images matches.

Query 3: set1 image0.png hog 4

The output shows the image-7 , image-8, image-2, image-6 to be the most similar to image-0. This is expected as the images have all the faces front-facing and so the gradient orientation is the same for all of them.

Query 4: set2 image0.png cm8x8 4

The output shows the image-2 , image-4, image-1, image-6 to be the most similar to image-0. This is expected as the image-2 , image-4, image-1 are of the same person and so have the same color moment vector.

Query 5: set2 image0.png elbp 4

The output shows the image-2 , image-6, image-7, image-4 to be the most similar to image-0. This is expected as the image-2 is of the same person and also front facing. As other images of person in image 0 are not front facing we don't get them as similar images.

Query 6 : set2 image0.png hog 4
The output shows the image-2 , image-1, image-4, image-3 to be the most similar to image-0. This is expected as the image-2 , image-1, image-4, image-3 is of the same person.

Query 7 : set3 image0.png cm8x8 4
The output shows the image-70 , image-110, image-60, image-90to be the most similar to image-0. This is because all of the people have the same skin color shade and similar shape.

Query 8: set3 image0.png elbp 4
The output shows the image-70 , image-90, image-80, image-110 to be the most similar to image-0. This is because all of the people have similar face features and similar texture.

Query 9: set3 image0.png hog 4
The output shows the image-110 , image-130, image-100, image-120 to be the most similar to image-0. This is expected as the images have similar face structure.

TASK 4:

Query 1: set1 image0.png 4
The output shows the image-7 , image-2, image-6, image-8 to be the most similar to image-0. This is expected as all of them are front facing images.

Query 2: set2 image0.png 4
The output shows the image-2 , image-4, image-1, image-3 to be the most similar to image-0. This is expected as the image-2 , image-4, image-1, image-3 is of the same person.

Query 3: set3 image0.png 4
The output shows the image-110 , image-70, image-90, image-100 to be the most similar to image-0. This is expected as the image-110 , image-70, image-90, image-110 and can also be seen visually. These images have almost the same textures and color shades and are almost at the same angle.

Bibliography:

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_olivetti_faces.html
[2] https://en.wikipedia.org/wiki/Color_moments
[3] https://en.wikipedia.org/wiki/Local_binary_patterns
[4] https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients
[5] Mdakane, L and Van den Bergh, F. 2012. Extended local binary pattern features for improving settlement type classification of quickbird images. In: PRASA 2012: Twenty-Third Annual Symposium of the Pattern Recognition Association of South Africa, Pretoria, South Africa, 29-30 November 2012. Link
[6]N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177. Link
[7]