Project Report Phase 2:

Jiaxuan Pang
Kuan-Ting Shen
Bandon Tseng
Ching-Wen Tu
Ting Xia
Ming Yi

Abstract

In this phase of the project, we worked on, implemented, and experimented on four feature dimension reduction and matrix factorization methods. Principal component analysis (PCA), singular value decomposition (SVD),non-negative matrix factorization (NMF), and latent dirichlet analysis (LDA), all of which, are combined with four feature extraction methods CM, HOG, LBP, and SIFT from phase 1. At this phase, we also used different similarity metrics to calculate similarity between images under different latent features. Our implementations are based on 11K hands dataset, which consists the hand images in different shape, light, contract and skin color [1]. It also consists an annotated file with meta-data, including left or right hand, dorsal or palmar, with or without accessories, and male hand or female hand. We implemented new demo program, which combines demo program from phase to extract features using CM, HOG, LBP, and SIFT. And further applying them on the four feature dimension reduction and matrix factorization methods proposed previously.

Keywords— Image Retrieval, SIFT, CM, LBP, HOG, PCA, SVD, NMF, LDA, Image Processing, Image Searching, Image Similarity, Multimedia

1 Introduction

Image retrieval is a term for browsing, searching and retrieving images from a larges database of digital images. By adding metadata to images such as title or descriptions, the traditional image retrieval can be performed over these annotate data [5]. However, manually annotate data is considered time consuming; as a result; finding a way to automatically extract image features and clustering similar image becomes important. In the previous phase, we implemented and experimented on four feature extraction algorithms, which are Color Moment(CM), Local binary patterns (LBP), Histograms of oriented gradients(HOG), and Scale-invariant feature transform(SIFT). We compared the performance, and complexity of different algorithms. However, to further discover deeper latent semantic under different feature representations and to reduce the computation complexity, different feature dimension reduction methods should be involved. In this phase, we first proposed our methods and implementations of combining 4 feature extraction methods with 4 dimension reduction methods. To exam the performance of different combination, We further proposed and implemented measurement methods to calculate the similarity between images under different latent semantics, with and without binary image meta-data information.

1.1 Terminology

It is well known that the image data can be represented by different color models. For example, the RGB color model describes colors in terms of combinations of the intensities of Red, Green, and Blue colors, and it's popularly used in displaying digital image. The YUV color model, which can be transformed to and from RGB color model, is good for compressing. In this phase, each image are converted to the YUV color model, and fed to one of four feature extraction algorithms from phase 1. There are certain technique needed to further reduce the feature variables and let us focus on most important ones. There are two majority ways of reducing the dimensional, **feature elimination** and **feature extraction/feature dimension reduction**. Unsurprisingly, **feature elimination** will be easiest way to reduce the feature dimension by deleting features. However, the disadvantages is significant, the variables being dropped will definitely contribute no information. **feature extraction/feature dimension reduction** addresses the problem that feature elimination exposed. Intuitively speaking, the feature extraction/feature dimension reduction algorithms create new independent feature variables that being mapped by the original feature variables. Principal component analysis(PCA) is one of representative feature dimension reduction algorithm [2]. Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's usually used to make data easy to explore and visualize. Singular value decomposition(SVD) is another feature dimension reduction technique. Its key concept is matrix decomposition. It describes a given matrix using its constituent elements. The general formalize of SVD is $M = U\Sigma V^T$, which M input matrix, Σ is singular values and is diagonal. Latent dirichlet analysis(LDA) is a generative probabilistic model of a collection of composites made up of parts. "Its uses include Natural Language Processing (NLP) and topic modelling among others" said by [3]. Since LDA is mostly used for Natural Language Processing, its input is particularly in the bag-of-word format, which represents the count of the words. However how to adapt the image data with LDA algorithm is a challenge, we will discuss in detail in the following chapters. The last dimension reduction methods we implemented and experimented is Non-negative ma-

trix factorization(NMF). NMF is also another matrix factorization method such that the input matrix should be non-negative. The input matrix X should be factorized into two matrices W and H so that $X \approx WH$, which W is the weight vector and H is the component vector. Bringing these basic concepts, in next chapter, we will describe the problem specification and the goal.

1.2 Goal description (problem specification)

Use different image feature dimension reduction methods include principal component analysis, singular value decomposition, non-negative matrix factorization and latent dirichlet analysis to calculate and analyze the feature descriptors extracted by different feature extraction algorithms like color moment, local binary patterns, histogram of gradient and scare-invariant feature transform to analyze and retrieval images. In order to do so, we need to implement a program which not only adapt implementations from phase 1, but also be able to easily expand for current phase. In addition, an appropriate level of program interface and inherently is required for group development.

For task 1, we use one of the feature dimension reduction methods to deal with the feature descriptor extracted by one of the four feature models we choose, get corresponding vector space and identify the top k latent semantics. Due to the fact that each dimension reduction method expects inputs with different semantic,the main challenge is to decide the format of feature models, and how to transform the input matrix to be the most meaningful.

For task 2, the system will identifies the most related m images in the given k-dimension latent space after the user choose one feature model and one dimension reduction technique. The most challenge is to find the most relevant k latent semantics and workout a methods to measure distance/similarity between latent components

For task 3, Similar to task 1, given an image label, we need to use one dimension reduction method to identify k latent semantics for images with the corresponding metadata after the user choose one feature extraction model.

For task 4, for the given label, identify the most related m images using the given latent semantics associated with the given label after the user choose one feature model and one dimension reduction technique.

For task 5, label an unlabeled image ID by comparing and matching with the given k latent semantics associated with the given labels by using one feature dimension reduction technique and one feature extraction model.

For task 6, given a subject ID, using feature model, dimension reduction method and latent semantics to find the most related 3 subjects. The first challenge is to generate subject-image, and image-subject corresponding dictionaries. Secondly, it's important to select good combination of feature model and dimension reduction methods. In addition, to make sure the similarly is calculable, the number of latent component(k) should be selected to fit both subject-image dataset.

For task 7, We shall follow similar stategе as task 6 to construct subject-subject similarity matrix. The matrix should be symmetrical. Then we need to perform NMF on a subject-subject similarity matrix and reports the top k latent semantics in a subject-weight pairs in decreasing order.

For task 8, performs NMF on a binary image-metadata matrix and reports top k latent semantics in the image-space and top k latent semantics in the metadata-space in decreasing order of weights.

1.3 Assumptions

A list of assumptions must be made to satisfies the algorithm requirements and to allow program runs smoothly.

1. Before running phase 2, the feature models should be extracted by either CM, HOG, SIFT, and LBP and stored to database(Local Drive) by calling "p1task2.py" and pass set of parameters.
2. The input of any dimension reduction methods should be $n * m$ matrix. SIFT feature should be further processed to satisfy the requirement(See next section).
3. For task 3 to 8. The image name in given dataset should be contained by metadata file. And the given image file name should also be contained by metadata file and the given dataset.
4. For task 6, the given subject Id should be in the metadata file, and the corresponding images belongs to the subject id should be also in the data set.
5. We support only **csv** format dataset.

2 Description of the proposed solution and implementation

As per project specifications, the very first key challenge is to combine each of four feature models with different dimension reduction algorithm. The input of all four dimension reduction algorithms in general is the subject-feature matrices. As the result, for CM, HOG, and LBP models, it's easy to obtain subject-feature matrix, because for different images, the dimension of feature models extracted by CM, HOG, and LBP are the same. We first flatten the feature model for each image(subject). After flatting each image's(subject) feature model, we can get a $1 \times n$ vector. Then we stack all flattened image feature models to build the subject-feature matrix. However, for the SIFT feature models, it requires more engineering to satisfy the subject-feature matrix requirement, due to the reason that SIFT algorithm may extract different number of key points for different images, they cannot simply be concatenated and stacked. The methods we used to address this issue will be described first in the following chapter.

We need to further process the subject-feature matrices we obtained, since each of them may have different specification. Both Principal Component Analysis and Singular Value Decomposition don't have the preference of input matrix as long as they're subject-feature matrix. Since Latent Dirichlet Analysis is originally applied on the bag-of-word data, the input matrix is supposed to be word sequence matrix. To further apply the LDA on some particular feature models, some subject-feature models should be further processed. In additional, as shown in the name, the Non-negative matrix factorization(NMF) requires non-negative number in its input matrix, we need to further process that as well. Both of them will be discussed in following chapter.

2.1 Pre-process on SIFT Feature Model

To construct subject-feature matrix from SIFT feature model, and maximize preservation of key point position and descriptor information, we decide to partition the image similar as color moment. Note here that the descriptor information is encoded by a

1×128 vector. For each image we split the image into 100×100 window, meaning for an image with the size of 1200×1600, we can obtain 12×16 windows. For each key point, we can find which particular window it lays in, we then assign the corresponding descriptor value to that window. We took mean of the descriptors if there exits more than one key points in one window, and we set a zero descriptor if there has not key point belong to the window. After flatting the matrix constructed previously, we can obtain a $1 \times 12 \times 16 \times 128$ vector. Each image's SIFT feature model in the dataset will be processed in the same way. Finally, the subject-feature matrix can be obtained by stacking all of the vectors. This method not only maximize the position information for key point, but also not loosing any description information of the key points.

2.2 Pre-process on CM Feature Model for LDA method

Due to the special usage of LDA algorithm, its input matrix is supposed to be the sequential information instead of float numbers or even negative number. The feature model extracted by CM definitely does not satisfy the input requirement since it not only contains variance of float numbers but also negative numbers. To overcome the issue, we further processed the CM subject-feature matrix, and let it contains the value of occurrence in a certain range. Recall the CM feature model obtained from task 1, each image with size of 1200×1600 is divided into 12×16 window, and each window consists size of 9 vectors, containing, mean, standard deviation, and skewness for all Y,U,V channels. The values mean, standard deviation, and skewness are having really large range. We first normalized the value. The value normalization mean, standard deviation, and skewness under different Y,U,V channels are the same; we only write the example for mean under one channel. For the set of mean value $\mathbf{X}$ in each each image under one channel, let x represents the value of one window before normalization, and x' represents the value of one window after normalization. We calculate the x' by

$$x' = \frac{x - min(\mathbf{X})}{max(\mathbf{X}) - min(\mathbf{X})}$$

The normalization will normalize all values between 0 to 1. We divide 0 to 1 into 10 bucket with step of 0.1, and we will simply count how many windows' value lay in one of these 10 ranges. Finally, for each image, we will obtain a 1×90 vectors, that contains range counts between mean, standard deviation, and skewness under Y, U, V channels.

2.3 Pre-process on CM Feature Model for NMF method

NMF can only process matrix which includes all elements are non-negative. We, therefore, have to ensure this assumption can be satisfied every time when we use NMF. Color moments is a special case among all model we implemented in this project. Color moments in our project computes each color channel into three moments: mean, standard deviation, skewness. Since the color channel are all positive, it is impossible to have negative mean value, so as standard deviation. The skewness value, however, can be negative if the value of pixel less than mean. So that we ignore skewness values before we using NMF with CM. The vector size of CM for other dimension reduction method is $12 \times 16 \times 3 \times 3 = 1728$ - split an image into 12×16 with 3 channel(YVU), compute three moments for each channel. After pre-processed, the vector size of CM would be $12 \times 16 \times 3 \times 2 = 1152$ - utilize two moments for each channel instead.

2.4 For each task, specify how to achieve the goal

2.4.1 Task 1

In this task, given image descriptors, we need to identify the top k latent semantics in the corresponding latent space.

For the feature extraction models, we have already define and test in phase 1, so for this task, we need to define and complete the four feature dimension reduction technique from text description into real code.

For Principal component analysis(PCA), Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's usually used to make data easy to explore and visualize. For a given image, we used various feature models to extract the feature descriptor and form a matrix which can be processed using dimension reduction methods to extract the latent semantics. If using principle component analysis method, we first standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis. Mathematically, this can be done by substracting the mean and dividing by the standard deviation for each value of each variable. Next, we compute the covariance matrix to identify the correlations of the variables with each other. The covariance matrix is a pxp symmetric matrix (where p is the number of dimensions) that has as entries the covanriance associated with all possible pairs of the initial variables. Next, we compute the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors of the covariance matrix are actually the directions of the axes where there is the most variance and that we call Principal components. And the eigenvalues are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each principal component. By ranking eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

For Singular value decomposition(SVD) , Its key concept is matrix decomposition. It describes a given matrix using its constituent elements. The general formalize of SVD is $M = U\Sigma V^T$, which M input matrix, Σ is singular values and is diagonal. U and V are orthogonal matrices. The columns of U are called the left-singular vectors of M, while the columns of V are the right singular vectors.The values along the diagonal of Σ are the singular values of M. The matrix U, Σ, and V can be found by transforming M in a square matrix and by computing the eigenvectors of this square matrix. The square matrix can be obtained by multiplying the matrix M by its its transpose. U corresponds to the eigenvectors of MM^T. V corresponds to the eigenvectors of M^TM. D corresponds to the eigenvalues MM^T. The singular values are ordered by descending order. They correspond to a new set of features (that are a linear combination of the original features) with the first feature explaining most of the variance.

For Latent dirichlet analysis(LDA) , the feature descriptors are preprocessed to form a matrix of data which represents the occurrences of each feature. Each image is represented as a bag of words, where each word is a segment. LDA model can be used to learn the inter-segment and inter-image relationships and cluster images as different categories. After inputting the data into LDA model and specify the number of topics, the model will group each image into different categories as probability.

For Non-negative matrix factorization(NMF), The input matrix X should be factorized into two matrices W and H so that $X \approx WH$, which W is the weight vector and H is the component vector. Basically, we can interpret xi to be a weighted sum of some components, where each row in H is a component, and each row in W contains the weights of each component. When multiplying matrices, the dimensions of the

factor matrices may be significantly lower than those of the product matrix and it is this property that forms the basis of NMF. NMF generates factors with significantly reduced dimensions compared to the original matrix. For example, if X is an m n matrix, W is an m p matrix, and H is a p n matrix then p can be significantly less than both m and n. In order to conduct NMF we formalize an objective function and iteratively optimize it. NMF is an NP-hard problem in general, so we will aim for a good local minimal. A generally used measures of distance is the frobenius norm (the sum of element-wise squared errors). Formalizing this, we obtain the following objective: minimize $||X - WH||^2$ with W, H ¿=0. There are a couple of optimization methods like multiplication update, hierarchical alternating least squares.

2.4.2 Task 2

Task 2 is built on top of task 1. After getting the latent semantics, we project descriptors into the latent space. After that, the distances between the query image and all the other images are measured by L2-norm. Therefore, scores in the results are Euclidean distances.

2.4.3 Task 3

The task 3 is very similar to task 1 except that the images are filtered by their labels before decomposition.

2.4.4 Task 4

Like task 3, we add the filtering process in task 2 to create the task 4.

2.4.5 Task 5

In this task, we need to use the given k latent semantics with the provided label to decide the tag for the unlabeled image. We tried several approach to achieve the better label prediction of the given unlabeled image.

In the beginning, we wanted to project the unlabeled image to new space by the latent semantics and compare it with labeled images space to determine if the unlabeled and the labeled images are in a same cluster. Since we know the dimension reduction methods would reduce the dimension of the original image space to a new space by latent semantic. We can also project a new image to the new space. We would like to calculate the distance of the unlabeled image feature with the reduced image space. For example, if we have 200 image with CM feature model, the original image space matrix would be 200×1728 - 200 images and 1728 features. After doing SVD decomposition with $k = 5$, the new reduced image space would be 200×5. Now, we have an unlabeled image and would like to project to the new space, the projected vector would be 1×5. We calculated the distance between the unlabeled projected vector - the dimension is 1×5 - to each row of the reduced image space - 200×5. If there were some pattern in the distances, for example the distance in same group would much lower then in another group, we could determine whether the unlabeled image had same label. But we found this method could not help us to determine whether the image was in the same group. On other hand, we also learned from piazza that we have no access to the image-space information. We, therefore, did not continue to explore this approach.

For the given label's latent semantics, we know the matrix size is $k \times m$, which k is the top k latent semantics and m is the number of features which provided by the feature extraction model that we selected. Now, we also use the same feature extraction model for the unlabeled image and flatten the features that we extract by the model. Then, we get the $1 \times m$ features for the unlabeled image. We discover that both the latent semantics and the unlabeled image have m features. Thus, we calculate the average of the top k latent semantics to represent the center of the latent-feature space. After we conduct the average of the latent space, we gain the $1 \times m$ matrix which is the same as the unlabeled image. We use $cosine - similarity$ to calculate the similarity between the unlabeled image and the average latent space. If the similarity score is over 0.5, we marked the unlabeled image as the same label with the given label. On the other hand, if the similarity score is lower to 0.5, we marked it with the opposite label of the given label.

Even though we have tried several methods to discover the opt method to decide the label for the unlabeled image, we still can not have the higher accuracy for predicting the label. We still need other improvement for our method and maybe we can use better combination of the latent semantics which can help us to have a better prediction of the image labeling.

2.4.6 Task 6

For this task, instead of directly finding image similarity, we need to find subject similarity under latent semantics. To maximal exposing the statistical information and avoid pattern observation, we choose CM feature model combine with LDA to extract the latent semantic. In the first step, given subject ID, with help of meta data information, we first locate the image set belong to subject ID. Note here that the given small dataset may not fully contain all images in the image set or it contains no image in the image set. If it falls in the second case, the program will be terminated with an error. The second step is go over rest of image formations in the given small dataset, with the help of metadata information, they can be grouped by subject ID. Similar to previous tasks, once we obtain a group of images, the CM algorithm could be easily applied and construct subject-feature matrix for each subject image group. To be able to calculate the similarity among latent semantic of target subject and other subject's latent semantics, the value of k, which is the number of component must be determined. In this case, the k is dynamically selected by the smallest number of image in either target subject image group or other subject image groups. We then used the smallest k to apply LDA obtaining all the latent semantics. To obtain the single similarity value, each of obtained latent semantic are flattened. Since the both flattened the latent semantic of the target subject and other subject's latent semantic have uniform length, we simply apply dot product $sim_i = S_{target} \cdot S_i^T$ to obtain unnormalized similarity value.

2.4.7 Task 7

The first part of achievement for this task is similar to task 7, we choose CM feature model combine with LDA to extract the latent semantic for building subject-subject similarity matrix. With help of metadata information, we first go over the given small dataset and group all images by the subject ID they belong to. The smallest k is obtained by smallest number of image amount all subject groups. We apply CM and LDA with k value sequentially to construct latent semantics which have

the same dimension. We obtain the subject to subject un-normalized similarity by applying dot product $sim_i = S_{target} \cdot S_i^T$, and build subject-subject similarity matrix among these value. In the final step, we apply NMF algorithm on the subject-subject similarity matrix with user provided k_{prov} value, and report obtained latent semantic components.

2.4.8 Task 8

In this task, we are performing NMF dimension reduction on a matrix containing image-space and metadata-space. For the former space, we extract the features from the image by using the models built in the previous tasks. To obtain the metadata-space, we begin to read the metadata file. However, it is evident to notice that data in the file is not in numeral, which leads us to find an encoding method to transform the data. Thus, we adopt one-hot encoding to represent each attribute in 0 and 1 form and build the flatten array for each row in the file. The size and format of the one-hot vector would be 1×8 and the following:

[left-hand, right-hand, dorsal, palmar, accessories, no-accessories, male, female]

For example, if an image is labeled as left-hand, palmar, with accessories, and female. The ont-hot vector of this image would be:

$$[1, 0, 0, 1, 1, 0, 0, 1]$$

We then combine both image-space and metadata-space into one array and apply NMF algorithms on this matrix with k value provided by the user. At last, we report top-k latent semantics for image and metadata space in a consecutive way.

2.5 Document-Term Matrix

Because the dimension reduction methods we used in this phase require different types of input. For example, the LDA is used to find the weights according to the frequency of key words among documents. On the other hand, the input of SVD is a document-term matrix. We, therefore, have to convert each model to the corresponding format which the dimension reduction methods can handle with.

In $document - term$ matrix, each row represent a document whose features are represented in each columns. The features of each model, however, are represented as matrix. Take color moments as an example, we split the image into several 100×100 windows. The feature matrix would has 12 row and 16 columns if the resolution of the image is 1200×1600. However, each cell contains 9 channels which including *mean*, *std* and *skewness* for YUV channel. We therefore have to convert the matrix to a vector. After that, we combine vectors of different documents as a matrix so that we could have a document-term matrix.

3 Interface specifications (with focus on the goal and problem specification)

There are several parts of our program includes **module**, **model** and each task. Each functions in module and models folder can work independently.

3.1 models

In the **models** folder, we implemented all four models, Color Moments, Local Binary Patterns, histogram of oriented gradients, and SIFT. We also introduced **modelFactory** in this phase to let us get model more easily.

- **modelFactory.py** - In this module, user can get all supported models' name by method **getSupportModel()** and get a model instance by **creatModel()** with a *str* type argument which is the name of the model, **createModel("sift")** for example.
- **model.py** In the the modules, we firstly craft an interface which is an abstract class for other implementations -**cm.py**, **lbp.py**, **hog.py**, and **sift.py**. In those models, we implemented some common interfaces for our tasks.
  - **extractFeatures()** - Extract features from an image. This function will split image to smaller windows if necessary.
  - **serializeFeatures()** - As we mentioned in the database module, not all data structure can be serialized by pickle. Therefore, the feature must know how to convert their data structure to a data format which can be used for python pickle.
  - **deserializeFeature()** - The load function of python pickle returns a data structure which is output of the **serializeFeatures** function. But the data may not be used directly, so this function will deserialize to the format which can be used as a feature to other functions.
  - **flattenFeature()** - Since the feature extract by **extractFeatures()** is a matrix, to be used as document-term vector, we need to flat the feature matrix as an one-dimensional vector. Each module would implement their method to flat the feature matrix according to the value of their matrix or some restriction. For example, the color moment need to ignore the third moment, skewness, when the dimension reduction method is non-negative matrix factorization(NMF). This function returns an one-dimensional numpy array.

3.2 module

We implemented several modules in **modules** folder for dimension reduction, distance functions, metadata reader, database and image reader.

- **DimRed.py** - We implemented SVD, PCA, LDA, and NMF in this file. There are also some functions for convenience such as $registerMethod()$, $getSupportedMethods()$ and $createReduction()$.
  - **DimRed.registerMethod()** - This function is used to register a new method to our program automatically.
  - **DimRed.getSupportedMethods()** - This function return a **list** of supported methods. Each supported method would be store as lower case **str**: $["svd", "pca", "lda", "nmf"]$
  - **DimRed.createReduction()** - We can get a dimension reduction method instance by this method. the argument would be a str which indicates what method we wanted, **createReduction("svd")** for example.

  - **printLatentSemantics()** - This function is used to print the image according to the latent semantics.
  - **transform()** - We can use this function to transform a vector or matrix to the new space.
  - **getLatentSemantics()** - This function returns the latent semantic. We use this latent semantic function to estimate the similarity of each subject in $task6$ and $task7$.

- **database.py** - The database module includes an abstract class, Database. When initialize a database, we have to assign which table we would like to access. The following is the interfaces of a database.
  - **addData()** - Add data to the table with a given key. The existed data can be overwritten if we set overwrite parameter.
  - **getData()** - Get data from the table for a given key. Return None if the data is not available in the database.
  - **keys()** - Get all keys in the table. This could be utilized for traversal the table.

  The **FilesystemDatabase** is an implementation of the abstract class Database. **FilesystemDatabase** stores whole DB in **FileDB** folder. We can also specify the table name when we initialize the database and the table would be a folder inside the **FileDB**. **FilesystemDatabase** basically convert the data into python **pickle** format, which is a built-in python library, and save it as a binary file that filename is key ID we given such as **Hand_0001102.pkl**. To get data from a table of database, the **FilesystemDatabase** will try to find the file in the folder and load the pickle file. The pickle, however, is unable to convert some data formats, such as **KeyPoints** of OpenCV, directly. The **FilesystemDatabase** is not responsible for such situation. We, therefore, have to make sure the input data is usable for pickle.
- **distanceFunction.py** - Regarding the distance estimation, distance module includes classes for calculating distance. We implemented a P-Norm distance as **Norm** and entropy function. We have to pass the order of P-norm when we initialize it. Using it by calling the class instance with two vectors to get the distance. We use this module in **task 2**, **task 4**, **task 5**, **task 6** and **task 7** to calculate the distance among reduced features.
- **handMetadataParser.py** - Because we need information in the matadata for **task 3**, **task 4**, **task 5**, **task 6**, **task 7**. This module is used for metadata parsing. We only support **.csv** format metadata in this project.
  - **getFilelistByLabel()** - This function can be used to get file list of specific label such as left-hand or male. It returns a list of image IDs in the list. This function can be utilized for **task 3**, **task 4**, **task 5**.
  - **getFilelistByID()** - We compare the similarity of subjects for **task 6** and **task 7**. We, therefore, implemented this function for getting the list of image IDs by subject ID. It returns two dictionaries - the key and value of first one are subject ID and list of image IDs, respectively, the second one includes image ID as key and subject ID as value.
  - **getFilelistWithOneHot()** - Since we need one-hot vector for the metadata space in **task 8**, this function generates one-hot vector according to the metadata.

3.3 tasks

The task files, p1task2.py, p2task1.py ... etc, are the program for each task of phase 1 and phase 2. We introduce the execution instructions of these programs in the following section.

4 System requirements and installation and execution instructions

In this phase of project, we use python as our language. To do dimension reduction, we utilize several libraries such as $Numpy$ and $scikit-learn$.

4.1 Environment Setup

Anaconda is a distribution of *Python* and *R* which aims to simplify package management mainly for scientific computation. Based on the operating system on the computer, the corresponding version can be easily downloaded from the website. After the installation, we can create our own clean environment.

To create a new environment:

```
conda create -n <environment name> python=3.7
```

We then activate the environment:

```
conda activate <environment name>
```

As the new environment is initialized, we begin to install our necessary packages. The packages installed are listed below

- **OpenCV** - An open-source library that support plenty of computer vision algorithms, such as extracting SIFT or HOG features from images
- **Numpy** - A package providing high-performance multidimensional array object
- **Scikit-learn** - An open-source library providing efficient tools for data-mining and data analysis
- **Scikit-image** - An image processing toolbox for SciPy, such as extracting LBP features from the image
- **Matplotlib** - A data visualization library built on NumPy arrays and designed to work with SciPy

To make installation process simple, all the packages are written in *requirements.txt* and can be installed with following command:

```
conda install --yes --file requirements.txt
```

4.2 Execution instructions

In the phase 1 project, we implemented database module to store image features of each model. Loading image features from database can save us a lot of computation time. Therefore, before we begin executing phase 2 project, running the task 2 of phase 1 for building database is necessary.

- All tasks There are some arguments are shared among all tasks. Including the following:
  - **-m model** - The model we would like to use. We support the following models.
    - **cm** - Color Moments
    - **sift** - Scale-invariant feature transform
    - **lbp** - Local binary patterns
    - **hog** - Histogram of oriented gradients
  - **-t table** - The name that has been used when creating descriptor database.
  - **-k k**: The number of latent semantics.
  - **-d method** - Decomposition method. We support the following decomposition methods.
    - **svd** - Singular value decomposition
    - **pca** - Principal component analysis
    - **lda** - Latent Dirichlet allocation
    - **nmf** - Non-negative matrix factorization
  - **-mi number** - The number of the query results.
  - **-meta metadata** - The path of metadata(HandInfo.csv).
  - **-l label** - The label used to find corresponding labeled images.
    - **l** - Left-Hand
    - **r** - Right-Hand
    - **d** - Dorsal
    - **p** - Palmar
    - **a** - With-accessories
    - **n** - Without-accessories
    - **m** - Male
    - **f** - female
- Phase 1 task 1
  - **-i** - The dataset folder path.
  - **-m** - The model we would like to use. We support the following models.
  - **-t** - The table name which would be created in database.

  Example:
  `python p1task2.py -i Dataset2 -t xtest -m cm`
- Phase 2 task 1
  - **-i** - The dataset folder path.
  - **-m** - The model we would like to use.
  - **-t** - The table name which would be created in database.

  Example:
  `python p2task1.py -m cm -t test -k 5 -d svd -p Dataset2`

- Phase 2 task 2
  - **-m** - The model we would like to use.
  - **-t** - The table name which would be created in database.
  - **-k** - The number of latent semantics.
  - **-d** - Decomposition method.
  - **-i** - The target image ID.
  - **-mi** - The number of the query results.
  - **-p** - The path of the dataset.

  Example:
  `python p2task2.py -m cm -t test -k 10 -d svd -i Hand_0008110 -mi 10 -p Dataset2`
- Phase 2 task 3
  - **-m** - The model we would like to use.
  - **-t** - The table name which would be created in database.
  - **-k** - The number of latent semantics.
  - **-p** - The path of the dataset.
  - **-d** - Decomposition method.
  - **-meta** - The path of metadata(HandInfo.csv).
  - **-l** - The label used to find corresponding labeled images.

  Example:
  `python p2task3.py -m cm -t test -k 5 -p Dataset2 -d svd -meta HandInfo.csv -l l`
- Phase 2 task 4
  - **-m** - The model we would like to use.
  - **-t** - The table name which would be created in database.
  - **-k** - The number of latent semantics.
  - **-d** - Decomposition method.
  - **-i** - The target image ID.
  - **-mi** - The number of the query results.
  - **-p** - The path of the dataset.
  - **-l** - The label used to find corresponding labeled images.

  Example:
  `python p2task4.py -m cm -t test -k 5 -d svd -mi 5 -p Dataset2 -i Hand_0008110 -meta HandInfo.csv -l l`
- Phase 2 task 5
  - **-m** - The model we would like to use.
  - **-t** - The table name which would be created in database.
  - **-k** - The number of latent semantics.

  - **-d** - Decomposition method.
  - **-l** - The label used to find corresponding labeled images.
  - **-i** - The target image ID.
  - **-ip** - If the target image is not in database, we can also use this argument to indicate the target image path.

  Example:
  ```
  python p2task5.py -m hog -t test -k 5 -d svd -l l -meta HandInfo.csv
  -i Hand_0000002
  python p2task5.py -m hog -t test -k 5 -d svd -l l -meta HandInfo.csv
  -ip Dataset2/Hand_0010646.jpg
  ```

- Phase 2 task 6
  - **-t** - The table name which would be created in database.
  - **-s** - The target subject ID.
  - **-meta** - The metadata path
  - **-p** - The image path

  Example:
  ```
  python p2task6.py -t test -s 27 -meta HandInfo.csv -p dataset
  ```

- Phase 2 task 7
  - **-t** - The table name which would be created in database.
  - **-k** - The number of latent semantics.
  - **-meta** - The path of metadata(HandInfo.csv).

  Example:
  ```
  python p2task7.py -t test -k 10 -meta HandInfo.csv
  ```

- Phase 2 task 8
  - **-t** - The table name which would be created in database.
  - **-k** - The number of latent semantics.
  - **-meta** - The path of metadata(HandInfo.csv).

  Example
  ```
  python p2task8.py -k 5 -meta HandInfo.csv -t test
  ```

4.3 Outputs

We use the provided sample query to execute out program in each task and print out the result. In order to execute the program properly, we need to use phase 1 task 2 to create the feature extraction database with the provided small dataset for each feature descriptor models.

4.3.1 Task 1

We use the most similar image to represent the top k latent semantics. Each image represent each row in our latent semantics matrix which is the extra credit for this phase.

- **Query 1**
  - Input
    - Model: SIFT
    - K: 20
    - Technique: Singular Value Decomposition (SVD)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0001102	2856.1469575016963
2	Hand_0011159	1632.0601225315577
3	Hand_0011158	1647.5452599154087
4	Hand_0001020	975.3785754400293
5	Hand_0011107	1147.6739359803644
6	Hand_0011600	1156.6884877769116
7	Hand_0000412	1813.375738284854
8	Hand_0000282	1023.7817611036928
9	Hand_0002803	1247.6661858919554
10	Hand_0002802	916.2523574202858
11	Hand_0000412	1089.5424600097604
12	Hand_0000931	931.622598609412
13	Hand_0002802	979.6532708761367
14	Hand_0000889	977.7300363091592
15	Hand_0000890	912.2670904762676
16	Hand_0002887	871.6080168252784
17	Hand_0009782	809.4265086547819
18	Hand_0000889	1045.7091642183807
19	Hand_0009350	774.6333451251191
20	Hand_0000205	674.4687560416054

- **Query 2**
  - Input
    - Model: SIFT
    - K: 40
    - Technique: Singular Value Decomposition (SVD)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0001102	2856.1469575063475
2	Hand_0011159	1632.059690884643
3	Hand_0011158	1647.5420068835228
4	Hand_0001020	975.2113117383658
5	Hand_0011107	1147.7132187809239
6	Hand_0011600	1156.697522854231
7	Hand_0000412	1813.502425224403
8	Hand_0000282	1024.5895473466012
9	Hand_0002803	1248.4356671342725
10	Hand_0002802	916.732066124938
11	Hand_0000412	1098.4976242969922
12	Hand_0000931	931.623280877406
13	Hand_0002802	983.6795227194082
14	Hand_0000889	995.5283742323339
15	Hand_0000890	919.4686315307238
16	Hand_0002887	867.8843984040343
17	Hand_0009782	828.2873959794588
18	Hand_0000889	1074.8167095036035
19	Hand_0011602	756.9594436812375
20	Hand_0000123	752.1308756203899
21	Hand_0000931	762.0984484856705
22	Hand_0001000	720.789194795047
23	Hand_0000245	778.6252201912342
24	Hand_0000890	806.5561745020113
25	Hand_0000281	689.0014865492676
26	Hand_0001428	686.1461944776457
27	Hand_0000255	777.3839422017365
28	Hand_0001358	741.5207474325596
29	Hand_0002887	588.2900935879059
30	Hand_0000323	658.6650093710132
31	Hand_0011104	661.9450249646352
32	Hand_0001582	622.6587740580271
33	Hand_0001103	609.7412495410861
34	Hand_0001394	777.1014329513594
35	Hand_0001468	913.4229076525721
36	Hand_0000122	566.1945964228196
37	Hand_0009781	604.6354768267131
38	Hand_0000270	590.1141502621388
39	Hand_0000246	700.3483839852063
40	Hand_0001001	549.0669109532699

- **Query 3**
  - Input
    - Model: Color Moments
    - K: 20
    - Technique: Latent Dirichlet Analysis (LDA)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0000411	86.40000000000015
2	Hand_0011406	371775.1574469844
3	Hand_0000203	4253487.440508978
4	Hand_0001622	1106336.2222600258
5	Hand_0001435	2089249.08964434
6	Hand_0001145	1642023.6997721118
7	Hand_0000488	620112.2665563016
8	Hand_0006561	4192630.246553893
9	Hand_0001325	816460.1981262073
10	Hand_0011105	780881.3582068707
11	Hand_0001154	599092.7880009054
12	Hand_0000411	86.40000000000015
13	Hand_0000990	3350238.3270284776
14	Hand_0009388	2599444.9014301402
15	Hand_0000411	86.40000000000015
16	Hand_0009198	1496190.7837002294
17	Hand_0007671	1322758.397399793
18	Hand_0011406	3087439.5616259873
19	Hand_0002868	1195021.3850071551
20	Hand_0007672	2279653.809254481

- **Query 4**
  - Input
    - Model: Color Moments
    - K: 40
    - Technique: Latent Dirichlet Analysis (LDA)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0000891	43.20000000000001
2	Hand_0001497	1220794.1875886666
3	Hand_0000488	101565.3310810086
4	Hand_0000891	43.20000000000001
5	Hand_0002778	1369510.6836686528
6	Hand_0000891	43.20000000000001
7	Hand_0002778	919280.8379752837
8	Hand_0001614	821245.4701924992
9	Hand_0007671	600061.1486194679
10	Hand_0001325	1262964.9626710457
11	Hand_0001153	1588822.4119628803
12	Hand_0000891	43.20000000000001
13	Hand_0001620	426282.4284350934
14	Hand_0000891	43.20000000000001
15	Hand_0000891	43.20000000000001
16	Hand_0000991	1135410.17332093
17	Hand_0007672	3542680.1582880244
18	Hand_0000891	43.20000000000001
19	Hand_0011105	506517.84302699985
20	Hand_0000990	2291447.7330353977
21	Hand_0001091	2453408.3038908755
22	Hand_0000891	43.20000000000001
23	Hand_0001624	170648.66916707606
24	Hand_0006561	2295041.6451742845
25	Hand_0000891	43.20000000000001
26	Hand_0009388	2200753.22032974
27	Hand_0000891	43.20000000000001
28	Hand_0000891	43.20000000000001
29	Hand_0000891	43.20000000000001
30	Hand_0011406	411613.13543157117
31	Hand_0000891	43.20000000000001
32	Hand_0000891	43.20000000000001
33	Hand_0001145	1750722.8626788966
34	Hand_0000203	1322424.3562894787
35	Hand_0000891	43.20000000000001
36	Hand_0000323	480241.02773827733
37	Hand_0000891	43.20000000000001
38	Hand_0006561	2752967.8429544372
39	Hand_0001427	1823423.1159421206
40	Hand_0001622	350094.71999661694

4.3.2 Task 2

- **Query 1**
  - Input
    - Model: HOG
    - K: 10
    - M: 10

    - Image ID: Hand_0000111.jpg
    - Technique: Principle Component Analysis (PCA)
  - Output

Rank	Image ID	Score
1	Hand_0000112	53.40855427027667
2	Hand_0000108	96.62662273653109
3	Hand_0001091	296.1756171571381
4	Hand_0001145	300.24298189180865
5	Hand_0001144	308.9392721346958
6	Hand_0000132	371.22002475148076
7	Hand_0001092	433.5463165097332
8	Hand_0001154	433.82603167733555
9	Hand_0000990	439.8645959573205
10	Hand_0001153	441.88994284787765

- **Query 2**
  - Input
    - Model: HOG
    - K: 40
    - M: 10
    - Image ID: Hand_0000111.jpg
    - Technique: Principle Component Analysis (PCA)
  - Output

Rank	Image ID	Score
1	Hand_0000112	143.28629617863902
2	Hand_0000108	233.73947098111566
3	Hand_0001145	532.6011828467886
4	Hand_0001091	549.7442964977355
5	Hand_0001144	582.2071163776272
6	Hand_0001092	682.4028813378678
7	Hand_0000132	739.2007370489652
8	Hand_0000133	751.1336847978355
9	Hand_0001154	773.5831225906549
10	Hand_0000990	798.788806568867

- **Query 3**
  - Input
    - Model: LBP
    - K: 10
    - M: 10
    - Image ID: Hand_0000200.jpg
    - Technique: Non-negative Matrix Factorization (NMF)
  - Output

Rank	Image ID	Score
1	Hand_0000199	6.448178930562091
2	Hand_0000330	13.105361895226634
3	Hand_0000488	13.345362489504932
4	Hand_0000490	24.822382622237626
5	Hand_0000491	25.987328221689733
6	Hand_0000329	73.42871617734434
7	Hand_0000282	88.14633797448356
8	Hand_0000890	93.19767817153422
9	Hand_0011158	93.73685973185191
10	Hand_0011159	96.94654506593521

- **Query 4**
  - Input
    - Model: LBP
    - K: 40
    - M: 10
    - Image ID: Hand_0000200.jpg
    - Technique: Non-negative Matrix Factorization (NMF)
  - Output

Rank	Image ID	Score
1	Hand_0000199	13.106368073758807
2	Hand_0000488	27.277962233152824
3	Hand_0000330	28.17666717471286
4	Hand_0000490	45.69758814125859
5	Hand_0000491	61.20326092183295
6	Hand_0000282	113.29009890612906
7	Hand_0000329	145.86856234956326
8	Hand_0000123	146.92916764503934
9	Hand_0001469	147.67409231046406
10	Hand_0001466	148.14586202049134

4.3.3 Task 3

We use the most similar image to represent the top k latent semantics. Each image represent each row in our latent semantics matrix which is the extra credit for this phase.

- **Query 1**
  - Input
    - Model: HOG
    - K: 20
    - Label: Dorsal
    - Technique: Principle Component Analysis (PCA)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0000145	595.3364837533883
2	Hand_0002804	445.1540905206259
3	Hand_0011599	688.1693681987047
4	Hand_0001534	686.5139194333074
5	Hand_0002803	550.8728427567871
6	Hand_0002803	396.6573701189984
7	Hand_0000111	379.3942453893302
8	Hand_0001012	489.82216501179926
9	Hand_0000931	321.0895746162793
10	Hand_0000400	266.90911037161135
11	Hand_0009389	502.23850351644353
12	Hand_0000207	254.85775776566805
13	Hand_0000145	333.5122352160689
14	Hand_0001534	203.33506033310334
15	Hand_0000880	339.49968537363407
16	Hand_0011104	305.3977983145031
17	Hand_0009388	306.714617506856
18	Hand_0011103	127.3838072809749
19	Hand_0000144	328.65895430904374
20	Hand_0000373	343.1773644158709

- **Query 2**
  - Input
    - Model: HOG
    - K: 30
    - Label: Dorsal
    - Technique: Principle Component Analysis (PCA)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0000145	595.340607242701
2	Hand_0002804	445.14836188033973
3	Hand_0011599	688.1754013182657
4	Hand_0001534	686.5572855768743
5	Hand_0002803	551.1591365983143
6	Hand_0002803	396.4084045350081
7	Hand_0000111	379.3733024756217
8	Hand_0001012	489.41315892227516
9	Hand_0000931	324.05192827365937
10	Hand_0000400	267.8378566039751
11	Hand_0009389	503.80337720523187
12	Hand_0000207	251.45901919007923
13	Hand_0000145	333.4362930782038
14	Hand_0001534	200.2885126195717
15	Hand_0001466	235.91343282849994
16	Hand_0011104	309.8318727947083
17	Hand_0009388	326.89514970104494
18	Hand_0011103	139.57606442387154
19	Hand_0000144	315.5676878793609
20	Hand_0000373	340.87912424907245
21	Hand_0000880	229.79775287087494
22	Hand_0001394	360.6546576758491
23	Hand_0001624	278.1362305259443
24	Hand_0001394	220.7746091788798
25	Hand_0001466	389.41693117531764
26	Hand_0001435	160.23681689348714
27	Hand_0001394	303.30143553163407
28	Hand_0007740	194.18899739768796
29	Hand_0000322	240.0536358103193
30	Hand_0001144	325.95300378128735

- **Query 3**
  - Input
    - Model: Color Moments
    - K: 20
    - Label: Left
    - Technique: Latent Dirichlet Analysis (LDA)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0000281	1005424.0167059416
2	Hand_0000122	86.40000000000006
3	Hand_0000122	86.40000000000006
4	Hand_0009388	835871.1120827026
5	Hand_0001435	719789.3864737866
6	Hand_0000122	86.40000000000006
7	Hand_0000203	1644377.1026027147
8	Hand_0000122	86.40000000000006
9	Hand_0007672	1081749.4806749364
10	Hand_0000122	86.40000000000006
11	Hand_0007671	771020.9379370701
12	Hand_0000488	348954.5567176015
13	Hand_0000122	86.40000000000006
14	Hand_0006561	2997127.3487485135
15	Hand_0000122	86.40000000000006
16	Hand_0000203	1643119.6533988672
17	Hand_0000122	86.40000000000006
18	Hand_0000122	86.40000000000006
19	Hand_0002868	1132154.7097109058
20	Hand_0006561	1853733.756471112

- **Query 4**
  - Input
    - Model: Color Moments
    - K: 30
    - Label: Left
    - Technique: Latent Dirichlet Analysis (LDA)
  - Output

Order	Image ID	Dot Product Value
1	Hand_0001325	188590.97006659186
2	Hand_0000162	457517.94051380915
3	Hand_0000253	57.60000000000001
4	Hand_0009388	718000.5061194033
5	Hand_0000253	57.60000000000001
6	Hand_0009388	1148738.7867346718
7	Hand_0000253	57.60000000000001
8	Hand_0009390	514603.12473922875
9	Hand_0000253	57.60000000000001
10	Hand_0001325	385053.3497997402
11	Hand_0000253	57.60000000000001
12	Hand_0001614	259575.15017893727
13	Hand_0002868	664141.4627999421
14	Hand_0000253	57.60000000000001
15	Hand_0000322	534960.6141399656
16	Hand_0001435	318822.22445675475
17	Hand_0007672	474723.47667278786
18	Hand_0000253	57.60000000000001
19	Hand_0000372	279582.7222086479
20	Hand_0001435	989841.4276009023
21	Hand_0000323	236477.69829397276
22	Hand_0007671	325363.5946771598
23	Hand_0007672	1224540.4488535798
24	Hand_0006561	795386.8483799084
25	Hand_0002803	621874.9141659769
26	Hand_0000488	157406.3809048661
27	Hand_0000203	1026854.6944694472
28	Hand_0000253	57.60000000000001
29	Hand_0007671	273105.627407611
30	Hand_0006561	2361669.103454426

4.3.4 Task 4

- **Query 1**
  - Input
    - Model: LBP
    - K: 10
    - M: 10
    - Label: Palmer
    - Technique: Non-negative Matrix Factorization (NMF)
    - Image ID : Hand_0000200.jpg
  - Output

Rank	Image ID	Score
1	Hand_0000199	7.071139728422938
2	Hand_0000488	8.500712193019883
3	Hand_0000330	14.464788575011756
4	Hand_0000490	50.656596711270886
5	Hand_0000491	57.44929760368825
6	Hand_0000282	72.79559916272522
7	Hand_0000283	97.3060130512401
8	Hand_0000329	105.46263636902204
9	Hand_0000303	131.25966063401367
10	Hand_0000348	135.5114443038098

- **Query 2**
  - Input
    - Model: LBP
    - K: 30
    - M: 10
    - Label: Palmer
    - Technique: Non-negative Matrix Factorization (NMF)
    - Image ID : Hand_0000200.jpg
  - Output

Rank	Image ID	Score
1	Hand_0000199	24.200571054294787
2	Hand_0000488	34.61209639798031
3	Hand_0000330	60.93352277385878
4	Hand_0000490	82.60660945095432
5	Hand_0000491	98.53508888849633
6	Hand_0000329	222.52036117522826
7	Hand_0000282	257.8303557802476
8	Hand_0000253	295.466783292878
9	Hand_0001469	302.5298503676372
10	Hand_0011158	317.2741764513058

- **Query 3**
  - Input
    - Model: SIFT
    - K: 10
    - M: 10
    - Label: With Accessories
    - Technique: Singular Value Decomposition (SVD)
    - Image ID : Hand_0011160.jpg
  - Output

Rank	Image ID	Score
1	Hand_0000488	1224.4573147459892
2	Hand_0011159	1319.874160141761
3	Hand_0000490	1393.4310452518948
4	Hand_0011158	1562.8523131587417
5	Hand_0000491	1584.569194581401
6	Hand_0001394	1678.5460986676842
7	Hand_0001395	1862.8855941303727
8	Hand_0000482	1864.3941134745999
9	Hand_0000483	1941.0207580718331
10	Hand_0001390	2081.478780834056

- **Query 4**
  - Input
    - Model: SIFT
    - K: 30
    - M: 10
    - Label: With Accessories
    - Technique: Singular Value Decomposition (SVD)
    - Image ID : Hand_0011160.jpg
  - Output

Rank	Image ID	Score
1	Hand_0011159	2867.4918288217486
2	Hand_0001390	3207.3035287333205
3	Hand_0000929	3215.403932905931
4	Hand_0011158	3221.046526725135
5	Hand_0000490	3263.7832380421955
6	Hand_0000491	3325.890496199957
7	Hand_0001092	3341.218038546994
8	Hand_0000482	3342.7773127026194
9	Hand_0001395	3365.5359908856594
10	Hand_0000401	3390.406887914212

4.3.5 Task 5

We use the given k latent semantics provided by the given label and the unlabeled image features to calculate the $cosine - similarity$ between them. After we get the similarity score, if it is over 0.5, we mark the unlabeled image have the same label with the given latent semantics. On the other hand, it will have different label with the provided latent semantics. For example, if the given latent semantics is Left, and the output is True, which means the unlabeled image is also Left. However, if the output is False, which means the unlabeled image is Right. All the other labels are all return in the same way with their corresponding labels.

- **Query 1**
  - Input
    - Model: LBP
    - K: 10

        * Label: Right
        * Technique: Non-negative Matrix Factorization (NMF)
        * Image ID : Hand_0000111.jpg
    - Output

      True (Labeled as Right)

    - Metadata Label

      Right

- **Query 2**
    - Input
        * Model: LBP
        * K: 30
        * Label: Right
        * Technique: Non-negative Matrix Factorization (NMF)
        * Image ID : Hand_0000111.jpg
    - Output

      True (Labeled as Right)

    - Metadata Label

      Right

- **Query 3**
    - Input
        * Model: SIFT
        * K: 10
        * Label: With No Accessories
        * Technique: Singular Value Decomposition (SVD)
        * Image ID : Hand_0001395.jpg
    - Output

      False (Labeled as With Accessories)

    - Metadata Label

      With Accessories

- **Query 4**
    - Input
        * Model: SIFT
        * K: 30
        * Label: With No Accessories
        * Technique: Singular Value Decomposition (SVD)
        * Image ID : Hand_0001395.jpg
    - Output

      False (Labeled as With Accessories)

    - Metadata Label

      With Accessories

4.3.6 Task 6

- **Query 1**
  - Input
    - Subject ID: 27
  - Output

Rank	Subject ID
1	0
2	594
3	1506

- **Query 2**
  - Input
    - Subject ID: 55
  - Output

Rank	Subject ID
1	1560
2	1506
3	92

4.3.7 Task 7

We use the most similar subject to represent the top k latent semantics. Each subject represent each row in our latent semantics matrix which is the extra credit for this phase.

- **Query 1**
  - Input
    - K: 10
  - Output

Order	Subject ID	Dot Product Value
1	1011	18231159867.48359
2	559	14742081238.64081
3	1000000	8606146721.701862
4	1543	21073785086.211945
5	1011	16647262644.135942
6	1011	27662473682.64925
7	1011	7506984637.044907
8	1560	9334734660.765158
9	1506	3708420601.428402
10	559	14845093662.350027

- **Query 2**
  - Input
    - K: 20
  - Output

Order	Subject ID	Dot Product Value
1	1011	20827959402.213074
2	1543	13635392757.635218
3	1513	18779472600.238285
4	0	10367010264.222277
5	1011	18831600378.757812
6	1543	17084522558.463657
7	1011	18944106206.08175
8	1011	9058606007.389267
9	1011	10776810266.435337
10	1543	4144745366.623598
11	1011	27510263485.74466
12	1513	10694518978.868021
13	57	8756798173.819035
14	1011	13311686490.438301
15	1543	18010990170.70129
16	1011	1003583391.5413244
17	1513	6741688956.796759
18	1011	7290183453.917957
19	1513	879441013.4034349
20	1011	10372628707.551208

4.3.8 Task 8

- **Query 1**
  - Input
    - K: 4
  - Output

Image-Space

Order	Term
1	[8.7172806 , 9.45670185, 4.06431232, 3.89475471]
2	[0. , 0.67750995, 0.20186786, 0.81805925]
3	[0. , 0. , 0. , 0.04589157]
4	[0. , 0.05962139, 0.07150824, 0.50939923]

Metadata-Space

Order	Term
1	[3.44943581e+00, 5.25582506e+00, 6.74596952e+00, ..., 5.71735878e-03, 0.00000000e+00, 8.71449859e-03]
2	[4.97656673e-01, 3.88090466e+00, 1.66589756e+00, ..., 4.67248132e-03, 1.89965068e-04, 5.18192622e-03]
3	[0.00000000e+00, 5.23307109e+00, 0.00000000e+00, ..., 4.83271014e-03, 3.92026712e-03, 4.52277565e-03]
4	[0.27927784, 3.01230644, 0.81671622, ..., 0.00396105, 0.00556956, 0.]

- **Query 2**
  - Input
    - K: 6

– Output

Image-Space

Order	Term
1	[6.05781275, 7.30821289, 3.2806701 , 4.37131486, 3.41993745, 8.03597759]
2	[0. , 0. , 0. , 0. , 0. , 2.29911951]
3	[0. , 0. , 0. , 0. , 0. , 1.47510768]
4	[0. , 0. , 0. , 0.0875188 , 0. , 3.44274291]
5	[0.16965313, 0.70102334, 0.23247851, 0.711904 , 0.44359494, 3.01051946]
6	[9.9568791 , 9.87394018, 4.09193735, 2.57187571, 2.77232162, 2.30134854]

Metadata-Space

Order	Term
1	[7.33434061e+00, 1.01887375e+01, 0.00000000e+00, ..., 4.28772920e-03, 0.00000000e+00, 7.69851387e-03]
2	[0.00000000e+00, 0.00000000e+00, 6.40693530e+01, ..., 2.02833558e-03, 1.75465924e-03, 2.24542598e-03]
3	[0.00000000e+00, 0.00000000e+00, 7.26713119e+01, ..., 4.68628868e-03, 7.91645383e-03, 0.00000000e+00]
4	[8.36750243e-02, 0.00000000e+00, 4.38686525e+01, ..., 3.86570417e-03, 4.98950654e-03, 0.00000000e+00]
5	[1.42272530e+00, 1.57997709e+00, 2.17869608e+01, ..., 3.01939935e-03, 2.42397933e-06, 3.64060352e-03]
6	[4.74974131, 9.07046875, 0. , ..., 0.00991852, 0. , 0.01046857]

5 Conclusions

In this project, we learn how to apply dimension reduction to a real-world problem. Even though we know the document-term matrix can be applied to dimension reduction methods, and the results of those methods, we still could not understand how to apply those method to a problem. We, therefore, spent a lot of time to discuss, read documents on internet, and discussed with Professor how to overcome some limitations - such as building key word for LDA.

However, even we tried several methods to improve the performance on our project, we still cannot find a best way to made a reliable querying system. Especially in task 5, although one of our analyses seems revealed some clues on multiple data projection, but the prediction accuracy was still bad. There are must some better way to overcome this problem, such as SVM or other machine learning model to help us predict the unlabeled data and clustering data in latent semantic space.

6 Bibliography

References

[1] Mahmoud Afifi. 11k hands: gender recognition and biometric identification using a large dataset of hand images. *Multimedia Tools and Applications*, 78(15):20835–20854, 2019.

[2] Matt Brems. A one-stop shop for principal component analysis, Jun 2019.

[3] Lettier. Your guide to latent dirichlet allocation, May 2019.

[4] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[5] B. E. Prasad, A. Gupta, H. D. Toong, and S. E. Madnick. A microcomputer-based image database management system. *IEEE Transactions on Industrial Electronics*, IE-34(1):83–88, Feb 1987.

Appendices

Roles of the group members

Jiaxuan Pang	SIFT, CM-LDA Pre-procesing implementation. Task 6,7 implementation. Report writing.
Ting Xia	NMF implementation. Task 1 implementation. Report writing.
Ching-Wen Tu	PCA, LDA implementation. Task 5 implementation. Propose solutions to task 6-8. Report writing.
Bandon Tseng	SVD, CM-NMF Pre-processing, Task 1, 2, 3, 4, 5, 6, 8, Metadata parser implementation. Project code architecture. Report writing.
Kuan-Ting Shen	Task 7, 8 implementation. Metadata parser implementation. Report writing.
Ming Yi	Propose solutions to task 1-6 ; Refine coding framework in dimensional reduction and task 1-4 .