Video Retrieval Techniques

Matthew Mendez
Mukund Manikarnike
Pawan Mahalle
Runyu Jin
Victor Garcia

Multimedia Information Systems
CSE 408/598 – Phase – 3
Fall 2015

school of computing, informatics, & decision systems engineering

ASU IRA A. FULTON SCHOOLS OF engineering

Contents

Tables and Figures

Abstract

Several of the Video encoding standards including MPEG-2 have searching for similar regions of interest in frames of or searching for similar frames built in to them. If these searches are accurate, we can encode these videos in a better fashion and hence be able to achieve better compression. A major part of achieving these searches efficiently involves feature extraction from frames and identifying most similar ones among them. Feature extraction could involve applying several of the transform coding techniques and searching could involve matching the extracted features among the frames to be compared. This project explores all of these concepts and implements certain schemes to perform feature extraction and frame matching techniques.

Keywords - Histogram, 2D-DCT, 2D-DWT, Difference Histogram

Introduction

Feature extraction and similarity matching are important topics in the field of multimedia processing. There are multiple features associated with multimedia like images and videos. These features can be used to calculate various measurements like similarity matching, classification and retrieval.

In this project, we focus on extracting five features specified in goal description from image frames and perform similarity matching based on all five features.

Goal Description

This project performs the following tasks

1. Representation of a video as a set of Frames which contain a set of 8 x 8 blocks.
2. Performs Feature extraction by
    a. Quantizing the Y Color space of each pixel in each block in each frame
    b. Applying 2D Discrete Cosine Transform to each block of each frame
    c. Applying 2D Discrete Wavelet Transform to each block of each frame
    d. Quantizing the block-wise differences in the Y color space of adjacent frames.
    e. Applying 2D Discrete Wavelet Transform to each frame
3. Retrieves the best matching frames in a given video using each of the above features.

Assumptions

The project assumes that

1. Videos of high file size won't be provided as an input because optimization in terms of speed hasn't been the focus of this project.
2. Videos would contain frames with dimensions that are divisible into 8 x 8 frame blocks. In other words, frames with dimensions that aren't multiples of 8 x 8 blocks won't be handled by using appropriate padding.

Requirements Specification

This section introduces each of the requirements that are required to be implemented as part of the project.

Task I Requirements

[REQ_1] The program shall be able to read a video file
[REQ_2] The program shall treat the content of the video file on a frame by frame basis.
[REQ_3] The program shall divide each frame in the video file into 8 x 8 blocks

Task I (a) Requirements

[REQ_4] The program shall create n-bin gray scale histograms by quantizing the Y color space for each cell block of each frame in the video.
[REQ_5] The quantized outputs shall be written to an output file in the following format. *<frame_id, block_coord, gray_instance_id, num_pixels>*
[REQ_6] The quantized outputs shall be written to an output file with the name in the following format *Video_filename_hist_n.hst*

Task I (b) Requirements

[REQ_7] The program shall apply 2-D DCT on the Y Component of each block of each frame
[REQ_8] The program shall select n most significant frequency components on applying 2-D DCT.
[REQ_9] The program shall output the selected frequency components into an output file in the following format. <*frame_id, block_coord, freq_comp_id, value>*
[REQ_10] The program shall name the output file in the following format. *Video_filename_blockdct_n.bct*

Task I (c) Requirements

[REQ_11] The program shall apply 2-D DWT on the Y Component of each block of each frame
[REQ_12] The program shall select n most significant wavelet components on applying 2-D DWT
[REQ_13] The program shall output the selected frequency components into an output file in the following format. <*frame_id, block_coord, wavelet_comp_id, value>*
[REQ_14] The program shall name the output file in the following format. *Video_filename_blockdwt_n.bwt*

Task I (d) Requirements

[REQ_15] The program shall create n-bin gray scale histograms by quantizing the difference in the Y color space of each block in adjacent frames.
[REQ_16] The quantized outputs shall be written to an output file in the following format. *<frame_id, block_coord, diff_component_id, pixel_count >*
[REQ_17] The quantized outputs shall be written to an output file with the name in the following format *Video_filename_diff_n.dhc*

Task II Requirements

[REQ_18] The program shall apply 2-D DWT on the Y component of each image.

[REQ_19] The program shall select m significant wavelet components on applying 2-D DWT.

[REQ_20] The program shall output the selected frequency components into an output file in the following format. <*frame_id, wavelet_comp_id, value*>

[REQ_21] The program shall name the output file in the following format. *Video_filename_blockdwt_m.fwt*

Task III Requirements

[REQ_22] The program shall take a query video as an input.

[REQ_23] The program shall take a frame ID as an input

[REQ_24] The program shall take n and m as inputs to be used to match against the features extracted by using one of the several transformations discussed until now.

[REQ_25] The program shall display 10 best matching frames (except itself) for the selected frame in the selected video for each of the 5 features extracted above.

Design Choices

Modular Class design

[DSN_CH_1] The project uses various classes to perform critical operations. The classes are described as follows

a. **Class VideoInformationExtractor**
   i. Class VideoInformationExtractor reads a video file provided as user input and provides API to extracts the following critical information. Class VideoInformationExtractor converts the color frames to gray scale images and also stores the original pixel values into a text file. The class provides the following methods
      1. getFrame(frameId) : provides 2-D matrix in the form of ndarray[2] frame number frameId from video file
      2. getBlock(frameId, blockRow, blockCol): provides the 8 x 8 frame block with row number blockRow and column number blockCol from frame frameId extracted from input video.

b. **Class HistogramGenerator**
   i. Class VideoInformationExtractor reads a video file provided as user input and provides API to extracts the following critical information. Class VideoInformationExtractor converts the color frames to gray scale images and also stores the original pixel values into a text file. The class provides the following methods
      1. getFrame(frameId) : provides 2-D matrix in the form of ndarray[2] frame number frameId from video file
      2. getBlock(frameId, blockRow, blockCol): provides the 8 x 8 frame block with row number blockRow and column number blockCol from frame frameId extracted from input video.

c. **Class SimilarityMatcher**
   i. This class provides the methods to perform similarity matching for various featues extracted from video. The class currently provides the following similarity matching procedures
      1. NBinHistogramSimilarityMatcher
      2. BlockLevel2DDWTSimilarityMatcher
      3. FrameLevel2DDWTSimilarityMatcher
      4. NBinDiffHistogramSimilarityMatcher

d. **Class SimilarImageGenenator**
   i. This class generates the images of similar frames when frame numbers are provided.

N bin Gray Scale Histogram

[DSN_CH_2] The n-bin gray scale histogram for each cell block of each frame in the given video is carried out as described below using class VideoInformationExtractor and class HistogramGenerator

a. The program generates n-bin gray scale histogram for each frame block and returns the information in the form of list of tuples of the form <*frame_id, block_x_coord, block_y_coord, gray_instance_id, value*>

b. The information extracts are stored in a file with name format *video_filename_hist_n.hst.*

Frame matching using N bin Gray Scale Histogram

[DSN_CH_3] Since we can representing each frame as a vector and each value has same weight, we can use edclidean distance as measure of similarity.

Class SimilarityMatcher has method NbinHistogramSimilarityMatcher which calcuates the Euclidean distance of query frame from all other frame. Smaller values of distances indicate higher similarity. The distance is calculated as follows

Distance = $\sum (f_{query}(i,j,h) - g(i,j,h))^2$

$f_{query}(i,j,h)$ is the h^{th} histogram value for the block (i , j) in the query frame.

g(i, j, h) is the h^{th} histogram value for the block (i , j) in every other frame.

Discrete Cosine Transform

[DSN_CH_4] The Discrete Cosine Transform is carried out as described below.

a. For each 8 × 8 block in a frame, the following equation is applied. The application is described in the points that follow.

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right] .f(i,j)$$

where $$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

b. The basic operation of the DCT is as follows
   i. Given an input image whose dimensions are N × M
   ii. f(i, j) is the intensity of the pixel in row i and column j
   iii. F(u,v) is the DCT coefficient in row k1 and column k2 of the DCT matrix.
   iv. For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of the DCT.
   v. Compression is achieved since the lower right values represent higher frequencies, and are often small - small enough to be neglected with little visible distortion.
   vi. The DCT input is an 8 by 8 array of integers. This array contains each pixel's gray scale level.
   vii. 8 bit pixels have levels from 0 to 255.

[DSN_CH_5] The extraction of significant components is carried it out in a zig zag manner as shown in **Table 3**. The extraction method is shown for an 8 x 8 block. The approach followed for a larger region is also the same.

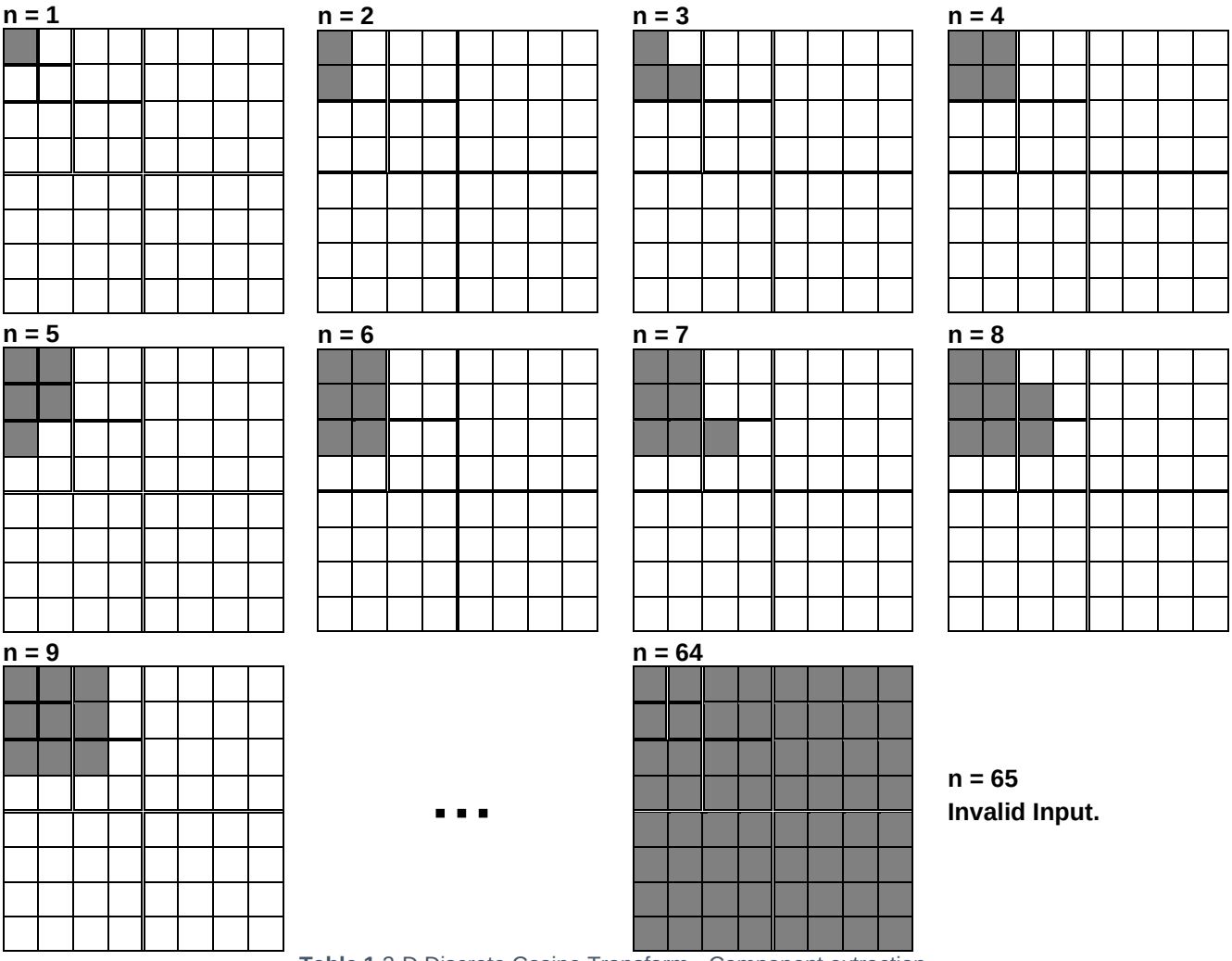


Table 1 2-D Discrete Cosine Transform - Component extraction

Frame matching using Discrete Cosine Transform

[DSN_CH_6] Following is the description of how frame matching is carried out using DCT.

a. Given a query frame and the entire video
    i. For each block in the query frame as well as every other frame in the video, a tuple containing the following is created.
        1. The DC component in the top left most quadrant
        2. The standard deviation of the AC components in all other regions.
b. For the query frame and each other frame in the video, there would be 'n' such tuples if there are 'n' blocks in the video.
c. For every frame in the video, the Euclidean distance between the query frame and every other frame in the video is obtained.
d. Sorting the frames based on the Euclidean distance between the averages first and then the standard deviation would give us a list of frames that match the given query frame. Sorting based on both the Euclidean distances with priority given to the average ensures that the average has a higher weight and this is important because the averages have more information about the image than the standard deviations. The standard deviations are used to resolve ties between to same averages.

Discrete Wavelet Transform

The approaches described in this section are referenced from [1]

[DSN_CH_7] Discrete Wavelet Transform is either applied to 8 x 8 blocks in each frame in the video input or to the entire frame as it is.

[DSN_CH_8] DWT assumes that the input frame or region of interest has dimensions as a power of 2.

[DSN_CH_9] In order to describe the 2 dimensional discrete wavelet transform, the understanding of the 1 dimensional discrete wavelet transform needs clear which is explained as follows

a. Given a row of pixels, the averages and details of alternating pixels are calculated as described in **Figure 1**. The pictorial representation describes the procedure for 8 such values. The procedure would remain the same for any length that is longer than that as well.

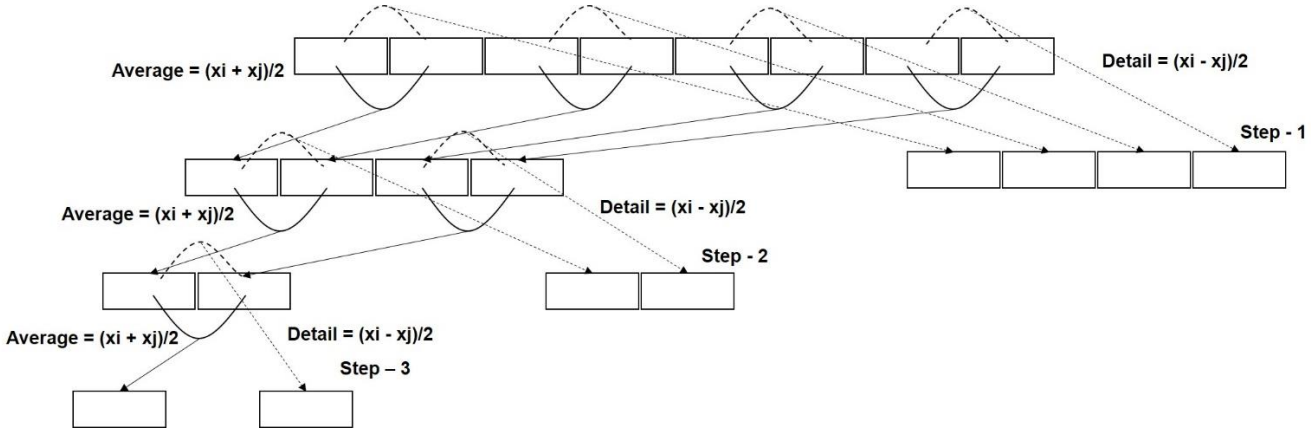


Figure 1 1D - Discrete Wavelet Transform

b. This process is repeated until a single average and single detail element are produced. This would mean that the process would run for $\log_2 n$ times.

[DSN_CH_10] The 2 dimensional discrete wavelet transform proceeds in the following way.

a. The procedure described for 1 dimensional DWT is carried out on each row in the region of interest and then applied to each column in the region of interest as shown in **Table 2.**

Step - 1

Row – wise approach

-	-	-	-	-	-	-	>
-	-	-	-	-	-	-	>
-	-	-	-	-	-	-	>
-	-	-	-	-	-	-	>
-	-	-	-	-	-	-	>
-	-	-	-	-	-	-	>
-	-	-	-	-	-	-	>
-	-	-	-	-	-	-	>

Column – wise approach

|	|	|	|				
|	|	|	|				
|	|	|	|				
|	|	|	|				
|	|	|	|				
|	|	|	|				
|	|	|	|				
v	v	v	v				

Output

A1	A1	A1	A1	D1	D1	D1	D1
A1	A1	A1	A1	D1	D1	D1	D1
A1	A1	A1	A1	D1	D1	D1	D1
A1	A1	A1	A1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1

Step – 2

Row-wise Approach

-	-	-	>				
-	-	-	>				
-	-	-	>				
-	-	-	>				

Column-wise Approach

|	|	|	|				
|	|	|	|				
|	|	|	|				
v	v	v	v				

Output

A2	A2	D2	D2	D1	D1	D1	D1
A2	A2	D2	D2	D1	D1	D1	D1
D2	D2	D2	D2	D1	D1	D1	D1
D2	D2	D2	D2	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1

Step – 3

Row-wise Approach

-	>						
-	>						

Column-wise Approach

|							
v							

Output

A3	D3	D2	D2	D1	D1	D1	D1
D3	D3	D2	D2	D1	D1	D1	D1
D2	D2	D2	D2	D1	D1	D1	D1
D2	D2	D2	D2	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1
D1	D1	D1	D1	D1	D1	D1	D1

Table 2 2-D Discrete Wavelet Transform

b. After each row-wise or column-wise step, the averages and details obtained are appended to the same row with the averages listed first and the details listed next.
c. As shown in **Table 2**, each such iteration produces the top left quadrant with average values and the other 3 quadrants with the detail information and every iteration focusses on applying the procedure described here on the smaller top left quadrant of average values.
d. The procedure is described for an 8 x 8 region. The process would be similar if the region were bigger. The only requirement would be that dimensions should be a power of 2. If not, we would need to carry out padding which isn't taken care of in this design.

[DSN_CH_11] The extraction of significant components is carried it out in a zig zag manner as shown in **Table 3**. The extraction method is shown for an 8 x 8 block. The approach followed for a larger region is also the same.

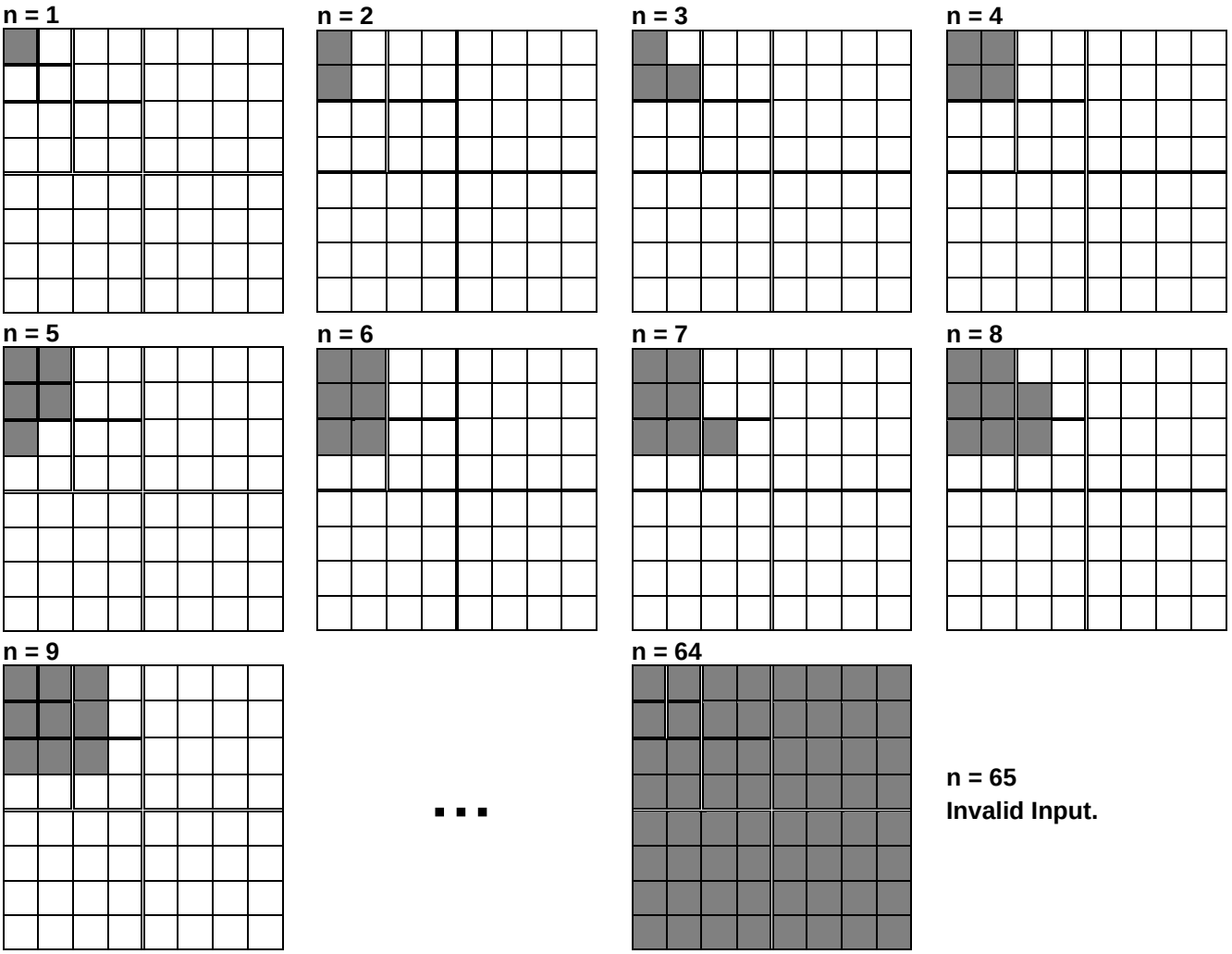


Table 3 2-D Discrete Wavelet Transform - Wavelet extraction

Frame matching using Discrete Wavelet Transform

The approaches described in this section are referenced from [2]

[DSN_CH_12] As shown in **Table 2**, the average information is concentrated in the top left corner of the region of interest and the details are present in the other 3 quadrants of the region of interest. Similarity matching is carried out in 2 different approaches for block level DWT and frame level DWT as described in

[DSN_CH_13] **Block Level DWT**

a. Given a query frame and the entire video
    i. For each block in the query frame as well as every other frame in the video, a tuple containing the following is created.
        1. The average in the top left quadrant
        2. The standard deviation of the detail information in all other regions.
b. The query frame and each other frame in the video, there would be ‘n’ such tuples if there are ‘n’ blocks in the video.
c. For every frame in the video, the Euclidean distance between the query frame and every other frame in the video is obtained.

d. Sorting the frames based on the Euclidean distance between the averages first and then the standard deviation would give us a list of frames that match the given query frame. Sorting based on both the Euclidean distances with priority given to the average ensures that the average has a higher weight and this is important because the averages have more information about the image than the standard deviations. The standard deviations are used to resolve ties between to same averages.

[DSN_CH_14] **Frame Level DWT**

a. Given a query frame and the entire video
    i. For each frame, tuples containing the following is created.
        1. The average in the top left quadrant
        2. The standard deviation of the detail information in all other regions.
b. For every frame in the video, the Euclidean distance between the query frame and every other frame in the video is obtained.
c. Sorting the frames based on the Euclidean distance between the averages first and then the standard deviation would give us a list of frames that match the given query frame. Sorting based on both the Euclidean distances with priority given to the average ensures that the average has a higher weight and this is important because the averages have more information about the image than the standard deviations. The standard deviations are used to resolve ties between to same averages.

N bin Difference Gray Scale Histogram

[DSN_CH_15] The n-bin difference gray scale histogram for each cell block of each frame in the given video is carried out as described below using class VideoInformationExtractor and class HistogramGenerator

a. The program generates n-bin gray scale histogram for each block of differenec frame and returns the information in the form of list of tuples of the form <*frame_id, block_x_coord, block_y_coord, diff_comp_id, value*>
b. The information extracts is stored in a file with name format *video_filename_diff_n.dhc.*

Frame matching using N bin Difference Gray Scale Histogram

[DSN_CH_16] Since we can representing each frame as a vector and each value has same weight, we can use Euclidean distance as measure of similarity.

Class SimilarityMatcher has method NbinHistogramSimilarityMatcher which calculates the Euclidean distance of query frame from all other frame. Smaller values of distances indicate higher similarity. The difference is calculate on cumulative sum of differences till frame index under consideration since the cumulative sum of differences indicate the absolute difference in pixel values from query frame. The distance is calculated as follows

$$\textbf{Distance} = \sum (f_{query}(I,j,h) - g_{cummulative}(I,j,h))^2$$

Where, f(I,j,h) is the h^{th} histogram value for (I , j) block of histogram for frame f.

$g_{cumulative}$ (I,j,h) is the h^{th} cumulative difference histogram value for (I , j) block of histogram for frame g.

Implementation

This project uses Python for implementation purposes. Third party libraries used are

1. OpenCV
2. NumPy

N bin Gray Scale Histogram

[IMP_CH_1] The n-bin gray scale histogram for each cell block of each frame in the given video is carried out as described below

a. Class VideoInformationExtracts is used to get all 8 x 8 frame blocks.
b. Class HistogramGenerator generates n-bin gray scale histogram for each frame block and returns the information in the form of list of tuples of the form <*frame_id, block_x_coord, block_y_coord, wavelet_comp_id, value*>
c. The information extracts is stored in a file with name format *video_filename_hist_n.hst.*

[IMP_CH_2] Note that, a video metadata is also added at the beginning of the file indicating the following information

a. Total number of frames in video
b. Number of frame blocks in x direction
c. Number of frame blocks in y direction
d. Number of gray scale bins present i.e. n

Discrete Cosine Transform

[IMP_CH_3] **Block Level DCT**

a. As described in the design, the implementation strategy is to read each frame from te video, treat them as a set of 8 x 8 blocks, apply 2D DCT per block and extract the number of significant frequency components requested by the user. This ensures that [REQ_7] and [REQ_8] are met.
b. This is carried out by using the class BlockLevel2DDCTExtractor in Task1b.py

Frame matching using Discrete Cosine Transform

[IMP_CH_4] Frame matching as described in the design is carried out as part of BlockLevel2DDCTSimilarityMatcher in SimilarityMatcher.py

Discrete Wavelet Transform

[IMP_CH_5] **Block Level DWT**

a. As described in the design, the implementation strategy is to read each frame from the video, treat them as a set of 8 x 8 blocks, apply 2D DWT per block and extract the number of significant wavelet components requested by the user. This ensures that [REQ_11] and [REQ_12] are met.
b. This is carried out by using the class BlockLevel2DDWTExtractor in Task1c.py

[IMP_CH_6] **Frame Level DWT**

a. As described in the design , the implementation strategy is to read each frame from the video, apply 2D DWT to the entire frame and extract the number of significant

wavelet components requested by the user. This ensures that [REQ_18] and [REQ_19] are met.

b. This is carried out by using the class FrameLevel2DDWTExtractor in Task2.py

[IMP_CH_7] The extracted outputs are output into a file with format as mentioned in [REQ_13] and [REQ_20]

[IMP_CH_8] The output filenames follow the formats mentioned in [REQ_14] and [REQ_21]

Frame matching using Discrete Wavelet Transform

[IMP_CH_9] Frame matching as mentioned in the design is carried out in different ways for block level DWT and frame level DWT as part of the classes BlockLevel2DDWTSimilarityMatcher and FrameLevel2DDWTSimilarityMatcher in SimilarityMatcher.py

Results

This chapter talks about how the results can be obtained and how the results compare with each other for each of the features discussed in the chapters prior to this. A few sample outputs are provided as part of this discussion. For all test results, please refer to the Test folder in the Project package.

The similarity matching results in this chapter illustrate how they meet [REQ_22] , [REQ_23] , [REQ_24] and [REQ_25]

[RES_1] N bin Gray Scale Histogram

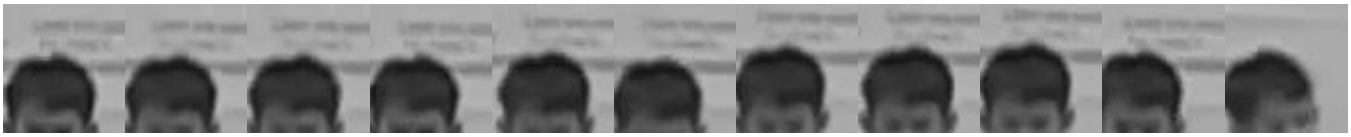
a. Run python file Task1.py
b. Provide the following inputs
    i. video file path = <folder_path_of_sample_videos>
    ii. video = R1.mp4
    iii. n = 4
    iv. Select option N bin gary scale histogram
c. The output file R2_hist_4.hst will be generated as shown in **Table 4**

264 8 8 4	[video metadata] <total_frames=264> <blocks_in_row=8> <blocks_in_col=8> <n=4>
0 0 0 0 0	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=0> <value=0>
0 0 0 1 0	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=1> <value=0>
0 0 0 2 64	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=2> <value=64>
0 0 0 3 0	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=3> <value=0>
0 0 1 0 0	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=0> <value=0>
0 0 1 1 0	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=1> <value=0>
0 0 1 2 64	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=2> <value=64>
0 0 1 3 0	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=3> <value=0>

Table 4 N Bin Gray Scale Histogram – Results

[RES_2] N bin Gray Scale Histogram Similarity Matching

a. Run python file Task3.py
b. Provide the following input
    i. video = R2.mp4
    ii. frameId = 150
    iii. n = 4
    iv. m = 0
c. The output consists of 10 most similar images based on n-bin gray scale histogram feature of frame blocks as shown in **Table 5**

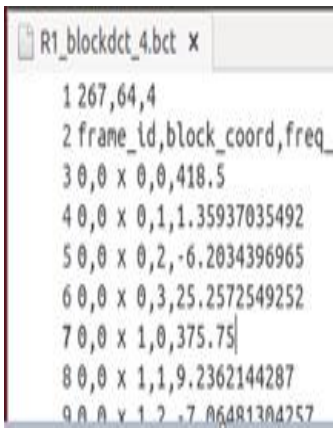


Query Frame	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	Rank 9	Rank 10

Table 5 N bin Gray Scale Histogram Similarity Matching – Results

[RES_3] Discrete Cosine Transform

a. Run python file Task1.py
b. Provide the following inputs
    i. video file path = <folder_path_of_sample_videos>
    ii. video = R1.mp4
    iii. n = 4
c. The output file R1_blockdct_4.bct will be generated as shown in **Table 6**



[video metadata] <total_frames=267><blocks_in_row=8><blocks_in_col=8><n=4>
<frameId=0><block_coord_x=0><block_coord_y=0><freq_comp_id=0><value=418.5>
<frameId=0><block_coord_x=0><block_coord_y=0><freq_comp_id=1><value=1.3593>
<frameId=0><block_coord_x=0><block_coord_y=0><freq_comp_id=2><value=-6.2034>
<frameId=0><block_coord_x=0><block_coord_y=0><freq_comp_id=3><value=25.2572>
<frameId=0><block_coord_x=0><block_coord_y=1><freq_comp_id=0><value=375.75>
<frameId=0><block_coord_x=0><block_coord_y=1><freq_comp_id=1><value=9.2362>
<frameId=0><block_coord_x=0><block_coord_y=1><freq_comp_id=2><value=-7.0648>

Table 6 Discrete Cosine Transform – Results

[RES_4] Discrete Cosine Transform Similarity Matching

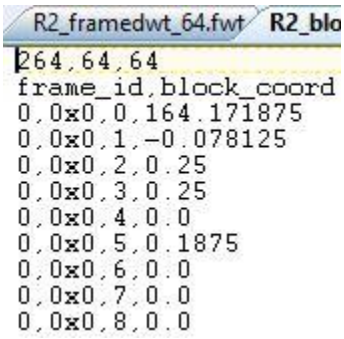
a. Run python file Task3.py
b. Provide the following inputs
    i. video = R1.mp4
    ii. frameId = 80
    iii. n =3
    iv. m = 0
c. The output consists of 10 most similar images based on n-bin gray scale histogram feature of frame blocks as shown in **Table 7**

Query Frame	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	Rank 9	Rank 10

Table 7 Discrete Cosine Transform Similarity Matching – Results

[RES_5] Discrete Wavelet Transform – Block Level

a. Run python file Task1.py
b. Provide the following inputs
    i. Video = R2.mp4
    ii. n = 64
c. The output file R2_blockdwt_64.bwt will be generated as shown in **Table 8**
    i. This ensures that it meets [REQ_13] and [REQ_14]



[video metadata] <total_frames=264> <blocks_per_frame=64> <n=64>

Frame No	Block X	x	Block Y	Wavelet Component ID	Value
0	0	x	0	0	164.171875
0	0	x	0	1	-0078125
0	0	x	0	2	0.25
0	0	x	0	3	0.25
0	0	x	0	4	0.0
0	0	x	0	5	0.1875
0	0	x	0	6	0.0
0	0	x	0	7	0.0
0	0	x	0	8	0.0

Table 8 Discrete Wavelet Transform Block Level – Results

[RES_6] Discrete Wavelet Transform – Block Level Similarity Matching

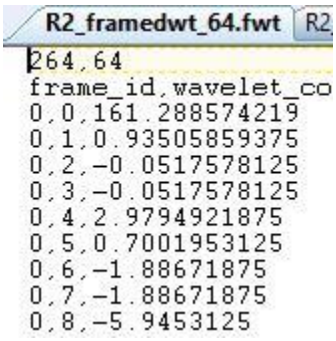
a. Run python file Task3.py
b. Provide the following inputs
    i. video = R2.mp4
    ii. frameId = 150
    iii. n = 64
    iv. m = 64
c. The output consists of 10 most similar images based on discrete wavelet transformation per block of each frame in the video as shown in **Table 9**

Query Frame	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	Rank 9	Rank 10

Table 9 Discrete Wavelet Transform – Block Level Similarity Matching – Results

[RES_7] Discrete Wavelet Transform – Frame Level

a. Run python file Task1.py
b. Provide the following inputs
    i. Video = R2.mp4
    ii. n = 64
c. The output file R2_framedwt_64.fwt will be generated as shown in **Table 10**
    i. This ensures that it meets [REQ_19] and [REQ_20]



[video metadata]	<total_frames=264>	<n=64>
Frame No	**Wavelet Component ID**	**Value**
0	0	161.288574219
0	1	0.93505859375
0	2	-0.0517578125
0	3	-0.0517578125
0	4	2.9794921875
0	5	0.7001953125
0	6	-1.88671875
0	7	-1.88671875
0	8	-5.9453125

Table 10 Discrete Wavelet Transform – Frame Level – Results

[RES_8] Discrete Wavelet Transform – Frame Level Similarity Matching

a. Run python file Task3.py
b. Provide the following inputs
   i. video = R2.mp4
   ii. frameId = 150
   iii. n = 64
   iv. m = 64
c. The output consists of 10 most similar images based on discrete wavelet transformation of each frame in the video as shown in **Table 11**

Query Frame	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	Rank 9	Rank 10

Table 11 Discrete Wavelet Transform – Frame Level Similarity Matching – Results

[RES_9] N bin Difference Gray Scale Histogram

a. Run python file Task1.py
b. Provide the following inputs
   i. video file path = <folder_path_of_sample_videos>
   ii. video = R1.mp4
   iii. n = 4
   iv. Select option N bin difference gray scale histogram.
c. The output file R2_hist_4.hst will be generated as shown in **Table 12**

264 8 8 4	[video metadata] <total_frames=264> <blocks_in_row=8> <blocks_in_col=8> <n=4>
0 0 0 0 0	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=0> <value=0>
0 0 0 1 0	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=1> <value=0>
0 0 0 2 64	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=2> <value=64>
0 0 0 3 0	<frameId=0> <block_coord_x=0> <block_coord_y=0> <bin=3> <value=0>
0 0 1 0 0	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=0> <value=0>
0 0 1 1 0	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=1> <value=0>
0 0 1 2 64	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=2> <value=64>
0 0 1 3 0	<frameId=0> <block_coord_x=0> <block_coord_y=1> <bin=3> <value=0>

Table 12 N Bin Difference Gray Scale Histogram – Results

[RES_10] N bin Difference Gray Scale Histogram Similarity Matching

a. Run python file Task3.py
b. Provide the following inputs
   i. video = R3.mp4
   ii. frameId = 150
   iii. n = 4
   iv. m = 0
c. The output consists of 10 most similar images based on n-bin gray scale histogram feature of frame blocks as shown in **Table 13**

Query Frame	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	Rank 9	Rank 10

Table 13 N Bin Difference Gray Scale Histogram Similarity Matching – Results

Conclusion

As part of this project, different feature extraction algorithms were implemented and tested on small videos. A few key observations that we've made as a result of this project are described below

1. We have been successful in comparing images based on all the features extracted.
2. We have seen that with increase in the number of significant components extracted, we see better matching results being obtained.
3. By transforming images, we've seen that the features of an image can be represented purely numerically with only a few pixels which is great because it gives us a mechanism to search for images from an image database.

Further enhancements

This section talks about a few enhancements that could be performed.

1. **Frame Dimensions**
    a. We have assumed that the videos would contain frames that
        i. Can be divided into 8 x 8 blocks.
        ii. Dimensions of a frame are of the order of a power of 2.
    b. A more complete approach would be to perform these transformation by padding appropriate pixel values if the frame dimensions don't meet the requirement.
2. **Comparison Metrics**
    a. Some of the comparison metrics have been carried out using references [3][4]. A more detailed analysis on these metrics would give us better comparison results.

Bibliography

[1] Ze-Nian Li, Mark S. Drew, Jiangchuan Liu, in *Fundamentals of Multimedia*

[2] "Python documentation for ndarray"
http://docs.scipy.org/doc/numpy-1.10.0/reference/arrays.ndarray.html

[3] Chu-Hui Lee and Zheng-Wei Zhou, *Comparison of Image Fusion based on DCT-STD and DWT-STD*

[4] "Similarity Measure: Cosine Similarity or Euclidean Distance or Both"
http://semanticvoid.com/blog/2007/02/23/similarity-measure-cosine-similarity-or-euclidean-distance-or-both/