Video Analysis and Indexing

# Abstract

The vast amount of data delivered from a video stream provides the need to identify the contents in a video. A higher level of content abstraction and investigation in the distinct region is required to identify the video content. A video can contain several points of interest in every frame. We extract the Sift key points from a set of videos and use the extracted “feature description” to find video similarities and do indexing to find the nearest and most similar object. Similarity search in high-dimensional spaces has become increasingly important in databases and search engines, particularly for the feature rich video data. Since the multimedia objects are of huge size, we initially reduce the dimensionality of the individual data sets using PCA (Principal component analysis) and do similarity and use Locality Sensitive Hashing to index and find the nearest neighbor for the given query video frame. We also use page rank algorithm and ASCOS to find the most important frames in the given data set. We present m-nearest neighbor of the query object using image representation. We also present how quick an Locality sensitive hashing helps in finding the nearest object within O(Log n) instead of linear search. The resulting neighbor frames are visualized.

Group Members:

Anoop Jatavallabha Vijayakumar

Deepak Soundararajan

Narendra Kumar Sampath Kumar

Ravikiran Tangirala

Santosh Mandya Jayaram

Keywords:

Video analysis, SIFT keypoints, Principal Component Analysis, Nearest Neighbor, PageRank, ASCOS, Locality Sensitive Hashing.

Introduction:

The two biggest challenges in Multimedia data management is data redundancy and query complexity. When the input data to an algorithm is too large to be processed and it is suspected to be redundant, it directly has an impact on the reliability of the product. To overcome the data redundancy, the input data can be transformed into reduced set of feature vectors. The other challenge is the complexity. The time taken to find the most similar objects in a database. In Many cases, Indexing overcomes the complexity. Indexing the data helps in finding the most similar data at a faster rate compared to linear search. However, we need to compromise between accuracy and complexity to a certain extent. Feature Selection largely relies on the purpose of the application. For example, sports video summarization usually focuses more on the motion of foreground athletes and balls, while generic-purpose applications may concern more about the background scenes. This phase of the project focuses on utilizing the extracted SIFT vectors from the previous phase and finds the similar frames, use Locality sensitive hashing (An Approximate Nearest neighbor search) to search the frames with most similar objects, PageRank and ASCOS to compute the most significant frames in the given dataset.

Terminology:

*Feature:*

A Feature is a piece of information which is relevant for solving the computational task related to a certain application.

*Feature vectors:*

When two or more different features are extracted, resulting in two or more feature descriptors at each image point. The organized collection of the information provided by two or more feature descriptors as the elements of one single vector is called Feature vector

*Feature Detection:*

Feature detection refers to methods that aim at computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not

*Feature Extraction:*

Feature Extraction is a process of extracting features from every frame in a video, independently of past and future frames.

*SIFT (Scale Invariant-feature transform):*

Scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images

*Euclidean distance:*

This is the straight-line distance between two points*.* Lower the value of Euclidian distance higher will be the similarity between the vectors.

Euclidian distance for n dimensions is defined as:  
deuclidian (p, q) = ((p1 - q1)2+( p2 - q 2)2+( p3 - q3)2+( p4 - q4).+. . . . . . +( pn - qn)2)1/2

Goal Description:

* Given the SIFT dataset, the goal is to reduce the dataset to ‘d’ dimensions using Principal Component Analysis. The descriptors along with the scale and orientations should be reduced into a d-dimensional vector space.
* Given the PCA reduced dataset, we need to find the k similar frames for every frames in the given video set along with their similarity percentage. The data is saved to a graph file filename\_d\_k.gspc
* Given the graph file filename\_d\_k.gspc, we need to find most significant ‘m’ frames using PageRank and ASCOS measure and visualize the selected ‘m’ frames for both approaches
* Given the graph file filename\_d\_k.gspc, we need to find most significant ‘m’ frames using personalized PageRank and suitably modified version of ASCOS measure to account for seeded frames and visualize the selected ‘m’ frames for both approaches
* Given the PCA reduced data from first step, we need to implement a Locality Sensitive Hashing Tool, which takes No of Layers L and bucket value is computed from the given ‘K’ value. It maps each SIFT key point to a bucket. The result is stored to a file filename\_d.LSH
* Given the data from the LSH, we implement a similarity based video object search tool which finds the similar frames containing in the entire dataset. Also, since the search is optimized through LSH, we present the number of unique SIFT vectors considered compared to total number of vectors and number of bytes of data from the Index accessed to process the query.

Assumptions:

Assuming the input SIFT database filename.sift is generated for the given set of videos

Task 1:

* The Scale and Orientation are of different scale compared to SIFT key point descriptor. We need to normalize the data before PCA transform. Co-relation or mean-centering gives the proper result. Z-score scales the data to have mean 0 and scaled to have standard deviation 1, so we use Z-score to normalize. [1] [2]
* The entire dataset containing key point samples is taken as object and Scale, Orientation and 128 descriptor values as feature in object-feature matrix. We have totally 130 dimensions to reduce.
* Assuming all videos are in same directory.

Task 2:

* Assuming the task -1 output i.e pca reduced output is provided ,dimensions d its reduced , k most similar to be found , directories are provided.
* Assuming VL-Feat library is pre-installed in matlab.
* We are using K-Means to create clusters to further prune/reduce the descriptor space and use the cluster centroids of each frame for finding the similar descriptors. This is done to decrease the computation speed and it doesn’t hamper the accuracy and is on par if we consider all the descriptors.
* We are creating 50 clusters ( found by trial and error to be optimal number of clusters that’s giving better results ).
* We use kd - tree to create the index for the centroids created for each frame and then we query the kd -tree with other frames to find the nearest neighbour matches. The number of descriptors matched is found using Lowe’s approach [ 7 ] and is as explained below.
* Suppose you have a point P in Frame 1(F1) and you want to find the "best" match in Frame 2(F2). We compare the descriptor of P in F1 to all the descriptors in F2. For comparison we used Euclidean distance as suggested by Lowe in his paper “Distinctive Image Features from Scale-Invariant Key points” [7]. Then, we find two points in F2, say U & V which have the least and second-least distance (say, Du and Dv) from P respectively. Here's what Lowe recommended: if Dv/Du >= threshold (We used values between 1 - 1.5), then this match is acceptable; otherwise, it's ambiguously matched and is rejected as a correspondence and we don't match any point in F2 to P. Essentially, if there's a big difference between the best and second-best matches, you can expect this to be a quality match.
* We are using 0.7 as the threshold for the descriptors to match. The reason and the explanation for the same is given above.

Task 3:

PageRank:

* The similarity percentage is taken as edge weight
* The page rank scores are initially taken to be 1/NoofFrames
* Alpha is a scalar damping factor taken to be 0.85, which controls the relative importance between neighbors [4]
* The difference between the (j+1)th iterated PageRank scores and the jth iterated Page Rank scores is taken as threshold which is power(1/10,j) , where J = iteration. E.g.: Iteration 1 has threshold 0.1, iteration 2 has 0.01 as threshold… [3]

ASCOS:

* Assuming graph data is already known.
* Assuming initial guessing similarity is the similarity values found in the graph file.
* Assuming the c value, the relative importance factor between direct and indirect neighbor is 0.85.
* The difference between the (j+1)th iterated PageRank scores and the jth iterated Page Rank scores is taken as threshold which is power(1/10,j) , where J = iteration. E.g.: Iteration 1 has threshold 0.1, iteration 2 has 0.01 as threshold… [3]

Task 4:

Personalized PageRank:

* The video number and frame number given for seed frames should be within the range of frames.
* The page rank scores are initially taken to be 1/NoofFrames
* Alpha is a scalar damping factor taken to be 0.85, which controls the relative importance between neighbors [4]
* For every seeding frame, the teleportation value is 1/|S| and for other frames it is taken as zero. [5]
* By default, all the nodes have PageRank value to be 1/NoofFrames except for the seeded nodes and their adjacent nodes which has default value and personalized teleportation values (Random walk and teleportation values 1/k) [5]
* To prevent the under accounting of seed node, we introduce self-loop so that all the seed nodes so that the seed nodes and their neighbor will have total value of 1/NoofFrames + 1/k+1 [5]

ASCOS with Seed frames:

* Assuming graph data is already known.
* Assuming initial guessing similarity is the similarity values found in the graph file.
* Assuming the c value, the relative importance factor between direct and indirect neighbor is 0.85.
* Assuming the teleportation value is 1/k+1 accounting for the seed frames as well as neighbors of the seed frames and for other frames it is taken as zero.

Task 5:

* The number of Layers L and K values are given beforehand. Number of buckets are calculated using 2^k
* The hash function can also be viewed as partitioning the space into two half-spaces by a randomly chosen hyperplane [12]
* The dimensions of the random hyperplane are generated from the range of -1 to +1 to get balanced 0/1 split [6]
* Assuming the number of hyperplane to be the ‘k’ value so we get 2^k binary values concatenated together denoting the bucket hash values

Task 6:

* Assuming the LSH file and PCA transformed SIFT data is given
* The assumptions for Task2 is taken into consideration as we find the frame containing the most similar object using the same measure
* The X1,Y1,X2,Y2 provided should contain object i.e. The SIFT descriptors should be present in the given query range. If there are no descriptors the comparison would not be made
* Assuming that the unique video and frame present in all the BucketLists has more collided to the query set than others [12]
* Total number of descriptors denote the number in Combined BucketLists obtained each Layer
* Unique number of descriptor denote the number of descriptors in the unique Video(i) and Frame(j) [ViFj set] in the BucketList collection from all the Layers.
* Total Bytes of data indexed is the bytes taken in the unique Video(i) and Frame(j) set [ViFj set]

Description of proposed Solution:

Task 1:

Prerequisite

1. The SIFT data points are generated for the given video sets

Algorithm

Read SIFT Database

X ← Take 128 descriptors and Scale, Orientation for PCA transform

Zscore(X) // for Normalization

PCAtransformed ← PCA(X)

Write\_to\_File(PCAtransformed)

end

Implementation:

Task 1 is like the PCA reduction that we did in earlier phase except for the fact that the given dimensions are of different scales (128 descriptors, scale, orientation). The data should be normalized before doing a PCA – transform. We normalize the data using Z-score, the preferred approach [1][2]. Then we do a PCA transform for the given dimension ‘D’. The D-dimensional vector space is written to a file in the format {<i,j,l,x,y>,[dim1,…,dimd]}.

[coeff,projection] = pca(X) - principal component ‘projection’ are the representations of X in the principal component space

The program also reports the d dimensions in terms of the input vector space. The d dimensions are to be reported in the form of <original index, score> in non-increasing order of scores

Output sample:

Dimension is reduced from 130 to 10 in this case

{<1,1,3,41,317,>,[-4.933081,-4.057873,-2.864079,2.654173,0.129546,0.659032,1.247459,-0.281763,0.949607,0.715954,]}

The scores are sorted as shown for the first dimension, only 5 out of 130 is shown here

<1;73;1.950961e-01>

<1;65;1.950042e-01>

<1;37;1.945760e-01>

<1;93;1.942677e-01>

<1;81;1.700276e-01>

Task 2

Prerequisite:

1. Task -1 output has to be present.
2. VL - Sift library has to be pre-installed in matlab.
3. dimensions ( d ) of the Task - 1 it has been reduced.
4. Value for k ( most similar videos ).
5. Output file directory and output filename to be generated.

Implementation:

The reduced pca output is taken as input for this task. We apply k - means clustering algorithm to create clusters for the given descriptors of each frame. The reason for applying k- means is to reduce/prune the descriptor space further for faster computation. The results obtained are on par. The centroids of the clusters of each frame are then used for comparison. kd - tree is constructed for each of the frame centroids and stored . The given frames centroids are then compared with the kd - tree of the other frame to find nearest points . The number of points matched is found based on Lowe’s[7] paper. The percentage similarity is calculated based on the number of descriptors matched with overall descriptors . Then k - most similar frames that has highest similarity percentage is provided as ouput.

Algorithm :

P -> PCA reduced descriptors

d - > dimensions its reduced to.

k -> most similar videos

for each frame

create clusters using k-means and store it

create a kdtree using k-means centroids

end for;

for each framei

for each framej

kd-treequery(framei,framej);

calcuate similarity percentage using lowe’s approach

end for;

end for;

sort the similarity percentage matrix and o/p the k-most similar videos

Correctness:

The o/p generated is verified manually and found to be almost accurate ( 95%) .

E.g.: The noise video 20.mp4 is matched mostly with 23.mp4 and 32.mp4

Output sample:

{<1,1>,<2,1>,96.000000}

{<1,1>,<61,1>,96.000000}

{<1,2>,<8,22>,93.000000}

{<1,2>,<2,7>,92.000000}

{<1,3>,<2,17>,93.000000}

{<1,3>,<34,12>,90.000000}

{<1,4>,<8,21>,85.000000}

{<1,4>,<5,18>,84.000000}

Task 3:

PageRank

Prerequisite:

1. The similarity graph G(V,E) is given

Algorithm:

P ←PageRank Matrix with the value 1/NoofFrames

T ← Adjacency Matrix

K ← No of edges each node has

Wij ← Weight of the edge between i and j

T (i,j) ← Wij, if an edge exist between i and j

0 if no edge exists between i and j

Damping factor Alpha ← 0.85

//Test for convergence using the PageRank score difference

While (I+1)th (A) – (I)(A) > power (1/10,iteration)

//PageRank is calculated using the following formula

Teleportation ← (1-alpha)/TotalFrames for all Vi

P ← Teleportation + Alpha \* (T \* (P/k));

Sort Desc according to PageRank scores

Visualize

end

Implementation:

PageRank relies on the random walks. Let Wij denote the weight of the edge between I and j. T is the adjacency matrix with value Wij if an edge exists between I and j, else 0. The PageRank vector P has all entries filled with 1/|V| where V is the number of vertex ( No of frames in this case). The teleportation vector is filled with (1-alpha)/NoofFrames. Test for convergence is implemented by placing a threshold. The difference between the (I+1)th iterated PageRank scores and the Ith iterated Page Rank scores is taken as threshold which is power(1/10,I) , where J = iteration. E.g.: Iteration 1 has threshold 0.1, iteration 2 has 0.01… [3]. The PageRank score for the top most node alone is taken into consideration for the test of convergence [3]. PageRank takes 2 iteration in total with values initially being 1/NoofFrames.

Correctness:

For the given set of videos, the most repeated content is ranked more, as the frame is bound to have many similarity matches

Output sample:

The M-most significant frames are visualized as videos and images

|  |  |  |
| --- | --- | --- |
| PageRank Score | Video Num | Frame Num |
| 0.845880806854498 | 33 | 7 |
| 0.556281883047620 | 33 | 16 |
| 0.514386350473545 | 33 | 23 |
| 0.503449444277778 | 20 | 11 |
| 0.472465869005291 | 1 | 4 |

Task 3:

ASCOS

Prerequisite:

1. The Graph data is already computed as we use the graph data to find the number of in-neighbors to each given frame and the initial guessing similarity matrix
2. The directory where the video files are present should be know in order to visualize the m most significant frames.

Algorithm:

Get the m-most significant frames and the graph file from the user as input

SimMat ← Initial Similarity Matrix

C ← 0.85 - Relative importance between direct and indirect neighbors

N(i) ← Number of in neighbors of node i

Function ASCOS()

Read the graph file given as input as an adjacency matrix SimMat of size N x N

While(SimMat(I,J) not converges)

For each I = 1 to N do

For each J = 1 to N do

If(I==J)

Update SimMat(I,J)=1;

Else

Find in-neighbors of I as N(I)

For each K=1 in N(I)

Sum=Sum+SimMat(K,J);

Update SimMat(I,J) = C / N(i) \* Sum;

End

Sum SimMat along columns and return m most significant node.

Implementation:

The ASCOS function computes the similarity between 2 nodes at a time by computing the Similarity between nodes I and J by using the node I’s in neighbors to node J. First, we read the graph file and created an adjacency matrix of size N\*N using the data in the graph and create initial similarity matrix using the similarity percentages between each node that can also be found in the graph. Now using the formulae given in Chen’s paper for describing ASCOS we update the similarity between I and J as 1 if I and J represent the same frame or by using the formula c/N(i) ∑ ∀k∈N(i) skj where N(i) is the in neighbors of the node I. We iterate the same until the similarity matrix converges.

Correctness:

On checking the top 10 most significant nodes of the given video data using PageRank and ASCOS algorithms for the same data it is found that 8 nodes in most significant 10 nodes using ASCOS are also found in top 10 nodes significant nodes found using PageRank.

Output sample:

M-most significant frames in the given video data visualized as a video and image.

|  |  |  |
| --- | --- | --- |
| ASCOS score | Video Num | Frame Num |
| 82 | 33 | 7 |
| 52 | 33 | 16 |
| 51 | 33 | 23 |
| 34 | 20 | 11 |
| 26 | 52 | 5 |

Task 4:

Personalized PageRank

Prerequisite

1. The similarity graph G(V,E) is given
2. The seed frames are given
3. No of seed frames are taken as 3

Algorithm

|S| ← No of seed nodes

P ← PageRank Matrix with the value 1/|S+1| if Vi is seed node else 0

P ← Fill all the adjacent nodes of seed with the same value 1/|S+1|

T ← Adjacency Matrix

K ← No of edges each node has

Wij ← Weight of the edge between i and j

T (i,j) ← Wij, if an edge exist between i and j

0 if no edge exist between i and j

Damping factor Alpha ← 0.85

//Test for convergence using the PageRank score difference

While (I+1)th (A) – (I)(A) > power (1/10,iteration)

Teleportation ← 1-alpha/|S| if Vi is a seed node

Else

Teleportation ← 0

//PageRank is calculated using the following formula

P ← Teleportation + Alpha \* (T \* (P/k));

Sort descending as per PageRank scores

Visualize

end

Implementation:

Personalized PageRank depends on Random walk and Teleportation values. Let Wij denote the weight of the edge between I and j. T is the adjacency matrix with value Wij if an edge exists between I and j, else 0. The PageRank vector P has seed entries filled with 1/NoofFrame + 1/|S+1| (Random walk and teleportation value) where S is the number of seed frames, also all the adjacent nodes of the seed nodes are filled with the same value, for others the values are only random walk 1/NoofFrame. The value S+1 taken because to prevent the underestimate of self-nodes [5]. The teleportation vector is filled with 1/|S| only for the seed nodes and zero for the other nodes. . Test for convergence is implemented by placing a threshold. The difference between the (I+1)th iterated PageRank scores and the Ith iterated Page Rank scores is taken as threshold which is power(1/10,I) , where J = iteration. E.g.: Iteration 1 has threshold 0.1, iteration 2 has 0.01… [3]. The PageRank score for the top most node alone is taken into consideration for the test of convergence [3]. PageRank takes 2 iteration in total with values initially being 1/NoofFrames when Vi ∉ S and 1/NoofFrames + 1/|S+1| when Vi € S.

Correctness:

The personalized PageRank will lift the seed nodes and the adjacent nodes of it to a considerable rise in the overall score.

Output:

The M-frames are visualized as videos and images

Tested for the following personalized frames

|  |  |  |  |
| --- | --- | --- | --- |
| Video i | Frame j | PageRank position | Personalized PageRank Position |
| 22 | 24 | 99 | 50 |
| 47 | 11 | 175 | 51 |
| 6 | 16 | 100 | 49 |

ASCOS with Seeding

Prerequisite

1. The similarity graph G(V,E) is given
2. The seed frames are given
3. No of seed frames are taken as 3

Algorithm

Get the m-most significant frames and the graph file from the user as input

SimMat ← Initial Similarity Matrix

C ← 0.85 - Relative importance between direct and indirect neighbors

N(i) ← Number of in neighbors of node i

k ← Number of seed nodes.

Function ASCOS()

Read the graph file given as input as an adjacency matrix SimMat of size N x N

While(SimMat(I,J) not converges or number of iterations =50)

For each I = 1 to N do

For each J = 1 to N do

If(I==J)

Update SimMat(I,J)=1;

Else

Find in-neighbors of I as N(I)

For each K=1 in N(I)

Sum=Sum+SimMat(K,J) + 1/(k+1);

End for

Update SimMat(I,J) = C / N(i) \* Sum;

End if

End For

End For

End While

Implementation:

The ASCOS function computes the similarity between 2 nodes at a time by computing the Similarity between nodes I and J by using the node I’s in neighbors to node J and also using the seed node data. First we read the graph file and created an adjacency matrix of size N\*N using the data in the graph and create initial similarity matrix using the similarity percentages between each node that can also be found in the graph. Now using the formulae given in Chen’s paper for describing ascos and giving weightage to seed nodes we update the similarity between I and J as 1 if I and J represent the same frame or by using the formula c/N(i) ∑ ∀k∈N(i) skj +1/(k+1) where N(i) is the in neighbors of the node I and k denotes the number of seed nodes. We iterate the same until the similarity matrix converges.

Correctness:

The ASCOS with seed frames will lift the seed nodes and the adjacent nodes of it to a considerable rise in the overall score.

Output:

The M-frames are visualized as videos and images

Tested for the following personalized frames

|  |  |  |  |
| --- | --- | --- | --- |
| Video i | Frame j | PageRank position | Personalized PageRank Position |
| 22 | 24 | 85 | 19 |
| 47 | 11 | 126 | 24 |
| 6 | 16 | 220 | 57 |

Task 5:

Prerequisite:

The PCA transformed Sift dataset is required for this task.

Algorithm:

D ← dimension

K ←2^k buckets

L ← Number of layers

Generate Random Hyperplane HPi for Dimension D where i = 1..k

DotProduct ←Dot (HPi,SIFT)

If DotProduct > 0 BinaryVal ← 1

else BinaryVal ← 0

BucketNum ← concat BinaryVal

Implementation:

The hash function can also be viewed as partitioning the space into two half-spaces by a randomly chosen hyperplane [12]. The Hyperplanes are generated randomly and the value for every dimension is chosen between -1 to +1 as it provides the balanced partition between the hyperplane. The binary data are concatenated together to form the bucket hash. The results are written to a file

Correctness:

It can be verified by querying an object (Task 6)

Output sample:

The first 10 samples for 10 Dimension SIFT vectors with K value 5, Layer 3

{1,5,<1;1;3;41;317;>} {1,8,<1;1;3;41;317;>} {1,21,<1;1;3;22;315;>} {1,16,<1;1;3;36;315;>} {1,21,<1;1;3;43;322;>} {1,16,<1;1;3;43;336;>}{1,20,<1;1;1;23;23;>} {1,21,<1;1;1;23;23;>}{1,20,<1;1;1;23;35;>} {1,21,<1;1;1;23;35;>}

Task 5 - Another Approach:

Prerequisite:

1. The output file for task 1 should be provided
2. The user will provide value for no of dimensions D
3. The user will provide value for no of Layers L
4. The user will provide value for bucket size K

Algorithm:

D <- No of dimensions

L <- No of layers

K <- Bucket size

no\_Of\_Buckets = 2^K

for(i = 1 to L)

salt\_Vector = rand(L\_index,D)

end for

ip\_Data <- Read data from reduced dimension from task  1

LSH[][] = Extract ip\_Data in matrix format

for(each row in LSH[][])

dp\_Data = dotProduct(row\_data, salt\_Vector[L\_index])

bucket\_Num\_Interm = dp\_Data + L\_index\*10

bucket\_Num = floor(mod(bucket\_Num\_Interm,no\_Of\_Buckets)

end for

Implementaion

Given a reduced SIFT dimension file, Layers, and bucket size. We generate salt vectors for each layer.

The salt vectors are generated basd on a random function of Layer index and dimensions. Generate a data matrix from the input file. For each row in the data matrix compute the follwing function.

Hash\_function = └ (v.r + b)/w ┘

Where

“v” is the salt vector.

“r” is the row data.

“b” is a constant (Layer index \* 10)

“w” is the number of buckets

Based on the results from Task 6, Hyperplane projection (Approach 1) gives more accurate results than (Approach 2)

Task 6:

Prerequisite:

The PCA transformed data and LSH datasets are given

Algorithm:

LSH\_Data ← {layer num, bucket num, <i; j; l; x; y>}

PCA\_Data ← {i, j, l, x, y, [dimi, . . . , dimd]}

Given X1,Y1,X2,Y2

D1 ← Query Descriptors from PCA\_DATA

BucketList ᵢ ← Combination of all buckets containing the query descriptor in Layerᵢ

M← Common ViFj from BucketList i to n

D2ᵢ ← Descriptors of Unique ViFj

SIM←Similarity(D1,D2i..n)

UNIQ\_SIFT←the number of unique SIFT vectors considered,

OVERALL\_SIFT←the overall number of SIFT vectors considered

BYTES←the number of bytes of data from the index accessed to process the query.

Implementation:

Given the LSH data and PCA transformed data, x1,y1,x2,y2. The algorithm finds the query descriptors from the PCA transformed data which forms the descriptor set D1. Next, the algorithm finds the bucketLists where the query descriptor falls using the LSH dataset. All the neighbor points from the BucketList is combined into a matrix. This is done for every layer Li and finally the BucketLists from all the Layer are combined into matrix M. From M, the algorithm finds the unique videos and frames present in all the BucketList. Then the descriptors are taken from the Task1 data. This will be D2i … D2n. Similarity is computed for (D1, D2i..D2n) using the Task 2 algorithm and find the frame which contains the query object.

Correctness:

Visualized the n-frames and found that the algorithm finds exact match in most cases (Because of object repition in videoset) and approximate match for videos containing Human objects.

Output sample:

For M = 5:

Query Frame: Video: 19 Frame: 8

Match1: Video: 24, Frame :15

Match 2: Video: 24, Frame:18

Match 3: Video: 24, Frame: 21

Match 4: Video:24, Frame: 23

Match 5: Video:24, Frame: 24

User Interface specification:

Display description:

The user can visualize the k most similar sequence of videos in the given Database.

System requirements/installation:

Matlab 2016

Visual Studio 2015 or any equivalent IDE

Related Works:

There are many approaches to measure the video similarity and video classification. One popular video representation technique is to represent each video sequence with frames [2]. Recently, the technique for measuring the video similarity based on the percentage of visually similar frames between the two sequences has been proposed in [13],[14],[15]. One commonly used technique for video similarity measurement is the Naıve Video Similarity (NVS) [12],[15]. This technique first finds the total number of frames from each video sequence which has at least one similar frame with the other sequence. Then, the ratio of these numbers will be computed to the total numbers of frames. After that, the threshold is used to compare the difference between the frames. Expectation-based measuring video similarity [5] can measure similarity of video efficiently by using expected value to average distance of video frames instead of the threshold. Each video sequence was represented with frame and each frame was represented with the color histogram to help enhance feature reduction. After that, categorization was performed using the nearest neighbor classifier with the 𝐿1 metric to measure distance by comparing each sampling frame of the training videos with all sampling frames of the test videos.

Principal component analysis (PCA) is a statistical procedure that transforms a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. The primary goal of PCA is to extract the dominant patterns exhibited in the data, making it an efficient technique for linear dimension reduction and a crucial preprocessing step in high dimensional statistical analyses and machine learning based classification or clustering. One of the major feature used to compare two images are SIFT key points.

Clustering (a.k.a. unsupervised classification) is the task of categorizing data into groups without using any labels (training data), and it is omnipresent in various application domains such as data mining, pattern recognition, and signal processing [7]–[9]. Among numerous clustering algorithms, K-means is the most prominent one [7]–[9]. It thrives on “tight” groups of data points that can be linearly separated [via (hyper)planes], but it further has important extensions to kernel K-means, which can be rendered equivalent to spectral clustering – the popular graph-partitioning tool that can cope even with nonlinearly separable data points [10].

Chen et al describes ASCOS Asymmetric Structure COntext Similarity measure which finds the similarity between the pairs of nodes in the network. The author suggests that ASCOS overcomes the problem in SimRank, which considers only the similarity between nodes where they can be reached in even number of steps., (i.e) the similarity between neighbors will be zero[11]. Another problem of SimRank that is overcome by using ASCOS is that SimRank considers similarity between 2 nodes to be symmetric (i.e) Sij=Sji which some studies have shown may not be appropriate for a few objects such as Human Faces etc.

**Screenshots:**

Task 3: Example

60 Dimension, K = 10, Most Significant frame

PageRank ASCOS

|  |  |
| --- | --- |
| C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Image_PR1.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Image_ASCOS1.jpg |

Task 4:

Personalized Page Rank (Video47, Frame11)

Moved from 175th position to 51th position

Personalized ASCOS (Video47, Frame11)

Moved from 126th position to 24th position



Task 6:

|  |  |  |
| --- | --- | --- |
| Query Object Video 19, Frame 8 | First : Video 24 Frame 15 | Second:Video 24 Frame 18 |
| C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_query19_8.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_24_21.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_24_18.jpg |
| C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_24_23.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_24_24.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_24_15.jpg |
| Third: Video 24 Frame 21 | Video 24 Frame 23 | Video 24 Frame 24 |

|  |  |  |
| --- | --- | --- |
| Query Video 52 Frame 10 | First: Video 51 Frame 9 | Second: Video 51 Frame 10 |
| C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_query52_10.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_51_9.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_51_10.jpg |
| C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_53_5.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_59_5.jpg | C:\Users\naren\AppData\Local\Microsoft\Windows\INetCacheContent.Word\task6_result_59_4.jpg |
| Third: Video 53 Frame 5 | Fourth: Video 59 Frame 4 | Fifth: Video: 59 Frame: 5 |

Conclusion:

We have found the K similar frames for the given set of video files and implemented PageRank and ASCOS algorithm both for random probability and personalized probablility to find the most significant frames in the given dataset. In order to make the searching more efficient we have implemented the locality based Hashing using random hyperplane to query the object in the video set to find the accurate and most similar object with very efficient complexity O(logn) compared to Linear search.

Bibliography:

[1] - <https://goo.gl/VdRGxv>

[2] - <https://goo.gl/isAVKG>

[3] - "Chun Liu" and "Yuqiang Li", "A Parallel PageRank Algorithm with Power Iteration Acceleration", International Journal of Grid Distribution Computing, Vol.8, No.2 (2015), pp.273-284

[4] - 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

[5] - 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

[6] - <https://goo.gl/vykxHz>

[7] - Distinctive Image Features from Scale-Invariant Keypoints” Lowe, D.G. International Journal of Computer Vision (2004) 60: 91. doi:10.1023/B:VISI.0000029664.99615.94

[11] Chen, Hung-Hsuan, and C. Lee Giles. "ASCOS++: An Asymmetric Similarity Measure for Weighted Networks to Address the Problem of SimRank." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.2 (2015): 15.

[12] - 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