# PHASE – 3

Project Report for Image Similarity Detection

By -

Krishna Vamsi Nandamuru

Krima Doshi

Nisarg Shah

Harshita Verma

Chinmay Sai Krishna Atluri

Eshita Khandelwal

**SUBJECT:** CSE 515 Multimedia and Web Databases

Arizona State University

28th, November, 2021

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| Sr. No | Heading | Page No. |
| 1. | Abstract |  |
| 2. | Introduction |  |
| 3. | Description of Proposed Solution |  |
| 4. | Interface Specification |  |
| 5. | Outputs |  |
| 6. | System Requirements |  |
| 7. | Related Works |  |
| 8. | Conclusions |  |
| 9. | Bibliography |  |

# ABSTRACT

With the introduction of modern technology, Media has become common place in today’s world. Increasing number of devices are being connected to the wider network and are sharing all forms of information that transcend the traditional textual data sharing. From services that provide on-demand videos via the web to personal cloud storage technologies, Multimedia is the new format to share information in this digital age.

With the emergence of multimedia and its dominant presence in the world, Content-based information retrieval has become a necessity to organise and maintain information. It has become a topic of interest for researchers to develop methods to query multimedia data and return relevant results.

A popular implementation of the above defined case is reverse image search, in which a person provides an input image and the recommender systems retrieve similar/relevant images and provide results based on the image.

The main issue we encounter as soon as we deal with multimedia is the constraint of space. As multimedia has a heavy computation cost, we use Dimensionality Reduction Methods as a way to reduce the dimensionality of the data and speed up the processing.

In this project, we describe one such application of multimedia, which is storing and retrieval of images. We use Olivetti faces dataset for working with as a proof of concept and implement an image similarity detection algorithm that extracts specified feature vectors from the input image and uses metric measures to compare the distances between images and determine the most similar.

The Phase-2 of this project concentrates on finding the latent features of a dataset using dimensionality reduction methods, especially Singular Vector Decomposition, Principal Component Analysis, K-means and Latent Dirichlet Allocation to find the latent features and comparing them to a query image to obtain the results.

Phase – 3 of this project is mostly based on search space pruning and image classification. We have utilised Support Vector Machines, Decision Trees and Personalised Page Rank for classification of images. Later, we prune the search space using Local Sensitive Hashing and VA Files Data Structures. Finally, an interface is developed that can accept a query image from the user and takes into account user feedback by classifying the images as “Relevant” or “Irrelevant” and returning the results.

Keywords Phase-1: Olivetti face, Feature Extraction, Color Moments, Histogram of Oriented Gradients, Extended Local Binary Patterns, Manhattan Distance, Cosine Distance, Cosine Similarity, Earth Movers Distance, L2 Norm.

Keywords Phase-2: Singular Vector Decomposition, Principal Component Analysis, K-Means, Centroids, Clusters, Object-Feature weights, Latent Dirichlet Allocation, RandomWalk, PageRank, Seed nodes, ASCOSS++, Similarity Matrix, Transition Matrix.

Keywords Phase-3: Support Vector Machine, Cross Validation, Kernel, Support, Decision Trees, branch, Root, Label, Local Sensitive Hashing, Buckets, Gaussian Distribution, Random Function, PageRank, Seed nodes, Vector Approximation Files, Binary Representation, Relevance Feedback

# INTRODUCTION

**1. Terminology**

**Feature Vector**

A Feature Vector can be described as an abstraction of an image used to characterize and numerically quantify the contents of an image. Usually, it takes a real, integer or binary value. It can be described as a list of numbers used to represent an image.

**Color Moments**

Color Moments are simple measures that can be applied to images and differentiate them based on the composition of colours within the image. They are measures that describe a probability distribution of colours for a particular block of pixels. They are mainly used as an indexing feature to quickly retrieve similar images from a database of images, based on their colour features. There are three main components to calculating the color moment feature. They are:

1. Mean

2. Standard Deviation

3. Skewness

The first colour moment “Mean” can be defined as the average colour in the image or the average of the pixel values of each color channel of the image. Mean is calculated using the following formulae:

(N corresponds to the number of pixels in the image & is the value of j’th pixel of i-th color channel)

The second colour moment “Standard Deviation” can be defined as the amount of deviation in the pixel value between corresponding pixels.

( is the mean value or the first color moment and N is the number of pixels in the image)

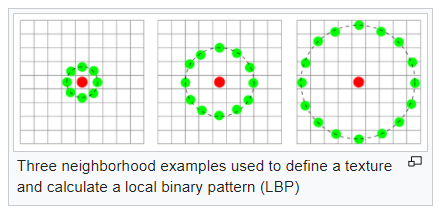
The third colour moment is the “Skewness” which measures the asymmetry of the colour distribution. Skewness provides information about the shape of the colour distribution of an image. Skewness is calculated as shown below:

( is the mean value or the first color moment and N is the number of pixels in the image)

**Extended Local Binary Patterns**

Local Binary Patterns or LBP is a feature that can be extracted from an image and gives information about the textures found in the image. It is a powerful feature used to classify and index images based on their texture analysis. An LBP Vector is calculated using the following steps:

* Each current window in consideration is divided into a set of cells.
* Each pixel in a cell is compared with n of its neighbours (value of n can be changed).
* If the neighbouring pixel’s value is greater than central pixel, value is considered “1”, otherwise value is “0”.
* Histogram of the frequency of occurrence of these “1”s and “0”s are calculated and normalized.



*(Example of the LBP being calculated)*

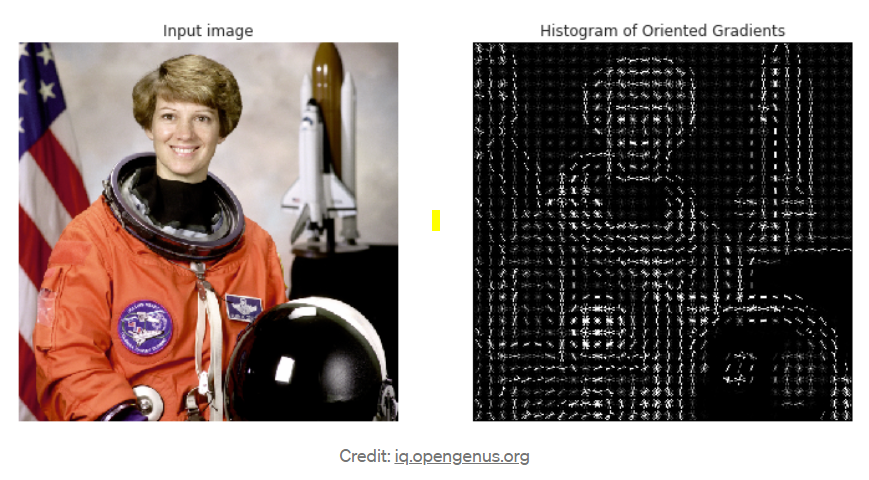
All such histograms for all blocks of the images are then concatenated into a single vector. The extension to the above described LBP is called uniform pattern, which reduces the length of feature vector. A local binary pattern is called uniform if there are at-most two ‘0-1’ or ‘1-0’ transitions in the patterns. All such patterns are binned into a single bin and rest of patterns are put in separate bins. This greatly reduces the space used to store the feature vector for LBP. Such an operation is called ELBP and is determined to be rotation invariant.

**Histogram of Oriented Gradients**

A gradient can be defined as a smooth or sharp change in the hue and colour of an image in a particular direction. The histogram of oriented gradients or HOG is a feature vector that computes the direction of change depending on the user’s value of the number of directions. The HOG is computed in the following manner:

* Input image is taken
* The gamma and colour values are normalized to [0-1]
* Gradients are computed over the given window
* Weighted votes are cast by each pixel as per the values
* Contrast is normalized over overlapping blocks
* Final HOG is computed over the selected window

The output of HOG is similar to that of edge detection algorithms. HOG also outputs a vector that if visualized more or less computes the edges of the objects in an image as per the colour gradients. A sample image and its HOG Output is as given below:



**Principal Component Analysis**

In a data distribution of points plotted on a real coordinate space, Principal components can be defined as a sequence of vectors that denote the maximum variation among points. These components are orthogonal to each other and constitute an orthonormal basis space on which all the points of the space can be projected with a minimal loss in information. Principal Component Analysis (PCA) is the process of computing these components then transforming the data by projecting it on to the most significant among the computed components. The process for computing PCA for a given matrix is as follows:

* For the given n x m matrix, calculate the covariance matrix (i.e. an n x n matrix that gives the covariance of each element of the matrix with every other element of the matrix). This captures all the discrimination power in the data.
* Now for the given covariance matrix A, we calculate the eigen vectors and eigen values as follows: **A.x = C.x**
* **After finding the Eigen values and Eigen vectors, we discard the ones with insignificant amount of discrimination power and retain only the significant ones.**
* **This leads to a dimensionality reduction in which the features with least discrimination powers have been eliminated.**
* **The Matrix factorization can be denoted as follows:**

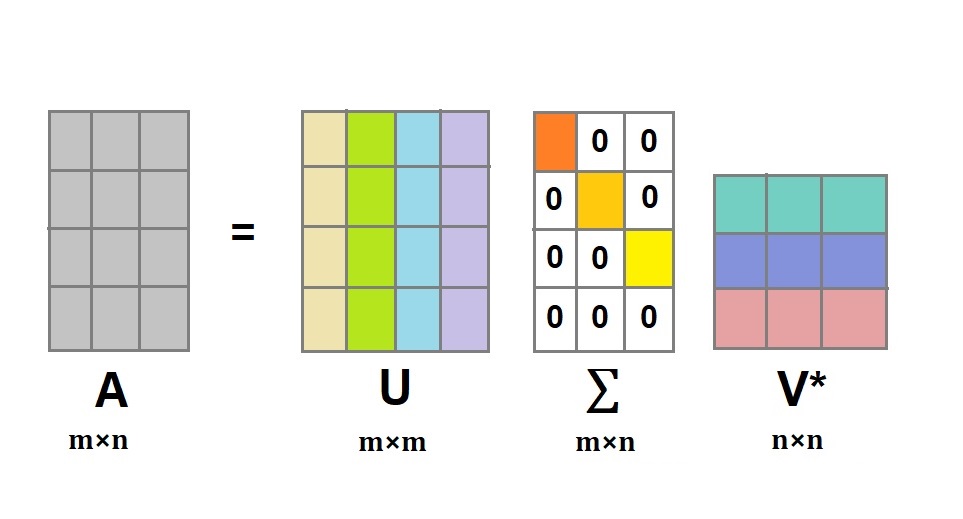
1. **Left factor matrix:** Describes the k old features in terms of k latent semantics
2. **Core matrix:** Describes the importance of k latent features.
3. **Right factor matrix:** Describes k latent features in terms of m old features

PCA is a dimensionality reduction technique which optimally preserves variance.

**Singular Value Decomposition**

SVD or Singular Value Decomposition is a factorization of matrices that generalizes the Eigen decomposition over any ‘m x n’ matrix. Given a matrix M, the SVD factorization is of the form, where U is the m x k object-latent feature matrix, Σ is the k x k diagonal eigenvalue matrix and V is the k x n latent feature-feature matrix. To obtain the U matrix we take the D matrix and multiply it with transpose of Database matrix. Then we find Eigen vectors of U matrix. To find feature-feature matrix we multiply transpose of database matrix with database matrix. And this matrix is feature-feature matrix. Eigen vectors are calculated for this matrix which is V(t). The database matrix is decomposed into U,S,V(t).

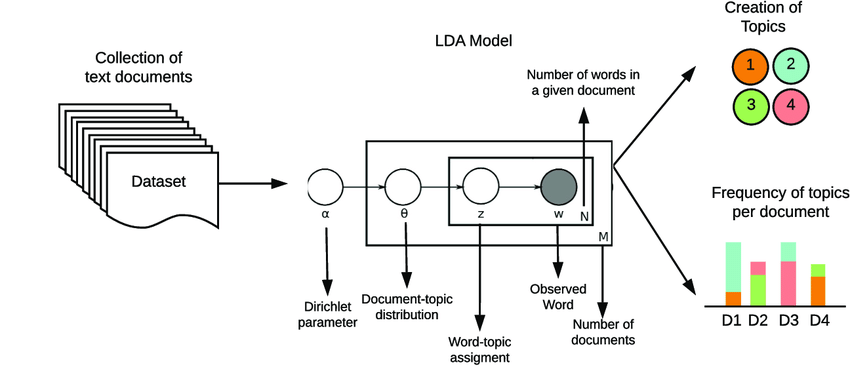
SVD preserves data and distances.



**Latent Dirichlet Allocation**

LDA is a generative model that maps a particular feature to an object based on its latent feature set. It will generate the probability that a particular feature is present in an object and ties back the probabilities to a probability maximization function that determines the mapping. The generative model used in LDA makes it richer in domain knowledge than PLSA, SVD. These generative models are built using background and domain knowledge. Each object has set of features. Observed variables in LDA are n objects and m features. k is the latent features which are unobserved variables. To measure likelihood of a feature in the database Poisson distribution is used to assign the probabilities. ‘Dirichlet’ indicates LDA’s assumption that the distribution of topics in a document and the distribution of words in topics are both Dirichlet distributions. ‘Allocation’ indicates the distribution of topics in the document.

Dirichlet distribution is associated to likelihood of multinomial distribution in the database.



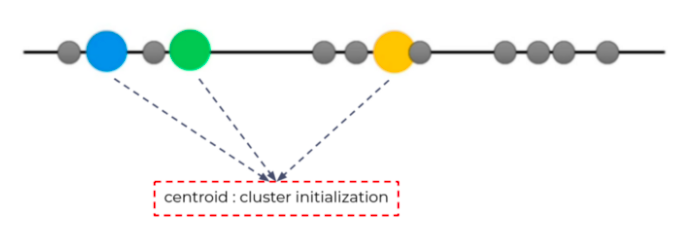
*Sample of LDA performed on a text document database and topic modelling*

**K-Means**

K-means Clustering is a vector quantization approach that divides n observations into k clusters, with each observation belonging to the cluster with the closest mean (cluster centres or cluster centroid), which serves as the cluster's prototype. To analyse the learning data, the K-means method starts with a set of randomly chosen centroids that serve as the starting points for each cluster, and then performs iterative (repetitive) calculations to optimize their positions.

It stops constructing and optimizing clusters if one of the following conditions occurs:

* The centroids have stabilized — there is no change in their values because the clustering has been successful.
* The defined number of iterations has been achieved.



The following steps depict the working of K-Means algorithm:

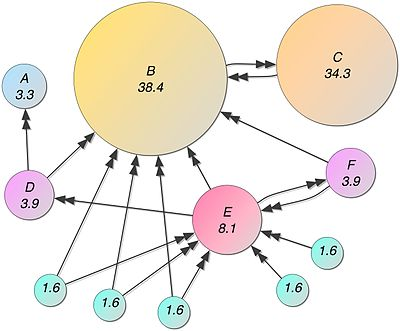
1. Determine the value “K”, the value “K” represents the number of clusters.
2. Randomly select 3 distinct centroid (new data points as cluster initialization)
3. Measure the distance (Euclidean distance) between each point and the centroid.
4. Assign each point to the nearest cluster.
5. Calculate the mean of each cluster as a new centroid.

Repetition of step 3–5 with the new center of cluster.

**Personalised PageRank**

PageRank algorithm, named after the google founder Larry Page is used to rank the websites as per their connectivity to other webpages. Two major concepts in this algorithm is the ‘Hubs’ and ‘Authorities’. An authority is defined as a node to which many hubs point to. A hub is defined as a node which points to many authorities. These both type of nodes are considered to be of significant value to the structure of the graph. PageRank algorithm attempts to assign a rank to a node in the graph based on its significance to the topology of the graph.

Personalised PageRank is an extension to the normal PageRank in which we choose a set of initial seed nodes, which can be described as nodes that are of points of interests, and find the ranking of the graph based on the given seed nodes. This enables us to modify the similarity and ranking scores based on user preferences on a database.



**Support Vector Machine**

Support Vector Machines (SVM) are commonly thought of as a classification strategy, however, they may be used to solve both classification and regression problems. It can handle both continuous and categorical variables with ease. To differentiate various classes, SVM creates a hyperplane in multidimensional space. SVM iteratively generates the best hyperplane, which is then utilized to minimize an error. The goal of SVM is to find a maximum marginal hyperplane (MMH) that splits a dataset into classes as evenly as possible.

Diagram

Description automatically generated

The SVM method is an excellent classification algorithm. It's a supervised learning algorithm that's primarily used to categorize data into several groups. A set of label data is used to train SVM. SVM has the advantage of being able to solve both classification and regression issues. To divide or classify two classes, SVM creates a decision boundary, which is a hyperplane between them. SVM is also utilized in picture classification and object detection. Some terminology of SVM is as follows:

* Support Vector: The data points nearest to the hyperplane are called support vectors. By computing margins, these points will better define the separation line. These points are more relevant to the classifier's construction.
* Hyperplane: A hyperplane is a decision plane that distinguishes between a group of objects with distinct class memberships.
* Margin: A margin is a distance between the two lines on the class points that are closest to each other. The perpendicular distance from the line to the support vectors or closest points is determined. A bigger margin between the classes is regarded as a good margin, whereas a smaller distance is considered a bad margin.

The main goal is to separate the given dataset as efficiently as feasible. The margin is the distance between the two points that are closest to each other. The goal is to find a hyperplane that has the largest feasible margin between support vectors in the dataset. In the following steps, SVM looks for the largest marginal hyperplane:

1. Create hyperplanes that effectively separate the classes. Three hyperplanes, black, blue, and orange, are shown on the left side of the illustration. The blue and orange classes exhibit more classification errors, while the black class correctly separates the two classes.
2. As illustrated in the right-hand figure, choose the right hyperplane with the highest segregation from the nearest data points.

**Decision Trees**

Decision Trees (DT) are one of the most intuitive classification techniques. The DT is essentially a series of decision rules for converting continuous data, such as spectral information from an image, into discrete thematic data, such as a land cover class. If a pixel's spectral information (or spectral alterations like vegetation indices) meets certain criteria, it will be allocated to a land cover class. Its structure, which is organized hierarchically in simple binary (yes/no) decisions, gives it the name "Decision Tree."

For classification and regression, Decision Trees (DTs) are a non-parametric supervised learning method. The goal is to learn simple decision rules from data attributes to develop a model that predicts the value of a target variable. A tree is an approximation of a piecewise constant.

**VA Files Indexing**

VA files or Vector Approximation files are a form of indexing structure used to locate images easily in higher dimensions. Higher dimensional objects are usually difficult to perform searches on due to their large computational complexities. So, in order to escape the complex and costly computations (dimensionality curse), VA files are used to prune the search space to a subset of the database for faster querying.

The idea behind VA Files is portioning of the space instead of the data. Each vector is approximated based on the cell in which it lies. This approximation is called a “signature” of the file and is used to filter on the search space. While evaluating the query, all approximations or signatures are scanned and only those that are having a good chance to be in the neighbourhood of the given query object are considered. The solution is searched only in this sub space instead of the entire space.

The vector approximation is generated as follows:

* Each feature of the image space is divided into equally distributed regions based on their values.
* Each region is then assigned a particular value based on the total number of bits.
* This process is repeated until all the features are divided into sub regions based on the number of bits.
* The final string is a concatenation of the approximated arrays, and the resulting VA-file is returned to the output.

The query is performed as follows:

* A query image is given by the user.
* The query image is then featurized as per the stored database model.
* A signature file is then computed for the query which is used as a filter to select the potential candidates for comparison.
* The actual distance from the query to the objects is computed only for the potential candidates and the top – k images from them are returned as the outputs.

**Local-Sensitive Hashing**

Local Sensitive Hashing or LSH is a hashing technique that puts similar items into “buckets” with a high probability if that object belongs to that bucket. It is a simple algorithm that is quite effective is pruning search spaces when the dataset to be searched is very large. The LSH Search for the nearest neighbour has two parameters, the width to be searched and the number of Hash tables (layers).

In the first step, we define a new family of hash functions g, where each function g is obtained by concatenating k functions (h1…hk) from F. In other words, a random hash function g is obtained by concatenating k randomly chosen hash functions from F. The algorithm then constructs L hash tables, each corresponding to a different randomly chosen hash function g.

In the pre-processing step we hash all n d-dimensional points from the data set S into each of the L hash tables. Given that the resulting hash tables have only n non-zero entries, one can reduce the amount of memory used per each hash table O(n) using standard hash functions. Given a query point q, the algorithm iterates over the L hash functions g. For each g considered, it retrieves the data points that are hashed into the same bucket as q. The process is stopped as soon as a point within distance cR from q is found.

LSH outputs the number of buckets it searched while finding the nearest neighbours, along with the miss rates and positive.

**2. Goal Description (Problem Specification)**

The project has been divided into eight main tasks. They are defined as follows:

1. **Task 1** - The aim of this task is to design a program that takes two folder of images, one of the three feature models, a user-specified value of X, a user-specified value of k, and one of the classifier models (SVM, Decision Trees, PPR based classifier) chosen by the user and outputs false positives and miss rates.
2. **Task 2 -** The aim of second task is to design a program that takes two folder of images, one of the three feature models, a user-specified value of Y, a user-specified value of k, and one of the classifier models (SVM, Decision Trees, PPR based classifier) chosen by the user and outputs false positives and miss rates.
3. **Task 3 –** Third task is to implement a program that takes two folder of images, one of the three feature models, a user-specified value of Z, a user-specified value of k, and one of the classifier models (SVM, Decision Trees, PPR based classifier) chosen by the user and outputs false positives and miss rates.
4. **Task 4 -** The fourth task is to implement locality sensitive hashing tool and implement similar image search using index structure.
5. **Task 5 -** Task 5 involves implementing a VA-file Index tool that stores the given folder of images in a VA-file data structure and when given an image and t, the VA-file index tool outputs t most similar images and outputs number of buckets searched as well as the unique images considering false positives and miss rates.
6. **Task 6 –** Task – 6 involves implementing a feedback mechanism using Decision Trees, that involves requesting the user to classify the returned results as “Relevant” or “Irrelevant” and based on the feedback, returns a set of revised results.
7. **Task 7 -** Task – 7 is similar to Task – 6, but instead of a Decision Tree, we use Support Vector Machine to implement a feedback mechanism, that involves requesting the user to classify the returned results as “Relevant” or “Irrelevant” and based on the feedback, returns a set of revised results.
8. **Task 8 –** Task – 8 is implementing an interface for Task-6 and Task – 7 that enables the user to provide a query image and returns n similar images. The user then provides feedback for the images and the query is revised and returns a set of revised results.

**3. Assumptions**

As the project is being developed as a proof of concept, there are several assumptions that have been made in-order to achieve the results. They are as follows:

* One of the main assumptions in the project is that the given set of images are all of the same nature and size within the image database folder i.e., all images are of size 64X64 (pixels).
* The second assumption is that all the given images are greyscale and there are no separate Color components to these images.
* The third assumption is that the database all contains significantly relevant images i.e. All images are of a single dataset and will not vary significantly in context or features from one-other. As such all given folders will be considered to be subsets of the “faces\_olivetti” dataset.
* The final assumption is that the given input image will also match the size and context of the images in the database.

# DESCRIPTION OF THE PROPOSED SOLUTION

In order to quantitatively reduce the dataset size and find latent features, the proposed solution is to transform the image in-order to obtain the latent features which are then represented into vector formats, enabling us to reduce the dataset size by representing only the top significant features and ignoring the insignificant ones, the premise being that the significant features represent the database effectively without losing a lot of information.

The subsequent idea is to classify the images provided in the folder. In order to achieve this, we use three classifiers:

1. Support Vector Machines
2. Decision Trees
3. Personalised Page Rank

One of these classifiers is chosen and trained on the dataset present in Folder 1 and used to classify the images stored in Folder 2. Once the images are classified, we use two methods to find the nearest neighbours:

1. Local Sensitive Hashing
2. Vector Approximation Files

There are two major process flows within the proposed solution architecture, one is to generate the feature vectors for all the images present in the folder. The second one is to generate the feature vectors for the given folder of images and classify them based on the trained model. Also, code design for each given task is as described below:

**1. Task – 1, 2 and 3:**

Diagram

Description automatically generated

Tasks 1,2 and 3 have similar dataflows, the only change is in the label to be predicted. Task – 1 asks for the prediction of image type. Task – 2 asks for the prediction of subject ID. Task – 3 asks for the prediction of sample ID of the given dataset of images.

The pseudocode for Decision Tree is as follows:

*Decision Tree*

*create\_decision\_tree(vectors\_df, counter=0, min\_support=0.6, min\_samples=2, max\_depth=5):*

*count\_df <- get frequency of all labels in vectors\_df of current tree node*

*support <- get probability of most frequent label*

*if len(classes in node:vectors\_df)==1 or len(rows in node:vectors\_df) < min\_samples or max\_depth reached or support > min\_support:*

*return most frequent class label*

*else:*

*all\_splits <- get all column splits*

*split\_column, split\_value <- determine best split from node:vectors\_df*

*data\_below, data\_above <- split data in node:vectors\_df using split\_column, split\_value*

*sub\_tree <- initialise empty sub-tree*

*left\_sub\_tree <- create\_decision\_tree(data\_below, counter + 1, min\_samples, max\_depth)*

*right\_sub\_tree <- create\_decision\_tree(data\_above, counter + 1, min\_samples, max\_depth)*

*APPEND(sub\_tree, left\_sub\_tree)*

*APPEND(sub\_tree, right\_sub\_tree)*

*return sub\_tree*

The pseudocode for SVM is as follows:

Design Decisions for Tasks-1,2 and 3:

**2. Task – 4:**

Task – 4 consists of pruning the search space using the LSH algorithm to find the t-nearest neighbours. LSH is more effective in searching as it searches only a subspace of the entire dataset instead of a sequential scan of the entire database.

Block Diagram for Task – 4

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Pseudocode for Task – 4:

*LSH(L, K, vectors, num\_obj):*

*LSH <- empty list*

*for all L:*

*APPEND(LSH, Create LSH\_family(K, num\_obj, vectors))*

*LSH\_family(K, num\_obj, vectors):*

*plane\_norms <- Create random vectors between 0 to 1*

*plane\_norms <- subtract 0.5 from plane\_norms for range between -0.5 to 0.5*

*LSH\_family <- empty list*

*for all K:*

*APPEND(LSH\_family, Create LSH\_node(w, vectors, b, plane\_norms))*

*LSH\_node(w, vectors, b, plane\_norms):*

*buckets <- empty dictionary*

*for all rows in vectors:*

*v\_dot <- apply transformation ((vectors-0.5)\*plane\_norms\*w + b) / w*

*v\_dot <- apply v\_dot > 0*

*row\_hash <- STRING(INT(v\_dot))*

*ADD buckets with row\_hash(key) and index(value)*

Design Decisions for Task – 4:

**3. Task – 5:**

Task – 5 consists of developing a VA File data structure, that can be used to prune the search space. It calculates a signature file based on the user input and uses that signature file as a simple filter to prune the results.

Diagram

Description automatically generated

**Diagram, timeline

Description automatically generated with medium confidence**

The pseudocode for Task –5(VA file generation) is as follows:

*Start Program*

*Get the image input matrix*

*B = get number of bits to be used for VA from the user*

*Function va\_gen(inpMat, b):*

*nRegions = Compute the number of regions for each dimension/feature*

*buckets = obtain the buckets of values for each dimension/feature*

*generate the VA file based on the buckets*

*return VAfile,limits*

*end va\_gen*

The pseudocode for Task –5(top k values query) is as follows:

*Table

Description automatically generated*

Design Decisions and details noted while program execution:

* It has been noted that the number of bits “b” value, must be provided in such a way that it is a multiple of the number of dimensions of the feature model chosen. (i.e., If Color Moments is chosen as feature model, then the min number of bits must be 192\*2 = 384). So, the optimal values for the models have been noted below:

|  |  |  |
| --- | --- | --- |
| Model | Number of bits to be represented for each dimension in VA | |
| 1st Optimal Value | 2nd Optimal Value |
| Color Moments | 786 (192\*4) | 576 (192\*3) |
| ELBP | 12288(4096\*3) | 16384(4096\*4) |
| HOG | 7056(1764\*4) | 8820(1764\*5) |

* For VA files, distance method chosen is L3-norm or Minkowski norm, as it has been established via multiple runs that computing the L3 norm provides the best results with minimal performance trade-off.
* It has also been noted that with increasing the VA file size, the search subspace decreases. However, the benefits of having a larger VA file diminishes after the 6th multiple. For best results, any multiple between 3 and 5 is preferred [1].

**4. Task – 6:**

Task – 6 consists of incorporating the user feedback after receiving the nearest neighbour results and using them to revise the results obtained using a DT based paradigm and display the new results:

Block diagram for Task – 6:

Diagram

Description automatically generated with medium confidence

Pseudocode for Task – 6:

*Start Program*

*inp = Get input from user for image*

*t = Get input from user for t value*

*nn = LSA (database, t) (‘or) VA (database, t)*

*feedback = Get feedback from user for the images in ‘nn’*

*train DT(nn,feedback)*

*revised\_results = predict (“nn”)*

*return revised\_results*

*End Program*

Design Decisions for Task – 6:

**4. Task – 7:**

Task – 7 consists of incorporating the user feedback after receiving the nearest neighbour results and using them to revise the results obtained using an SVM based paradigm and display the new results.

Block diagram for Task – 7:

Text

Description automatically generated with low confidence

Pseudocode for Task – 7:

*Start Program*

*inp = Get input from user for image*

*t = Get input from user for t value*

*nn = LSA (database, t) (‘or) VA (database, t)*

*feedback = Get feedback from user for the images in ‘nn’*

*train SVM(nn,feedback)*

*revised\_results = predict (“nn”)*

*return revised\_results*

*End Program*

**5. Task – 8:**

Task – 8 is simple. It just acts as an interface to enable the user to provide feedback to Tasks 6 and 7.

# INTERFACE SPECIFICATIONS

The entire code has been developed and been tested using Python programming language, running on PyCharm and VS Code IDEs. The main.py script is the primary runtime script for the entire code.

The instructions to run the program are as follows:

1. Navigate to the src folder in the deliverable.

Graphical user interface, application

Description automatically generated

1. Open the “config.config” file in any IDE of choice (Visual Studio Code).

Text

Description automatically generated

1. Edit the path variables as necessary in “base0\_dir”.

Text

Description automatically generated

1. Navigate to main.py script in the Code folder

Text

Description automatically generated

1. Run the “main.py” file
2. All computed outputs will be in the respective Output folders.

# OUTPUTS

1. **Output for Task – 1 & 2**: (These are the csv files that contain the latent features)

2. **Output for Task-3&4**: (We get a subject-subject or type-type similarity)

3. **Output for Task – 5:** Output of ‘k’ similar images

4. **Outputs for Task- 6 and 7:**

6. **Output for Task- 8:**

# SYSTEM REQUIREMENTS

The main requirements for the Image Similarity program to run is the python programming language and any IDE of choice. Python can be installed in the system using the given steps:

1. Open web browser of choice and navigate to <https://www.python.org/downloads/>
2. Select the most recent/desired version of python version.
3. After the EXE file finishes downloading, click on Run.
4. Follow the instructions on the screen to setup python on the machine.

Python usually comes with a package manager called “pip”. This can be used to install further dependencies. First is the IDE Jupyter notebooks. This can be installed with the command:

pip3 install notebook

After installing Jupyter, we have to install the required libraries for the program. Each library can be installed using a standard command

pip install <library name>

*(Replace with respective library)*

1. **Numpy:** This is the standard python library for dealing with arrays. It can be installed with the command : pip install numpy
2. **Pandas:** Pandas library enables to store arrays as Dataframes, which make it easier to work with multi-dimensional data.
3. **Matplotlib:** This is the library required to visualize the vectorized images as normal 2-D images on screen.
4. **Math:** Basic math functions library. Used for rounding to the nearest decimal places.
5. **Scipy:** Library used to perform scientific calculations. In this instance it is used to access skew function.
6. **Skimage:** This library is used to compute the ELBP and HOG vectors. It contains many feature vectorization methods that can be used to extract features from images.
7. **PIL (Pillow):** This is the python imaging library. It contains many functions that make working with images easier.
8. **Glob:** This library is used to create iterable lists of file paths that can then be used in standard looping functions.
9. **OS:** Standard python library that contains functions to deal with operations related to Operating systems. It is used to access filepaths and write to files.
10. **Cvxopt:** Standard python library that contains efficient Python classes for dense and sparse matrices (real and complex), with Python indexing and slicing and overloaded operations for matrix arithmetic.

Also, the above dependencies are mentioned with a view in mind that the underlying operating systems is Windows. In case of MAC or Linux, the installation procedures may differ slightly.

# RELATED WORK

There have been references to several related works in achieving the final outputs of the program.

For implementation of the K-Means Data Dimensionality reduction, a blog on Image compression by Himanshu Sharma [1], which speaks on how to compress images using color space reduction using K-means and visualizing the results. A K-means is applied on the image space and each separate colour is replaced by its centroid reducing the color space used to represent the image.

For K-Means Clustering, we also referred to “A New Method for Dimensionality Reduction Using K-Means Clustering Algorithm for High Dimensional Data Set” [2]. Clustering is the process of finding groups of objects such that the objects in a group will be similar to one another and different from the objects in other groups. K-means clustering algorithm often does not work well for high dimension, hence, to improve the efficiency, apply PCA on original data set and obtain a reduced dataset containing possibly uncorrelated variables. In this paper principal component analysis and linear transformation is used for dimensionality reduction and initial centroid is computed, then it is applied to K-Means clustering algorithm.

For identifying m most significant subjects using Personalised Pagerank measure, the major point of reference was Shengyu Huang, Xinsheng Li. K, Selcuk Candan, Maria Luisa Sapino(2013)[4] work on “Reducing seed noise in Personalised Page Rank”. In this paper, the authors have concluded that conventional personalised page rank algorithm associate unnecessarily high bias to the seed nodes and this negatively affects the node rankings when the seed set is incomplete or noisy. To deal with this problem the authors have proposed alternative robust personalised pagerank algorithm that eliminates the potential noise in the seed set. The experiment results confirm that the seed-set maximal approach is reuse promoting in that it is possible to divide the work relative to individual seed nodes and teleportation discounting technique provides additional robustness against noise introduced during graph-partitioning (and block diagonalization) based approximate random walk computation processes.

# BIBLIOGRAPHY

[1] K-Means for Dimensionality Reduction by Himanshu Aggarwal <https://analyticsindiamag.com/beginners-guide-to-image-compression-using-k-means-clustering/>)

[2] Dhavamany, Napoleon & S, Pavalakodi. (2010). A New Method for Dimensionality Reduction Using K-Means Clustering Algorithm for High Dimensional Data Set. International Journal of Computer Applications. 13. 10.5120/1789-2471.

[3] Chen, Hung-Hsuan & Giles, C.. (2015). ASCOS++. ACM Transactions on Knowledge Discovery from Data. 10. 1-26. 10.1145/2776894.

[4] Huang, S., Li, X., Candan, K.S. et al. Reducing seed noise in personalized PageRank. Soc. Netw. Anal. Min. 6, 6 (2016). https://doi.org/10.1007/s13278-015-0309-6

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res. (3/1/2003), 993–1022