Significant or Relevant Frame Selection with Clustered Data and Similar Video Object Search with Locality Sensitive Hashing

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*—SIFT, Principal Component Analysis, Dimensionality Reduction, Locality Sensitive Hashing, Relevance Feedback**

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].

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.

ACOS++: It is an Asymmetric Similarity Measure for Weighted Networks. For addressing the problems of SimRank, we follow new similarity measures ASCOS and ASCOS++; these processes outputs a more complete similarity score than SimRank and SimRanks families. ASCOS++ enriches ASCOS to include edge weight into themeasure, giving all edges and network weights an opportunity to make their contribution. ASCOS++, which enriches Asymmetric Network Structure Context Similarity (ASCOS) by including all paths between nodes and the weights of the edges along the paths in the calculation.

II. Description of the Proposed Solution/Implementation

A. Video dimensionality reduction using PCA

Video dimensionality reduction using PCA produces two outputs including an output database and the reduced dimensions in terms of the original dimensionality in non-increasing order of scores.

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

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

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.

$$X = D - \bar{D} \tag{2}$$

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.

$$S = PCP^{-1} \tag{3}$$

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 1.

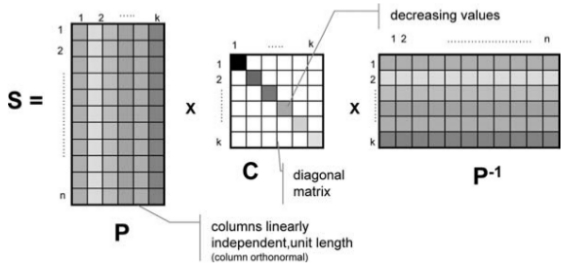

Fig. 1: Eigen decomposition [1, p.157]

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 2.

$$S' = P'C'P'^{-1} \tag{4}$$

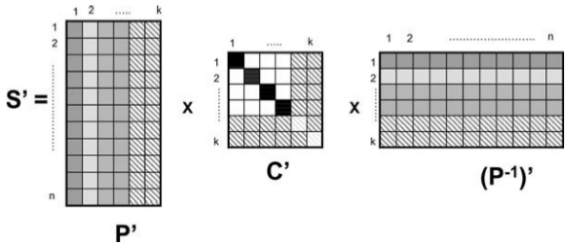

Fig. 2: How to remove eigenvectors with small eigenvalue [1, p.159]

Here, P' will be used and stored after being sorted in non-increasing order 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

$$D' = DP' \tag{5}$$

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 database. Then, D' is stored with a corresponding label as

$$\{\langle i, j, l, x, y \rangle, [dim_i, ..., dim_d]\}$$

where i is the video number, j is the frame number, l is the cell number, the pair x and y is the position of the SIFT keypoint in the frame, and $[dim_i, ..., dim_d]$ are the reduced dimensional output database.

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. To sort $score$ in non-increasing order, we used $sort$ function in MATLAB, and then output B and I. See Figure 3.

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

[B,I] = sort(V, 'descend');
% write score to file
new = zeros(130:2*dimensionality);
for i = 1:dimensionality
    new(:,1+2*(i-1)) = I(:,i);
    new(:,2+2*(i-1)) = B(:,i);
end
```

Fig. 3: Code of PCA

B. Similarity Graph Generation

C. PageRank

D. ASCOS++

For implementing ASCOS++, we followed the below steps-

n = total videos X total number of frames in each video

I = Identity matrix

S = Similarity matrix

A = Adjacency Matrix

- We have the csv file from Task2.

- We generate the adjacency matrix, which contains the similarity measure (considering edge weights) between any two frames.
- Find a matrix P which is obtained upon normalizing the adjacency matrix.See Figure 4.

```
// For normalizing the P matrix- A) Summation part
for(int i = 0; i < row; i++){
    float sum = 0.00;
    for(int j = 0; j < row; j++){
         sum = sum + adj_matrix[j][i];
    }
    column_sum[i] = sum;
}
//normalizing P matrix
for(int i = 0; i < row; i++){
    for(int j = 0; j < row; j++){
        if(column_sum[i] != 0){
        P[j][i] = adj_matrix[j][i] / column_sum[i];
    }
        else{
            P[j][i] = 0;
        }
    }
}
```

Fig. 4: Code of Normalization

- We find a matrix Q, using the formula : Q(i,j)=P(i,j)(1-exp(-A(i,j));See Figure 5.

```
for(int i = 0; i < row; i++){
for(int j = 0; j < row; j++){
    int v=exp(-adj_matrix[i][j]);
Q[i][j]=P[i][j]*(1-v);
```

Fig. 5: Calculation of Q matrix

- Find transpose of Q
- Perform following computation to find column matrix of similarity using (I-cQ(i))S(i)=(1-c)* I(i) where i=1,2,..n. For faster way to calculate S(i), we use Gauss siedel approach. See Figure 6.
- S(i) is the column matrix. Perform this for n times, to get S1,S2,S3Sn. This will give following Similarity matrix of nXn.
- For selecting top m frames- For each column of similarity matrix, add the similarity weights for that column aka frame with all other frames of the graph. (do it column-wise as well as row-wise as this is assymetric matrix)
  Follow the formula- Total weight * (number of nodes having non-zero similarity weight)/(total number of nodes) See Figure **??**.
  7
  This way we get the significance of the similarity measure from the frame having contributing weights.
- Store the choosen m frames with their corresponding video number, in a text file.

```
do{
    big = 0;
for(int col = 0; col < row; col++){
        for(int i = 0; i < row; i++){
            float sum = 0;
            for(int j = 0; j < row; j++){
                if(i != j){
                    sum = sum + (Q[i][j] * similarity_score[j][col]);
                }
            }
            temp = (identity_matrix[i][col] - sum)/Q[i][i];
            error = fabs(identity_matrix[i][0] - temp);
            if(error > big){
                big = error;
            }
            similarity_score[i][col] = temp;

        }
    }
    m--;
}while(m >= 0);
```

Fig. 6: Gauss siedel approach

```
for (int t=0;t<row;t++)
{
    column_summ=0;
    nonzero_value=0;

    for (int y=0;y<total_columnns;y++)
    {

        if (similarity_score[y][t]!=0)
        {
            nonzero_value=nonzero_value+1;
            column_summ=column_summ+similarity_score[y][t];
        }

    }// row is read
        int temp=similarity_frame_array[count_similarity_frame_array];
        similarity_frame_array[count_similarity_frame_array]= (column_summ)*nonzero_value/total_columnns;
        similarity_frame_array[count_similarity_frame_array]=similarity_frame_array[count_similarity_frame_array] + temp;
        count_similarity_frame_array=count_similarity_frame_array+1;
} //column is read
        count_similarity_frame_array=count_similarity_frame_array -1;
```

Fig. 7: Top m frame calculation(a)

```
    std::priority_queue<std::pair<float, int> > q;

for (int i = 0; i <=count_similarity_frame_array; ++i) {
   q.push(std::pair<float, int>(similarity_frame_array[i], i));
 }

// number of indices we need
 int top_m_frame[k];
int count_top_m_frame=0;

 for (int i = 0; i < k; ++i) {
  float ki = q.top().second;
   std::cout << "index[" << i << "] = " << ki << std::endl;
   top_m_frame[count_top_m_frame]=ki;
   count_top_m_frame=count_top_m_frame+1;
  q.pop();
}
```

Fig. 8: Top m frame calculation(b)

E. Personalized PageRank

F. Personalized ASCOS++

For Personalised ASCOS++, we follow the same process for ASCOS++, with the only difference in using the Identity matrix with altered fashion.

Instead of normal Identity matrix, design a different identity matrix- 1. Go column-wise 2. If the given column i.e frame is one of the points of starting and is part of given input seed frame, then assign the value of 0.33 to the identity_matrix[seed frame][column]

Else you need to assign value of 0.25 to the three seed nodes and the node i.e frame from where we are starting. See Figure **??**.

3. Once identity matrix is obtained, follow the same process

```
// initializes identity matrix, going column wise, row=column
for(int j = 0; j < row; j++){
    if (j==node1 || j==node2 || j==node3)
    {
        identity_matrix[node1][j] = 0.33;
        identity_matrix[node2][j] = 0.33;
        identity_matrix[node3][j] = 0.33;
    }
    else
    {
        identity_matrix[j][j] = 0.25;
        identity_matrix[node1][j] = 0.25;
        identity_matrix[node2][j] = 0.25;
        identity_matrix[node3][j] = 0.25;
    }
}
```

Fig. 9: Identity matrix creation

of ASCOS.

G. Locality Sensitive Hashing

We use hash function which Brian Kulis et al. presented in their paper [2] using the well-known inner product similarity function based on rounding.

$$h_r(x) = \begin{cases} 1 & 1, \text{ if } r^T x \geq 0 \\ 0 & 0, \text{ otherwise} \end{cases} \tag{6}$$

where r is a random hyperplane from standard normal distribution $N(0,1)$ of the same dimensionality as the input x. Goemans and Williamson et al.[3] presented that above hash function satisfies LSH property.

As following a paper of Alexandr Andoni et al.[4], we concatenate hash functions in order to amplify the gap between P_1 and P_2 in parameters (R, cR, P_1, P_2). For parameters K and L, we choose L layers and maximum 2^K number of buckets per each layer as below

$$g_j(q) = (h_{1,j}(q), ..., h_{k,j}(q))$$

where $h_{t,j}$ $(1 \leq t \leq k, 1 \leq j \leq L)$. Then, we assign the bucket number to each point as follwoing below step.

1) Select unique buckets in each layer for all points.
2) Sort unique buckets in increasing order.
3) assign the index of unique buckets to corresponding point as the bucket number.

The code of LSH is below. See figure 10.

```
% do for-loop as many as the number of layers.
for j=1:l

    % Select random hyperplane from standard normal distribution N(0,1)
    % as many as "k", and each the size of each r is the same as
    % dimensionality.
    r = randn(dimensionality,k);

    % do inner product similarity hash based on rounding
    h =  0 <= hist*r;

    % assign the bucket number to each point
    [~, ~, bucket] = unique(h,'rows');
```

Fig. 10: Code of LSH

Finally, The output is stored with a corresponding label as

$$\{layer\_num, bucket\_num, \langle i, j, l, x, y \rangle\}$$

where i is the video number, j is the frame number, l is the cell number, the pair x and y is the position of the SIFT keypoint in the frame.

H. Similar Video Object Search

III. Interface Specifications

A. Task 1

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

The input file must be the csv format which follows

$$\{\langle i, j, l \rangle, [x, y, scale, orientation, a_1, ..., a_1 28]\}$$

where i is the video number, j is the frame number, l is the cell number, $[x, y, scale, orientation]$ are the sift detectors where the pair x and y is the position of the SIFT keypoint in the frame, and $[a_1, ..., a_1 28]$ are the sift descriptors.

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 including an output database and d dimensions in terms of the input vector space in non-increasing order. The file for an output database is named as '$filename_d$_spc.csv' where $filename$ is an input file name and d is the target dimensionality. The format is

$$\{\langle i, j, l, x, y \rangle, [dim_i, ..., dim_d]\}$$

where i is the video number, j is the frame number, l is the cell number, the pair x and y is the position of the SIFT keypoint in the frame, and $[dim_i, ..., dim_d]$ are the reduced dimensional output database.

The file for d dimensions in terms of the input vector space in non-increasing order is named as '$filename_d$_spc.csv' where $filename$ is an input file name and d is the target dimensionality.

The format is

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

which means contribution of original dimensions to new dimensions.

B. Task 2

C. Task 3

For ASCOS++, the user takes the CSV file from Task 2 and asks for the top m most similar and relevant frames. The CSV will be having 5 columns, having 5th column

indicating the similarity measure value of a video number and its corresponding frame compared with other videos and their frames.

The format of CSV is

$$\langle video1, frame1, video2, frame2, similarity\_value \rangle$$

Final output, after computation, will provide us a similarity matrix with nXn matrix where n is the total number of videos x total number of frames of each video. Also we will get a file where we get the frame number and corresponding video number from the entire graph having most similarity and relevancy with other frames of other videos.

The format of TXT is

$$\langle frame, video \rangle$$

D. Task 4

For Personalised ASCOS++, we repeat the sam eprocess as we do for ASCOS with only difference that we calculate the similarity measure only for the seed frames. For this, instead of creating identity matrix, we create a matrix where we provide value only for the seed frames matching row and column. And rest will be zero.

E. Task 5

For LSH, the SIFT vector file in reduced dimensionality created in task 1 as input file and the directory for output files will be selected while running the application.

The input file must be the csv format which follows

$$\{\langle i, j, l, x, y \rangle, [dim_i, ..., dim_d]\}$$

where i is the video number, j is the frame number, l is the cell number, the pair x and y is the position of the SIFT keypoint in the frame, and $[dim_i, ..., dim_d]$ are the reduced dimensional output database.

As user input, the application accepts the number of layers L and the number (2^K) of buckets per layer as user input from commend line. L and K must be positive integer.

After that, the application will generate a output file. The file for an output database is named as '$filename$_K_K_L_L_lsh.csv'

where $filename$ is an input file name, L is the number of layers, and K is the number (2^K) of buckets per layer. The format is

$$\{layer\_num, bucket\_num, \langle i, j, l, x, y \rangle\}$$

where i is the video number, j is the frame number, l is the cell number, the pair x and y is the position of the SIFT keypoint in the frame.

F. Task 6

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 dimensionality reduction with PCA, from MATLAB prompt

1) Open "Task1.m" in matlab.
2) Input "Task1" to execute application.
3) Select input a file generated from phase 1 with UI.
4) Input the new dimensionality to which you want to reduce the original dimensionality.
5) Select directory to store output files with UI.

B. Task 2

C. Task 3

For task 3B,

1) Open the "ascos_3b.cpp" file in DevCPP or any applicaiton which can support GCC Compiler. In the code- as for the CSV file location and enter the value of top m frames required.
2) Store the top m frame selection with their corresponding video in a txt file.

D. Task 4

For task 4B,

1) Open the csv file "ascos_4b.cpp" in DevCPP or any application which can support GCC compiler. In the code- hard code the CSV file location and enter the value of top m frames required. Also enter the seed frames.
2) Store the top m frame selection with their corresponding video in a txt file.

E. Task 5

For Locality Sensitive Hashing, from MATLAB prompt

1) Open "Task5.m" in matlab.
2) Input "Task5" to execute application.
3) Select input a file generated from task 1 with UI.
4) Input the number(2^K) of buckets per layer, and the number of layers, L.
5) Select directory to store an output file with UI.

F. Task 6

V. Related work

A. 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.

B. Similarity Graph Generation

C. PageRank

D. ASCOS++ and Personalised ASCOS++

[5] ASCOS++ is an Asymmetric Similarity Measure for Weighted Networks. For addressing the problems of SimRank, we follow new similarity measures ASCOS and ASCOS++; these processes outputs a more complete similarity score than SimRank and SimRanks families. ASCOS++ enriches ASCOS to include edge weight into themeasure, giving all edges and network weights an opportunity to make their contribution. ASCOS++, which enriches Asymmetric Network Structure Context Similarity (ASCOS) by including all paths between nodes and the weights of the edges along the paths in the calculation.

E. Personalized PageRank

F. Locality Sensitive Hashing

According to Alexandr Andoni et al. [4], The Locality Sensitive Hashing algorithm depends on the existence of Locality Sensitive Hash functions. They define the LSH algorithm in there paper as follows. Let $\mathcal{H}$ be a family of hash functions mapping $\mathbb{R}^d$ to some universe U. For any two points p and q, consider a process in which we choose a function h from $\mathcal{H}$ uniformly at random, and analyze the probability that $h(p) = h(q)$. The family $\mathcal{H}$ is called locality sensitive (with proper parameters) if it satisfies the following condition.

A family $\mathcal{H}$ is called (R, cR, P_1, P_2)-sensitive if for any two points $p, q \in \mathbb{R}^d$.

- if $\|p - q\| \leq R$ then $Pr_{\mathcal{H}}[h(q) = h(p)] \geq P_1$,
- if $\|p - q\| \geq cR$ then $Pr_{\mathcal{H}}[h(q) = h(p)] \leq P_2$.

In order for a locality-sensitive hash (LSH) family to be useful, it has to satisfy $P_1 > P_2$.

In addition, they also present the amplification step for LSH to supplement its drawback which the gap between P_1 and P_2 can be very small with one LSH function. Therefore, they tried to achieve the desired probabilities of collision by amplifying hash functions. They explain this step as follows.

- Choose L functions g_j, $j = 1, ...L$, by setting $g_j = (h_{1,j}, h_{2,j}, ...h_{k,j})$, where $h_{1,j}, h_{k,j}$ are chosen at random from the LSH family $\mathcal{H}$.
- Construct L hash tables, where, for each $j = 1, ...L$, the j^th hash table contains the dataset points hashed using the function g_j.

G. Similar Video Object Search

VI. Conclusions

References

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

[2] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *2009 IEEE 12th international conference on computer vision*, pages 2130–2137. IEEE, 2009.

[3] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[4] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Communications of the ACM, vol. 51, no. 1*, pages 117–122. ACM, 2008.

[5] Hung-Hsuan Chen 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):15, 2015.

Appendix A
Roles of the group members

1) Anurag Arora : Implemeting Task 6 and making the report.
2) Bineeta Gupta : Implemeting Task 3b and 4b and making the report.
3) Kyungyong Han : Implementing Task 1 and 5 and making the report.
4) Neha Prasad : Implemeting Task 3a and 4a and making the report.
5) Nishant Jagadeesan : Implemeting Task 3b and 4b and making the report.
6) Ujjwal Dasu : Implementing Task 2 and making the report.