CSE 515: MULTIMEDIA AND WEB DATABASES
Project Phase 1

Krishna Chaitanya Bogavalli, Balavenkat Gottimukkala,Vatsala Janardhan, Vinay Matcha, Vinisha Sukameti, Ganesh Vanama

September 16, 2019

Abstract

In this phase of the project we experiment with image features obtained using feature descriptors, and similarity / distance measures with the goal of being able to familiarize with the storing and retrieving of multimedia data objects with focus of image processing. The dataset used was provided by *Mahmoud Afifi. 11K Hands: Gender recognition and biometric identification using a large dataset of handimages. M. Multimed Tools Appl (2019) 78: 20835* .This phase aims at feature extraction for images in dataset using feature descriptors like, Local Binary Pattern and Scale Invariant Feature Transform and finding similar images using the obtained features.

Keywords

Image Processing, feature descriptors, 11K Hands dataset, SIFT, Local Binary Pattern

1 Introduction

Image analysis is often crucial in computer vision and image processing applications. Images are often converted into different feature vector space for better understanding of patterns, edges, corners. This project aims in understanding the images features and their vector representations and find the similarities between images. These features can widely be used as a pre-processing step in building image search engines, deep learning and machine learning applications

1.1 Terminology

- *Feature Descriptors* : They encode interesting information into a series of numbers and act as a sort of numerical "fingerprint" that can be used to differentiate one feature from another. Ideally this information would be invariant under image transformation, so we can find the feature again even if the image is transformed in some way.
- *LBP* : Local Binary Patterns.
- *SIFT*:Scale Invariant Feature Transform.
- *Image gradient* : A directional change in the intensity or color in an image. The gradient of the image is one of the fundamental building blocks in image processing.
- *Histogram Of Gradients* : In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features.
- *Cosine Similarity* : Cosine similarity measures the orientation of two n-dimensional sample vectors irrespective to their magnitude. It is calculated by the dot product of two numeric vectors, and it is normalized by the product of the vector lengths, so that output values close to 1 indicate high similarity. Cosine Similarity between two points in a space is given by:

$$\text{Cosine Similarity(X,Y)} = cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||} \tag{1}$$

- *Euclidean distance* : Euclidean distance, p-2 norm, between 2 points in a space is given by:

$$\text{Euclidean distance(X, Y)} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

1.2 Problem Specification

1. **Task 1** : The goal is to get the feature descriptor/vector for a given image using any two of the two models i.e, Color Moments,Local binary patterns,Histograms of oriented gradients,Scale-invariant feature transform.

2. **Task 2** : Taking inputs as image ID,color model type extract feature vectors for all images in the dataset using a proper format for storage and retrieval of the same.

3. **Task 3** : For a given image and a value$k$ get the k similar images using the feature vectors obtained from the previous task.

1.3 Assumptions

- A dataset given has more than 11,000 hand images which is already proposed.
- There is no need to create any new feature.
- Images are not corrupted
- Paths are designed as per the local machine
- The features vectors are obtained using the predefined functions from python packages opencv and scikit-image.
- The similarity between images are obtained using implementation of any one distance/similarity metrics.

2 Implementation

2.1 Task 1

Objective of this task is to get the feature vector for a given image using Local Binary Patterns and Scale Invariant Feature Transform.

2.1.1 Local Binary Pattern

Local binary patterns are among the recent texture descriptors. The original LBP operator replaces the value of the pixels of an image with decimal numbers, which are called LBPs or LBP codes that encode the local structure around each pixel. Each central pixel is compared with its nearest neighbors (no of nearest neighbors = 8) with 2-hop radius; the neighbors having smaller value than that of the central pixel will have the bit 0, and the other neighbors having value equal to or greater than that of the central pixel will have the bit 1. For each given central pixel, one can generate a binary number that is obtained by concatenating all these binary bits in a clockwise manner, which starts from the one of its top-left neighbor. The resulting decimal value of the generated binary number replaces the central pixel value. The histogram of LBP labels (the frequency of occurrence of each code) calculated over a region or an image can be used as a texture descriptor of that image.Bin size for the histogram is specified by taking no of nearest-points+2 Implementation steps in LBP are as follows:

1. Divide image into 100*100 blocks.
2. Convert the actual image into Gray Scale image.
3. Calculate decimal value for each pixel. And find the histogram of bins for each window blocks of entire image and concatenate them to get unified feature descriptor. (No of bins of histogram is measured using no of nearest points + 2) For a given image,

the feature descriptors are obtained using the local-binary-pattern function under the scikit-image module in python.

2.1.2 Scale Invariant Feature Transform

Key points refer to the corners that are present in the image. These corners are certain points or pixels in an image that are invariant to any changes resulting in another image with the same object but with a change in property such as illumination, rotation.SIFT (Scalar Invariant Feature Transform) is a special type of algorithm where both key points and feature descriptors can be calculated. The SIFT implementation is used to detect key points as well as calculate the descriptors of those key points. Even though SIFT is computationally expensive yet is highly expressive than other descriptors and hence it is used for applications like tracking and recognition. SIFT gets its name from the Scale Invariant property where a set of convolutions are computed between the input image and increasing the size of Gaussian Kernels. Then the difference of these Gaussian Kernels is used for obtaining the descriptors. SIFT includes both a detector and a descriptor. The detector is based on the difference-of-Gaussians (DoG), which approximates the Laplacian. The DoG detector detects centres of blob-like structures. The SIFT descriptor is a based on a histogram of gradient orientations. The steps in SIFT are as follows:

1. Finding the keypoints.
2. Keypoint Matching : After obtaining the keypoints and the descriptors for both images, we need to find the best matched keypoints in both images. In order to obtain the best matches suitable matching algorithm is required. Some of the matching algorithms available in OpenCV are Brute Force and FLANN Matchers. In this phase we used FLANN Matcher. After matching, we will have several one-to-many matches among these key points. We use Lowes ratio to find best matches.

2.2 Task 2

Task2 aims to generate feature vector for the given descriptor LBP or SIFT for a folder of images. The features vectors for LBP were saved in csv format. The feature vectors for SIFT were saved in a pickle file. It is similar to implementing task1 on batch of images.

2.3 Task 3

Task 3 aims in finding the first k similar images to a given query images using the corresponding similarity measures. These features of the query images were compared with those of features in files saved from task2.

2.4 Visualization

3 Interface Specification

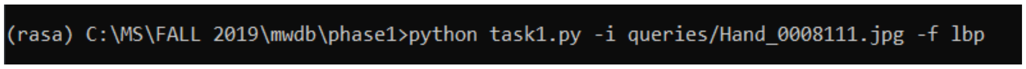
3.1 System and Data

This phase of the project has been implemented on both Windows and Linux OS and uses Python 3.6 as the programming language with code written and commented following PEP-8 standards. Code has been tested on machines with RAM of 8 GB and 16GB.

3.2 Query Specification

User can run each of the tasks as simple python scripts, with different options as command line arguments.Below are the sample command line executions for each tasks

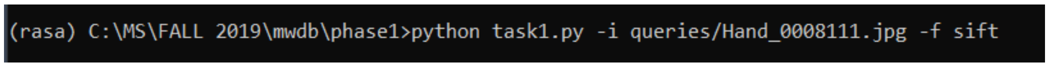
1. Task1: -i indicates the input image path, -f indicates the feature
   (a) LBP- python -i small_db/Hand_0008110.jpg -f ”lbp”



```
(rasa) C:\MS\FALL 2019\mwdb\phase1>python task1.py -i queries/Hand_0008111.jpg -f lbp
```

Figure 1: Query Specification Task-1

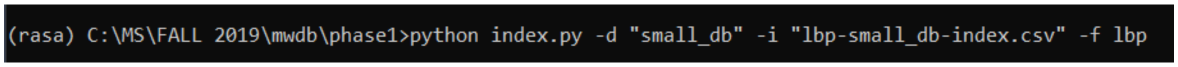
   (b) SIFT- python -i small_db/Hand_0008110.jpg -f ”sift”



```
(rasa) C:\MS\FALL 2019\mwdb\phase1>python task1.py -i queries/Hand_0008111.jpg -f sift
```

Figure 2: Query Specification Task-1

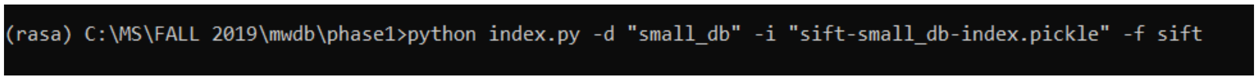
2. Task2. Runs through the folder of images specified by ”-d” and stores the features in a file specified by ”-i”. Descriptor is specified by ”-f”
   (a) LBP- python index.py -d ”small_db” -i ”lbp-small_db-index.csv” -f lbp



```
(rasa) C:\MS\FALL 2019\mwdb\phase1>python index.py -d "small_db" -i "lbp-small_db-index.csv" -f lbp
```

Figure 3: Query Specification Task-2

   (b) SIFT- python index.py -d ”small_db” -i ”sift-small_db-index.pickle” -f sift

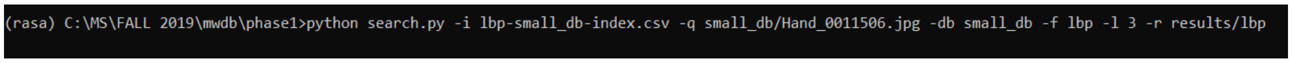


```
(rasa) C:\MS\FALL 2019\mwdb\phase1>python index.py -d "small_db" -i "sift-small_db-index.pickle" -f sift
```

Figure 4: Query Specification Task-2

3. Task3. Searches through the stored feature file specified by ”-i” for query image specified by ”-q”. Finds the k similar images specified by ”-l” in the folder specified by ”-db”. Descriptor is specified by ”-f”. Saves the similar images in the folder specified by ”-r”

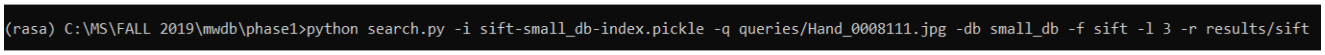
   (a) LBP- python search.py -i ”lbp-small_db-index.csv” -q queries/Hand_0008111.jpg -db small_db -f lbp -l 3 -r results/lbp



```
(rasa) C:\MS\FALL 2019\mwdb\phase1>python search.py -i lbp-small_db-index.csv -q small_db/Hand_0011506.jpg -db small_db -f lbp -l 3 -r results/lbp
```

Figure 5: Query Specification Task-3

   (b) SIFT- python search.py -i ”sift-small_db-index.pickle” -q queries/Hand_0000046.jpg -db small_db -f sift -l 3 -r results/sift



```
(rasa) C:\MS\FALL 2019\mwdb\phase1>python search.py -i sift-small_db-index.pickle -q queries/Hand_0008111.jpg -db small_db -f sift -l 3 -r results/sift
```

Figure 6: Query Specification Task-3

4 Installation and Execution

The system successfully runs on an windows and linux machines with Anaconda3. Programs leverage use of the pandas package for Dataframes, numpy for the arrays, to ensure the direct functional use of feature descriptors i.e,SIFT and LBP we used packages directly available from the opencv-python and scikit-image libraries respectively. Therefore, one needs to install the following packages:

```
conda create -n myenv python=3.6
conda activate myenv
conda install -c menpo opencv # for SIFT
conda install -c anaconda scipy
conda install -c anaconda numpy
conda install -c conda-forge scikit-image # LBP
```

Alternatively one can install these packages using pip. It is advised to install packages that are current and stable.The version of the packages may vary ,compatablilty should be considered.

5 Results

1. Task1

   (a) Local Binary Pattern

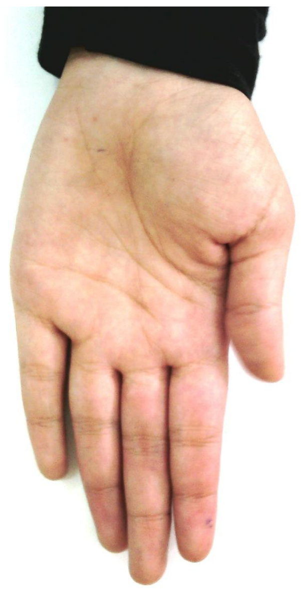


Figure 7: Input Image

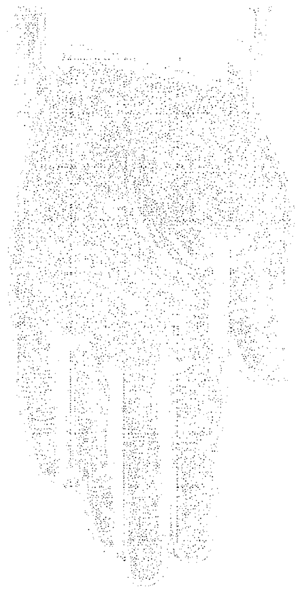


Figure 8: LBP code

(b) SIFT

Figure 9: Input Image

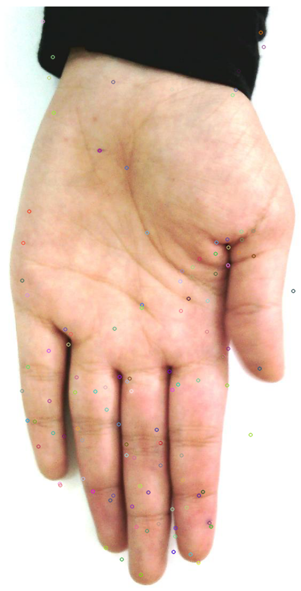


Figure 10: SIFT code

2. Task3 - 3 similar images for input image

(a) Local Binary Pattern

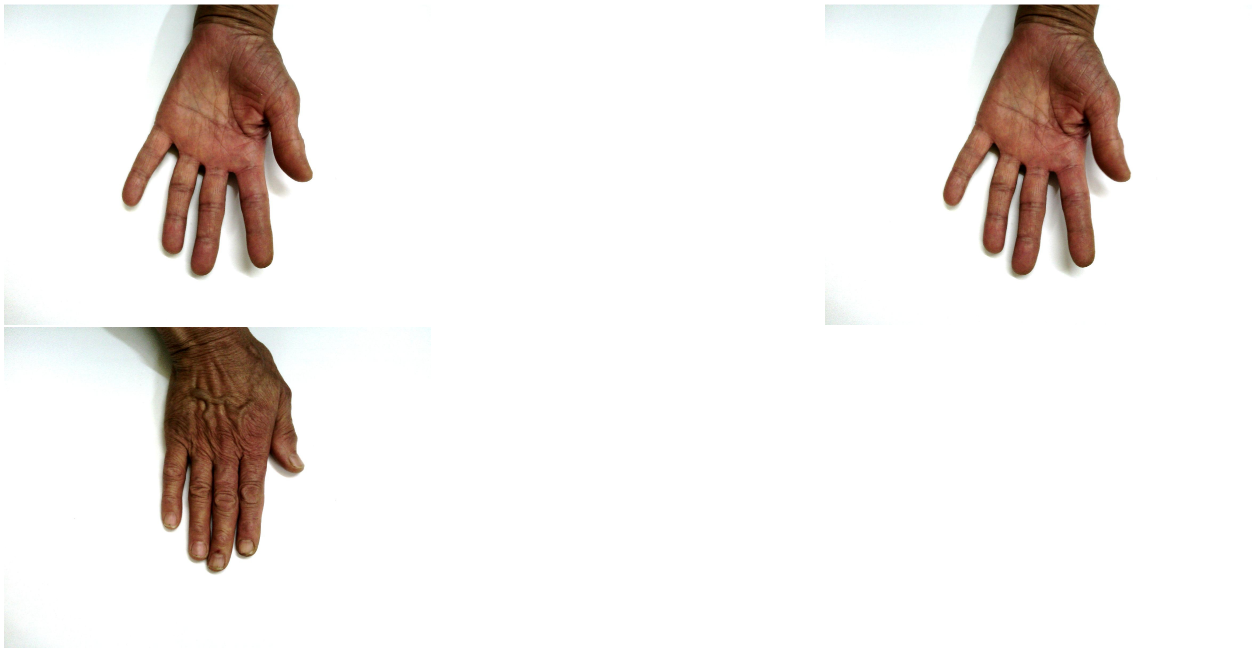


Figure 11: 3 Similar Images using LBP

(b) SIFT

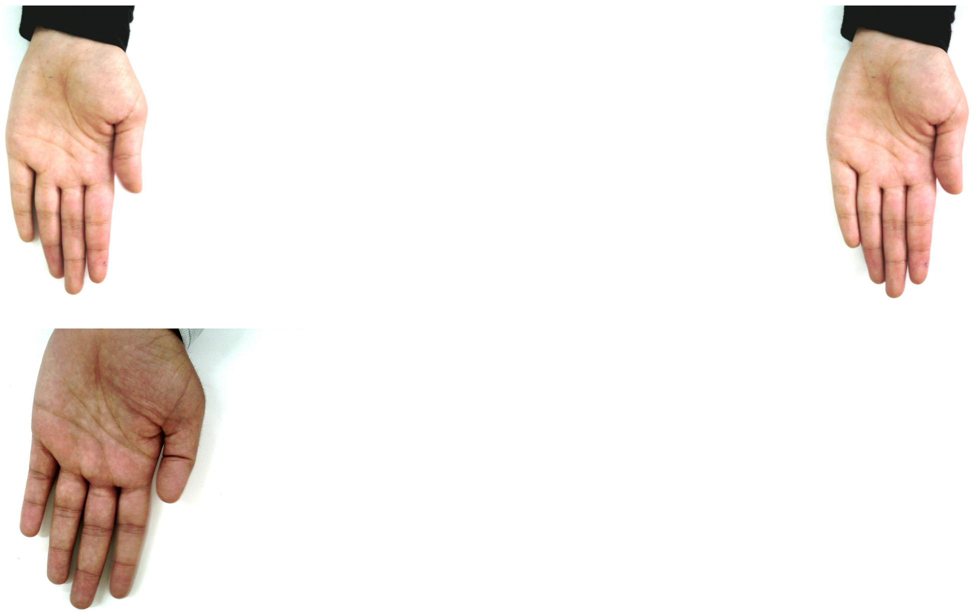


Figure 12: 3 similar images for SIFT

6 Conclusions

The intricacies of a dealing with vector based representation of multimedia objects was observed in areas spanning that can help retrieve results based on distance measures. Efficient representation of multimedia objects using feature vectors using feature descriptors-LBP and SIFT was implemented, storing the obtained results in a file system and to see how it helps the ease the process of comparison.The similarity metric used here may not give the best results now , but in further phases analysis of the same will be improved .For now, Chi square distance was used to measure similarity between LBP and FLANN Matcher was used to match the keypoints in SIFT.

References

[1] Mahmoud Afifi (2018) “11K Hands: Gender recognition and biometric identification using a large dataset of hand images”, arXiv:1711.04322, 17 (Spetember 2018), pp : 1–3

[2] Di Huang ; Caifeng Shan ; Mohsen Ardabilian ; Yunhong Wang ; Liming Chen (2011) “Local Binary Patterns and Its Application to Facial Image Analysis: A Survey”,Published in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 28 (March 2011), pp : 765–781

[3] D.G. Lowe.(2005), “Distinctive image features from scale-invariant keypoints”,Published in IJCV, 5 (January 2005), pp : 91–110

[4] Adrian Rosebrock, ”Local Binary Patterns with Python OpenCV”, Blog, https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

[5] Adrian Rosebrock, ”The complete guide to building an image search engine with Python and OpenCV”, Blog, https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/

7 Appendix

7.1 Specific roles of team members

Please see below how the models to be implemented was divided among the team :

1. Krishna Chaitanya Bogavalli : Color Moments & Local binary patterns
2. Balavenkat Gottimukkala : Histograms of oriented gradients & Scale-invariant feature transform
3. Vatsala Janardhan : Color Moments & Histograms of oriented gradients
4. Vinay Matcha : Histograms of oriented gradients & Scale-invariant feature transform
5. Vinisha Sukameti : Local binary patterns & Histograms of oriented gradients
6. Ganesh Vanama : Local binary patterns & Scale-invariant feature transform