CSE 515: Multimedia and Web Databases

Phase 1

**Fetching Similar Images Corresponding To Its Feature Descriptors Using Distance/Similarity Measure**

**Team Members:**

|  |  |  |
| --- | --- | --- |
| Arijit Panda | apanda7@asu.edu | 1217155056 |
| Mansi Panpaliya | mpanpali@asu.edu | 1217179197 |
| Asmi Bijaya Pattnaik | apattnai@asu.edu | 1217240531 |
| Rekha Yale Gnanaprakash | ryalegna@asu.edu | 1217158670 |
| Anantha Krishnan Kumar | akuma218@asu.edu | 1217112208 |
| Harika Kondru | hkondru@asu.edu | 1217032934 |

**Abstract:**

In this Big data world when people manually annotate images by giving metadata, it becomes time-consuming and may not even narrate the image. To address this issue, given an image this project deals with efficient retrieval of similar images from a large dataset. Our project team has utilized the image features, vector models (CM,LBP,SIFT,HOG), and similarity/distance measures to accomplish similar images successfully. The whole project is divided into 3 tasks which encompasses the overall concept.

**Keywords:** LBP, Color Moments, Feature Descriptor, Vector Model , Euclidean Distance

**Introduction**

Thanks to the growth of data, storing and recovery of multimedia like images can be challenging. If humans manually describe each image, textual data about images can be easily searched using traditional technologies. Even this becomes infeasible if the images are generated in a large database automatically e.g. surveillance cameras. There is also the danger of missing the images which use different synonyms in their metadata. Furthermore, comparisons of multimedia objects like images can't be imagined to be performed in a traditional manner.

Images have different features e.g color, texture, edge, shape which can be represented as vectors. The overall objective of the project is related to searching images in a large dataset and returning similar kind of images effectively. The hand image dataset provided for this project is (Mahmoud Afifi. 11K Hands: Gender recognition and biometric identification using a large dataset of hand images). The dataset contains different hand images in various orientations and gestures. Our task was to extract features and map them into a vector for each image. The search algorithms (similarity/distance measures) are applied to compare the images accordingly. Given a particular hand image, the algorithm will search the entire dataset and return the k similar images according to the distance/similarity measures. This method can be used to find matching multimedia objects based on the information given after applying search algorithms.

* **Terminologies:**
* Color Moments:

Color moments are the evaluation of the color distribution in an image e.g mean, variance, skew etc. It works well under different lighting conditions as it encodes shape and color. <https://en.wikipedia.org/wiki/Color_model>

* Local Binary Pattern:

It is a type of visual descriptor which is used for texture classification. <https://en.wikipedia.org/wiki/Local_binary_patterns>

* Euclidean Distance:

The straight line distance between points A and B is called Euclidean Distance (Here A and B are two different pixels) <https://en.wikipedia.org/wiki/Euclidean_distance>

* **Goal Descriptions:**

The object of the project is to find similar hand images by applying search algorithms on different models. Here we map the extracted features of the images into vectors on top of which search algorithms can be applied taking distance/similarity measures into account. The features of the images are calculated according to the vector model chosen. The distance/similarity functions are selected according to the accuracy and feasibility of the outputs we got.

* **Assumptions:**
  + Pre requisite knowledge of vector models and Python programming.
* Data provided is consistent in its form i.e images remain unchanged.
* The mode for LBP remains same throughout the time.

**Proposed Solution:**

The project consists of 3 tasks, where we are basically searching for k similar images when image, model and k are passed as arguments to the program. Each task can be independently performed and the inputs are also different for every task.

* **Task1:**

Here one of the vector models and an image id was given as user inputs and we had to accordingly show the feature descriptors of that image using the given model in a human-readable form. The procedures vary according to the different Color moments we are given. According to the requirements we have utilized different methods to convert the image to different color models and then extract the feature descriptors. The algorithm is provided below for this task.

* + For both Color Moments and LBP(Local Binary Patterns) read the given image into an image object.
  + For Color Moments convert the image into YUV model and split it into 100\*100 windows. In LBP convert the image into gray scale and then split it into 100\*100 windows. (Splitting is performed to avoid loss of data in the process)
  + Furthermore create a new directory and store these cropped images. This process is the same for both the models.
  + For Color Moments calculate the mean, variance and skew for each of the color channel for all the cropped images and store it an array. Additionally concatenate these features (append these elements to the array) and show it in a human readable form. (There will be 192\*9 values as the given hand images are of 1600\*1200 resolution)
  + In Case of LBP, LBP features are computed according to our supplied radius and number of points. LBP features are a 2D array with the same width and height as our input image and in turn not usable as a feature vector. To construct the actual feature vector, we take the help of histogram which counts the number of times each of the LBP prototypes appears. The returned histogram is number of points + 2-dimensional for each window. (Note radius is taken 1 and in turn 8 number of points are taken respectively for faster calculation and efficiency)
  + We are deleting the cropped images as we want all the tasks to be independent of each other.
* **Task2:**

One folder with images is provided as user input and we have to extract the feature descriptors and store it in a folder. We are extracting feature descriptors the way we did in task1. The difference here is that we are doing it for each image and storing the feature descriptors as a single row in a CSV file along with the image name. The algorithm is provided below for better understanding.

* For both Color Moments and LBP implement the task1 for each image in the dataset(folder) with the exception of not deleting the cropped image folder. For each image create one folder is then store all cropped images.
* In Color Moments after extracting feature descriptors(192\*9 values) for an image store these values in a 1D array and then write it into a CSV file along with its image name as 2 columns.
* In LBP construct feature vectors for all the images as done in task1 for one image. Similarly store it in a 1D array(192\*10 values) write it into a CSV file along with its image name as 2 columns.
* **Task3:**

Here the user provides the vector model to utilize, a particular image and a value k (k: number of similar images the user wants to checkout). The program performs task2 here and then show k similar images and also list the overall matching score. We used Euclidean Distance here to calculate the similarity between images . The algorithm for both Color Moments and LBP is same and provided below.

* Perform the task 2 first and store feature descriptors in a CSV file according to the vector model. According to the image provided compare the image name with the first column of the CSV file and get corresponding feature descriptors for that image.
* Then find visit the 2nd column of the CSV file to get the feature descriptors of other images and calculate the Euclidean distance. Furthermore the store the distance values and image names as tuples in a list.
* Sort the list according to the distance values and then show the k similar images along with the distance as an overall matching score.

**Interface Specifications:**

Command Line Interface is used to run all the programs. Pycharm is used to execute the programs. The program can be opened and then by pressing (ctrl+shift+f10) the program will be executed.

**System Requirements:**

The list of prerequisites for the project is provided below.

* Operating System: Windows 10
* Programming Languages : Python 3.7
* Python Libraries Used : os, PIL, cv2, numpy, csv, json, skipy, shutil
* Integrated Development Environment :Pycharm

**Installations:**

* **Install Python 3.7:**

For installing python please follow this link <https://www.python.org/downloads/>.

* **Install Pycharm:**

For installing Pycharm please follow this link <https://www.jetbrains.com/help/pycharm/installation-guide.html>

* **Install Python Libraries.**
* Download pip.py and Open a command prompt and navigate to the folder containing get-pip.py. Run the following command to instal

python get-pip.py

* Install all the necessary libraries like a similar command below in command prompt according to your requirement.

pip install PIL

**Execution Instructions:**

As the source code is written in Python and you already have python environment no installation or compilation is required.

* **Task1:**

It takes one of the vector models and an imageid as input like below.

Provide the model you want to work with lbp/cm: lbp

Tell us which task you want to perform: task1

Provide the image name: Hand\_0000002.jpg

* **Task2:**

It takes the folder name as an argument as below.

Provide the model you want to work with lbp/cm: lbp

Tell us which task you want to perform: task2

Enter the folder: E:\\LBP

* **Task3:**

It takes vector model, image name , number of similar images you want to show as mentioned below

Provide the model you want to work with lbp/cm: cm

Tell us which task you want to perform: task3

Provide the image name: Hand\_0000002.jpg

number of similar image ids you want to show: 3

**Related Work:**

For LBP [Adrian Rosebrock](https://www.pyimagesearch.com/author/adrian/) has worked regarding extraction of Local Binary Patterns from images and use them (along with a bit of machine learning) to perform texture and pattern recognition. <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>

**Conclusions:**

We have implemented all the tasks as specified in the report document. We designed all the tasks precisely and got to know the implementation of theoretical concepts taught in class. We had to optimize our code multiple times to improve the performance in order to efficiently execute for the large data set. We got more insights on theoretical concepts and practical execution of vector model. After facing several problems regarding storing LBP features we then put it in a 1D array. Extracting values and comparing from CSV files were challenging and we got to learn a lot about images and distance/similarity measures about actually how they work.

**Bibliography:**

* <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>(Author: [**Adrian Rosebrock**](https://www.pyimagesearch.com/author/adrian/)**)**
* <https://en.wikipedia.org/wiki/Color_model>
* <https://en.wikipedia.org/wiki/Local_binary_patterns>
* <https://en.wikipedia.org/wiki/Euclidean_distance>
* <https://sites.google.com/view/11khands> (Author: **Mahmoud Afifi**)

**Appendix:**

We have worked as a group on all tasks sharing our conceptual ideas throughout this phase and helping each other in understanding every task. On higher level, below are the details of tasks we divided among ourselves, but not tightly bound as almost everyone was involved in every task.

|  |  |  |
| --- | --- | --- |
| S No. | Task | Team Member |
| 1. | Research on libraries to be used | All |
| 2. | Project Report | All |
| 3. | Task1, Task2, Task3 | All |
| 4. | Testing | All |