CSE 515 MWDB PROJECT PHASE 2

Report Submitted Under the Guidance Of

Dr. K. Selcuk Candan

TA Yash Garg

Group 21

Meha Shah

Prashant Garg

Priyanka Dubey

Karan Uppal

Sweta Singhal

ABSTRACT

There were two main objectives of this task. First was to find the similar frames and find the most significant frame by well-known algorithms PageRank and ASCOS. Then since the sift vectors are too large in number to find the similarity by brute-force, we opted to reduce the dimensions of the data and prune our search space by using the Locality Sensitive Hashing algorithm. It divides the data points into buckets and based upon how similar they are place them into same or different buckets. Finally, we used this pruned file to find similar objects in the database given the query by the user.

TERMINOLOGY: -

Sift : Scale Invariant feature transform (SIFT) is an algorithm designed by David Lowe to detect key points in an image [6]. SIFT feature can be used for cluster identification [8], feature matching, indexing and nearest neighbor search [9] amongst various applications.

Keypoints: A selected circular image region with an orientation is called keypoint of a SIFT vector which is associated with a descriptor [6]. Keypoints are extracted by SIFT detector.

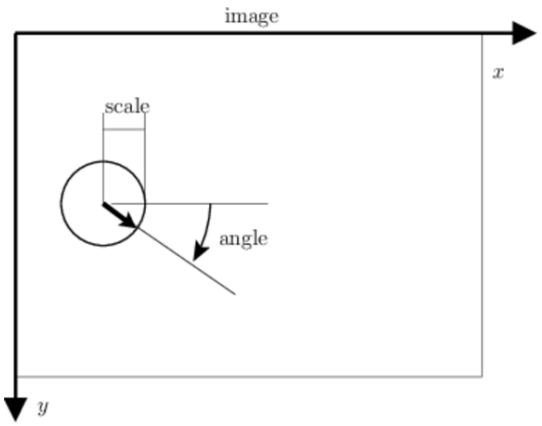


FIGURE 1: SIFT KEYPOINT[10]

Descriptors: A SIFT descriptor is a 3-D spatial histogram which is represented as a single 128 dimensional vector of the image gradients in characterizing the appearance of a keypoint [10].

Similarity matrix [15] : This is a matrix which gives the measure of similarity between objects of the data . d(i,j) is the similarity between i and j elements , similarity here can be computed with a lot of methods like cosine similarity , intersection similarity , Euclidian intersection , Manhattan intersection etc . the diagonal is all ones as the self-similarity is 1 .

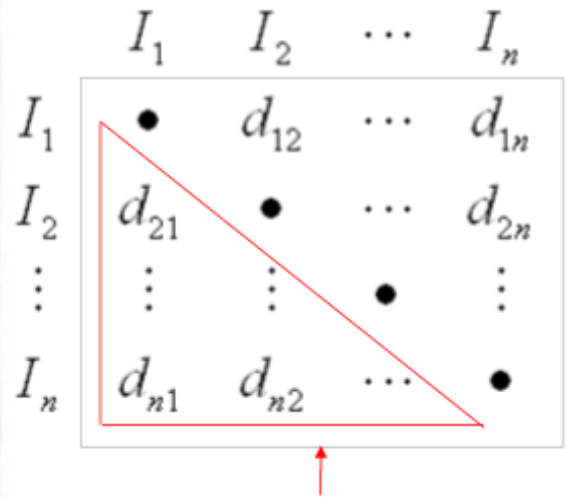


FIGURE 2: MATRIX SHOWING SIMILARITY MATRIX [15]

Nearest Neighbor Search(NNS): - It is the search related to finding the closet point with respect to a data point. The search can involve one or more neighbor, which is known as K-nearest neighbor search. [51]

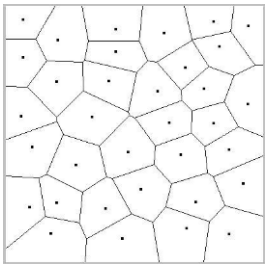


Figure 3 Voronoi Cell [53]

LSH (Latent Sematic Hashing): - Th goal of LSH is to create a binary code representation of the data given. In LSH we generate random hyperplanes to encode data into binary form. [52]

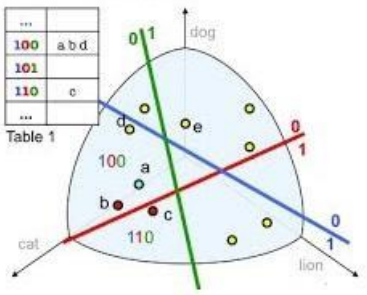
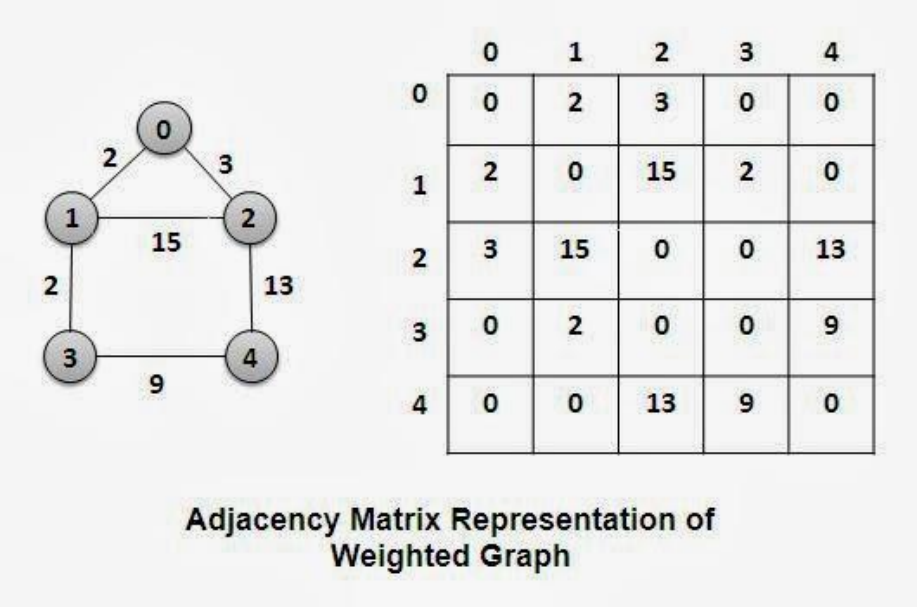


Figure 4 Visualization of LSH [54]

Page Rank: It is an algorithm used to compute the relevance of each node (Here, node signifies the frame) in the graph (Here graph signifies the frame collection). The idea that Page Rank brought up was that, the importance of any web page can be judged by looking at the pages that link to it. If we create a web page *i* and include a hyperlink to the web page *j*, this means that we consider *j* important and relevant for our topic. If there are a lot of pages that link to j, this means that the common belief is that page j is important. If on the other hand, *j* has only one backlink, but that comes from an authoritative site k, (like www.google.com, www.cnn.com, www.cornell.edu) we say that k transfers its authority to j; in other words, k asserts that j is important. Whether we talk about popularity or authority, we can iteratively assign a rank to each web page, based on the ranks of the pages that point to it. [5]

ASCOS: ASCOS i.e. 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-neighbours to node j. [6]

Adjacency Matrix: It is a m*m square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. [7] With respect to this project, m will represent the total number of frames in the video data collection and the value of each position will represent the similarity value/distance between two frames.



	0	1	2	3	4
0	0	2	3	0	0
1	2	0	15	2	0
2	3	15	0	0	13
3	0	2	0	0	9
4	0	0	13	9	0

Figure 1[8]

GRAPH: A graph is made up of vertices, nodes, or points which are connected by edges, arcs, or lines. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another. An edge between 2 nodes indicates that they are related based on the context in which the graph is made. [9]

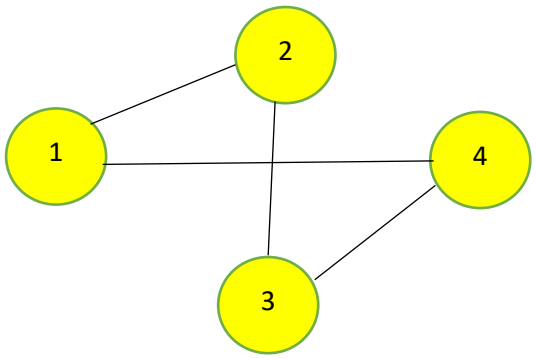


Figure 2 : undirected graph

Task 1

PCA- Principal Component Analysis

Problem Specification-

This task was very similar to what we did in the last phase. A set of videos were given to us and sift vectors were calculated from the program in phase 1. We then had to reduce the number of dimensions of the file. For demo purposes the targeted dimensions were given to us as 10 and 60. Additionally we had to report the new dimensions in terms of scores of old dimensions just like we did in the last phase.

Theory

A statistical procedure which makes use of the orthogonal transformation to produce a set of linearly independent vectors. It converts the originally dependent variable using the orthogonal transformation into independent variables thus removing redundancy and ambiguity from the data [1]. The basis of PCA lies in maximizing the variance across first principal component.

PCA helps in identifying patterns in the data [2]. With the data given it tries to identify the directions in which the variance is maximum. It then allots it the first principal component. It repeats the procedure for rest of the directions too till the number of original dimensions are matched. Since these new dimensions are a better representation of the data, we can plot the original data points in this new vector space. Now, we can perform the analysis on the new data points and calculate distances, similarities etc. on them [3].

Implementation

1. Gather data- In a simple language, we need to gather some data to perform PCA. [29]

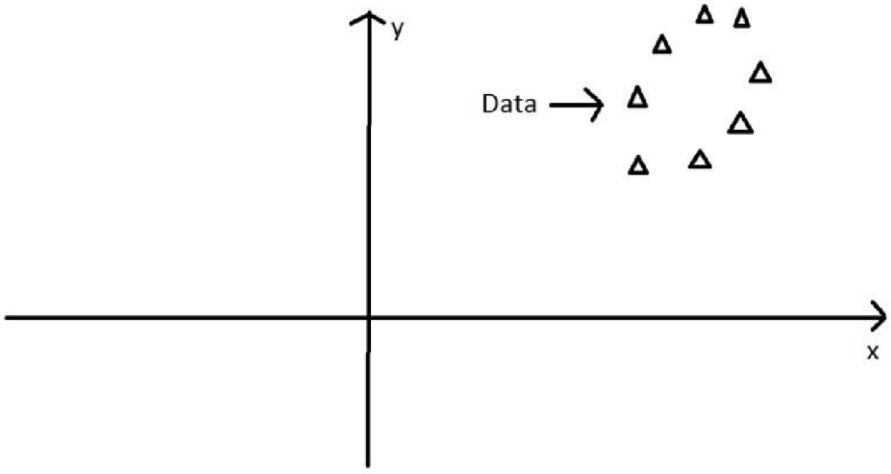


Figure 5: Sample data point plotted. [29]

2. Subtract the means- Calculate the mean of each column and then subtract it from each of the data points in that column

Data =

<x, y> = <2.5, 2.4> <0.5, 0.7> <2.2, 2.9> <1.9, 2.2> <3.1, 3.0> <2.3, 2.7> <2, 1.6> <1, 1.1> <1.5, 1.6> <1.1, 0.9>
DataAdjust =
<x, y>= <.69, .49> <-1.31, -1.21> <.39, .99> <.09, .29> <1.29, 1.09> <.49, .79> <.19, -.31> <-.81, -.81> <-.31, -.31> <-.71, -1.01>

3. Calculate the Covariance Matrix- Suppose the data is n-dimensional, then the covariance matrix would be n*n matrix. Diagonal values tell us about the variance of that dimension whereas the non-diagonal elements tell us about the correlation that exists between the two dimensions. [28]

VARIANCE-COVARIANCE CALCULATIONS

Rates of return for 7 assets over 10 years

	AMR	BS	GE	HR	MO	UK	SP500
1974	-0.3505	-0.1154	-0.4246	-0.2107	-0.0758	0.2331	-0.2647
1975	0.7083	0.2472	0.3719	0.2227	0.0213	0.3569	0.3720
1976	0.7329	0.3665	0.2550	0.5815	0.1276	0.0781	0.2384
1977	-0.2034	-0.4271	-0.0490	-0.0938	0.0712	-0.2721	-0.0718
1978	0.1663	-0.0452	-0.0573	0.2751	0.1372	-0.1346	0.0656
1979	-0.2659	0.0158	0.0898	0.0793	0.0215	0.2254	0.1844
1980	0.0124	0.4751	0.3350	-0.1894	0.2002	0.3657	0.3242
1981	-0.0264	-0.2042	-0.0275	-0.7427	0.0913	0.0479	-0.0491
1982	1.0642	-0.1493	0.6968	-0.2615	0.2243	0.0456	0.2141
1983	0.1942	0.3680	0.3110	1.8682	0.2066	0.2640	0.2251

Variance-Covariance Matrix

	AMR	BS	GE	HR	MO	UK	SP500
AMR	0.2060	0.0375	0.1077	0.0493	0.0208	0.0059	0.0533
BS	0.0375	0.0790	0.0355	0.1028	0.0089	0.0406	0.0390
GE	0.1077	0.0355	0.0867	0.0443	0.0194	0.0148	0.0471
HR	0.0493	0.1028	0.0443	0.4435	0.0193	0.0274	0.0467
MO	0.0208	0.0089	0.0194	0.0193	0.0083	-0.0015	0.0094
UK	0.0059	0.0406	0.0148	0.0274	-0.0015	0.0392	0.0178
SP500	0.0533	0.0390	0.0471	0.0467	0.0094	0.0178	0.0359

Figure 6: Calculating the variance-covariance matrices.

4. Calculate the eigenvectors and eigenvalues of the covariance matrix- Both these eigenvalues and eigenvectors provide useful information about the data. This process of taking the eigenvectors of the covariance matrix, we can extract the lines that characterize the data. [28]

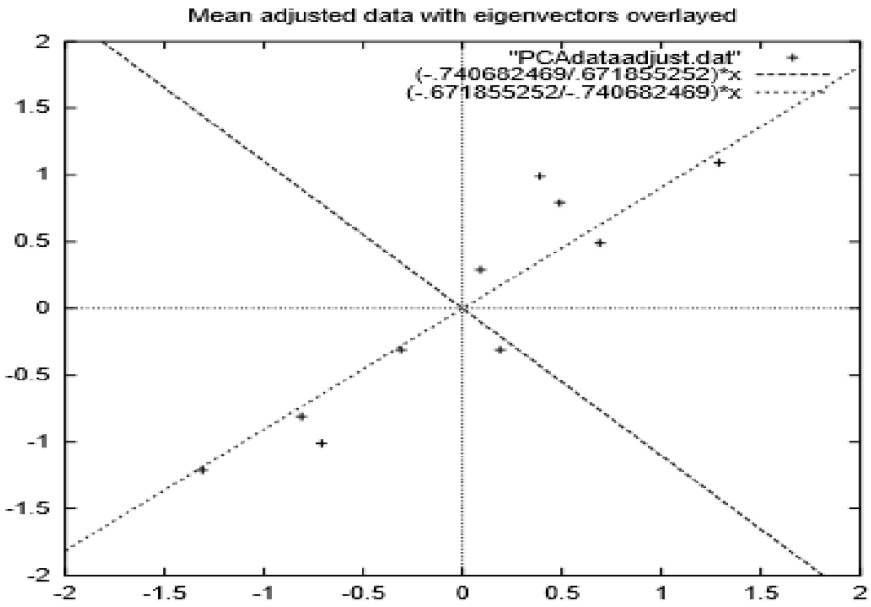


Figure 7: The eigenvectors and eigenvalues on the sample data.

5. Choosing components and forming a feature vector- Now, we arrive to the important concept of data compression and dimensionality reduction. Once we obtain the eigenvectors from the covariance matrix, we need to order them from highest value to lowest value. This provides us with the order of significance.

6. Deriving the new data set- Once we have reduced the dimensions as per our desire, we now have to convert it to new data set.

   FinalData = RowFeatureVector * RowDataAdjust

Assumptions

Following are the assumptions made in the task.

1. Sift vector file we got was made with r value in the sift library as 1.05. It has been reduced from 10 to reduce the number of sift vectors in the videos.
2. The input file is assumed to be in correct format.
3. Perl binaries have been installed on the system to run the code.
4. The value of r is taken as 2. That means each frame has been divided into 4 cells.
5. The input of number of vectors is always an integer less than the original dimensions of the file.
6. For the purpose of reduction, we take into consideration the scale, orientation and 128 descriptors. We have ignored the dimensions such as video number, frame number and cell number.

Interface Specification

Input

The input of this task is the sift vector file which was generated as part of phase 1.

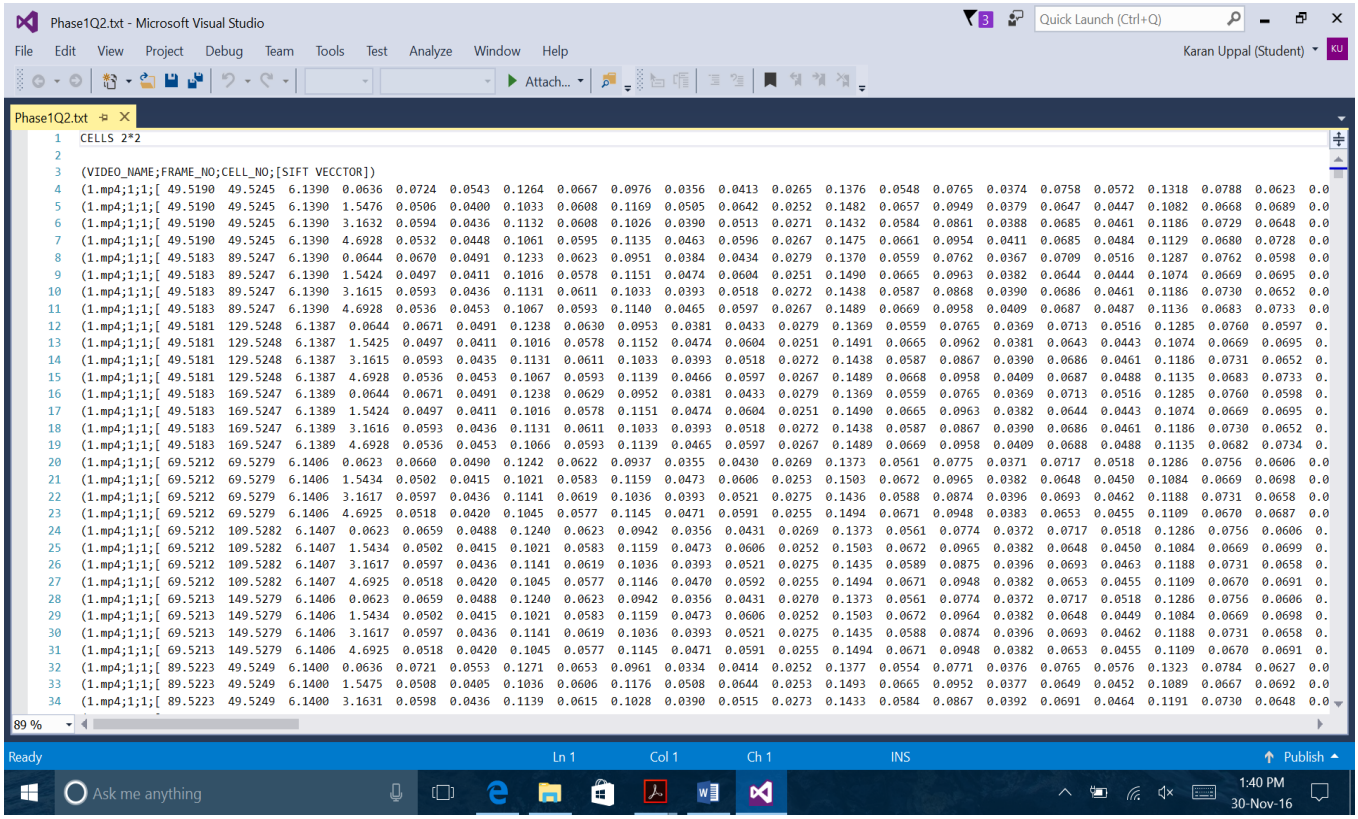


Figure 8: The input file for task 1.

Output

The output of this task are the two files for demo dimensions as 10 and 60 each.

Dimensions- 10

New data points- The format is:

<Video; Frame; Cell; nd1; nd2; …….; ndn>

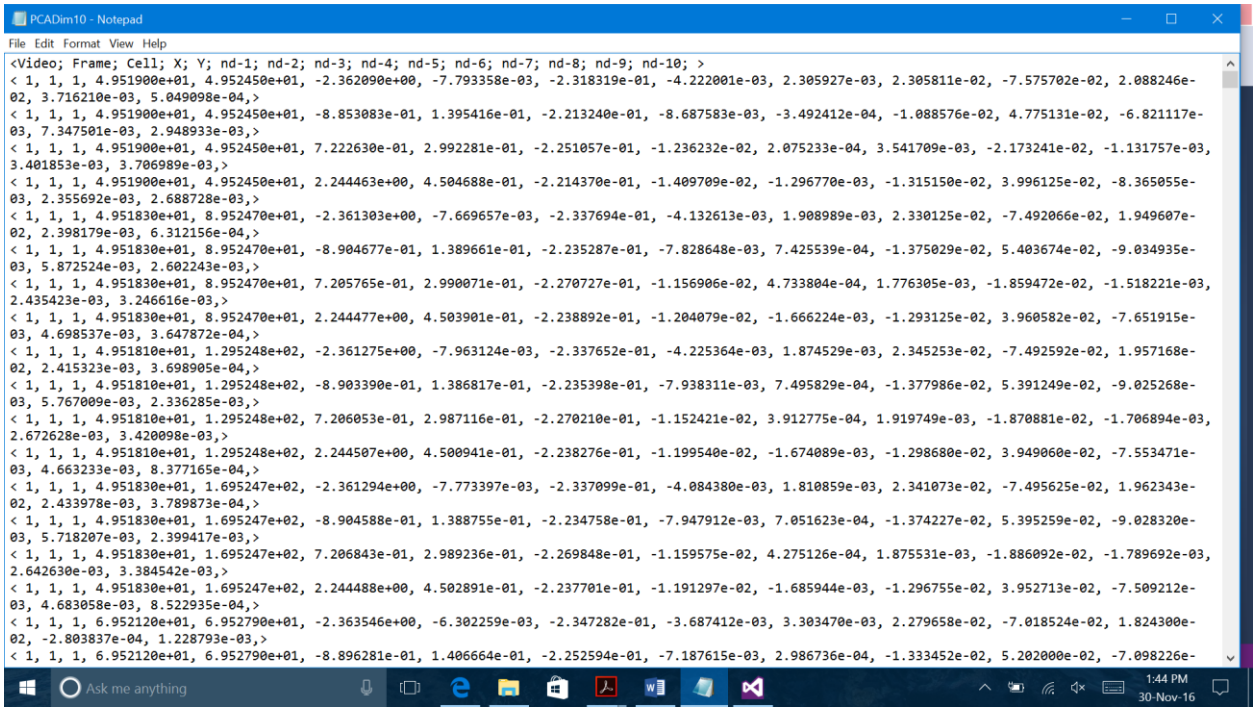


Figure 9- Showing the data points in new vector space.

Additionally, the new dimensions had to be shown in terms of old dimensions. Their score was to be reported.

Format of the output is:

New Dimension- I = <od1 = 0.3; od5=0.2; …….; odn=-0.23 >;

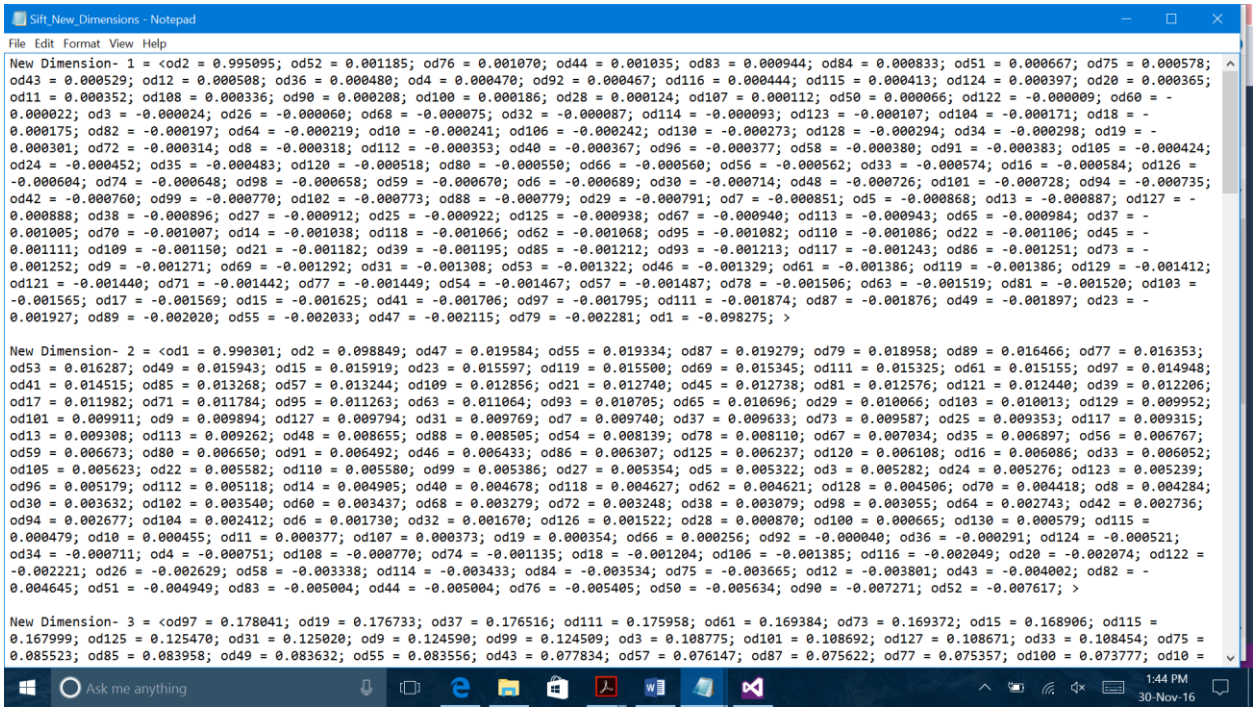


Figure 10: New dimensions in terms of old dimensions

Dimensions- 60

New data points- The format is:

<Video; Frame; Cell; nd1; nd2; …….; ndn>

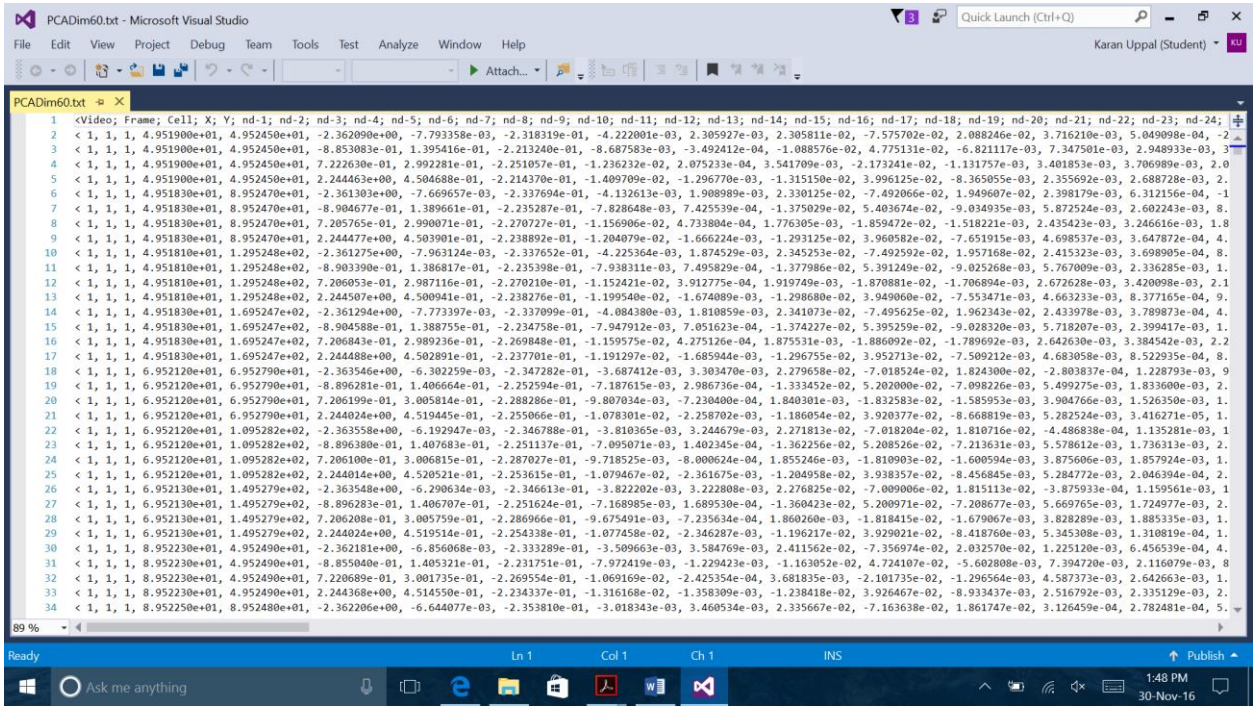


Figure 11: New data points in terms of new dimensions.

Additionally, the new dimensions had to be shown in terms of old dimensions. Their score was to be reported.

Format of the output is:

New Dimension- I = <od1 = 0.3; od5=0.2; …….; odn=-0.23 >;

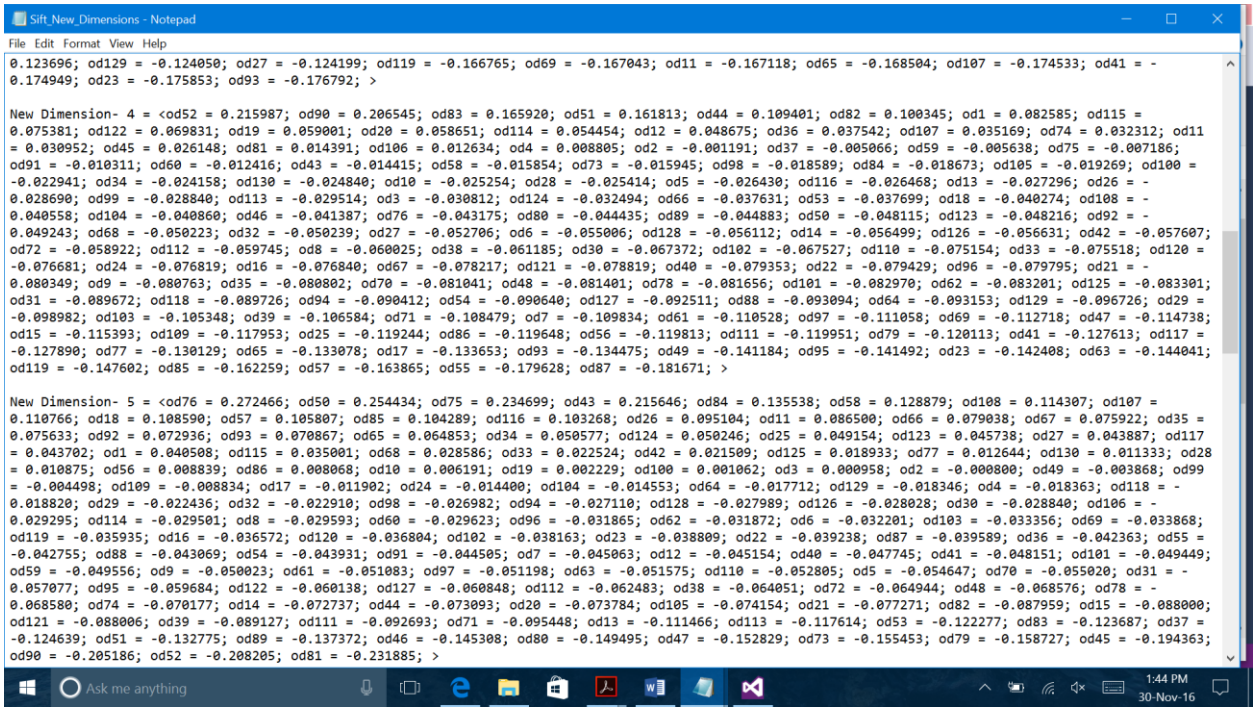


Figure 12: New data points in terms of new data points.

Verification-

To verify that the dimensions have been correctly reduced, we can reverse the process of the PCA and try and obtain the original dimensions with original data points. This is only possible if we keep all the new dimensions. But if some dimensions have been reduced it may not be possible to get back the original data points.

Task 2 (Video Frame Similarity Graph Generation)

Keywords: SIFT, SIMILARITY, EUCLIDEAN, COSINE

Problem Specification:-

In this task, given an integer k, and using the output of Task 1 which is the reduced sift keypoints stored in the file, we need to create a similarity graph G(V,E) where V corresponds to frame of videos in directory and E is the edge pair set $<v_a,v_b>$ such that, for each video frame v_a, v_b is one of the k most similar frames to v_a that is not already in the same video file.

The output needs to be stored in a file, where each line is of the form :

$$<v_a, v_b, \text{sim}(a, b)>$$

where $v_a = <i_a, j_a>$ and $vb = <i_b, j_b>$are two frames and sim(a, b) is the degree of similarity between these two frames. i and j are video number and frame number respectively.

Implementation:-

First of using a perl script (Formatting_new.pl), the output of task1 is read and stored in a variable Z. Then from Z, all the videos are separated and stored in variable A. So until now A{1} will have first video, A{2} second video and so on. This will be used for calculating similarities further.

The first step in implementing this task was to reuse the *Sift_Similarity_Matrix* function which we created in Phase2. This function, given two variables with two videos, return back the frame to frame similarity between them. B – is a cell array used which contains the frame similarity matrix between all possible combinations of videos. After getting the Frame similarity between video, for every frame we are iteration over all the other frames from all other videos (except its own) and then storing it in another matrix. This matrix is called *FrameSimilarityMatrix*. Sorting is done in descending order on this matrix to find the best matches of this frame and the top k matches are stored in *TotalFrameSimilarityMatrix*. This process is repeated for all frames in all videos. When we store the similarities values in *FrameSimilarityMatrix*, we also simultaneously store the video number and frame number of the frames for which this similarity belongs. This helps us in figuring out in the end when the best matches are available that for which videos it belong.

In the end, the *TotalFrameSimilarityMatrix* is iterated over and printed in a file.

SIFT similarity is calculated on the basis of the descriptors as mentioned in [37][38][39][36] and David Lowe himself who designed the SIFT algorithm. Keypoints between two images are matched by identifying their nearest neighbors [37]. The SIFT algorithm sometimes gives false matches also. To avoid this problem [39] David Lowe experimented on a large database and found that if we ignore the keypoints where, the ratio of closest-distance to second-closest distance is greater than 0.8, it removes 90% of the false matches while discards only 5% correct matches. This is incorporated in the algorithm for sift similarity.

In implementation of sift similarity, we are maintaining a count of how many vectors are matched in a cell based on best match and second best match. This gives us the cell percent match for every frame. After this we are calculating the frame similarity based on cell similarity

percent. Now this frame – frame similarity value is stored in the similarity matrix.

Similarity methods selected for SIFT is Euclidean. We selected this method because David Lowe suggests this approach to match SIFT descriptors in his paper. He mentions in his paper that correct matches have normal distribution and while incorrect matches have uniform noise distribution. Least-square matching is preferred way of matching for normal distribution. Therefore, we selected Euclidean (L-2 norm) as the first similarity measure.

Assumptions: -

1. It is not necessary that Va has 0.5 similarity value from Vb then Vb will also have the same similarity value to Va. The similarities are not symmetric. The reason behind this that the similarity value is calculated depending on how many vectors of Va are similar and found in Vb. There might be a case where Va finds all the vectors in Vb so it will have 1 similarity value. Vb has mre vectors than Va, so the similarity of Vb to Va would be less than 1 in that scenario.
2. Threshold value for good sift matches will be 0.8.

3. We are comparing cells Frame by Frame, but each Frame to other Frame Based on cell level spatial data. Matching ith Cell from Reference Video Frame with ith Frame of comparison Video.

Input: -

1. Phase:3 Task:1 output file "Sift_Reduced_Formatted.txt" is required as input.

Figure 6: Input from Task1

2. Input from user: K i.e. number of similar frames for each frame.

```
Command Window
  Reading..How many "K"-Similar Nodes, Enter the value of K :- 2
```

Figure 7: Input from user: K similar nodes

Output: -

The output is saved in Phase3Q2.txt. Below is the snippet of the output format for K = 2 and on 10 videos from Phase2 Demo. The output is of the format:

<Va, Vb, Similarity(Va, Vb)>

Va comprises of $<i_a, j_a>$, where i_a is the video number and j_a is the frame number. This signifies one node of the graph. For every node two most similar nodes are found whose value is Similarity(Va, Vb). Similarity(Va, Vb) is the similarity value between node Va and Vb.

Phase3Q2.txt

Va	Vb	Similarity(a,b)
(1,1)	(4,1)	1.000000
(1,1)	(10,1)	0.958895
(1,2)	(4,2)	1.000000
(1,2)	(10,2)	0.947475
(1,3)	(4,3)	1.000000
(1,3)	(10,3)	0.938353
(1,4)	(4,4)	1.000000
(1,4)	(10,4)	0.934975
(1,5)	(4,5)	1.000000
(1,5)	(10,5)	0.880718
(1,6)	(4,6)	1.000000
(1,6)	(10,6)	0.916591
(1,7)	(4,7)	1.000000
(1,7)	(4,8)	0.901715
(1,8)	(4,8)	1.000000
(1,8)	(4,7)	0.869076
(1,9)	(4,9)	1.000000
(1,9)	(4,8)	0.849452
(1,10)	(4,10)	1.000000
(1,10)	(10,10)	0.848178
(1,11)	(4,11)	1.000000
(1,11)	(10,11)	0.837254
(1,12)	(4,12)	1.000000
(1,12)	(10,12)	0.835242
(1,13)	(4,13)	1.000000
(1,13)	(10,13)	0.812095
(1,14)	(4,14)	0.811912
(1,14)	(10,13)	0.794778
(1,15)	(4,17)	0.759499
(1,15)	(10,14)	0.751897
(1,16)	(4,17)	0.787953
(1,16)	(4,16)	0.787504
(1,17)	(10,16)	0.796304
(1,17)	(4,17)	0.784504
(1,18)	(7,16)	0.743383
(1,18)	(4,16)	0.743360
(1,19)	(10,19)	0.748736
(1,19)	(4,17)	0.730383
(1,20)	(4,16)	0.737114
(1,20)	(10,20)	0.732108
(1,21)	(10,20)	0.719733
(1,21)	(10,21)	0.695198
(1,22)	(10,21)	0.692132
(1,22)	(4,15)	0.689033
(1,23)	(4,15)	0.717355
(1,23)	(4,14)	0.711367
(1,24)	(4,15)	0.724371
(1,24)	(10,15)	0.705566

Steps to execute: -

For input: - One is require to keep the file which is output from Task 1 of this Phase.

1> Open Task2.m

2> Run it

3> User will be asked about two inputs: -

   a. K number of similarity nodes.

4> Output File name is: - Phase3Q2.txt

TASK 3 (Most Significant Frame Selection)

Problem Specification:

In this task, we have to implement a program which takes as an input

- the graph file obtained as an output from task 2 of project Phase 3. This file contains the top k most similar nodes/frames with respect to each node/frame in the collection along with their similarity value.
- an integer, m

and identify the most significant m frames in the collection using

a. PageRank Algorithm
b. ASCOS measures

Also, we need to visualize the selected m frames for both approaches.

[A] PageRank

Implementation:

In order to achieve this, we will give input m and the output file of Phase 3 Task 2 'Phase3Q2.txt'. File 'Phase3Q2.txt' contains the video number, frame number of two frames as well as sim(a,b) i.e. similarity value between the two frames. The significance of this file is it contains top 'k' most similar frames of each frame in the database along with the similarity value with those 'k' frames.

Phase3Q2 - Notepad

File Edit Format View Help

Va	Vb	Similarity(a,b)
(1,1)	(4,1)	1.000000
(1,1)	(10,1)	0.958895
(1,2)	(4,2)	1.000000
(1,2)	(10,2)	0.947475
(1,3)	(4,3)	1.000000
(1,3)	(10,3)	0.938353
(1,4)	(4,4)	1.000000
(1,4)	(10,4)	0.934975
(1,5)	(4,5)	1.000000
(1,5)	(10,5)	0.880718
(1,6)	(4,6)	1.000000

Figure 31:Phase3Q2.txt

Then, we have created the Perl script that reads the input file to create the OutPut.txt which contains the video and frame number as well as similarity value between two frame for all the videos.

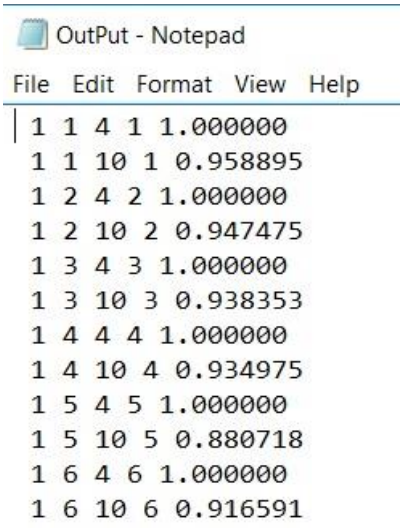


Figure 4 OUTPUT.TXT

In this project, each frame in the database will represent the node in the graph. Before discussing our implementation, we will first discuss the algorithm to calculate the page rank.

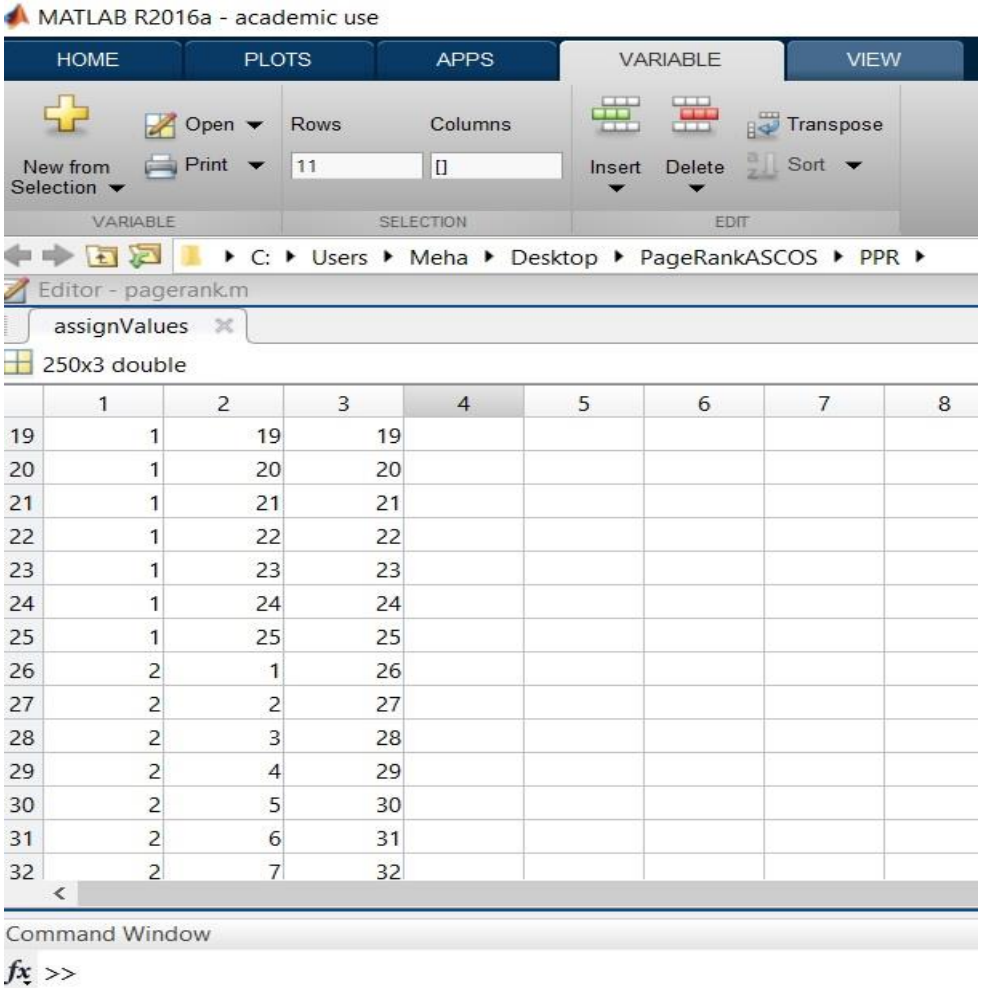
In the PageRank algorithm, we don't have the starting point or the frames for search. We take the probability of the frames/nodes occurring in the random walk as equal. The value of PageRank for frames is more when it has many incoming links from other nodes/frames or the nodes pointing towards the node i.e. frame which will be a part of 'k' most similar frames with respect to each frame is of more significance.

$$r = \frac{(1-P)}{n} + P(A'\left(\frac{r}{d}\right) + \frac{s}{n})$$

Here

- r is a vector of PageRank scores
- P is a scalar damping factor, which is the probability that a random surfer clicks on a link on the current page, instead of continuing on another random page. This is to stop the other frames having too much influence, this total factor is "damped down" by multiplying it by 0.85
- A' is the transpose of the adjacency matrix of the graph.
- d is the vector containing the outdegrees of each node in the graph . it is set to 1 when outdegree is 0 .
- n is the scalar number of nodes in the graph.
- s is the scalar sum of the PageRank scores for pages with no links

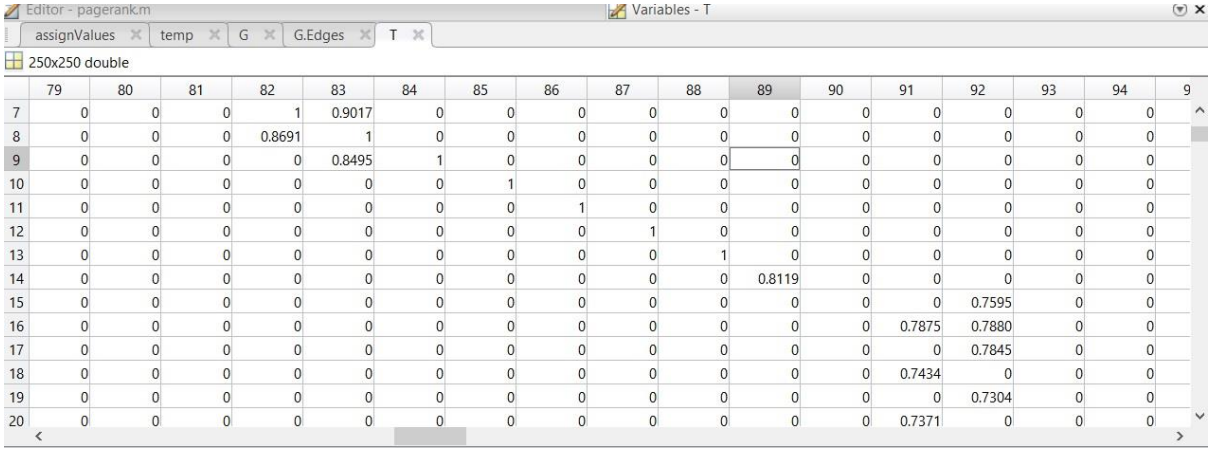
Our first task was to obtain the adjacency matrix T, as used in the above PageRank formula. To achieve it, we have created the ‘assignValue’ matrix that give a unique value to each of the video frame pair in our database.



	1	2	3	4	5	6	7	8
19	1	19	19					
20	1	20	20					
21	1	21	21					
22	1	22	22					
23	1	23	23					
24	1	24	24					
25	1	25	25					
26	2	1	26					
27	2	2	27					
28	2	3	28					
29	2	4	29					
30	2	5	30					
31	2	6	31					
32	2	7	32					

Figure 12 : AssignValue Matrix

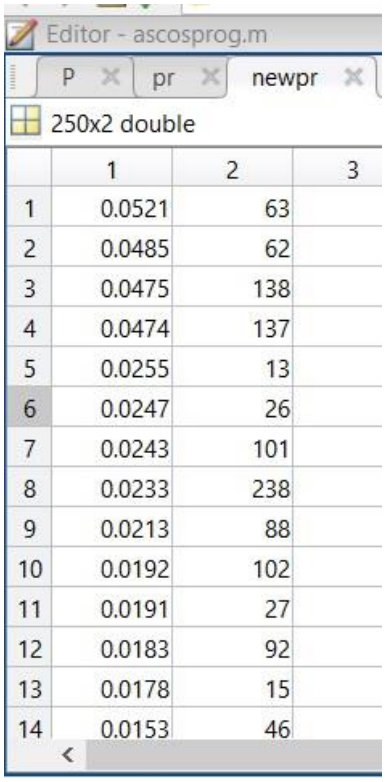
This matrix will also help us in identifying the unique nodes in our database. Then, we will create an adjacency matrix using OutPut.txt file (Phase3Task2 output) such that for each unique node/frame, its value in the matrix with respect to ‘k’ the similarity value will represent most similar frames. Please note that ‘k’ most similar frames and similarity value will be obtained from the OutPut.txt file. Apart from ‘k’ frames, similarity value between other nodes have been assumed as 0.



	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	9
7	0	0	0	1	0.9017	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0.8691	1	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0.8495	1	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0.8119	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7595	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	0.7875	0.7880	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7845	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0.7434	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7304	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0.7371	0	0	0	

Figure 15 : Adjacency Matrix

As soon as we obtain the adjacency matrix, we will implement the above mentioned recursive page rank algorithm. We have used damping factor=0.85 and initialized the PageRank matrix by eigen values. After observing the output for different values, we have taken 100 iterations to converge the solution.



	1	2	3
1	0.0521	63	
2	0.0485	62	
3	0.0475	138	
4	0.0474	137	
5	0.0255	13	
6	0.0247	26	
7	0.0243	101	
8	0.0233	238	
9	0.0213	88	
10	0.0192	102	
11	0.0191	27	
12	0.0183	92	
13	0.0178	15	
14	0.0153	46	

Figure 16 : PageRank scores. Second column denotes the unique number given to video and frame pair.

As soon as the solution is converged, we will obtain the page rank score with respect to each node and as specified by the user, we will visualize the output of ‘m’ most significant frames.

ASSUMPTIONS:

- The graph is assumed to be free of self-loops
- The value of P, damping factor is taken as .85 as suggested in the original Larry page research paper.
- The value of d is 3. as the input files contains 3 most similar frames for evry frame of every video
- The graph is assumed to be free of self-loops i.e. similarity value is taken as 0 between two same frames in the adjacency matrix.
- The value of P, damping factor is taken as .85 as suggested in "Sergey Brin , Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems, v.30 n.1-7, p.107-117, April 1, 1998"
- Apart from 'k' frames, we have assumed similarity value between other nodes as 0.

INTERFACE SPECIFICATION :

INPUT :

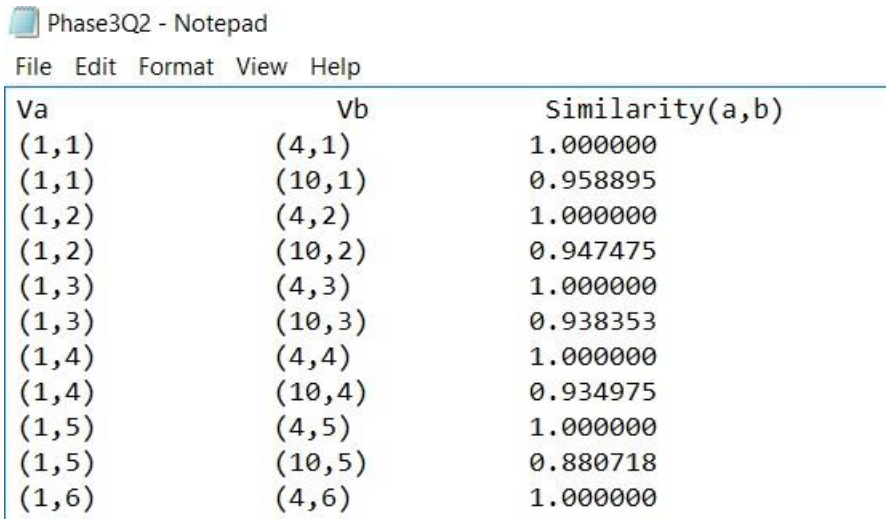
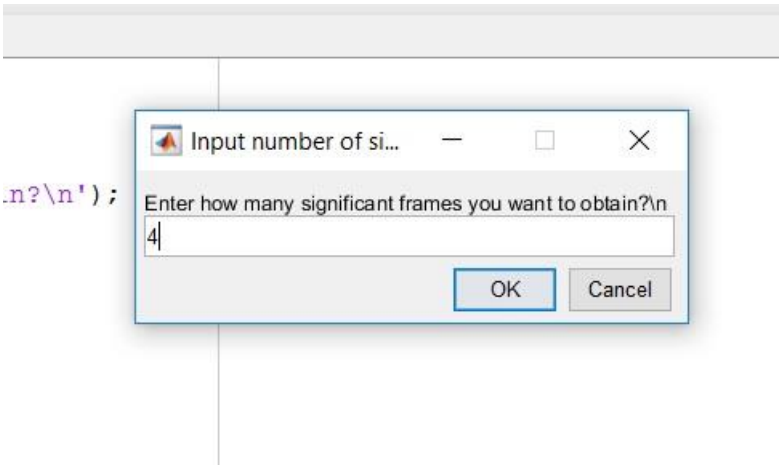


Figure 17 :phase3Task3.txt

- the user has to enter the value of m.


.n?\n');
Input number of si...
Enter how many significant frames you want to obtain?\n
4
OK
Cancel

OUTPUT:

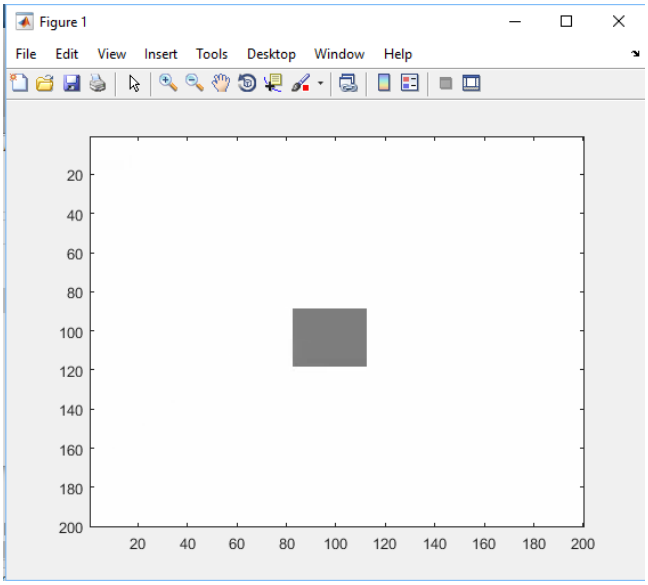


Figure 1
File Edit View Insert Tools Desktop Window Help
20 40 60 80 100 120 140 160 180 200
20 40 60 80 100 120 140 160 180 200

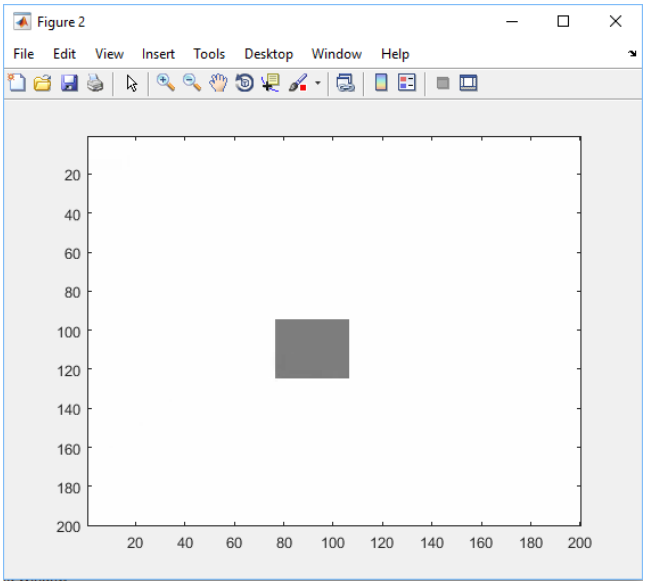
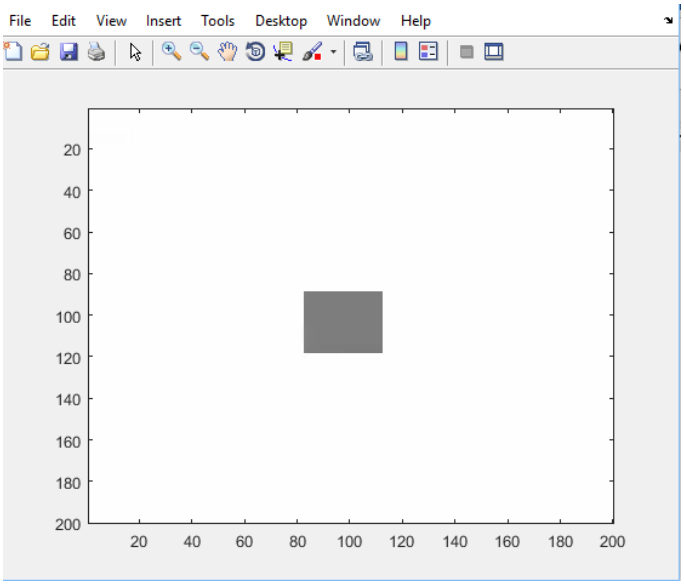


Figure 2
File Edit View Insert Tools Desktop Window Help
20 40 60 80 100 120 140 160 180 200
20 40 60 80 100 120 140 160 180 200



The figure above are the m most significant frames as found by pageRank algorithm

Task [B] ASCOS

Implementation:

In order to achieve this, we will give input m and the output file of Phase 3 Task 2 'Phase3Q2.txt'. File 'Phase3Q2.txt' contains the video number, frame number of two frames as well as sim(a,b) i.e. similarity value between the two frames. The significance of this file is it contains top 'k' most similar frames of each frame in the database along with the similarity value with those 'k' frames.

Phase3Q2 - Notepad

File Edit Format View Help

Va	Vb	Similarity(a,b)
(1,1)	(4,1)	1.000000
(1,1)	(10,1)	0.958895
(1,2)	(4,2)	1.000000
(1,2)	(10,2)	0.947475
(1,3)	(4,3)	1.000000
(1,3)	(10,3)	0.938353
(1,4)	(4,4)	1.000000
(1,4)	(10,4)	0.934975
(1,5)	(4,5)	1.000000
(1,5)	(10,5)	0.880718
(1,6)	(4,6)	1.000000

FIGURE 18: PHASE3Q2.TXT

Then, we have created the Perl script that reads the input file to create the OutPut.txt which contains the video and frame number as well as similarity value between two frame for all the videos.

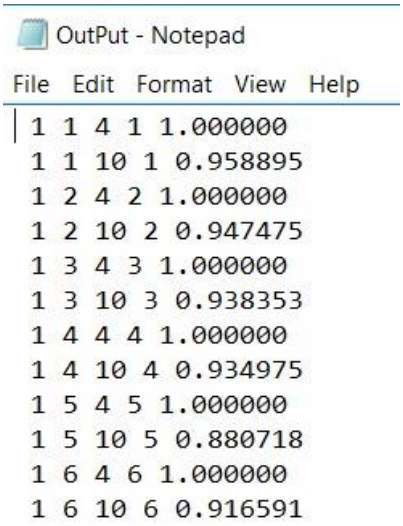


FIGURE 19: OUTPUT.TXT

In this project, each frame in the database will represent the node in the graph. Before discussing our implementation, we will first discuss the algorithm to calculate the ASCOS score.

In the ASCOS similarity measure, the similarity score from a node/frame i to a node/frame j is dependent on the similarity score from node i's in-neighbours to node j. This statement has two interesting properties:

1. the calculation excludes the out-neighbours.
2. the similarity score is asymmetric because it considers the in-neighbours of frame i but not the in-neighbours of frame j.

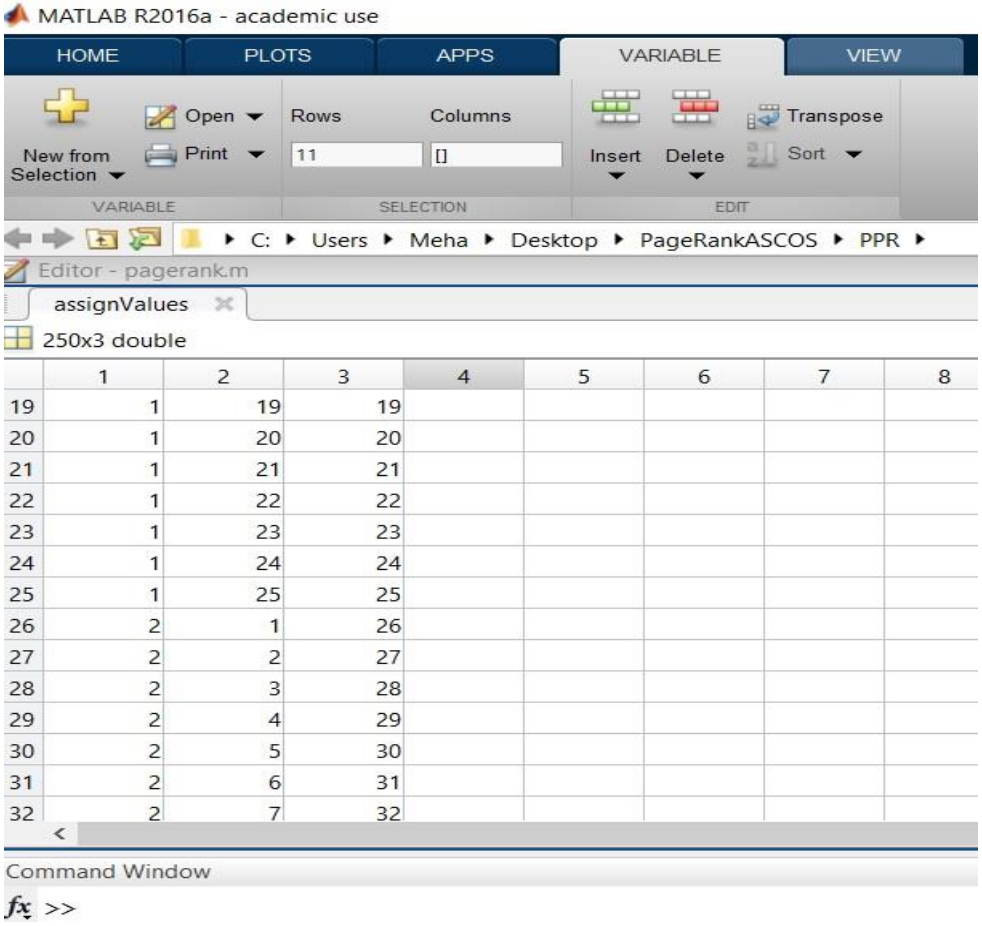
We have used the below recursive algorithm to calculate ASCOS score with respect to each frame.

$S = c*P(T)*S + (1-c)I$

Here,

a. S: ASCOS similarity score matrix for all the frames
b. c: Relative importance parameter. Its value lies between 0 and 1. It controls the relative importance between the direct neighbors and indirect neighbors, that is, neighbors' neighbors. The smaller the value, the less important the indirect neighbors are. In this project, we have taken its value as 0.9 to give high importance to the neighbor's neighbor while calculating the ASCOS score.
c. P: It is the column normalized matrix of adjacency matrix such that P =[p(i,j)] and p(i,j) = a(i,j) / sum(a(k,j)) where
   k -> in-neighbours of frame i
   a(i,j) -> similarity value between frame I and frame j
   a(k,j) -> similarity value between frame k and frame j
d. I: Identity matrix

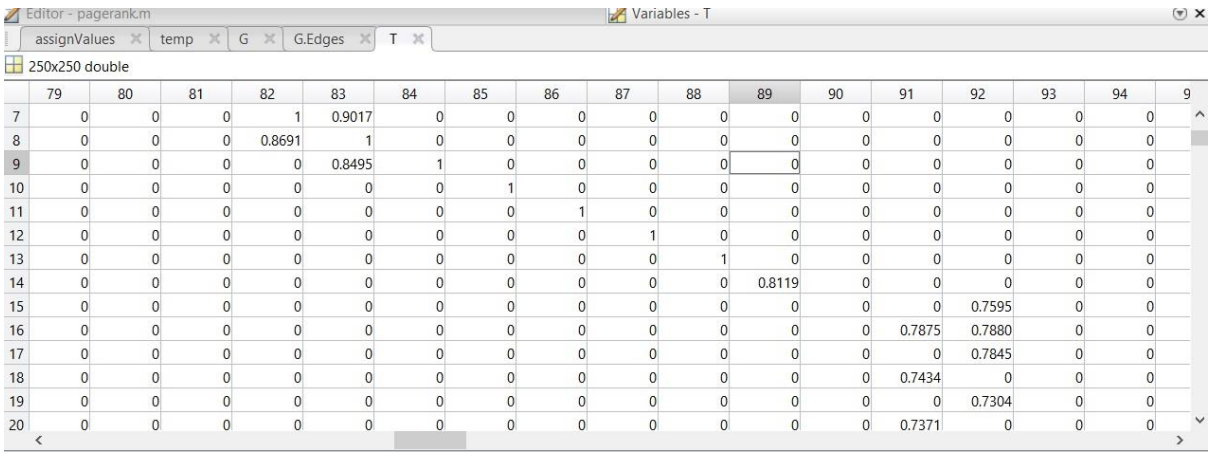
Our first task was to obtain the adjacency matrix A in order to obtain the matrix P as used in the above ASCOS formula. To achieve it, we have created the 'assignValue' matrix that give a unique value to each of the video frame pair in our database.



	1	2	3	4	5	6	7	8
19	1	19	19					
20	1	20	20					
21	1	21	21					
22	1	22	22					
23	1	23	23					
24	1	24	24					
25	1	25	25					
26	2	1	26					
27	2	2	27					
28	2	3	28					
29	2	4	29					
30	2	5	30					
31	2	6	31					
32	2	7	32					

Figure 20 : AssignValue Matrix

This matrix will also help us in identifying the unique nodes in our database. Then, we will create an adjacency matrix using OutPut.txt file (Phase3Task2 output) such that for each unique node/frame, its value in the matrix with respect to ‘k’ the similarity value will represent most similar frames. Please note that ‘k’ most similar frames and similarity value will be obtained from the OutPut.txt file. Apart from ‘k’ frames, similarity value between other nodes have been assumed as 0.



	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	9
7	0	0	0	1	0.9017	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0.8691	1	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0.8495	1	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0.8119	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7595	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	0.7875	0.7880	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7845	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0.7434	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7304	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0.7371	0	0	0	

Figure 21 : Adjacency Matrix

Then, in order to compute the P matrix we need the similarity values between two nodes say A and B as well as similarity value between in-neighbours of node A and B.

p(i,j) = a(i,j) / sum(a(k,j))

As we have assumed distance between two non-similar nodes i.e. nodes apart from k similar nodes as 0, there might be a case when sum of distance between each in-neighbours with the target node is 0. To handle that case when denominator becomes 0, we have assumed the denominator to be 100 so that overall value of p[i,j] minimizes and as a result its impact on the overall ASCOS score becomes negligible.

Editor - ascosprog.m | Variables - P

P

250x250 double

	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	9
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0.2029	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0.1273	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0.2040	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0.2045	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0.2045	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0.1450	0.1086	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0.1503	0.1132	0.1062	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0.0988	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0.2063	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	0.1347	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	0	0	0	0.1341	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	0	0.2081	0.1977	0	0.0985	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1999	0	0	

As soon as we obtain the P matrix, we will implement the above mentioned recursive page rank algorithm. We have used c=0.9 and initialized the ASCOS matrix by eigen values. After observing the output for different values, we have taken 100 iterations to converge the solution.

250x2 double

	1	2	3
1	0.1000	1	
2	0.0450	76	
3	0.0423	226	
4	0.0068	3	
5	0.0068	78	
6	0.0017	228	

Figure 22 : ASCOS scores. Second column denotes the unique number given to video and frame pair.

As soon as the solution is converged, we will obtain the ASCOS score with respect to each node and as specified by the user, we will visualize the output of ‘m’ most significant frames.

ASSUMPTIONS

- The graph is assumed to be free of self-loops i.e. similarity value is taken as 0 between two same frames in the adjacency matrix.
- The value of c, relative importance parameter is taken as 0.85 as suggested in "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)"
- Apart from 'k' frames, we have assumed similarity value between other nodes as 0.
- When the sum of similarity value between all the in-neighbours and target node is zero, we will assume it to be 100 so that overall value of p[i,j] minimizes and as a result its impact on the overall ASCOS score becomes negligible.

INTERFACE SPECIFICATION :

INPUT :

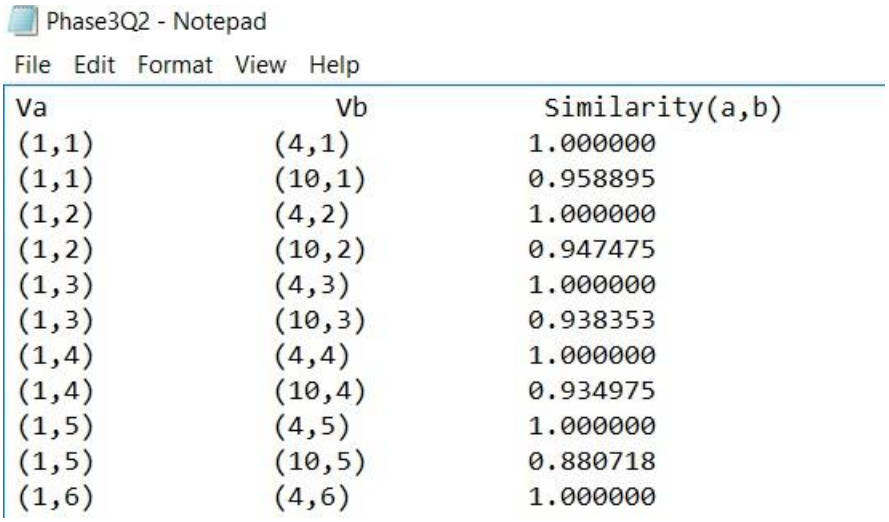
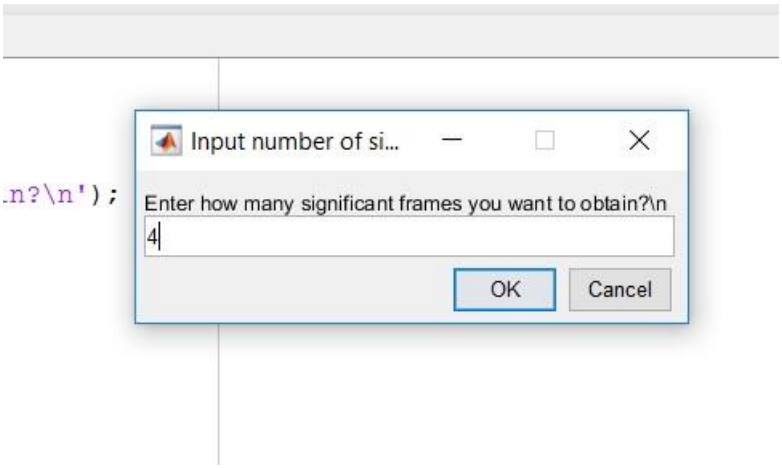
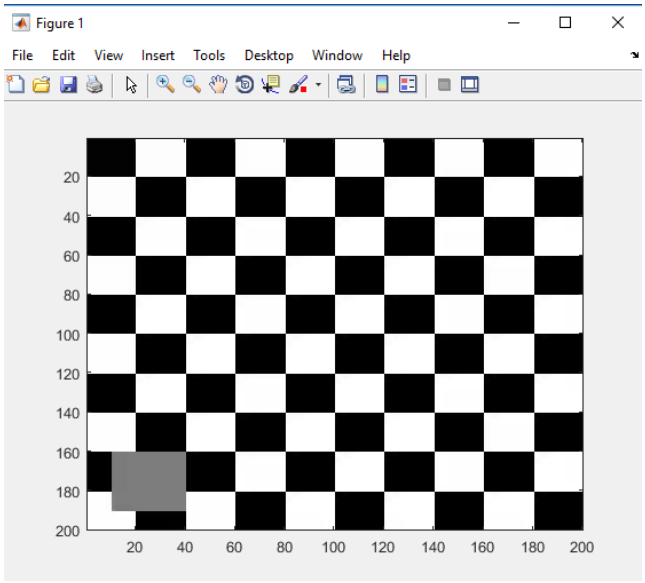


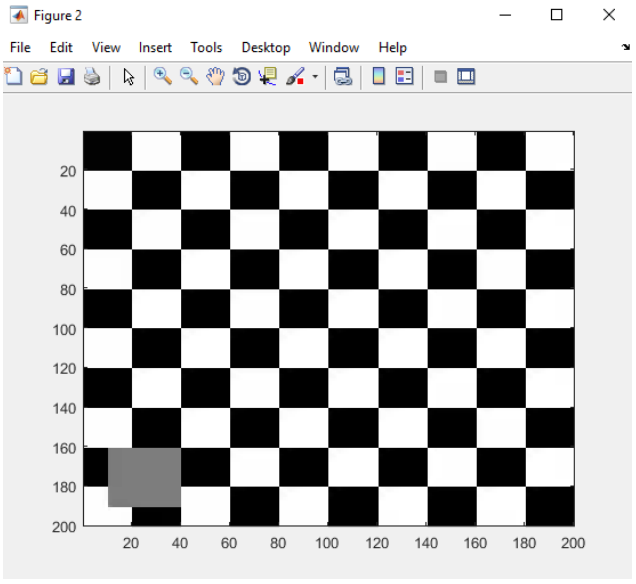
Figure 23 : Phase3Q2.txt

- the user has to enter the value of m.



OUTPUT:





The above figures are the m most significant frames.

TASK 4 (Most Relevant Frame Selection)

Problem Specification:

In this task, we have to implement a program which takes as an input

- the graph file obtained as an output from task 2 of project Phase 3. This file contains the top k most similar nodes/frames with respect to each node/frame in the collection along with their similarity value.
- an integer, m
- three input frames (each described as a pair of video/frame ids)

and identify the most significant m frames relative to the input frames in the collection using

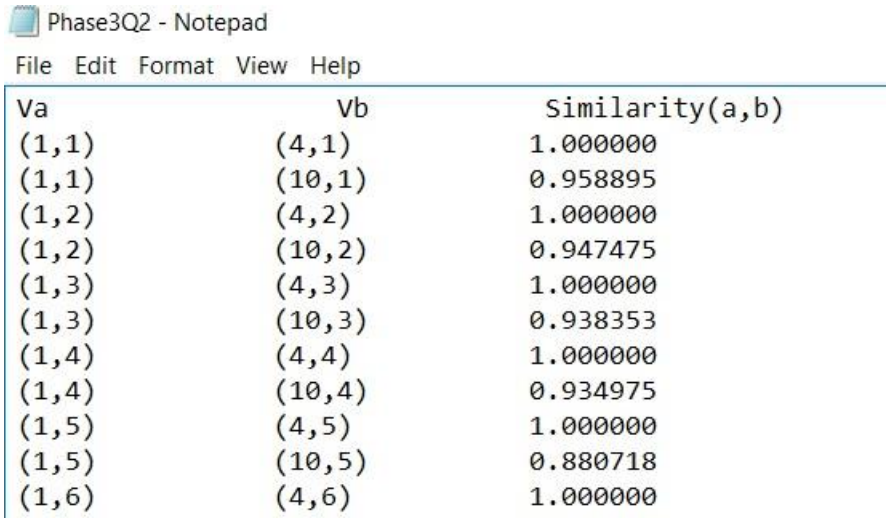
a. Personalized PageRank Algorithm
b. Modified ASCOS measure

Also, we need to visualize the selected m frames for both approaches.

[A] Personalized PageRank

Implementation:

In order to achieve this, we will give input m, the output file of Phase 3 Task 2 'Phase3Q2.txt' and three frames as a combination of video number and frame number. File 'Phase3Q2.txt' contains the video number, frame number of two frames as well as sim(a,b) i.e. similarity value between the two frames. The significance of this file is it contains top 'k' most similar frames of each frame in the database along with the similarity value with those 'k' frames.


Phase3Q2 - Notepad

File Edit Format View Help

Va	Vb	Similarity(a,b)
(1,1)	(4,1)	1.000000
(1,1)	(10,1)	0.958895
(1,2)	(4,2)	1.000000
(1,2)	(10,2)	0.947475
(1,3)	(4,3)	1.000000
(1,3)	(10,3)	0.938353
(1,4)	(4,4)	1.000000
(1,4)	(10,4)	0.934975
(1,5)	(4,5)	1.000000
(1,5)	(10,5)	0.880718
(1,6)	(4,6)	1.000000

FIGURE 24: PHASE3Q2.TXT

Then, we have created the Perl script that reads the input file to create the OutPut.txt which contains the video and frame number as well as similarity value between two frame for all the videos.

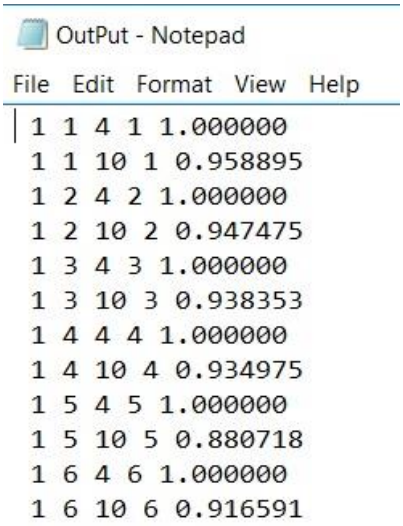


FIGURE 25: OUTPUT.TXT

In this project, each frame in the database will represent the node in the graph. Before discussing our implementation, we will first discuss the algorithm to calculate the personalized page rank.

We have implemented Personalized Page Rank with Global Seed Ranking. In the Personalized Page Rank algorithm, jumps are back to one of a given set of starting vertices called seed vertices. In this project, the input 3 frames will be considered as the seed vertices. In a way, the walk in PPR is biased towards (or personalized for) this set of starting vertices and is more localized compared to the random walk performed in PR.

In the personalized page rank with global ranking (PPR- G), we first measure the significance of the individual seed nodes in the overall graph, G i.e PageRank and, then, modulate the teleportation rates onto the seed nodes based on the relative significance values of the seeds. In other words, we would compute the PPR scores with global seed ranking (also referred to as PPR-G scores) as follows:

Given a graph, G(V,E), and a seed node

$$g = (1 - \beta)Tg + \beta\overrightarrow{s2}$$

Here

$g = Personalised\ page\ Rank\ with\ global\ ranking$

T= Transition/adjacency matrix

$\beta = damping\ factor$

$\overrightarrow{s2} = re - seeding\ vector$, if v_i $belongs\ to\ S\ then$ $\overrightarrow{s_2[i]} = \frac{\vec{p}[i]}{\sum_{v_j \varepsilon S} \overrightarrow{p[j]}}$

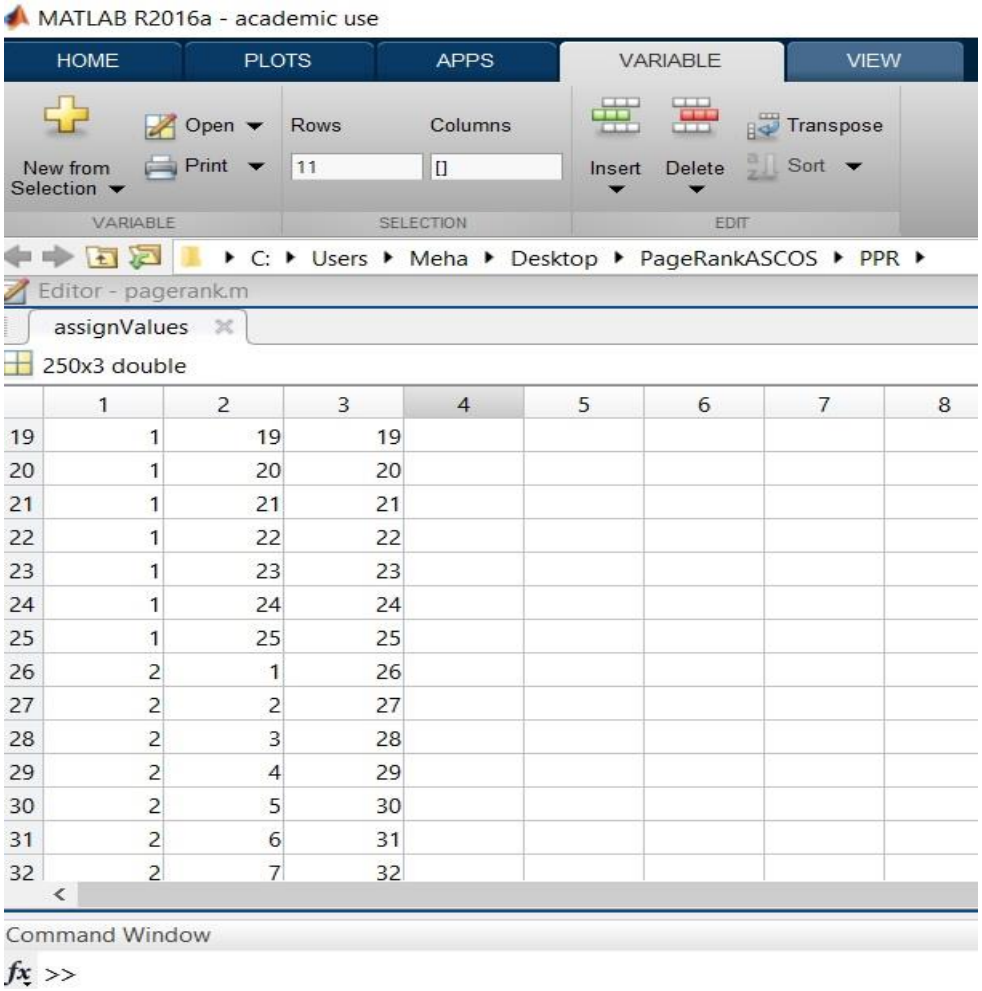
v_i $doesnt\ belong\ to\ S\ then\ \overrightarrow{s_2[i]} = 0$

(Here , S = Seed vector matrix)

Seed Vectors are the frames taken from the user. While calculating the personalized PageRank we take into account a user's interest by modifying the teleportation vector by considering a

given set of important nodes which are the target of the random jumps: given a set of nodes s ⊆ v , instead of jumping to a random node in v with probability β, the random walk jumps to one of the nodes in the seed set, S, given by the user.

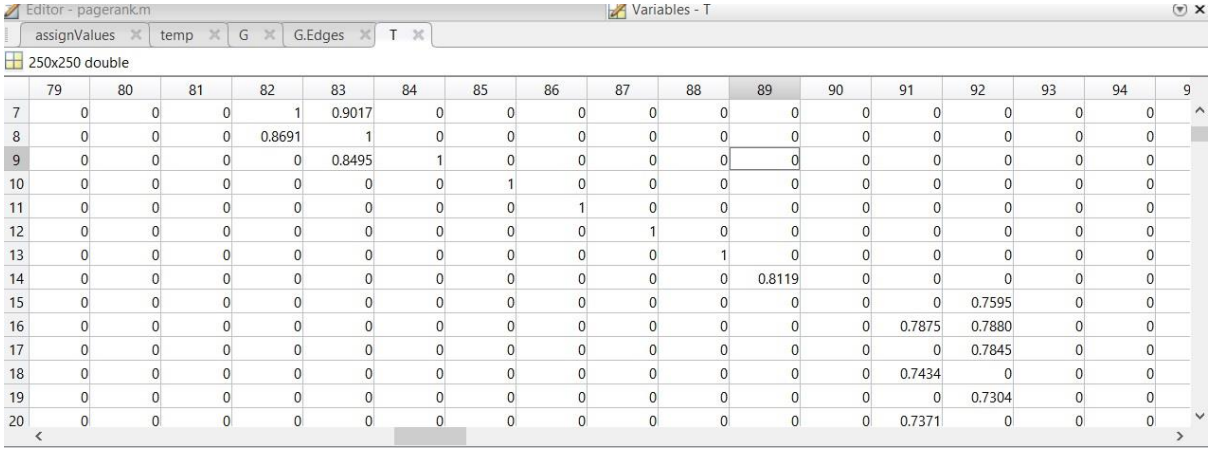
Our first task was to obtain the adjacency matrix T, as used in the above PageRank formula. To achieve it, we have created the 'assignValue' matrix that give a unique value to each of the video frame pair in our database.



	1	2	3	4	5	6	7	8
19	1	19	19					
20	1	20	20					
21	1	21	21					
22	1	22	22					
23	1	23	23					
24	1	24	24					
25	1	25	25					
26	2	1	26					
27	2	2	27					
28	2	3	28					
29	2	4	29					
30	2	5	30					
31	2	6	31					
32	2	7	32					

Figure 26 : AssignValue Matrix

This matrix will also help us in identifying the unique nodes in our database. Then, we will create an adjacency matrix using OutPut.txt file (Phase3Task2 output) such that for each unique node/frame, its value in the matrix with respect to 'k' the similarity value will represent most similar frames. Please note that 'k' most similar frames and similarity value will be obtained from the OutPut.txt file. Apart from 'k' frames, similarity value between other nodes have been assumed as 0.

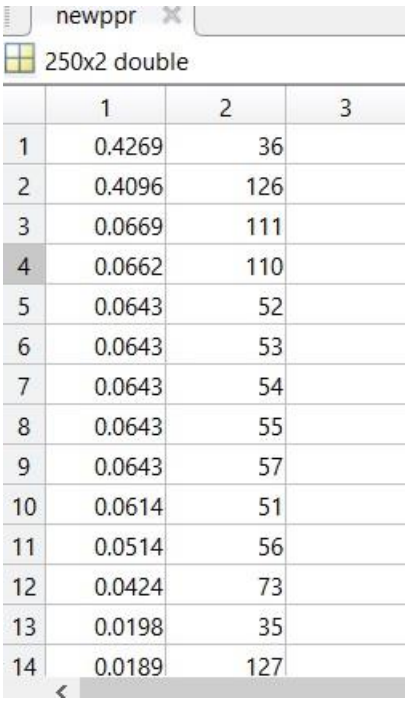


	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	9
7	0	0	0	1	0.9017	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0.8691	1	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0.8495	1	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0.8119	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7595	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	0.7875	0.7880	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7845	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0.7434	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7304	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0.7371	0	0	0	

Figure 27 : Adjacency Matrix

As soon as we obtain the adjacency matrix, we will implement the above mentioned recursive personalized page rank algorithm. We have used damping factor=0.85 and initialized the Personalized PageRank matrix by eigen values. We have calculated the page rank of all the nodes in order to compute the re-seeding vector in the above-mentioned formula.

After observing the output for different values, we have taken 100 iterations to converge the solution.



	1	2	3
1	0.4269	36	
2	0.4096	126	
3	0.0669	111	
4	0.0662	110	
5	0.0643	52	
6	0.0643	53	
7	0.0643	54	
8	0.0643	55	
9	0.0643	57	
10	0.0614	51	
11	0.0514	56	
12	0.0424	73	
13	0.0198	35	
14	0.0189	127	

Figure 28 : Personalized PageRank scores. Second column denotes the unique number given to video and frame pair.

As soon as the solution is converged, we will obtain the personalized page rank score with respect to each node and as specified by the user, we will visualize the output of ‘m’ most significant frames.

ASSUMPTIONS:

- The graph is assumed to be free of self-loops
- The value of P, damping factor is taken as .85 as suggested in the original Larry page research paper.
- The value of d is 3. as the input files contains 3 most similar frames for evry frame of every video
- The graph is assumed to be free of self-loops i.e. similarity value is taken as 0 between two same frames in the adjacency matrix.
- The value of P, damping factor is taken as .85 as suggested in "Sergey Brin , Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems, v.30 n.1-7, p.107-117, April 1, 1998"
- Apart from 'k' frames, we have assumed similarity value between other nodes as 0.

INTERFACE SPECIFICATIONS :

INPUT :

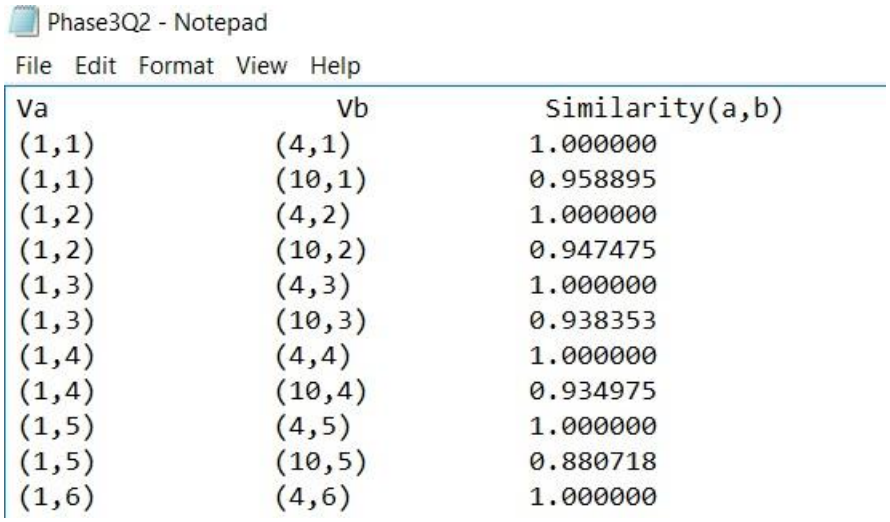
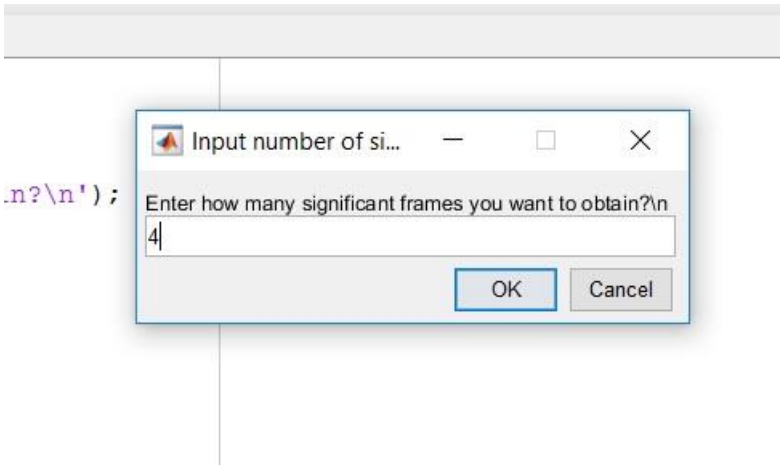
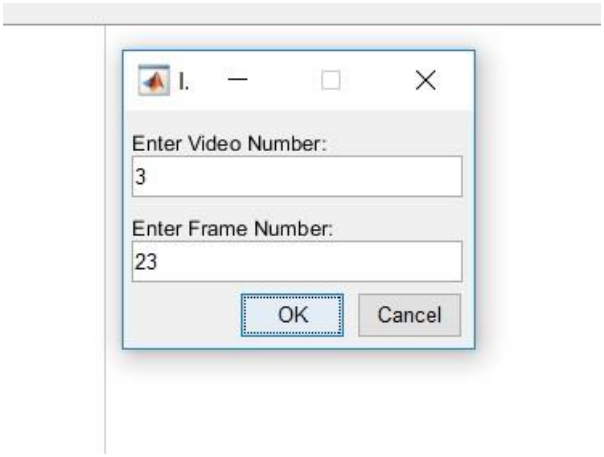


Figure 29 : Phase3Q2.txt

- the user has to enter the value of m.



- The user also has to enter the video and frame number of the 3 seed frames



OUTPUT :

- The frames below show the m most significant frames

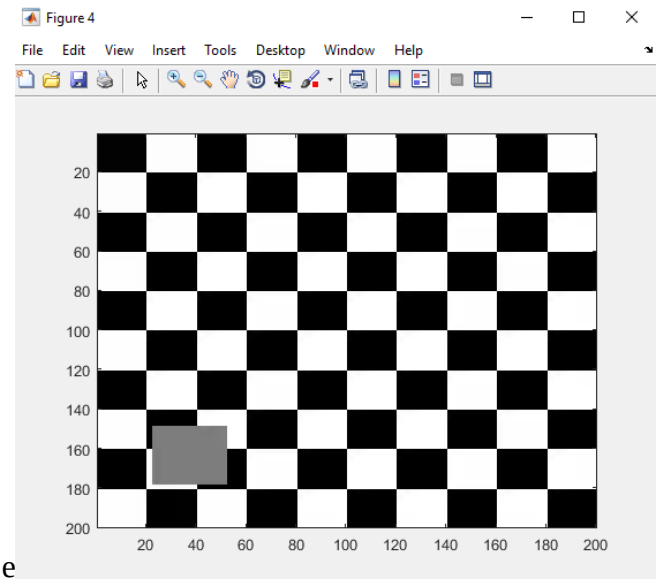


Figure 4
File Edit View Insert Tools Desktop Window Help
20 40 60 80 100 120 140 160 180 200
20 40 60 80 100 120 140 160 180 200

e

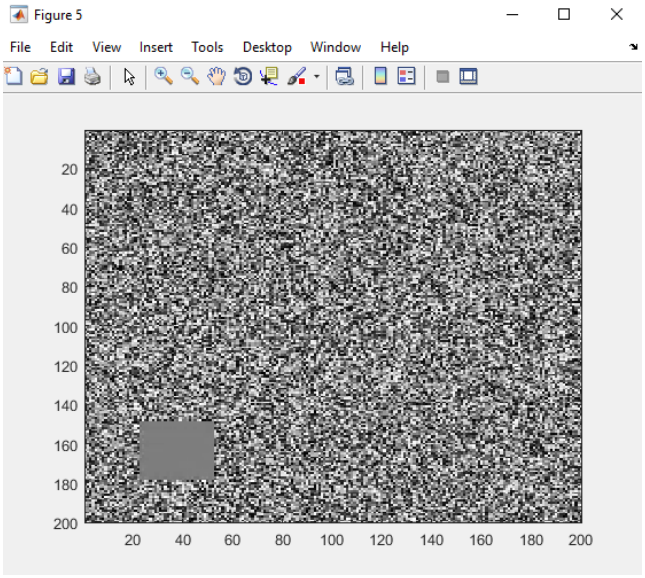


Figure 5
File Edit View Insert Tools Desktop Window Help
20 40 60 80 100 120 140 160 180 200
20 40 60 80 100 120 140 160 180 200

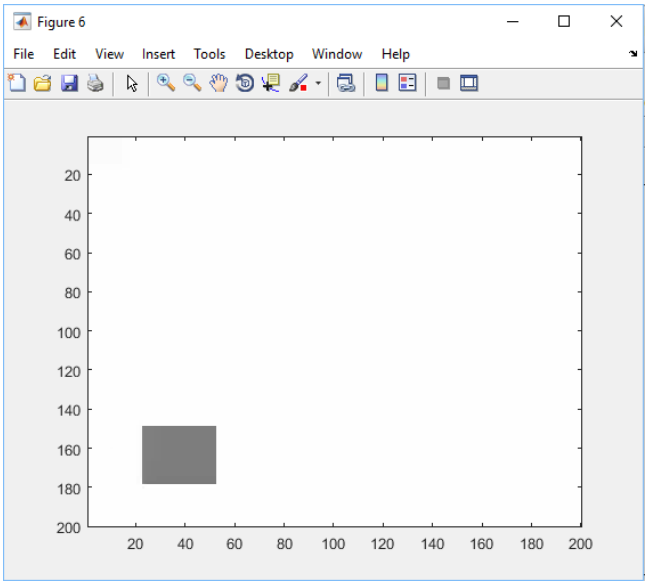


Figure 6
File Edit View Insert Tools Desktop Window Help
20 40 60 80 100 120 140 160 180 200
20 40 60 80 100 120 140 160 180 200

[B] Modified ASCOS

Implementation:

In order to achieve this, we will give input m and the output file of Phase 3 Task 2 'Phase3Q2.txt'. File 'Phase3Q2.txt' contains the video number, frame number of two frames as well as sim(a,b) i.e. similarity value between the two frames. The significance of this file is it contains top 'k' most similar frames of each frame in the database along with the similarity value with those 'k' frames.

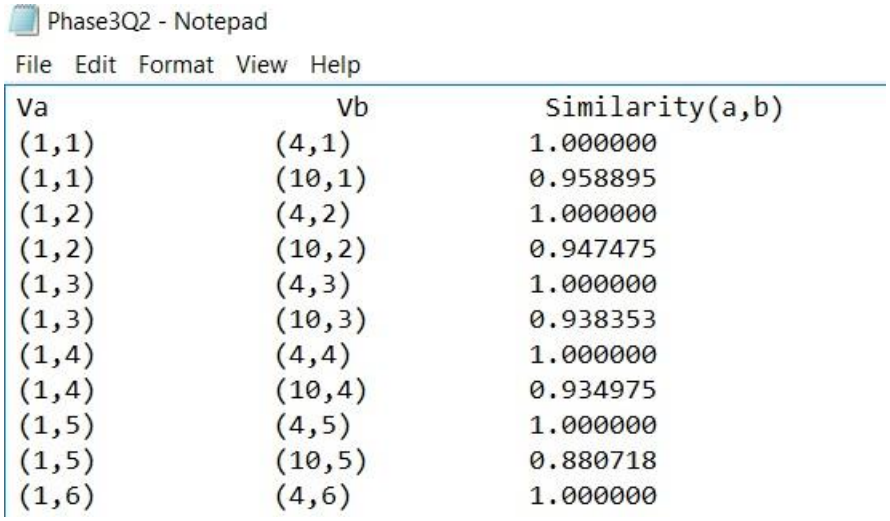


FIGURE 30: PHASE3Q2.TXT

Then, we have created the Perl script that reads the input file to create the OutPut.txt which contains the video and frame number as well as similarity value between two frame for all the videos.

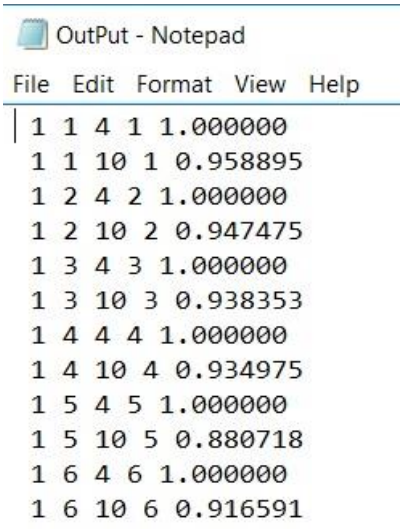


FIGURE 31: OUTPUT.TXT

In this project, each frame in the database will represent the node in the graph. Before discussing our implementation, we will first discuss the algorithm to calculate the ASCOS score.

In the ASCOS similarity measure, the similarity score from a node/frame i to a node/frame j is dependent on the similarity score from node i's in-neighbours to node j. This statement has two interesting properties:

3. the calculation excludes the out-neighbours.
4. the similarity score is asymmetric because it considers the in-neighbours of frame i but not the in-neighbours of frame j.

We have used the below recursive algorithm to calculate ASCOS score with respect to each frame.

S = c*P*S + (1-c)f

Here,

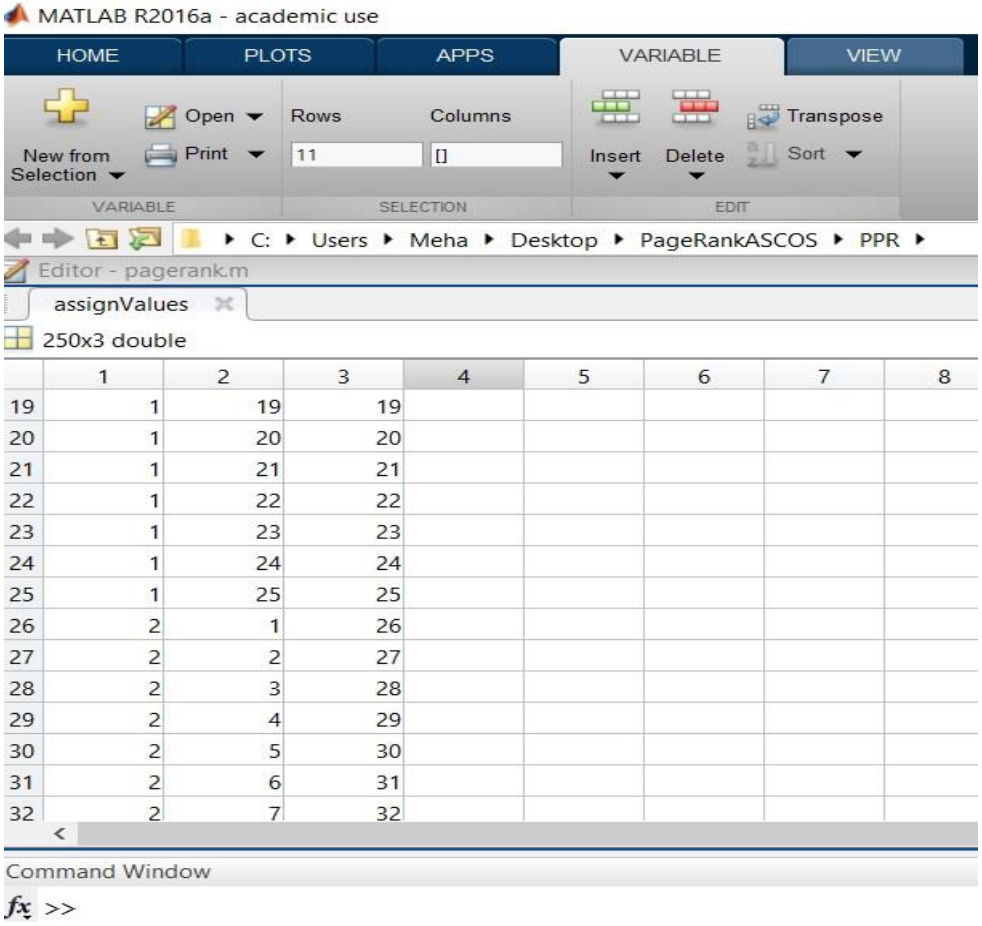
c. S: Modified ASCOS similarity score matrix for all the frames
d. c: Relative importance parameter. Its value lies between 0 and 1. It controls the relative importance between the direct neighbors and indirect neighbors, that is, neighbors' neighbors. The smaller the value, the less important the indirect neighbors are. In this project, we have taken its value as 0.9 to give high importance to the neighbor's neighbor while calculating the ASCOS score.
e. P: It is the column normalized matrix of adjacency matrix such that P =[p(i,j)] and p(i,j) = a(i,j) / sum(a(k,j)) where
   k -> in-neighbours of frame i
   a(i,j) -> similarity value between frame I and frame j
   a(k,j) -> similarity value between frame k and frame j
f. f: re-seeding vector, if $v_i \ belongs\ to\ S\ then\ \overrightarrow{f[i]} = \frac{\overrightarrow{ascos}[i]}{\sum_{v_j \varepsilon S} \overrightarrow{ascos[J]}}$

$$v_i \ doesnt\ belong\ to\ S\ then\ \overrightarrow{f[i]} = 0$$

(Here , S = Seed vector matrix)

Seed Vectors are the frames taken from the user. While calculating the modified ASCOS similarity measure we take into account a user's interest by modifying the teleportation vector ((1-c)f) by considering a given set of important nodes: given a set of nodes s ⊆ v , instead of jumping to a random node in v with probability c, the random walk jumps to one of the nodes in the seed set, S, given by the user.

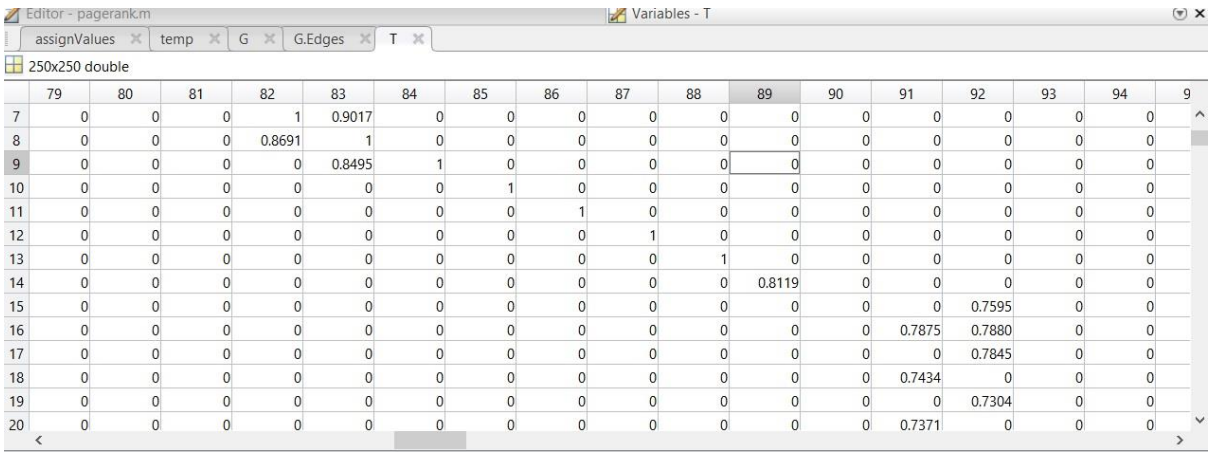
Our first task was to obtain the adjacency matrix A in order to obtain the matrix P as used in the above modified ASCOS formula. To achieve it, we have created the 'assignValue' matrix that give a unique value to each of the video frame pair in our database.



	1	2	3	4	5	6	7	8
19	1	19	19					
20	1	20	20					
21	1	21	21					
22	1	22	22					
23	1	23	23					
24	1	24	24					
25	1	25	25					
26	2	1	26					
27	2	2	27					
28	2	3	28					
29	2	4	29					
30	2	5	30					
31	2	6	31					
32	2	7	32					

Figure 82 : AssignValue Matrix

This matrix will also help us in identifying the unique nodes in our database. Then, we will create an adjacency matrix using OutPut.txt file (Phase3Task2 output) such that for each unique node/frame, its value in the matrix with respect to 'k' the similarity value will represent most similar frames. Please note that 'k' most similar frames and similarity value will be obtained from the OutPut.txt file. Apart from 'k' frames, similarity value between other nodes have been assumed as 0.



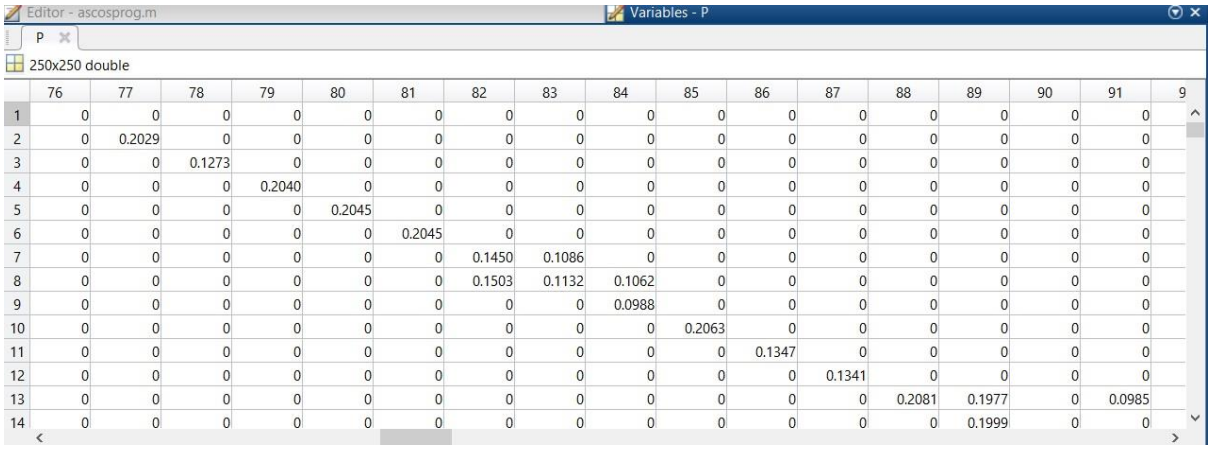
	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	9
7	0	0	0	1	0.9017	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0.8691	1	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0.8495	1	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0.8119	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7595	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	0.7875	0.7880	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7845	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0.7434	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7304	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0.7371	0	0	0	

Figure 93 : Adjacency Matrix

Then, in order to compute the P matrix, we need the similarity values between two nodes say A and B as well as similarity value between in-neighbours of node A and B.

p(i,j) = a(i,j) / sum(a(k,j))

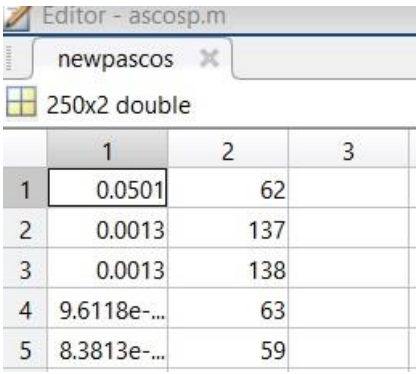
As we have assumed distance between two non-similar nodes i.e. nodes apart from k similar nodes as 0, there might be a case when sum of distance between each in-neighbours with the target node is 0. To handle that case when denominator becomes 0, we have assumed the denominator to be 100 so that overall value of p[i,j] minimizes and as a result its impact on the overall modified ASCOS score becomes negligible.



	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	9
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0.2029	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0.1273	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0.2040	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0.2045	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0.2045	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0.1450	0.1086	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0.1503	0.1132	0.1062	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0.0988	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0.2063	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	0.1347	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	0	0	0	0.1341	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	0	0.2081	0.1977	0	0.0985	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1999	0	0	

As soon as we obtain the P matrix, we will implement the above mentioned recursive page rank algorithm. We have used c=0.9 and initialized the modified ASCOS matrix by eigen values. We have calculated the ascos similarity value of all the nodes in order to compute the re-seeding vector in the above-mentioned formula.

After observing the output for different values, we have taken 100 iterations to converge the solution.



	1	2	3
1	0.0501	62	
2	0.0013	137	
3	0.0013	138	
4	9.6118e-...	63	
5	8.3813e-...	59	

Figure 101 : modified ASCOS scores. Second column denotes the unique number given to video and frame pair.

As soon as the solution is converged, we will obtain the modified ASCOS score with respect to each node and as specified by the user, we will visualize the output of ‘m’ most significant frames.

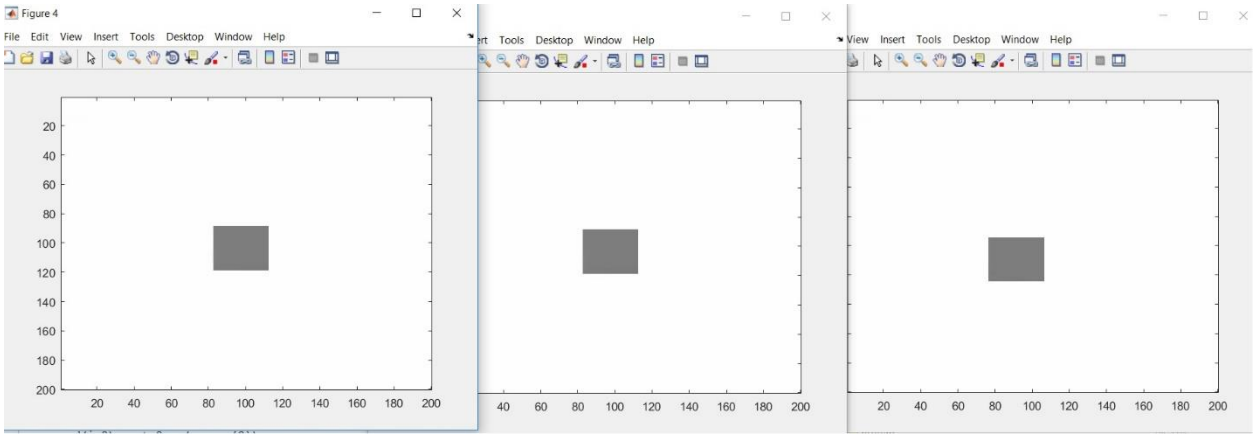


Figure 112 : Output-'m' most significant frames

Assumptions:

- The graph is assumed to be free of self-loops i.e. similarity value is taken as 0 between two same frames in the adjacency matrix.
- The value of c, relative importance parameter is taken as 0.85 as suggested in "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)"
- Apart from 'k' frames, we have assumed similarity value between other nodes as 0.
- When the sum of similarity value between all the in-neighbours and target node is zero, we will assume it to be 100 so that overall value of p[i,j] minimizes and as a result its impact on the overall ASCOS score becomes negligible.

INTERFACE SPECIFICATIONS :

INPUT :

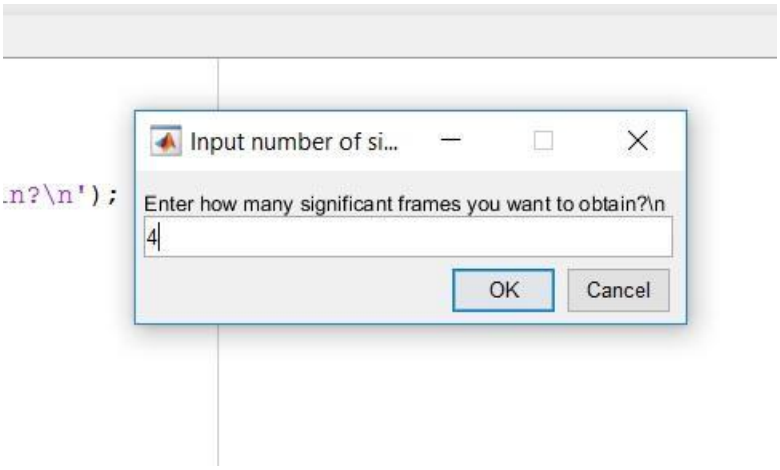
Phase3Q2 - Notepad

File Edit Format View Help

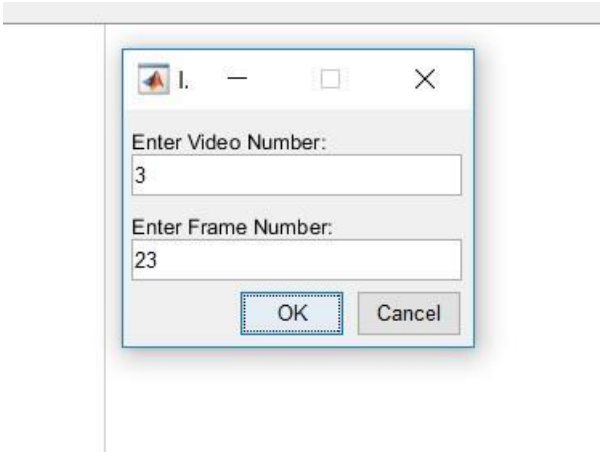
Va	Vb	Similarity(a,b)
(1,1)	(4,1)	1.000000
(1,1)	(10,1)	0.958895
(1,2)	(4,2)	1.000000
(1,2)	(10,2)	0.947475
(1,3)	(4,3)	1.000000
(1,3)	(10,3)	0.938353
(1,4)	(4,4)	1.000000
(1,4)	(10,4)	0.934975
(1,5)	(4,5)	1.000000
(1,5)	(10,5)	0.880718
(1,6)	(4,6)	1.000000

Figure 34 : Phase3Q2.txt

- the user has to enter the value of m.



- The user also has to enter the video and frame number of the 3 seed frames



OUTPUT :

- The frames below show the m most significant frames

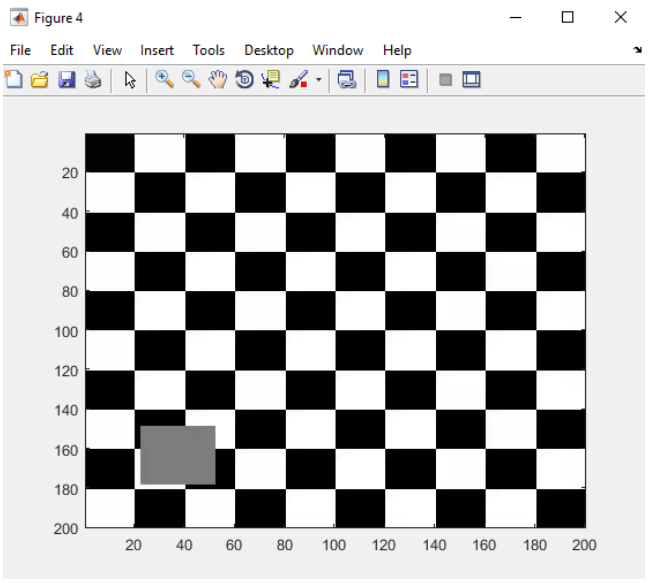


Figure 4
File Edit View Insert Tools Desktop Window Help
20 40 60 80 100 120 140 160 180 200
20 40 60 80 100 120 140 160 180 200

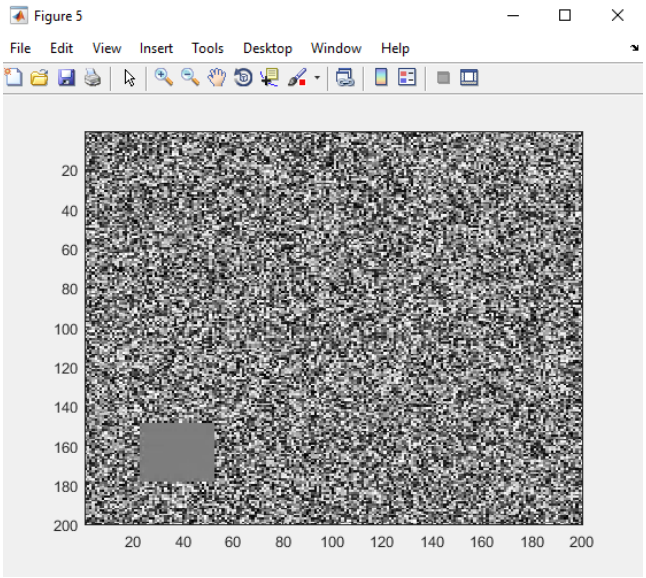


Figure 5
File Edit View Insert Tools Desktop Window Help
20 40 60 80 100 120 140 160 180 200
20 40 60 80 100 120 140 160 180 200

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

Keywords: - LSH, nearest neighbor search

Problem Specification: -

Implement a Locality Sensitive Hashing (LSH) tool, which takes as input

1> file filename d.spc,
2> the number of layers, L, and – the number (2K) of buckets per layer

Maps each key point into a hash bucket for each layer.

Output entries of the form: - {layer, bucket, <Video no; Frame no; Cell no; Loc X; Loc Y >}

Approach: -

As given in the theory of LSH, that LSH generates random planes and give binary value to each point in data set as 1 or 0 depending upon its location with respect to plane. This step is repeated K times for each layer. So I have created a random hyper plane using "rand()" function in MATLAB.

Keeping the value of coefficients between -1 and 1 and Value of c or constant to be between -100 and 100, so that all the possible quadrants can be randomly generated. Though limiting the range is not completely random, it's a pseudo random approach, get big difference in value of c and coefficients make the approach nearly random.

I am creating K planes for each Layer, as per above mentioned method. Then for each plane generated I am looking for sign of dot product of the normal of the plane with the data point vector. So, if the dot product is positive I know it lies in the region above the hyperplane, so I am assigning it the Value 1 and vice Versa. We can assign any value 0 or 1 it does not matter till you are using same criteria for each data point.

Steps to execute: -

For input: - One is require to keep the file which is output from Task 1 of this Phase.

1> Open Task5_Main.m

2> Run it

3> User will be asked about two inputs: -

a. Number of layers (L)

b. The number of (K) Hyper-Plane to form 2^K buckets

4> Output File name is: - Result.txt

Overview of various files and their purpose: -

1> Formatting.pl: - Takes input as "Sift_Reduced_Formatted.txt" (the output of task 1) and Formats the data in such a way that it can be directly loaded in MATLAB as a Single Matrix. Gives as output the "OUTPUT.txt".

2> Sift_Reduced_Formatted.txt: - This is the file which is formed as output from Task 1 and is used as input in this Task.

3> OUTPUT.txt: - This file is created as output of "Formatting.pl". It's an intermediate file. It's used as input to "Task5_Main.m". It contains Formatted data of "Sift_Reduced_Formatted.txt".

4> Result.txt: - The file which contains output of this Task.

5> Task5_Main.m: - The file which creates first thread of this program. This is the only file required by user to be execute for this Task.

Assumptions: -

1. The rand() function in MATLAB creates the data randomly, which is not possible not its pseudo random generation is assumed as near to random.
2. The random plane's equation coefficients are randomized in range -1 to 1.
3. Perl binaries are assumed to be present in the system along with the MATLAB.
4. Assuming we have writes to create files in "present working directory".
5. All input files are present in "present working directory".

Input: -

1. Phase:3 Task:1 output file "Sift_Reduced_Formatted.txt" is required as input.

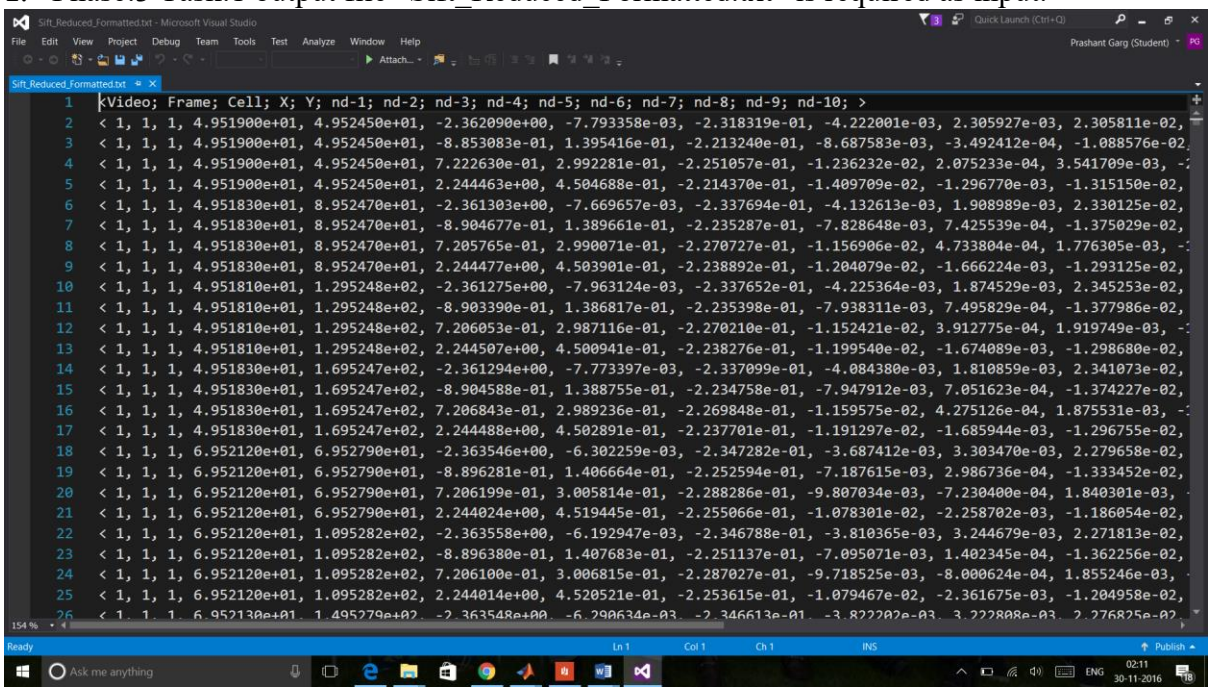


Figure 30: The Input file from TASK1

2. The input from the user: Layers(L) and Bucket size 2^k (k).

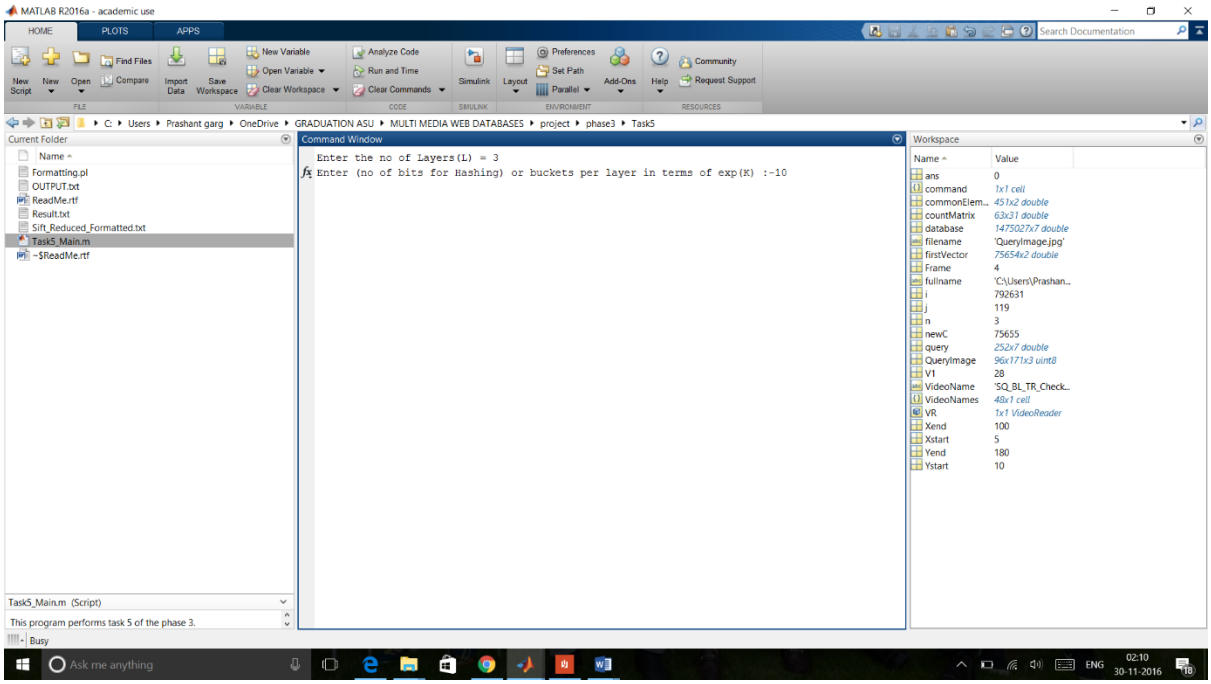


Figure 31: The sample screenshot of user input

Output: -

1. The Output is saved in file “Result.txt”.

Format: {layer_num,bucket_num,<i;j;l;x;y>}

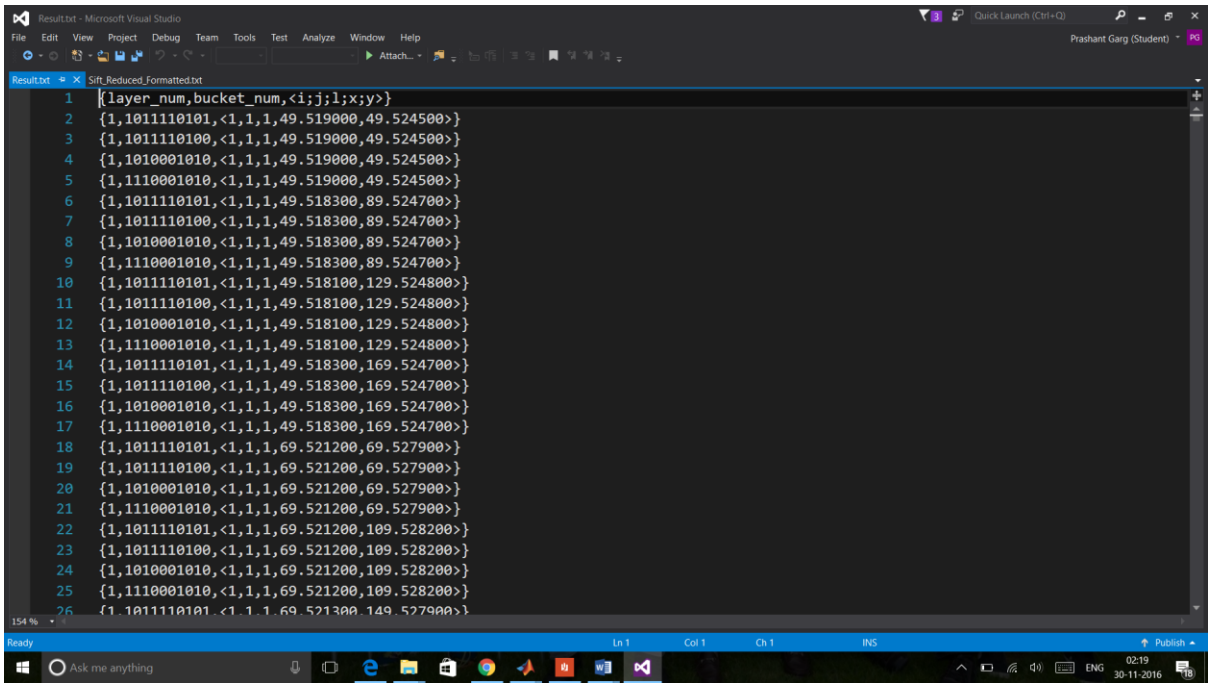


Figure 32: sample output

Software/ Libraries required: -

1. MATLAB 2016a
2. Perl Binaries (5.0 and above)
3. Text Editor for viewing the output.

Task 6

LSH- Locality Sensitive Hashing

Problem Specification-

In this task, we had to build a similarity based video object search tool which takes as input the following:

1. Output file generated from task 5.
2. An integer n, which is basically the top n results for the given query.
3. Query object with video number, frame number and x1, x2, y1, y2 given of the object.

Additionally, the constraint was that the top results should not be from the video from which the query is given.

Theory

Locality Sensitive Hashing is technique to reduce the dimensions of the data. In this technique, we divide the data points into different buckets, each of which acts as a new dimension. We try to ensure that the data objects in each bucket have maximum probability of collisions. In other terms, similar objects are placed into same bucket. In case of query, only that bucket is searched for giving the result [61].

Implementation-

In this task, we had the output from the task 5 as the input file. The various sift vectors were placed in the buckets based on their similarity. The value of n for top n results and object to be used in query were taken as input from the user.

1. Take the output file from task 5 as the input to this task. It will act as the database for this task.
2. Take the value of n as input. This will tell us top n results to be shown.
3. Take the query object as the input. We want to know the video number, frame number, x1, x2, y1, y2 of the object.
4. We will then extract all the sift vectors which will lie in this range.
5. Then taking each sift vector at a time, we extract the similar object from the database.
6. The video number and frame number of the similar object is incremented in the new similarity matrix.
7. Similarity matrix is video*frame matrix with videos as rows and frames as columns.
8. For each of the similar object 1 number is incremented in the matrix.
9. This process is done for each of the query sift vectors.
10. Lastly, the top n maximum numbers are extracted which tell us the top n results of the query.
11. The query video along with the top matches is shown to the user.
12. The results are stored in the current directory.
13. Additionally, the number of unique sift vectors considered, total number of sift vectors considered and byte of data accessed are reported on the command window.

Interface Specifications

Input

There are three inputs to this task.

1. Output file generated for task 5.

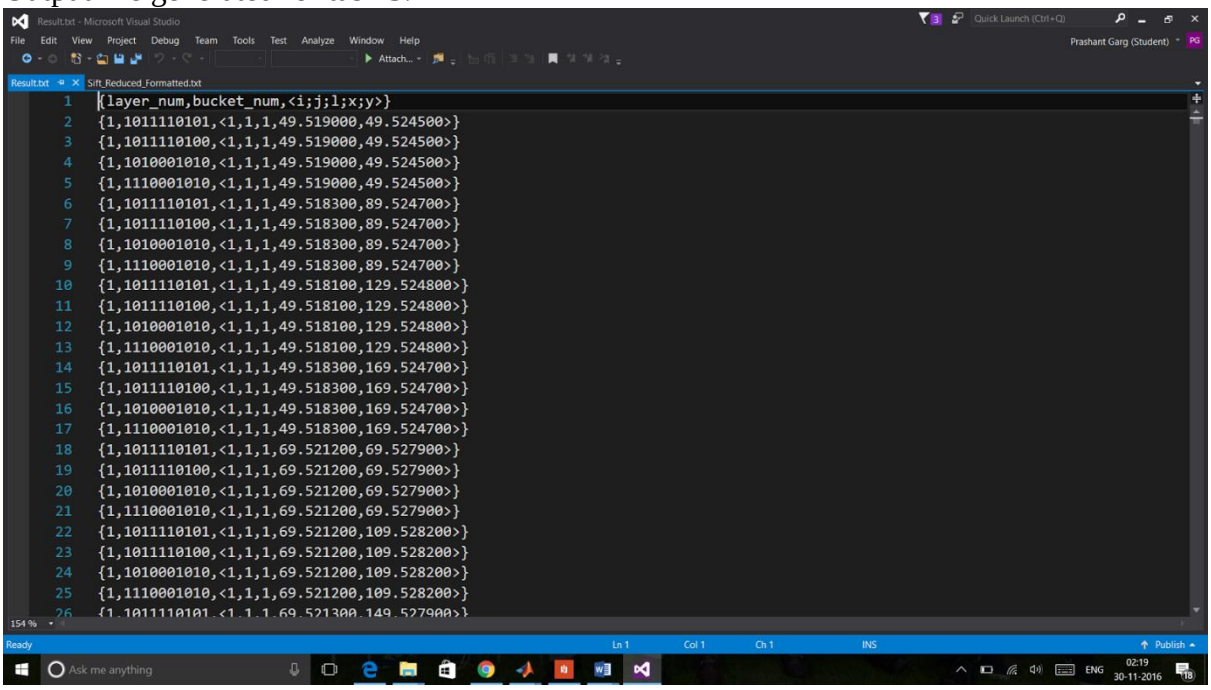


Figure36: Input file for task 6

2. N which is the value of top n results.
3. Object as query with video number, frame number and coordinates.

Figure37- Query Image

Output

The output for this task are the top n results of the query object. The total number of unique sift vectors considered, total number of sift vectors considered and bytes of data accessed.

Figure38: Top result for the query

Figure39: Second best result for the query.

Figure40: Third best result for the query.

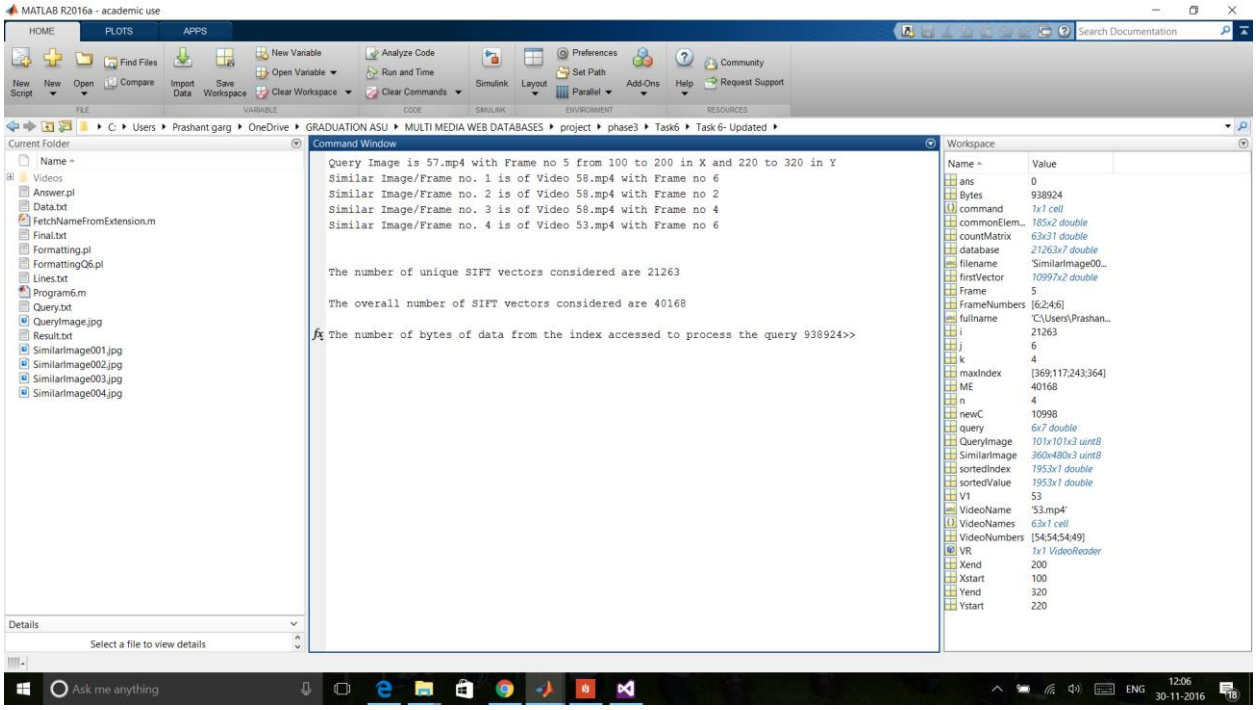


Figure41- Number of unique sifts, total sifts and bytes accessed.

Assumptions

1. The input files are in correct format.
2. The perl binaries are already installed in the system.
3. The value of n is an integer.
4. The video number, frame number and coordinates are in range and integers.

Steps to Execute:

1. Open the file 'Program6.m'
2. Run it on the matlab.
3. Input the values of Video Number, frame number, x1, x2, y1, y2.
4. Results would be displayed on command prompt and stored in the current directory.

Related Study: -

- Alexandr Andoni and Piotr Indyk. "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions". Communications of the ACM, vol. 51, no. 1, 2008, pp. 117-122.
- Sergey Brin , Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems, v.30 n.1-7, p.107-117, April 1, 1998
- 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)
- Huang, S., Li, X., Candan, K. S., Sapino, M. L. (2016). Reducing seed noise in personalized PageRank. Social Network Analysis and Mining, 6(1), 1-25.

Conclusion

In the end of the phase, we learnt a lot about finding the most significant frame and how to prune the search space using LSH. We performed dimensionality reduction using PCA on the sift vectors we found out during the phase 1. For demo, dimensions were given as 10 and 60. We created the similarity matrix in the second phase and used its output for finding the most significant frame using renowned algorithms page rank and ASCOS. We also implemented the personalized page rank algorithm and ASCOS. In task 5, we used LSH algorithm to prune our search space. We used the output of this task to find similar objects in task 6. With similar objects being in same bucket, finding the n-most similar frames became faster.

References: -

[1] https://en.wikipedia.org/wiki/Principal_component_analysis

[2] http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[3] http://www.nature.com/nbt/journal/v26/n3/full/nbt0308-303.html

[4]https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-[30]dummies-eigenvectors-eigenvalues-and-dimension-reduction/

[31] https://en.wikipedia.org/wiki/PageRank

[32] Hung-Hsuan Chen and C. Lee Giles. 2015. ASCOS++: An Asymmetric Similarity Measure for Weighted Networks to Address the Problem of SimRank.

[33] https://en.wikipedia.org/wiki/Adjacency_matrix

[34] http://www.thecrazyprogrammer.com/2014/03/representation-of-graphs-adjacency-matrix-and-adjacency-list.html

[35] https://en.wikipedia.org/wiki/Graph_theory

[36] Sergey Brin , Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems

[37] Huang, S., Li, X., Candan, K. S., Sapino, M. L. (2016). Reducing seed noise in personalized PageRank. Social Network Analysis and Mining,

[36] https://www.youtube.com/watch?v=NPcMS49V5hg, Lecture 05 - Scale-invariant Feature Transform (SIFT), Dr. Mubarak Shah

[37]http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html

[38]http://dsp.stackexchange.com/questions/10423/why-do-we-use-keypoint-descriptors

[39]http://www.cs.ubc.ca/~lowe/keypoints/

[51] Beyer, Kevin, et al. "When is “nearest neighbor” meaningful? “International *conference on database theory*. Springer Berlin Heidelberg, 1999.

[52] Zhou, Jile, Guiguang Ding, and Yuchen Guo. "Latent semantic sparse hashing for cross-modal similarity search." *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014.

[53] http://www.cs.wustl.edu/~pless/546/lectures/L11.html

[54] https://i.ytimg.com/vi/Arni-zkqMBA/hqdefault.jpg

[61] https://en.wikipedia.org/wiki/Locality-sensitive_hashing