CSE 515 (Fall2016) Phase 3 Report: Group 17

ANSHUMAN BORA, Arizona State University
VRAJ DELHIVALA, Arizona State University
SATYAM JAISWAL, Arizona State University
SAUMYA PARIKH, Arizona State University
TANMAY PATIL, Arizona State University

Abstract

Finding similarity of frames between multiple videos is not enough. They also have to be listed according to some kind of importance or relevance. In phase2, we listed them by sorting with respect to the similarity index. The frame with highest similarity came first and the top *k* frames were selected as most similar sub-sequence. In phase3, we have improved the listing logic by implementing intelligent algorithms which list similar frames by significance, relevance and also by nearest neighbor importance in high dimensional spaces. SIFT vectors in a reduced space have been used as input considering they describe important visual features. Algorithms like PageRank and ASCOS++ have been used for ranking according to significance. Relevance feedback based algorithms like personalized PageRank and personalized ASCOS ++ have been used and nearest neighbor based technique of Locality Sensitive Hashing has also been used. This report talks about all these implementations in detail.

Keywords

Feature Extraction, Similarity Graph, Significant Frame, Relevant Frame, Page Rank, ASCOSS, Personalized PageRank, Locality Sensitive Hashing, Multi-dimensional Indexing

1. INTRODUCTION

1.1 Terminology

a) Clustering: Clustering is defined as grouping data or points into groups such that more similar data are in the same group. Clustering comes under unsupervised learning.

b) Indexing: Is a mechanism with which access to large data managed in files. [1]

c) Classification: This is a method in which data is split into predefined classes. It comes under supervised learning.

d) Relevance feedback: Is a mechanism used in algorithms where the user gives feedback on the relevance of documents in an initial set of results. [2]

e) PageRank: PageRank is an algorithm which is used by Google to rank the webpages on the internet. [3]

f) ASCOS++: Asymmetric Network Structure Context Similarity, takes into account all the nodes and the weight of edges between the nodes for calculation of similarity measure. [4]

g) LSH: Locality sensitive hashing, algorithm for solving the approximate or exact Near Neighbor Search in high dimensional spaces. [5]

h) Random Walk: how soon a person walking from a random node will arrive at the destination with random intermediate nodes

i) Seed node: Special nodes which are given preference in order to provide a biased result in favor of the seed nodes.

j) Guessing Matrix: The initial similarity matrix chosen in such a way that it gives the algorithm to converge.

k) Hash function: A hash function is any function that can be used to map data of arbitrary size to data of fixed size. The values returned by a hash function are called hash values, hash codes, digests, hash sums, or simply hashes.

l) Hash table: a hash table (hash map) is a data structure used to implement an associative array, a structure that can map keys to values. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be found.

1.2 Goal Description

Phase 3 involves around experimenting with clustering, indexing, classification and relevance feedback. The detailed goal description are as follows:

1. Task1: In task1 we had to implement Video Feature Extraction algorithm where Principal Component Analysis (PCA) had to be applied to the original space to get a reduced vector space.
    a. Input: The input to this algorithm is '*d*' (the target reduced dimensionality) and *'filename.sift'* (file containing SIFT features)
    b. Output: the output would be a file named 'filename_d.spc' which would contain entries in the form of *{<i, j, l, x, y>, [dim-1 , . . , dim-d]}*, such that i = video, j = frame and l = cell indices. x and y and provide the position of the SIFT key point and [dim-1, . . , dim-d] are the reduced dimensions. The program is also expected to output the d dimensions in the form of <*original_index, score*> in non-increasing order of scores.

2. Task2: In task2 we have to create a similarity graph using the reduced SIFT vector space.
    a. Input: Integer *K* and SIFT key points in file *'filename_d.spc'*.

   b. Output: Graph(V, E) where V are the nodes corresponding to each frame and E are the edge pairs corresponding to <va, vb> where vb is one of the most similar frames to vb, from a different video that vb. This graph will be stored in *'filename_d_k.gspc'* in the format of *<va, vb, sim(a,b)>*

3. Task3: The goal of task3 is to list most similar frames according to significance. This is to be done using 2 methods namely, PageRank and ASCOS++.

4. Task4: This task is similar to the previous task. The only difference is that the similar frames are to be listed according to relevance. The algorithms to be used are Personalized PageRank and Personalized ASCOS++.
   a. Input: Filename *'filename_d_k.gspc'* and integer *'m'* where m is number of the most significant frames to be listed.
   b. Output: Visualize the *m* frames for PageRank and ASCOS++ both.

5. Task5: This task involves implementing multi-dimensional index structures and nearest neighbor search using Locality Sensitive Hashing (LSH) tool.
   a. Input: Filename *'filename_d.spc'* , number of layers *L*, the number of buckets *2^k.*
   b. Output: File *'filename_d.lsh'* containing entries in the form {layer_num, bucket_num, <i;j;l;x;y>}.

6. Task6: A similarity-based video object search tool is to be implemented.
   a. Input: LSH index file, integer n, object in the form *{i ,j, <x1, y1>, <x2, y2>}* such that <x1, y1> & <x2, y2> is a rectangle containing the object.
   b. Output: Visualize the top n frames. Also output should contain:
      i. Number of unique SIFT vectors considered.
      ii. Number of overall SIFT vectors considered.
      iii. Number of bytes of data from the index accessed to process the query.

1.3 Assumptions

For this phase, we have made the following general and specific assumptions:

Overall:

1. The frames of the video are divided in cells, where cell 1 is the top left cell of the frame.
2. While comparing two frames, we have performed the comparison on corresponding cellular level.
3. All the decimal values for SIFT, vectors have a precision of 6.
4. For every task in this phase, the file video_mappings_task3.csv is always available. This file contains mappings of the video numbers used in our code to the actual video indices in the database.
   Mapping format: Video Number, Actual Video Name.
   Example: 31, 37.mp4
5. Each node in the following tasks represents a unique frame of the input video files.
6. The similarity value is out of 100. So a similarity value of 95 suggests a similarity of 95%.

Task1:

1. The input values k and d will always be less than equal to the total number of dimensions.

Task 2:

1. We assume that dividing frames into cells takes care of spatial position of the SIFT vectors while calculating the similarity between cells and eventually calculating similarity between frames.
2. Task 2 assumes that it is given the same video map that is used to extract sift vectors from the videos

Task3:

1. The convergence criterion for PageRank is checked at every iteration.
2. The convergence criterion for ASCOS++ has been fixed at 25 iterations for faster calculations.
3. A guessing Matrix is needed as input in PageRank as well as in ASCOS++. The matrix is constructed with the following formulae: If i represents a node and j represents another node and ij gives the edge between the two nodes, then if there is an edge between the two nodes, ij will have a value of 1, else ij will be 0.
4. The Jacobi iterative method package is available for calculations of ASCOS++.
5. The value of Damping factor, c is 0.85 in PageRank calculations.
6. The value of discounted parameter, d is 0.9 in ASCOS++ calculations.
7. The PageRank for every node is initialized with a value = 1/n, where n is the total of nodes in our graphs.
8. Tolerance factor for Jacobi calculation is 0.9.
9. The value of m is 6 for this Task.
10. The output is an array of objects which could be can be accessed through the command line. The actual output is a set of images that are shown after the execution of the program. A TIFF file is also stored for future reference.
11. Input file is available from task2.
12. Convergence for PageRank is determined by checking the MaxError rate allowed at every iteration. Once the error falls under the MaxAllowedError, we stop the iterations.
13. Max error rate is set at 0.001 is a feasible choice

Task4:

1. The convergence criterion for PageRank is checked is 100 iterations.
2. The convergence criterion for ASCOS++ has been fixed at 25 iterations for faster calculations.
3. A guessing Matrix is needed as input in PageRank as well as in ASCOS++. The matrix is constructed with the following formulae: If I represents a node and J represents another node and S{IJ} gives the edge between the two nodes, then S{IJ} exists then it will have a value of 1, else will be 0 for that particular element in the matrix.
4. The Jacobi iterative method package is available for calculations of ASCOS++.
5. The value of Beta is 0.15 in PageRank calculations.
6. The value of discounted parameter, d is 0.9 in ASCOS++ calculations.
7. The PageRank for every node is initialized with a value = 1/n, where n is the total of nodes in our graphs.
8. Tolerance factor for Jacobi calculation is 0.9.
9. The value of m is 6 for this Task.
10. The output is an array of objects which could be can be accessed through the command line.

11. The actual output is a set of images that are shown after the execution of the program. A TIFF file is also stored for future reference.
12. Input file is available from task2.
13. The guessing matrix for personalized ASCOS++ we start with something similar to task 3 with the added job of increasing the weights (multiply by 5) of all the outgoing nodes of the given seed frames(nodes).

Task5:

1.We assume that a bucket at any layer can hold infinite number of SIFT vectors.

2. We assume the partition value of the space should be calculated considering the maximum and minimum values of each dimension and the number of buckets in each layer.

3. To calculate the partition value 'w' we divide the mean value by 16 being the default value of bins in hash table.

Task6:

1. Hash function family which was used in task 5, should be provided as '.mat' file for this task.
2. Output from Task 1 should be provided.

2. IMPLEMENTATION

2.1 Task 1: Video Feature Extraction

Nice. For vectors, such as SIFT vectors the number of dimensions can go as high 132. Computing similarity between video frames using such long vectors becomes a tedious task. And the extra work is wasted, as a lot of these dimensions are not important, or significant to the overall identification of the feature at that point. Thus, their contribution to the similarity measurement is negligible and can be removed. This process is done using various dimensionality reduction techniques. The most relevant features of the video were extracted after this task. Principle Component Analysis was applied on the dimensions of the video that we got from output of phase II.

This algorithm gives principal components from a set of dimensions. Each principal component returned will be a linear combination of the original columns or dimensions[6]. The pca() function of MATLAB returns various values. For dimensionality reduction, we consider the score, coefficients and latent. To express the same data in the new coordinates formed by the principal components, the new first dimension should be a linear combination of the original ones. This is given by the score, or by calculating M*coeff (where M is the input matrix). The importance of each principal component can be determined by how much variance of the data it explains. This is given by latent. These are in fact the eigenvalues. For calculating the PCA, we took the feature vectors of each frame of the videos. These were concatenated to form a matrix. So, if n = number of frames, and c = number of dimensions (c = bins for histogram, c = 132 for SIFT, etc.).

1. We would get an $n \times c$ matrix which would contain the entire feature vectors. This matrix would be then passed to the pca function, which would return the score, coefficients and latent, using the eigenvalue decomposition method.

2. The latent vector is then sorted, and the order of the indices is recorded. The score matrix is then rearranged, based on the order of the indices. This matrix represents the sorted principal components matrix. The dimensions can be reduced by removing the last j columns, which have the least variances, and are consequently the least significant dimensions. Thus an $n \times c$ matrix is reduced to $n \times (c - j)$.

Output Format

Output 1: {<video, frame, cell, x coordinate, y coordinate>,[descriptor_1, descriptor_2,..descriptor_d]}

Output 2: <original_index, score>

Example

Example 1: {<49,1,1,122.830970,151.154905>,[3.877580, 2.538420, 0.112462, 0.006702, 0.301454, -0.307601, -0.172356, -0.042774, -0.039154, -0.056550]}

Example 2: <135 ,0.884598>

2.2 Task 2: Video Frame Similarity Graph Generation

To create similarity graph of the video frames, we calculated the similarity value of the frame with every frame in other videos. For this once we had the reduced sift vectors from PCA, we processed the output into a map of the sift vectors. This map at level 1 has video map keys, then at level 2 has frame map keys, at level 3 has cell map keys and finally at level 4 has the sift vectors.

To calculate the frame to frame similarity we granulated the frames into and did cell-2-cell distance computation. For cell-2-cell distance computation we compared every vector in frame 1 cell 1 to every vector frame 2 cell 1. These distances are then averaged out. We chose to average the distances because, averaging gives a cell leader of the cell which can be used for calculating the distance.

Similarly, we averaged out once we have all cell-to-cell distances to find frame-2-frame. We chose to average the distances at frame level because contribution of each cell is important in find frame to frame distance.

All the frame to frame distances are then combined into one matrix and normalized followed by multiplying with 100 to give the similarity measure. We sorted using similarity and picked up the k similar frames.

Output Format

{<source video, source frame>, <destination video, destination frame>, similarity value}

Example

{<63, 14>, <17, 16>, 98.947183}

2.3 Task 3: Most Significant Frame Selection

2.3.1 PageRank:

PageRank(PR) algorithm calculates how important a page, or a node in our case, is by considering how many pages point to it. This is a random walker model, which tells us how soon a person walking from a random node will arrive at the destination with random intermediate nodes. It is an iterative approach, with each step bringing us closer to the desired result or convergence. We start by giving each node an equal PR value given by 1/n, where n is the total number of nodes in the graph. We introduce a damping factor, d, which is used to bring in the factor of randomly travelling to a different page. We take its value as 0.85, as this value give the best possible solution. The equation looks something like this:

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$

Where, PR(Tn) = PageRank of each page.
C(Tn) = Each page evenly spreading its PageRank among its outgoing links.

We have used power iteration method in this approach.

Code Level Implementation:

1. We create an adjacency matrix of size n x k(where n is the total frames and k is the no of most similar frames from task2)
2. We create an IncomingMap where the key is of the form Video,Frame and it stores all incoming Nodes in the form of array of objects.
3. We also create a PageRankMap where the key is of the form Video,Frame and it stores the PageRankValue.
4. So we basically used Array type data structures to index and Map Type data structures to search. This saves us a lot of computation time but we further improve this in the Personalised PageRank Approach

2.3.2 Ascos++:

ASCOS, Asymmetric Network Structure Context Similarity, states that the similarity score from a node i to a node j is dependent on the similarity score from node i's in-neighbors to node j. ASCOS++ is a similarity measure which enriches ASCOS by including the weighted paths between all the nodes of a network. Its general formula is as follows:

$$s_{ij:=} \begin{cases} c \cdot \sum_{\forall k \in N(i)} \frac{w_{ik}}{w_{i*}} (1 - exp^{-w_{ik}}) s_{kj}\ , & if\ i \neq j \\ 1 & if\ i = j \end{cases}$$

Where w_{ik} is the weight of edge e (i, k), and $w_{i*} = \sum_{\forall k \in N(i)} w_{ik}$, c is the discounted parameter which is used to control the relative importance of direct and the indirect neighbors. We take its value as 0.9 for our calculations. S_{ij} gives the similarity measure between two nodes i and j.

Initially we used the naïve approach for implementation. But the program was taking too much time to execute, more than an hour to be precise, to run a single iteration. Hence, we decided to move on to the distributed approach. The distributed approach uses the following formula:

$$(\boldsymbol{I} - c\boldsymbol{Q}^T)\boldsymbol{S}_i = (1 - c)\boldsymbol{I}_i, i = 1,2, \ldots , n$$

Where **A=** $[a_{ij}]$ is the adjacency matrix of a graph G, **P** is the column normalized matrix of **A** and **Q** = $[q_{ij}]$=$[p_{ij}(1 - \exp(-a_{ij}))]$**.** This turns ASCOS++ into a classic systems of linear algebra equations, in which $\boldsymbol{I} - c\boldsymbol{Q}^T$ is a coefficient matrix with dimension n by n; S_i is an unknown column vector with n variables to be solved; and $(1 - c)I_i$ is a constant column vector of size n. We choose the initial value of adjacency matrix such that if two nodes, i, j has an edge between them then the value of similarity, $S_{ij} = 1$ otherwise it is 0.

We then apply the Jacobi iterative method to solve this system of linear equation. We use the Jacobi because this approach possesses high degree of natural parallelism for distributed computation.

Code Level Implementation:

1. During the preliminary reading of the ASCOS++ paper we began implementing the Naive Approach that involved finding relations for every pair of nodes.
2. Due to the Astronomical computation time this took we shifted to Distributed Computation by solving N Linear Equations.
3. The paper had a mention of Jacobi Iteration which we used.
4. We had a Adjacency Matrix, A, Guessing Matrix, S a Column Normalized matrix ,P
we get the following equations by code

```
Q = P.*(Ones - exp(-(adjMatrix)));
   IdentityMat = eye(size(adjMatrix, 1));
   c = 0.9;
   AMatrix = (IdentityMat - (c .* Q'));

   clear Sim;
   tic;
   parfor i = 1:size(adjMatrix, 1)
      i
      BMatrix = (1 - c) * IdentityMat(:,i);
      Sim(:, i) = jacobi(AMatrix, BMatrix, GuessMatrix(:, i), c, 25);
   end
```

<u>Running Task 3.</u>

1. Add the entire task 3 folder to the path, making sure all other paths are removed since we have conflicting file names in different folders.
2. Open Task3new.m and in your console assign the following variables.
videoDirectory = Directory containing all the Demo videos
dataFileName = address of the input file- filename_d_k2_d10.gspc
m = no of most significant frames
3. Run the Following functions to see separate outputs for PageRank and ASCOS++
pageRankTask3(videoDirectory, dataFileName, m);
ascosMeasures(videoDirectory, dataFileName, m);

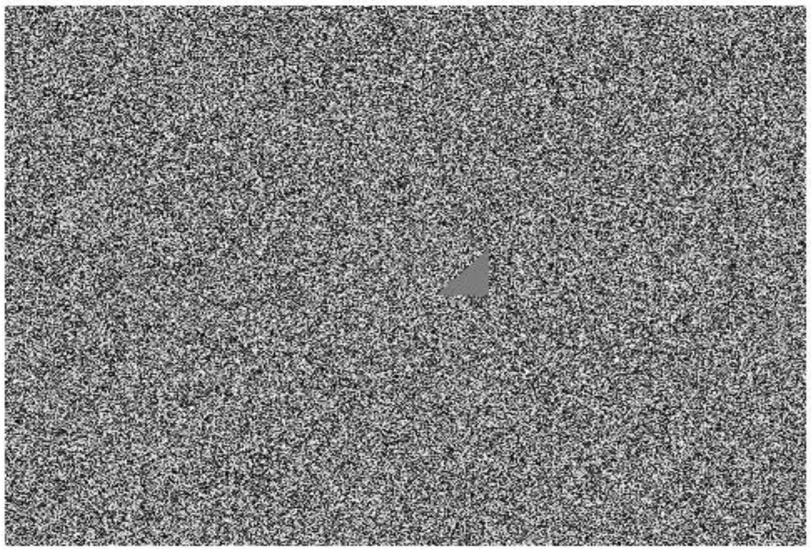
Output Format

Video Name-_ _ Video Number -_ _ Frame Number - _ _ PageRank - _ _[image]

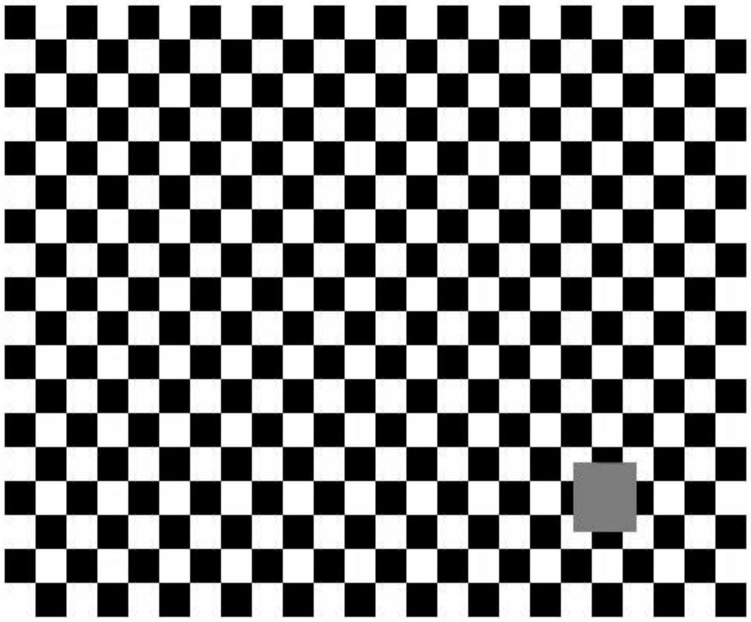
Video Name-_ _ Video Number -_ _ Frame Number - _ _ ASCOS++ - _ _ [image]

Visualization

Video Name -26.mp4 Video Number -19 Frame Number -20 ASCOS++ -370.598



Video Name -22.mp4 Video Number -15 Frame Number -28 PageRank -111.6273



2.4 Task 4: Most Relevant Frame Selection

In Task 3 ,we found most significant frames from a collection of frames but these frames might always not be relevant to a user. Hence we consider seed frames as reference/ relevance nodes to start off and the pagerank and ASCOS++ measures are measured according to this.
Personalization of these measures is basically making more information through these seed nodes, i.e instead of the random walk approach and power iteration in page rank we use these seed nodes where a random user can jump back to, hence we are ideally using a restart vector.

Personalized Page Rank

1. We have implemented the RPR-2 algorithm for the personalized pagerank method.
2. There are 3 seed nodes as input where they are a combination of video number and frame number.
3. A restart vector is used where all indices of the nodes are set to zero expect the indices of the seed nodes which are set to one.
4. For 3 seed nodes there will be 3 restart vectors.
5. The value for max iterations have been set to 100.
6. Out of the seed nodes, a subset is used to get the final pagerank vector. It is called as SCRIT.
7. If SCRIT is singleton then the pagerank vector associated with it is returned as the final pagerank vector.
8. If SCRIT is not singleton then the summation of all pagerank vectors is divided by the SCRIT set and that value is returned as the final pagerank vector.

Personalized ASCOS++

This is almost the same as normal ASCOS but instead we are multplying the entire corresponding row of a seed node in the guessing matrix by 5 so as to give them a higher starting guess. Else the code for ASCOS++ is same as that in task3.

<u>Running Task 4.</u>

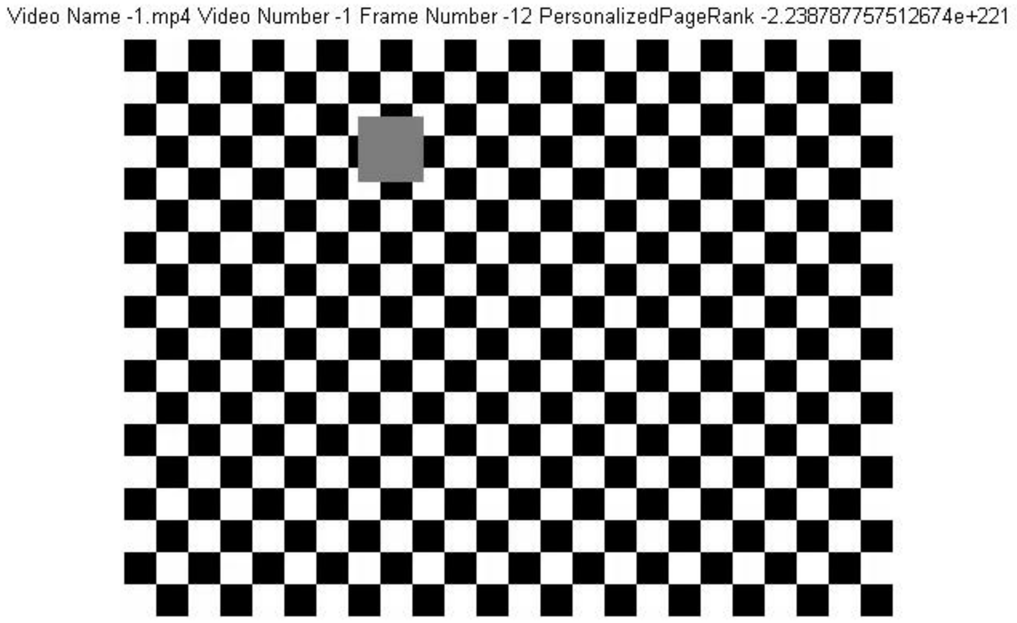
1. Add the entire task 4 folder to the path, making sure all other paths are removed since we have conflicting file names in different folders.
2. Open Task4Combo.m and in your console assign the following variables.
videoDirectory = Directory containing all the Demo videos
dataFileName = address of the input file- filename_d_k2_d10.gspc
m = no of most significant frames
seed1Str, seed2Str, seed3Str- the three seed nodes, in the format – ‘VideoNum,FrameNum’
3. Run the Following functions to see separate outputs for PageRank and ASCOS++

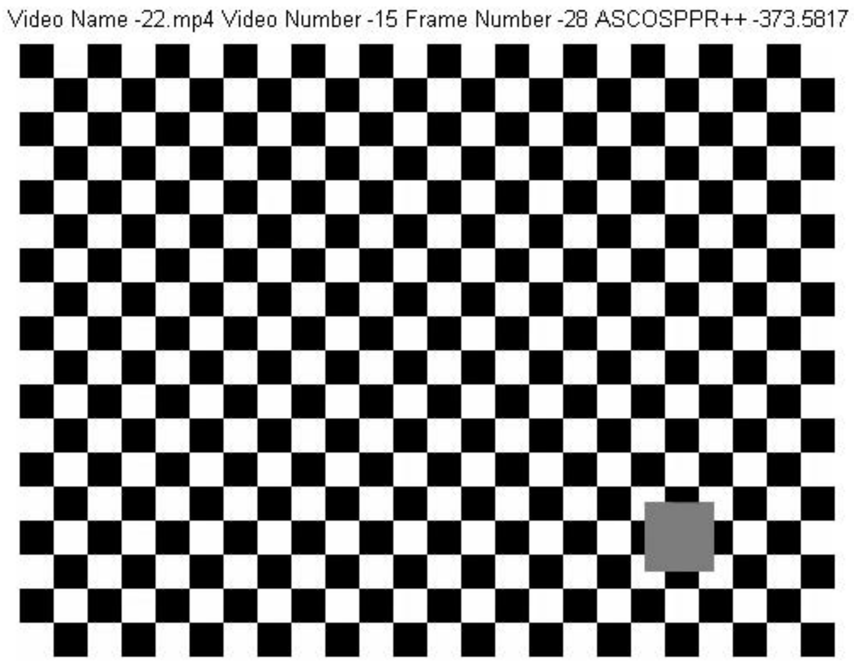
```
task4( videoDirectory, dataFileName, m, seed1Str, seed2Str, seed3Str );
personalizedAscosMeasures(videoDirectory, dataFileName, m ,seed1Str, seed2Str, seed3Str);
```

Output Format
Video Name-_ _ Video Number -_ _ Frame Number - _ _ Personalized PageRank - _ _[image]

Video Name-_ _ Video Number -_ _ Frame Number - _ _ Personalized ASCOS++ - _ _ [image]

Visualization


Video Name -1.mp4 Video Number -1 Frame Number -12 PersonalizedPageRank -2.238787757512674e+221


Video Name -22.mp4 Video Number -15 Frame Number -28 ASCOSPPR++ -373.5817

2.5 Task 5: Multi-dimensional Index Structures and Nearest Neighbor Search

We employ Euclidean locality sensitive hashing to build a data structure that solves the optimization problem to find a point which minimizes the objective function of distance to the query point.

Given the L and K we create L hash-tables and 2K buckets in each frame. Of all the PCA reduced SIFT vectors we map each vector to each hast-table and record its bucket number. This is done because the nearest neighbor search algorithm will not be efficient we don't use the family of hash functions. Applying the family of hash functions increases the gap between the probability that point is within distance R and outside distance cR. Here R is threshold or the required distance and cR is approximate distance.

To map the vector to a bucket of a hash table we use a family of hash functions H. Each layer is assigned a hash function. The hash functions are randomly generated. The sift vector is projected on a random 1D line in whose co-ordinates are picked from Gaussian distribution. This is then shift by a random value and finally the line is partitioned into sections of length w.

Formula: hr, b = $(r \cdot x + b)/w$,

h is hash function

r is random projection line

x is vector in consideration

b is random sift value

w is partition value

We implemented this by generating random values of the size of the dimension and multiplied it with the vector. Similarly, we generated a random value between a continuous distribution of 0 and w. The value w is calculated by first listing the max and min value of each dimension then took the max of the absolute value from max and min. We considered these values as minimum by negating them a maximum when positive. We then record the change in the minimum and maximum value of each dimension. This change is doubled and multiplied by the square root of the number of number of buckets in each frame. 'w' is then calculated by taking the mean of all values divided by 16. Number of buckets plays an important part in getting 'w' because considering the hash table to be the entire space dividing into number of buckets is important. Similarly, max and min values of each dimension is important because the space exists in that dimension and so are the vectors.

Output Format
{Layer number, bucket number, <video number, frame number, cell number, x,y>}

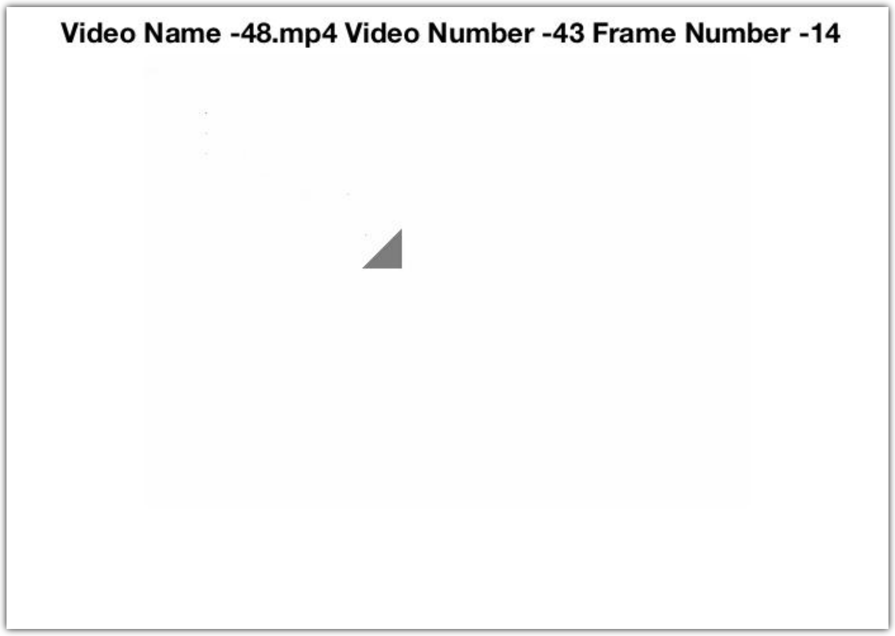
Example
{3, 7, <1; 1; 1; 23.434244; 23.434244>}

2.6 Task 6: Similar Video Object Search

We start with extracting the sift vectors of the section of the video marked with the given co-ordinates. These sift vectors are processed for dimensionality reduction with PCA. We then use the Hash-table and family of hash functions from task 5 to find the buckets where the sift vectors would fall. The vectors of found vectors are collected in a matrix. We then find the distances of each vector in input video frame section and the found vectors. The minimum value distances are picked grouped into frames and listed out as most significant frames. Special weightage has been given to a sift vector if the vector has been repeated multiple times.

For distance calculation we again use Euclidean distance function because it this suggested in many research papers as the best measure

Visualization



3. INTERFACE SPECIFICATION

1. All codes regarding this phase must run from MATLAB.
2. All the videos should also be placed in the same directory.
3. The output files will be created in the same folder.
4. Open the following files and input appropriate parameters into the functions for the corresponding tasks.
    - Task 1: pcaReductionSift.m
    - Task 2: run.m
    - Task 3: Task3new.m
    - Task 4: Task4Combo.m
    - Task 5: run.m
    - Task 6: task6.m

4. SYSTEM REQUIREMENTS THE OUTPUT FILES WILL BE CREATED IN THE SAME FOLDER.

1. The MATLAB software.
2. Minimum 16Gb RAM is required for performing all the tasks successfully.

5. RELATED WORKS

5.1 PageRank

PageRank algorithm is heavily used in a wide range of fields. It is used in recommendation systems like Amazon to recommend shoppers about their next purchase and Netflix users about which movie to watch next. This algorithm has really helped improve the user experience in these sites. Another major use of this algorithm is in ranking of tweets in the social networking site Twitter. It is not possible for Twitter, which generate millions of tweets on any given day, to show all tweets for a search query. It uses this algorithm to improve the search results.[7]

5.2 ASCOS++

ASCOS++ is a measure which have had very limited application so far. As this measure captures the similarity scores among any pairs of nodes in a network, it has been used in discovering similar objects in a social network. [8]

5.3 Locality Sensitive Hashing

LSH is used in Near Duplicate Detection, an approach in which is used in web crawling to determine similar kinds of websites. [9] It is also used for hierarchical clustering of data, an approach which seeks to build a hierarchy of clusters, either bottom up or top down. [10] LSH is also heavily used in Acoustic fingerprinting, a process in which audio signals are converted into a condensed state which helps in finding similar audio samples from a database. [11]

6. CONCLUSION

Hence this in this phase we experimented with clustering, indexing, classification and relevance feedback. In phase3, we have improved the listing logic by implementing intelligent algorithms which list similar frames by significance, relevance and also by nearest neighbor importance in high dimensional spaces. SIFT vectors in a reduced space have been used as input considering they describe important visual features. Algorithms like PageRank and ASCOS++ have been used for ranking according to significance. Relevance feedback based algorithms like personalized PageRank and personalized ASCOS ++ have been used and nearest neighbor based technique of Locality Sensitive Hashing has also been used

BIBLIOGRAPHY

1." LSH Algorithm and Implementation (E2LSH)". MIT, http://www.mit.edu/~andoni/LSH/. Accessed 30 November 2016.
2." The Google PageRank Algorithm and How It Works". Princeton, http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm. Accessed 30 November 2016.
3." Indexing", UC Davis, http://web.cs.ucdavis.edu/~green/courses/ecs165a-w11/7-indexes.pdf. Accessed 30 November 2016.
4. Hung-Hsuan Chen and C. Lee Giles. 2015. ASCOS++: An Asymmetric Similarity Measure for Weighted Networks to Address the Problem of SimRank. ACM Trans. Knowl. Discov. Data 10, 2, Article 15 (October 2015).
5. " Relevance feedback and pseudo relevance feedback". Stanford, http://nlp.stanford.edu/IR-book/html/htmledition/relevance-feedback-and-pseudo-relevance-feedback-1.html#sec:relevance-feedback. Accessed 30 November 2016.

6. Q/As related to PCA from stackoverflow.com

7.Goel, Ashish. "Applications of PageRank to Recommendation Systems." *Stanford*, web.stanford.edu/class/msande233/handouts/lecture8.pdf. Accessed 30 November 2016.

8.Hung-Hsuan Chen , C. Lee Giles, ASCOS: an asymmetric network structure COntext similarity measure, Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, August 25-28, 2013, Niagara, Ontario, Canada [doi>10.1145/2492517.2492539]

9.Gurmeet Singh Manku , Arvind Jain , Anish Das Sarma, Detecting near-duplicates for web crawling, Proceedings of the 16th international conference on World Wide Web, May 08-12, 2007, Banff, Alberta, Canada [doi>10.1145/1242572.1242592]

10.Koga, Hisashi, Tetsuo Ishibashi, and Toshinori Watanabe (2007), "Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing", Knowledge and Information Systems, 12 (1): 25–53, doi:10.1007/s10115-006-0027-5 .

11.“dejavu - Audio fingerprinting and recognition in Python”. GitHub, https://github.com/worldveil/dejavu/. Accessed 30 November 2016.