Video Sub-sequence Search Using Diverse Similarity Measures In The Original Space And The Reduced Dimensionality

Anurag Arora, Bineeta Gupta, Kyungyong Han, Neha Prasad, Nishant Jagadeesan, Ujjwal Dasu
Arizona State University
Ira A. Fulton School of Engineering
Tempe, AZ 85281,United State
aarora40, bkgupta, hkyungyo, nhprasad, njagade1, udasu@asu.edu

***Abstract*—Traditional data is more about text-based data. Therefore, Video classification techniques did not needed to analyze the data. Nowadays, however, We can search for a lot of video data on the Internet. Therefore, it is very important to classify or cluster the videos and images to utilize, retrieval and index them. In this paper, therefore, we would like to perform video sub-sequence by using diverse similarity measures using color histogram, SIFT, and motion vector as the feature in both the original space and reduced dimensionality space. The project scope is to calculate similarity between given two videos by using various similarity measures and to find the most similar video sub-sequence when the range of frames of a video is given in the original space and the reduced dimensional space.**

Keywords*—*Color Histogram, SIFT, Motion Vector, Chi Square, Intersection Similarity, Manhattan Distance, Hausdorff Distance, Euclidean Distance, Histogram of Motion Vector(HOMV), Squared Euclidean Distance, nearest neighbor distance ration, Video Sub-Sequence Search, Dynamic Programming, Principal Component Analysis, K-means, Orthogonal-Triangular Decomposition(QR decomposition), Dimensionality Reduction, Bag of Features

I. Introduction

Similarity or distance (dissimilarity) measures are used to match feature descriptors of two objects. Similarity measure is a function that returns larger value when the two objects are similar. On the other hand, distance measures give a smaller value for more similar objects. In this project, we manipulate Chi square and Intersection similarity for color histogram similarity, Hausdorff and Squared Euclidean Distance using nearest neighbor distance ration(NNDR) strategy for SIFT , Intersection similarity and Manhattan Distance using Histogram of Motion Vector(HOMV) for motion vector with original dimensionality, and Euclidean and Manhattan Distance for motion vector with reduced dimensionality, in this project. For overall similarity, we use linear combination. In addition, to reduece computation time, we use sliding windows algorithm with dynamic programming.

Video Sub-sequence search takes a lot of time to execute in huge database. In order to perform Video Sub-Sequence Search faster, Dynamic Programming is used in this project.

Many data analysis algorithm becomes significantly complicated if there are high dimensions and this is called curse of dimensionality. Not only for video similarity measure, for any other algorithm that process data to extract information will work more efficiently if the number of dimensions is reduced. Not all the dimensions of a given set of data will be useful. So, eliminating those dimensions helps to process the data even more efficiently. Less the number of feature, much easy it is to understand the data. Here, in this project we are going to do dimensionality reduction using Principal Component Analysis and K- means cluster. Principal Component Analysis(PCA) transform is a linear transform, which optimally decorrelates the input data [1, p.156-159]. K-means clustering is one of the commonly used clustering algorithm which finds k points as centroid [2]. Orthogonal-Triangular Decomposition(QR decomposition) is transformations to be used to reduce dimensionality like PCA [3].

The project goal is to calculate similarity between given two videos by using various similarity measures and to find the most similar video sub-sequence when the range of frames of a video is given in the original space and the reduced dimensional space.

For the project, we assume that

1) The format of input files is 'csv' file.
2) The name of input files is 'output_sift.csv', 'output_mvect.csv', and 'output_chst.csv' for task 1, 2 and 3.
3) In motion vector, minimum dimensions required are 2.
4) For SIFT vector, we do not consider key point locations such as x, y, scale, and orientation as a vector because the distance measures we are implementing ie, David lowe's Keypoint matching [4] and Hausdorff distance[5] only use SIFT descriptors.

II. Description of the proposed solution/implementation

A. Dynamic Programming for video similarity

Our approach to calculate video similarity involves using an algorithm [6], which applies a combination of spatial and temporal characteristic of the video to determine similarity, by conducting pairwise comparisons of individual frames from the two videos. The algorithm then selects a subset of frame-pairs, one each from the two videos being compared, using dynamic programming, to find such an alignment of the frame pairs which minimizes the total distance score (that

is, it maximizes the similarity score) of all the frame pairs selected. So, basically the algorithm compares and calculates a distance measure for pairs of frames and then uses this to find an aligned subset of frame-pairs that cumulatively have the minimum distance (or maximum similarity) between the two videos. See Figure 1.

```
for i=2:nFrames1+1
    for j=2:nFrames2+1
        matrixD(i,j)= min([matrixD(i-1,j-1)+matrix(i-1,j-1)*w2,matrixD(i-1,j)+
            matrix(i-1,j-1)*w1,matrixD(i,j-1)+
            matrix(i-1,j-1)*w1]);
    end
end
dist=matrixD(nFrames1+1,nFrames2+1);
```

Fig. 1: Code of Calculating Minimum Distance between two videos using Dynamic Programming

The approach uses the following ideas to include both spatial and temporal characteristics of the two videos during the comparison:

1. Temporal consideration:

1) The DP algorithm imposes the constraint that if frame f_1 of first video has matched with frame f_2 of second video, any subsequent frames of first video can only match with frames that appear after f_2 in second video (and vice versa).
2) The algorithm minimizes information loss / maximizes the number of frames selected. In order to use as much of the video information as possible, the dynamic programming algorithm penalizes the similarity score between frame pairs that are temporally father apart, that is, the distance scores between frame pairs that have skipped more frames from the last pair selected to be part of the alignment are assigned bigger weights(higher penalty).

2. Spatial consideration:

1) We have spatial consideration during comparison of frames for histogram and motion vector similarity comparison. For each frame, each cell's histogram/motion-vector is only compared with the corresponding cells histogram/motion-vectors in the other frame. This does not apply to SIFT similarity since that is being considered as a bag of features.

The DP algorithm, tries to minimize information loss, but due to the nature of how it decides the similarity, it does "skip" some of the frames while calculating the total distance score.

B. Color Histogram Similarity

For color histogram, based on the color intensity distribution of two frames, similarity between them is computed. If the two frames share almost similar color distribution then that means two frames are similar. For this purpose, we consider using Intersection Similarity and Chi Squared distance measure because both are good at finding the difference of color distribution between two images / frames as both, calculate the ratio of similarity or dissimilarity between images. Both the distances are good at finding overlaps and are particularly well suited for histograms since each dimension represents a count of pixels in that dimension.

The first measure used is chi-square distance. Mathematically, it is

$$\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)} \tag{1}$$

where x_i and y_i are the corresponding bin.

We compute frame level distance for a pair of frames of the two videos being compared by calculating and summing over the chi-square distances of each cell-pair. We finally normalize by dividing with the maximum cell distance found for this cell and the total number of cells in the frame. See figure 2, 3.

```
double parser::frame_dist_ch_sq(t_frame& f1, t_frame& f2)
{
    int total_cells=r*r;
    double dist=0,cell_dist,max_dist=0;
    //get count of pixels in first cell per frame
    int f1_pix_cnt=f1.get_pixel_per_cell();
    //get count of pixels in second cell per frame
    int f2_pix_cnt=f2.get_pixel_per_cell();
    // calculate the cell chi_square distance for each pair of cells in this frame
    for(int i=0;i<total_cells;i++)
    {
        cell_dist=cell_dist_chi_sq(f1.cells[i], f2.cells[i],f1_pix_cnt,f2_pix_cnt);
        if(cell_dist>max_dist)
            max_dist=cell_dist;
        dist+=cell_dist;
    }
    // Normalizing with max_distance between any 2 cells in this frame for a better measure
    if(max_dist>0)
        return (dist/(max_dist*total_cells));
    else
        return (dist/(total_cells));
    //return (dist/(total_cells));
```

Fig. 2: Code of normalized frame level distance of Chi Square

```
double parser::cell_dist_chi_sq(t_cell c1, t_cell c2, int pix_cnt1, int pix_cnt2)
{
    //get the histograms from the two cells to compare
    histogram h1=c1.hist;
    histogram h2=c2.hist;
    double dist=0,x1=0,x2=0;
    //calculate the normalised value of each bin in both the histograms
    //Use chisquare measure to calculate total distance
    for(int i=0;i<n;i++)
    {
        x1=(double(h1[i])/pix_cnt1);
        x2=(double(h2[i])/pix_cnt2);

        if(x1!=0 || x2!=0)
            dist+=(pow(x1-x2,2)/(x1+x2));
    }
    return dist;//no normalization here, normalise at the frame level
}
```

Fig. 3: Code of the chi square distances of each cell-pair

The second measure used is intersection similarity. Mathematically, it is

$$\sum_{i=1}^{n} \frac{min(x_i, y_i)}{max(x_i, y_i)} \tag{2}$$

We compute frame level distance for a pair of frames of the two videos being compared by calculating and summing over the cell-intersection similarity of each cell-pair. We finally normalize by diving with the maximum cell distance found for this cell and the total number of cells in the frame. Since we are using distance measures in the dynamic programming part, we should convert this to a distance measure for code reuse. To convert this to a distance measure we subtract this similarity measure from 1. See figure 4, 5.

```
double parser::frame_dist_intersect(t_frame& f1, t_frame& f2)
{
    int total_cells=r*r;
    double dist=0,cell_dist,max_dist=0;
    //get count of pixels in first cell per frame
    int f1_pix_cnt=f1.get_pixel_per_cell();
    //get count of pixels in second cell per frame
    int f2_pix_cnt=f2.get_pixel_per_cell();
    // calculate the cell intersection similarity measure for each pair of cells in this frame
    for(int i=0;i<total_cells;i++)
    {
        cell_dist=cell_dist_intersect(f1.cells[i], f2.cells[i],f1_pix_cnt,f2_pix_cnt);
        if(cell_dist>max_dist)
            max_dist=cell_dist;
        dist+=cell_dist;
    }
    // convert similarity to distance measure by normalizing with max_distance between any 2 cells in this frame
    if(max_dist>0)
        return 1-(dist/(max_dist*total_cells));
    else
        return 1-(dist/(total_cells));
    //return 1-(dist/(max_dist*total_cells));
```

Fig. 4: Code of normalized frame level distance of Intersection Similarity

```
double parser::cell_dist_intersect(t_cell c1, t_cell c2, int pix_cnt1, int pix_cnt2)
{
    // get histograms for both cells to be compared
    histogram h1=c1.hist;
    histogram h2=c2.hist;
    double dist=0,max=0,min=0;
    // calculate the sum of all max and sum of all min values respectively
    for(int i=0;i<n;i++)
    {
        max+=(double(h1[i])/pix_cnt1) > (double(h2[i])/pix_cnt2)?(double(h1[i])/pix_cnt1):(double(h2[i])/pix_cnt2);
        min+=(double(h1[i])/pix_cnt1) > (double(h2[i])/pix_cnt2)?(double(h2[i])/pix_cnt2):(double(h1[i])/pix_cnt1);
    }
    //calculate intersection similarity
    dist=min/max;
    return dist;
}
```

Fig. 5: Code of the chi square distances of each cell-pair

C. SIFT similarity

The first method is Hausdorff distance.

The directed Hausdorff distance between two sets of points in a space is defined as,

$$h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{3}$$

where A and B are the two sets of points, and $\|\cdot\|$ is a norm for points in the sets. In general, under this formulation $h(A,B)h(B,A)$. To address this, the bidirectional Hausdorff distance[7] is defined as,

$$\hat{h}(A,B) = max(h(A,B), h(B,A)) \tag{4}$$

The hausdorff distance measures the degree of mismatch between two point sets by calculating the distance of a point in A that is farthest to a distance in point in B [7]. See Figure 6.

The Hausdorff distance between two frames $frame_1$ and $frame_2$ is calculated as follows: Each frame is considered as a bag of feature descriptors. For each descriptor the minimum distance to the other descriptor set is calculated. The maximum of these minimum distances gives the hausdorff distance between the two point sets i.e. the frames.

The second method is Squared Eucliden distance using nearest neighbor distance ration (NNDR) strategy which is a matching strategy proposed by David Lowe [8]. Two feature vectors are considered a match if, f_1 the $1nn$ (first nearest neighbor) and f_2 $2nn$ (the second nearest neighbor) satisfy the condition:

$$f_1 < f_2 < t \tag{5}$$

```
for(i=0;i<col_D1;++i,D1+=dim){
    double minDist=MAX_DOUBLE;

    for(j=0;j<col_D2;++j,D2+=dim){
        double dist=0;
        int k;
        for(k=0;k<dim;k++){
            //L2 norm
            dist+=(D1[k]-D2[k])*(D1[k]-D2[k]);
        }
        //dist=sqrt(dist);
        //mexPrintf("Distance = %g\n",dist);
        //Update the nearest neighbour
        if(dist<minDist) minDist=dist;
    }

    //Update the maximum distance between the distribution
    if(maxDist<minDist) maxDist=minDist;
    //Reset the D2 pointer for next iteration
    D2-=col_D2*dim;
}
```

Fig. 6: Code of hausdorff

Where t is a threshold. This is done to avoid ambiguous matches. But as a result, repeating matches in the images being compared are not matched [5]. Euclidean distance can be used when all components of the descriptor are expressed in the same units. This distance can be used for SIFT, whose descriptor is a histogram and are $L2$ normalized [5]. We are using an extended approach of Lowes matching strategy to find the distance between two video frames [9],

$$d(A,B) = \frac{(n_A + n_B)}{2} - card(M_{ab}) \tag{6}$$

Where, M_{ab} is the set of feature vectors matched using NNDR strategy. This distance has been proved to be a metric. Only this SIFT descriptors are used for matching as proposed by Lowe [4] as they are location and scale invariant. See Figure 7.

Consider two video frames $frame_1$ and $frame_2$ each with n_1 and n_2 sift descriptors respectively. For each descriptor in $frame_1$ we calculate the squared Euclidean distance to every other descriptor in $frame_2$ and the top two matching descriptors are compared with each other. If their ratio is greater than 1.5 then it is considered a match. This thresholding is presented by Lowe as 1.5 which was calculated empirically to be a reasonably good matching results [4]. All points not satisfying this ratio are rejected.

D. Motion similarity

The first method is HOMV (Histogram of Motion Vectors) using Manhattan Distance or Intersection Similarity for the original dimensionality.

First, every cell of a frame has multiple motion vectors. Therefore, magnitude and angle of each vector must be calculated first with respect to basis vector. Also, normalisation is performed for getting an appropriate range. See Figure 8.

HOMV is histogram of motion vectors, where it contains both the phase angle and the magnitude information in a vector.

```
for(i=0 ; i<col_D1 ; ++i, D1+=dim) {
    double bestMatch=MAX_DOUBLE;
    double secondBest=MAX_DOUBLE;
    for(j=0; j<col_D2;++j, D2+=dim){
        loopCounter++;
        int k;
        double dist=0;
        for(k=0;k<dim;++k){
            double temp=D1[k]-D2[k];
            dist+=temp*temp;
        }

        //update best and second best
        if(dist<bestMatch){
            secondBest=bestMatch;
            bestMatch=dist;
        }
        else if(dist<secondBest){
            secondBest=dist;
        }
    }
    //reset D2 pointer;
    D2-=col_D2*dim;

    //Check with threshold
    if(bestMatch*thres<=secondBest){
        matchCount++;
    }
}
```

Fig. 7: Code of NNDR using square Euclidean Distance

```
sx= new_matrix_video1(x,5);
sy=new_matrix_video1(x,6);
dx= new_matrix_video1(x,7);
dy= new_matrix_video1(x,8);
magnitude=sqrt(abs(dx-sx) * abs(dx-sx)+abs(dy-sy) * abs(dy-sy));
theta = (atan((dy-sy)/(dx-sx))) * 57.33;
if (theta<0)
    theta=theta+180;
end
   if ( ( (dy-sy) ==0 && (dx-sx)==0) || (dx-sx)==0)
        theta = 0;
   end
```

Fig. 8: Code of how to calculate magnitude and angle in a cell

We are first taking bins as input from user. $HOMV(m,n)$ is a weighted histogram of $\theta(n,i)$.

$$HOMV(m,n) = \frac{L}{N} \sum_{\substack{i \\ \theta_{m-1} \leq \theta(n,i) < \theta_m}} r(n,i)\theta(n,i) \qquad (7)$$

where n is the frame index, m is the histogram bin index, $m \in (0, M)$, L is the number of angle regions , N is the total number of MVs. The phase angle region $(-\pi, \pi)$ is divided into M equal sub-regions with boundaries θ_m, with $\theta_0 = \pi$, and $\theta_m = \pi$. Weight of the $\theta(n,i)$ is the magnitude of the corresponding motion vector, $r(n,i)$.

Then, HOMV model is applied by following above formula and updated the values for each of the bin column. See Figure 9

```
initial_theta_m=abs(ceil(initial_theta/90));
if (initial_theta_m==0)
    initial_theta_m=1;
end
multiply_mag_theta=initial_theta*initial_mag*4/count_of_cell;
new_matrix_video1_HOMV(index_new_matrix_video1_HOMV,1)=cell_value;
if (initial_theta_m==1)
 sum_1=sum_1+multiply_mag_theta;
 new_matrix_video1_HOMV(index_new_matrix_video1_HOMV,initial_theta_m+1)=sum_1;
end
if (initial_theta_m==2)
 sum_2=sum_2+multiply_mag_theta;
 new_matrix_video1_HOMV(index_new_matrix_video1_HOMV,initial_theta_m+1)=sum_2;
end
if (initial_theta_m==3)
 sum_3=sum_3+multiply_mag_theta;
 new_matrix_video1_HOMV(index_new_matrix_video1_HOMV,initial_theta_m+1)=sum_3;
end
if (initial_theta_m==4)
 sum_4=sum_4+multiply_mag_theta;
 new_matrix_video1_HOMV(index_new_matrix_video1_HOMV,initial_theta_m+1)=sum_4;
end
```

Fig. 9: Code of HOMV model

Then, we have corresponding matrices for the given videos after applying HOMV model.

Finally, for finding the similarity, we use Manhattan and Intersection measures. See Figure 10, 11.

Manhattan Distance is below :

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i| \qquad (8)$$

Intersection Distance is below :

$$\sum_{i=1}^{n} \frac{min(x_i, y_i)}{max(x_i, y_i)} \qquad (9)$$

```
%final matrix construction
for i_new=0:nframes1
    for j_new=0:nframes2
        dot=0;
        for c_new=1:r
            mag1_1=new_matrix_video1_HOMV(r*(i_new)+c_new,2);
            mag1_2=new_matrix_video2_HOMV(r*(j_new)+c_new,2);
            mag2_1=new_matrix_video1_HOMV(r*(i_new)+c_new,3);
            mag2_2=new_matrix_video2_HOMV(r*(j_new)+c_new,3);
            mag3_1=new_matrix_video1_HOMV(r*(i_new)+c_new,4);
            mag3_2=new_matrix_video2_HOMV(r*(j_new)+c_new,4);
            mag4_1=new_matrix_video1_HOMV(r*(i_new)+c_new,5);
            mag4_2=new_matrix_video2_HOMV(r*(j_new)+c_new,5);
            dot=abs((mag1_1-mag1_2)+(mag2_1-mag2_2)+(mag3_1-mag3_2)+(mag4_1-mag4_2));
        end
            final_matrix(i_new+1,j_new+1)=dot/r;
    end
end
```

Fig. 10: Code of Manhattan Distance using HOMV

For reduced dimensionality, we applied another method. Since the HOMV Model specifically considers the dx, dy, sx, and sy as paarameters to implement the model, if we reduce dimensionality, these parameters might get dropped and the method will fail. For avoiding incorrect result, we will apply Euclidean and Manhattan distance on reduced dimensional space. See Figure 12, and 13.

Euclidean Distance is below :

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \qquad (10)$$

```
for i_new=0:nframes1
    for j_new=0:nframes2
        dot=0;
        maxx=0;minn=0;
        for c_new=1:r
            mag1_1=new_matrix_video1_HOMV(r*(i_new)+c_new,2);
            mag1_2=new_matrix_video2_HOMV(r*(j_new)+c_new,2);
            mag2_1=new_matrix_video1_HOMV(r*(i_new)+c_new,3);
            mag2_2=new_matrix_video2_HOMV(r*(j_new)+c_new,3);
            mag3_1=new_matrix_video1_HOMV(r*(i_new)+c_new,4);
            mag3_2=new_matrix_video2_HOMV(r*(j_new)+c_new,4);
            mag4_1=new_matrix_video1_HOMV(r*(i_new)+c_new,5);
            mag4_2=new_matrix_video2_HOMV(r*(j_new)+c_new,5);
            maxx=maxx+max(mag1_1,mag1_2);
            minn=minn+min(mag1_2,mag1_1);
            maxx=maxx+max(mag2_1,mag2_2);
            minn=minn+min(mag2_2,mag2_1);
            maxx=maxx+max(mag3_1,mag3_2);
            minn=minn+min(mag3_2,mag3_1);
            maxx=maxx+max(mag4_1,mag4_2);
            minn=minn+min(mag4_2,mag4_1);
        end

        if (maxx==0)
```

Fig. 11: Code of Intersection Distance using HOMV

```
%final matrix construction using Euclidean distance
for i_new=0:(nframes1-1)
    for j_new=0:(nframes2-1)
        dot=0;
        for c_new=1:r
          dot=dot+pdist2(new_vector_1_norm(r*(i_new)+c_new,:),new_vector_2_norm(r*(j_new)+c_new,:));

        end
          final_matrix(i_new+1,j_new+1)=dot/r;
    end
end
```

Fig. 12: Code of Euclidean Distance

```
nal matrix construction
 i_new=0:(nframes1-1)
  for j_new=0:(nframes2-1)
      dot=0;
      for c_new=1:r
        dot=dot+pdist2(new_vector_1_norm(r*(i_new)+c_new,:),new_vector_2_norm(r*(j_new)+c_new,:),'manhattan'

      end
        final_matrix(i_new+1,j_new+1)=1-dot/r;
  end
```

Fig. 13: Code of Manhattan Distance

E. Overall Similarity

In overall Similarity, we use linear combination [10] of motion similarity, SIFT similarity, and histogram similairty as below

$$dO_{ij} = w_C dC_{ij} + w_S dS_{ij} + w_M dM_{ij} \tag{11}$$

where w_c, w_s, and w_m are weight, but we give 1 as all weights. dC_{ij}, dS_{ij}, and dM_{ij} denote the distances between two video shots i and j of color histogram, SIFT, and motion vector. These distances are normalized by the mean distance values for each feature. The best match for the query is the one with the smallest overall distance.

F. Video sub-sequence search for Task 2 and 4

The subsequence search is an application of the dynamic programming approach discussed earlier, where, we run the DP algorithm to compare the query video with varied sized subsequences of each video in the database using a sliding window approach. We calculate the range of the sliding window size to be alpha and beta, where:

$$\alpha = 0.8 \times size(QueryVideo) \tag{12}$$

$$\beta = 1.2 \times size(QueryVideo) \tag{13}$$

The subsequence size from the video in the database, varies between alpha and beta. So, a DP based comparison is made between the query video and each such subsequence. See Figure 14.

```
for strt=1:steps:nFrames2-steps*2

    sSize=min(k,nFrames2-strt);
    if(strt+k>=nFrames2)
        break;
    end
    corresDistance=zeros(sSize,p);
    for i=a:b
        sSize2=min(nFrames2,strt+k-1);
        for j=strt:sSize2
            corresDistance(j-strt+1,i-a+1)=sift_hausroff(
        end
    end
```

Fig. 14: Code of Dynamic Programming

G. Video dimensionality reduction using PCA

Video dimensionality reduction using PCA produces two outputs such as an output database and the reduced dimensions in terms of the original dimensionality.

In order to perform this, PCA technique is introduced. First, a covariance matrix is calculated as following formula.

$$S = \frac{1}{(n-1)} \times X^T X \tag{14}$$

where S is m by m symmetric, square matrix with real values, X is n by m matrix, X^T is transposed matrix of X.

X is calculated as following below formula.

$$X = D - \bar{D} \tag{15}$$

where D is n by m matrix of the original database, and $\bar{D}$ is n by m matrix. Components of each column of $\bar{D}$ is mean of each column of the original database.

Second, PCA decomposition is performed.

$$S = PCP^{-1} \tag{16}$$

where S is an m by m symmetric, square matrix with real values, P is m by k (where $k \leq m$) column linearly independent eigenvector matrix of S, and C is real and diagonal eigenvalue matrix of S ordered by descending from the left and uppermost. See Figure 15.

Third, eigenvectors having small eigenvalues is removed which means we keep the number of eigenvectors in the same number as the reduced dimensionality we want. See Figure 16.

$$S' = P'C'P'^{-1} \tag{17}$$

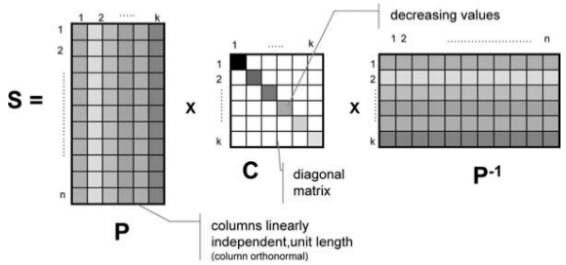

Fig. 15: Eigen decomposition [1, p.157]

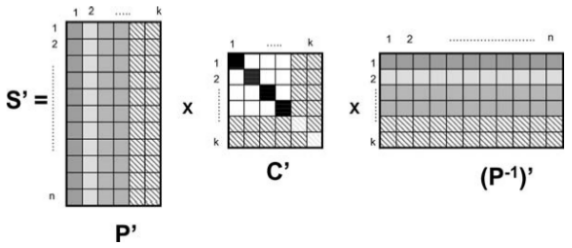

Fig. 16: How to remove eigenvectors with small eigenvalue [1, p.159]

Here, P' will be used and stored as

$$\langle original\ index, score \rangle$$

for the reduced dimensions in terms of the original dimensionality.

Finally, we calculate an output database by following below

$$D' = DP' \tag{18}$$

where D is n by m matrix of the original database, P' is m by d(d is the reduced dimensionality we want) matrix from above formula $S' = P'C'P'^{-1}$, and D' is n by d matrix of the reduced dimensional output databse. Then, D' is stored with a corresponding label as

$$\langle video, frame, cell, vector \rangle$$

In MATLAB, the output database is directly obtained by using $score$ in pca function. Here, $score$ is different from $score$ above mentioned as the reduced dimensions in terms of the original dimensionality. The $score$ as the reduced dimensions in terms of the original dimensionality is obtained by using V in $eigs$ function in MATLAB. See Figure 17.

H. Video dimensionality reduction using K-means

Video dimensionality reduction using K-means produces two outputs such as an output database and the reduced dimensions in terms of the original dimensionality like PCA.

In order to perform this, K-means clustering technique and orthogonal-triangular decomposition are introduced. First, the d(d is the reduced dimensionality we want) number of

```
% do pca
[coeff, score, latent] = pca(hist.data);

% create score.
X = bsxfun(@minus, hist.data, mean(hist.data,1));
covariancex = (X'*X)./(size(X,1)-1);
[V, D] = eigs(covariancex, dimensionality);
[nrows,ncols] = size(V);
```

Fig. 17: Code of PCA

centroid points are calculated. 'kmeans' functionn in MATLAB is utilized for this step like below

$$[idx, C] = kmeans(D, d) \tag{19}$$

where D is n by m matrix of the original database, d is the reduced dimensionality we want, C is d by m matrix of centroid locations which means each row is a centroid location in the original dimensions, and idx is n by 1 vector containing cluster indices of each observation which is not used in this project.

Second, orthogonal-triangular decomposition is used[11].

$$C^T = Q \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{20}$$

where C^T is an m by d transposed matrix of C, Q is m by m an orthogonal matrix with orthonormal columns with real values, and R is d by d upper triangular matrix. Then, the first d columns of Q are taken. Then we can get

$$C^T = Q_r R \tag{21}$$

Here, Q_r will be used and stored as

$$\langle original\ index, score \rangle$$

for the reduced dimensions in terms of the original dimensionality.

Finally, we calculate an output database by following below

$$D' = DQ_r \tag{22}$$

where D is n by m matrix of the original database, and D' is n by d matrix of the reduced dimensional output databse. Then, D' is stored with a corresponding label as

$$\langle video, frame, cell, vector \rangle$$

In MATLAB code, new_Q is $score$ as the reduced dimensions in terms of the original dimensionality, and k_means_data is an output database in Figure 18.

```
% do k-means
[idx,C] = kmeans(hist.data, dimensionality, 'MaxIter', 1000);

% reduce dimensionality by using orthogonal-triangular decomposition.
[Q,R] = qr(C.');
new_Q = Q(:,1:dimensionality);
k_means_data = hist.data*new_Q;
```

Fig. 18: Code of K-means

III. Interface specifications

A. Task 1

For Histogram Similarity, the user input below in MATLAB console

Task1_histogram_similarity(*db*, $file1$, $file2$, $Measure$)

db is the path to the .csv database of all videos histogram values. The input db files must be the csv format which follows

$$\langle video, frame, cell, vector \rangle$$

$file1$ is the first video file name whose histogram values need to be extracted from the db. $file2$ is the second video file name whose histogram values need to be extracted from the db. $Measure$ is either $Intersection$ for intersection similarity or $ChiSquared$ for chi square similarity. If any other similarity method is entered. The program would ask you to enter an appropriate distance measure.

The output would be a value of total histogram similarity/difference for the two videos, and it will be received by console.

For SIFT Similarity, the user inputs below in MATLAB console

Task1_sift_similarity(*db*, $file1$, $file2$, $Measure$)

db is the path to the .csv database of all videos histogram values. The input db files must be the csv format.

$$\langle video, frame, cell, vector \rangle$$

$file1$ is the first video file name whose sift descriptor values need to be extracted from the db. $file2$ is the second video file name whose sift descriptor values need to be extracted from the db. $eMeasure$ is either $Hausdorff$ for Hausdorff distance or $SquaredEuclidean$ for squared Euclidean Distance. If any other similarity method is entered. The program would ask you to enter an appropriate distance measure.

The output would be a value of total sift similarity/difference for the two videos, and it will be received by console.

For Motion Similarity, the user inputs below in MATLAB console

Task1_motion_similarity(*db*, $file1$, $file2$, $Measure$)

db is the path to the .csv database of all videos histogram values. The input db files must be the csv format.

$$\langle video, frame, cell, vector \rangle$$

$file1$ is the first video file name whose motion vector values need to be extracted from the db. $file2$ is the second video file name whose motion vector values need to be extracted from the db. $eMeasure$ is either $HOMV\_Manhattan$ for Manhattan distance using HOMV or $HOMV\_Intersection$ for Intersection Similarity using HOMV. If any other similarity method is entered. The program would ask you to enter an appropriate distance measure.

The output would be a value of total motion similarity/difference for the two videos, and it will be received by console.

For Overall Similarity, the user inputs below in MATLAB console

Task1_Overall_Similarity($directoryName$, $nameOfHistFile$, $nameOfSiftFile$, $nameOfMVFile$, $file1$, $file2$, $distanceMeasure$, $reduced$)

$directoryName$ is the path of the common directory path where the input files for histogram, sift and motion vectors exist. $nameOfHistFile$ is the name of the file which contain histogram values for all videos. $nameofSiftFile$ is the name of the file which contain sift vector data for all videos. $nameOfMVFile$ is the name of the file which contain motion vector data for all videos. The input db files must be the csv format.

$$\langle video, frame, cell, vector \rangle$$

$file1$ is Its the name of the first file to be compared whose data has to be extracted from the earlier provided histogram, sift and motion vector databases / files. $file2$ is the name of the second file to be compared whose data has to be extracted from the ealier provided histogram, sift and motion vector databases. $distanceMeasure$ is the distance combination to be used for sift, histogram and motion vector respectively. e.g. - Haus_Chi_HI means distance measure 'hausdorff' being used for sift, 'chisquared' being used for hisogram and 'HOMV intersection' is being used for motion vectors. $reduced$ is a flag variable which helps in differentiating between whether the simarity measured has to be put on original dimensions or reduced dimensions..

The output is a value calculated by a linear combination of all similarity measures(SIFT, Histogram and motion vector), and it will be received by console.

B. Task 2

For Task 2, the user input below in MATLAB console

Task2_Overall($directoryName$, $nameOfHistFile$, $nameOfSiftFile$, $nameOfMVFile$, $file1$,a, b, knn, $distanceMeasure$, $reduced$, $videoPath$)

$directoryName$ is the path of the common directory path where the input files for histogram, sift and motion

vectors exist. $nameOfHistFile$ is the name of the file which contain histogram values for all videos. $nameofSiftFile$ is the name of the file which contain sift vector data for all videos. $nameOfMVFile$ is the name of the file which contain motion vector data for all videos. The input db files must be the csv format.

$$\langle video, frame, cell, vector \rangle$$

$file1$ is Its the name of the first file to be compared whose data has to be extracted from the earlier provided histogram, sift and motion vector databases / files. $file2$ is the name of the second file to be compared whose data has to be extracted from the ealier provided histogram, sift and motion vector databases. a is the start of the frame range. b is the end of the frame range. knn is the number of top k most similar frame sequences that have to be computed and it would be the number of output files that would be generated. $distanceMeasure$ is the distance combination to be used for sift, histogram and motion vector respectively. $reduced$ is a flag which would define if the computation has to be made on original dimensions or reduced dimensions. $videoPath$ is path to output directory where the subsequence files would be generated.

The output is a video which have top knn(number) of subsequence frames. The format of video is 'avi'.

C. Task 3

For dimensionality reduction with PCA, the files created in phase 1 as input files and the directory for output files will be selected while running the application.

The input files must be the csv format which follows

$$\langle video, frame, cell, vector \rangle$$

As user input, the application accepts target dimensionality d as user input from commend line to reduce the original dimensionality to the inputted value. Target dimensionality d must be less than the original dimensionality, and must be positive integer.

After that, the application will generate two output files per input file including an output database and d dimensions in terms of the input vector space. The file for an output database is named as 'out_file_d_$x$pca.csv' where d is the target dimensionality and x is one of s for sift, c for color histogram, and m for motion vector. The format is

$$\langle video, frame, cell, vector \rangle$$

For color histogram and motion vector, a $vector$ in the reduced dimensionality is calculated by whole components from a vector in the original dimensionality.

For SIFT, however, a $vector$ in the reduced dimensionality is calculated only by descriptors from a vector in the original dimensionality, that is the detectors are not changed. Therefore, the format of SIFT follows

$$\langle video, frame, cell, x, y, scale, orientation, vector \rangle$$

The file for d dimensions in terms of the input vector space is named as 'out_file_score_d_$x$pca.csv' where d is the target dimensionality and x is one of s for sift, c for color histogram, and m for motion vector. The format is

$$\langle original\ index, score \rangle$$

which means contribution of original dimensions to new dimensions.

For dimensionality reduction with k-means, everything is the same as dimensionality reduction with PCA except for the output file names. In k-means, 'km' is used in the file name instead of 'pca'.

D. Task 4

For Task 4, the user input below in MATLAB console

Task2_Overall($directoryName$, $nameOfHistFile$, $nameOfSiftFile$, $nameOfMVFile$, $file1$,a, b, knn, $distanceMeasure$, $reduced$, $videoPath$)

$directoryName$ is the path of the common directory path where the input files for histogram, sift and motion vectors exist. $nameOfHistFile$ is the name of the file which contain histogram values for all videos. $nameofSiftFile$ is the name of the file which contain sift vector data for all videos. $nameOfMVFile$ is the name of the file which contain motion vector data for all videos. The input db files must be the csv format having vectors in reduced dimensional spaces.

$$\langle video, frame, cell, vector \rangle$$

$file1$ is Its the name of the first file to be compared whose data has to be extracted from the earlier provided histogram, sift and motion vector databases / files. $file2$ is the name of the second file to be compared whose data has to be extracted from the ealier provided histogram, sift and motion vector databases. a is the start of the frame range. b is the end of the frame range. knn is the number of top k most similar frame sequences that have to be computed and it would be the number of output files that would be generated. $distanceMeasure$ is the distance combination to be used for sift, histogram and motion vector respectively. $reduced$ is a flag which would define if the computation has to be made on original dimensions or reduced dimensions. $videoPath$ is path to output directory where the subsequence files would be generated.

The output is a video which have top knn(number) of subsequence frames. The format of video is 'avi'.

The input and output are exactly same as Task 2 except that Task 4 uses vectors in the reduced dimensional space as input instead of vectors in the original dimensional space.

IV. System requirements/installation and executuion instructions

Windows 10 is used as OS. MATLAB and C++(Visual Studios 2015, x-64 bits) are used to implement and execute the various tasks for this phase.

A. Task 1

For Histogram Similarity from MATLAB prompt

1) Open the file "Task1_histogram_similarity.m" in matlab.
2) Input histogram_similarity(db, $file1$, $file2$, $Measure$) in console with the actual parameters as explained Section III Interface Specifications.

For SIFT Similarity from MATLAB prompt

1) Open the file "Task1_sift_similarity.m" in matlab.
2) Input sift_similarity(db, $file1$, $file2$, $Measure$) in console with the actual parameters as explained Section III Interface Specifications.

For Motion Similarity from MATLAB prompt

1) Open the file "Task1_motion_similarity.m" in matlab.
2) Input motion_similarity(db, $file1$, $file2$, $Measure$) in console with the actual parameters as explained Section III Interface Specifications.

For Overall Similarity from MATLAB prompt

1) Open the file "Task1_Overall_Similarity.m" in matlab.
2) Input Overall_Similarity($directoryName$, $nameOfHistFile$, $nameOfSiftFile$, $nameOfMVFile$, $file1$, $file2$, $distanceMeasure$, $reduced$) in console with the actual parameters as explained Section III Interface Specifications.

B. Task 2

From MATLAB prompt

1) Open the file 'Task2_Overall.m' in matlab.
2) Task2_Overall($directoryName$, $nameOfHistFile$, $nameOfSiftFile$, $nameOfMVFile$, $file1$,a, b, knn, $distanceMeasure$, $reduced$, $videoPath$) in console with the actual parameters as explained Section III Interface Specifications.

C. Task 3

For dimensionality reduction with PCA, from MATLAB prompt

1) Open "Task3_PCA.m". in matlab.
2) Input "Task3_PCA" to execute application.
3) Select input files from phase 1 by using UI.
4) Input reduced dimensionality you want of each file.
5) Select directory to store output files by using UI.
6) If a file remains, go to step 3.

For dimensionality reduction with K-means, from MATLAB prompt

1) Open "Task3_Kmeans.m" in matlab.
2) Input "Task3_Kmeans" to execute application.
3) Select input files from phase 1 by using UI.
4) Input reduced dimensionality you want of each file.
5) Select directory to store output files by using UI.
6) If a file remains, go to step 3.

D. Task 4

The process is exactly same as Task 2.

V. Related work

A. Dynamic Programming

Dynamic programming is one of ways to resolve the time complexity issue [12]. Dynamic Programming break its own problem down into simpler sub-problems, then solve sub-problems only once, and store solution into storage. Therefore, when the same sub-problem occurs, it simply looks up the previously computed and stored solution without recalculating solution. So, Dynamic Programming reduce the time complexity issue by using overlapping sub-problems and optimal sub-structures.

B. Sliding Windows

The sliding window algorithm packs a pre-defined buffer-size called 'window' with the corresponding amount of data row[13]. A data stored in the window in advance is compared to the new one because each row is stored into the window. If two data are matched, both the new record to add and records already in the window have the same group ID. This comparison will continue until the new record is compared to all records in the window. Then, the window will finally reach its pre-defined value. From here, the window will slide.

C. Chi Square

Chi square finds the square of difference between each corresponding bin divided by sum of value of corresponding bin and does this for all the available bin [14]. The difference between corresponding bin value is squared to avoid having negative values. If two histogram are similar the distance obtained will be zero.

D. Intersection Similarity

Intersection similarity considers to what degree two vectors overall along each dimension and hence its is commonly used in color and texture histograms [1, p.105]. For intersection similarity, each corresponding bin of two images is taken and minimum value among corresponding bin is taken and in the same way it is computed for all corresponding bins and added all together and divide by maximum value among corresponding bin and it is computed for all available corresponding bin and the value is added.

E. Hausdorff Distance

Hausdorff distance is an asymmetric distance that says the extent to which each point of one set lies near some point of the target set and vice versa [7]. The idea behind hausdorff distance is that suppose if there are two polygons and the shortest distance between two polygons are found. Now one point among the shortest distance producing points is fixed and the longest distance from this point to the points in other polygon is found [15]. This is the idea behind hausdorff distance. Two sets are close if every point on one of the set is close to some point in the other set. Thus, this distance can be used to find the extent to which two objects or images resemble

similar. Hausdorff allows portions of one shape to be compared with another [7]. This distance is chosen for similarity measure using SIFT. SIFT has many dimensions that, even if more number of dimensions are reduced, hausdorff distance measure has good tolerance towards dimensionality reduction [7]. Another reason is that portion of one image can be considered with another which is useful because SIFT feature in one image can be at different location in another image. If two images A and B are there. Hausdorff distance of (A,B) and (B,A) will not be similar. One is called forward hausdorff distance and the other is backward hausdorff distance. We are computing the average of forward and backward hausdorff distance as both are not equal and prevent the result from being biased.

F. Nearest Neighbor Distance Ration(NNDR)

Keypoints between two images are matched to find their nearest neighbor. The probably density function for correct or incorrect matches are shown in terms of the ratio of closest to the second closest neighbors of each keypoint. However, many features from an image, the second neighbor will be very close to the first and we will not have a correct match because of background clutter. In such cases, the ratio of closest distance to second closest is taken. If that ratio is greater than 0.8, they are rejected. This way we can eliminate around 90% of false positives and less than 5% of correct frames [4].

G. Bag of Features

Bag of features in image representation can be defined as collection of various available features from an image. To represent image as bag of features, three steps are involved. They are feature detection, feature description and codebook generation [16]. Once feature is detected, an image is nothing but with several local features. Once those features are detected, we have to find a representational model which is actually a vector. Final step is to put a code word for all similar features [16].

H. Euclidean Distance

In simple terms, euclidean distance, is the square root of the sum of squared differences between corresponding elements of the two vectors [1, p.103-104]. It is commonly used for similarity comparison between two vectors in space and can also be understood as Minkowski distance of order 2. Squared Euclidean Distance is the same as the Euclidean distance with no square root.

I. HOMV

Histogram of Motion Vector is one of the way that is used to find similarity among videos based on the difference in motion vector among videos [17]. It considers both phase angle and magnitude of motion vectors there by ensuring that key information is preserved. As these two are the components that is essential in motion vectors. Instead of just putting the count of vectors in bin (discrete values of angles) it gives a weight by considering the magnitude of the vector. It sums all the magnitudes of vector for each bin. HOMV outcome is a matrix which contains histogram of each frame in each column and row contains temporal information.

J. Manhattan Distance

Manhattan is Minkowski distance of order 1. It can simply be understood by distance between two points measured across axis at right angles. This distance measure is commonly used for its computational efficiency and is used in color based comparisons as research suggest that it may capture human judgement of image similarity better than euclidean distance.[1, p.103-104]

K. Linear Combination

If a combination of different features is used, the distances are normalized to be in the same value range and then a linear combination of the distances is used to create the ranking. Linear combination is described from a set of features by multiplying the feature with a scalar value and summing up together. The set of vectors in which linear combination is done to arrive at a new feature is called spanning vector. We are using linear combination in overall similarity measure so that distances are normalized in the same value range [18].

L. Principal Component Analysis

Principal Component Analysis(PCA) is a linear algebra technique that is used to reduce the original dimensionality to a new dimensional space [1, p.156-159]. PCA is a linear transformation and all the new features should be linearly independent of each other. Because of these characteristics distance and angle between objects are preserved. PCA uses matrix decomposition on correlation matrix of the original dimension and decomposes into left eigenvector matrix, right eigenvector matrix, and a dimension matrix which has eigenvalues. These eigenvalues gives the importance of each new dimension. The feature elimination starts with a vector having low eigenvalue and goes on till the desired reduction in dimension. There are many ways like scree test, mean eigenvalue, and Kaiser-Guttman rule that is helpful for selecting dimensions that preserves most of the variance of the given set of data.

M. K-means

K-means clustering is one of the commonly used clustering algorithm [2]. The aim of the K-means algorithm is to divide M points in N dimensions into K clusters so that the within-cluster sum of squares is minimized. The algorithms is like below

1) Select k points as initial centroid.
2) Assign the data points to the closest centroid.
3) Again recalculate the centroid for the clusters formed
4) repeat until the centroid doesnt vary.

Better clustering can be obtained by not choosing centroid close to each other.

N. Orthogonal Triangular Decomposition(QR decomposition)

According to Kim et al. [3], The QR decomposition algorithm developed by Gould and Businger is transformations that are stable with respect to rounding errors. The QR decomposition is about two to ten times less computationally expensive than Singular value decomposition depending upon the matrix size. In mechanical dynamics applications, QR decomposition functions equally well as singular value decomposition to determine the rank and orthogonal subspaces of the matrix.

VI. Conclusions

As abovementioned, in this paper, we introduced the goal, the assumptions, our proposed solution to solve given tasks, the implementation, the way to execute the applications, and the output format of the project. From this project, we learned and implemented diverse similarity measures, the way to search for the most similar video sub-sequence, and the way to reduce dimensionality with less information loss. Finally, we provide you with sample outputs. For output, due to execution time and space, we use 6 videos out of given 16 videos including

1) 6x_SQ_BL_TM_BR_Check.mp4
2) 6x_SQ_BL_TM_BR_Noise.mp4
3) 6x_SQ_BL_TM_BR_White.mp4
4) 6x_TR_TL_BR_Check.mp4
5) SQ_BL_TR_Noise.mp4
6) TR_BL_TM_BR_75_25_White.mp4

From phase 1, the resolution is 4, and bin of color histogram is 10.

For task 1, color histogram similarity having parameters as '6x_SQ_BL_TM_BR_Check.mp4', '6x_SQ_BL_TM_BR_White.mp4', and 'Intersection' results in 0.1056. SIFT similarity having parameters as '6x_SQ_BL_TM_BR_Check.mp4', '6x_SQ_BL_TM_BR_White.mp4', and 'Intersection' results in 1.0886. Motion similarity having parameters as '6x_SQ_BL_TM_BR_Check.mp4', '6x_SQ_BL_TM_BR_White.mp4', and ''HOMV_Manhattan' results in 0.0625.

For task 2, We will provide sample output as a video file.

For task 3, the reduced dimensionality is 7 for color histogram, 50 for SIFT, and 5 for motion vector. The output file of task 3 will submit as csv format files with score files of csv format. Therefore, the number of output files of Task 3 is 12.

For task 4, We will provide sample output as a video file.

References

[1] K Selçuk Candan and Maria Luisa Sapino. *Data management for multimedia retrieval*. Cambridge University Press, 2010.

[2] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[3] SS Kim and MJ Vanderploeg. Qr decomposition for state space representation of constrained mechanical dynamic systems. *Journal of Mechanisms, Transmissions, and Automation in Design*, 108(2):183–188, 1986.

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

[5] Dimitri A Lisin. *Image Classification With Bags OF Local Features*. PhD thesis, University of Massachusetts Amherst, 2006.

[6] Yap-Peng Tan, Sanjeev R Kulkarni, and Peter J Ramadge. A framework for measuring video similarity and its application to video query by example. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 2, pages 106–110. IEEE, 1999.

[7] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[8] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10): 1615–1630, 2005.

[9] TUDOR BARBU. A novel image similarity metric using sift-based characteristics. In *Mathematical Models in Engineering and Computer Science: Proceedings of the 2nd International Conference on Computers, Digital Communications and Computing, ICDCC'13*, pages 15–18, 2011.

[10] Yining Deng and BS Manjunath. Content-based search of video using color, texture, and motion. In *Image Processing, 1997. Proceedings., International Conference on*, volume 2, pages 534–537. IEEE, 1997.

[11] Cheong Hee Park and Haesun Park. Nonlinear feature extraction based on centroids and kernel functions. *Pattern Re'cognition*, 37(4):801–810, 2004.

[12] Stuart E Dreyfus and Averill M Law. *Art and Theory of Dynamic Programming*. Academic Press, Inc., 1977.

[13] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM, 2002.

[14] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[15] Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3-4):251–265, 1995.

[16] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):591–606, 2009.

[17] Kasım Tasdemir. *Content based video copy detection using motion vectors*. PhD thesis, Citeseer, 2009.

[18] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

Appendix A
Roles of the Group Members

1) Anurag Arora : Implementing key frame extraction, sub-sequence video generation from frames and making the report.
2) Bineeta Gupta : Implementing motion similarity, using Original dimesnions and reducded dimensions each, for Task 1, 2, and 4.
3) Kyungyong Han : Implementing dimensionality reduction using PCA and K-means in Task 3 and making the report.

4) Neha Prasad : Implementing color histogram similarity in Task 1, 2, and 4.
5) Nishant Jagadeesan : Integrating Similarity measures to use it in Task 2 and 4 and making the report.
6) Ujjwal Dasu : Implementing SIFT similarity in Task 1, 2, and 4. Integrating Similarity measures to use it in Task 2 and 4.