Lab 4: Motion Compensation and CBIR

Rami Saad
Section 4
Toronto, Canada
rami.saad@ryerson.ca

***Abstract* - In this lab the effects of Motion Compensation and Content-Based Image Retrieval (CBIR) was observed. When comparing a frame difference, the predicted frame had a smaller entropy compared to the direct frame difference, most DPCM modes performed better when compared to the direct frame difference. In part II the Manhattan city block and Euclidean distance returned images that were visually similar but the Histogram Intersection returned images that were very dark in color.**

Motion Vector; Manhattan distance; Euclidean distance; Histogram Intersection; CBIR;

I. Introduction

The purpose of this lab was to understand the principles of Motion Compensation and CBIR. The effects of computing motion vectors using Mean Absolute Difference (MAD) to compare the resultant predicted frame against a direct frame difference was observed. Furthermore, using a suite of gathered images the effects of comparing city block, Euclidean distance, and histogram intersection was observed.

The lab was primarily conducted using Matlab. Matlab has a suite of built in function such as sort(), abs() and sum() were to simply the amount of code that needed to be written. Other in built functions such as imcomplement(), VideoReader() and rbg2gray were also used when handling and operating on image and video data.

II. Theory

In part 1 of this lab, two frames of a video were captured. The second frame was divided into 16x16 blocks and each of these blocks had a co-located block in the first frame with a search radius of 7. Then in each of these search regions a sequential search was performed. The formula for the number of operation necessary is provided below.

$$Operations = (2p+1)^2 \cdot N^2 * 3 \tag{1}$$

In the equation above, the $(2p+1)^2$ denotes the amount of positions that need to be evaluated in the co-located search region, N denotes the block size, it is multiplied by 3 because there are 3 operation MAD(i,j) used in the motion vector computation, subtraction, absolute value, and addition[1].

The calculation for the Mean Absolute difference is provided below.

$$MAD(i,j,p,q) = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n}|C_{n+1}[p,q] - C_n[p+i,q+j]|}{mn} \tag{2}$$

In the equation above, m and n refer to the size of the block, i and j are positions bounded by the search window [-p,p], C_{n+1} is the current frame and C_n is the previous frame.

For Content-based Image Retrieval the images are first converted from RGB to HSV and then quantized. Next the histograms are calculated and Each bin from one image is compared against the same bin from another image using one of three algorithms. These algorithms are described below.

$$Manhatatan, d_{L1}(h,g) = \sum_i |h(i) - g(i)| \tag{3}$$

$$Euclidean, d_{L2}(h,g) = \sum_i (h(i) - g(i))^2 \tag{4}$$

$$Euclidean, d_{L3}(h,g) = \frac{\sum_i min(h(i), g(i))}{min(|h|, |g|)} \tag{5}$$

III. Methodology

Motion Vector Computation

Subsection (1)

1. The "vipboard.mp4" video file was read into the Matlab workspace using the VideoReader() built in function.

Subsection (2)

1. Using the readFrame() built in Matlab function, 2 suitable frames (20_{th} and 21_{st} frames) from the video were stored into a variable named frames.

Subsection (3)

1. A function named blockByBlock16 was created which takes in 3 input arguments: frame, row, and col. They are a frame of the image, a row position the block belongs on in 2D space and a column position the block belongs on in 2D space respectively. The function has 1 output argument block which returns the 16x16 block.
2. The function returns a 16x16 block in the frame specified by the input row and col variables.
3. A function named blockByBlockSearch was created which take 4 input arguments: frame, row, col, and k. They are a frame of the image, row position the block belongs on in 2D space, a column position the block belongs on in 2D space, and the search radius respectively.
4. The function returns a block of varying size depending on if the co-located block is on the edge of the image or in the middle. This block contains all the values in the search radius.

Subsection (4)

1. A function named computeMotionVec was created which takes in 5 input arguments framePrev, frameCurr, p, q, and k. These input arguments represent the previous frame, current frame, the row position that the block belongs to in 2D space, the column position the block belongs to in 2D space and the search radius, respectively.
2. The function has 3 output variables dy, dx, and bestMatch which are the dy and dx motion vectors and the entire block characterized by those motion vectors.
3. The function first utilizes blockByBlock16 to get the current frame's block and blockByBlockSearch to get the full search area of the previous frame.
4. Then a sequential search is applied on all positions of the search area using a MAD calculation described in equation (2).
5. The function stores the dx and dy motion vectors of the smallest MAD calculation into their respective output variables as well as the entire block into the bestMatch variable which is used for debugging.

Subsection (5)

1. The motion vector computation was applied to every block position using the first two frames stored in subsection (2).
2. The motion vectors were used to construct the predicted frame block by block using the first frame of the video.
3. The bestMatch variable was used to verify that the correct block was being retrieved using the motion vectors.
4. The entropy of a direct frame difference was calculated using the myEntropy function from lab 3.
5. The entropy of a frame difference with the predicted block was calculated.
6. The entropy of DPCM modes 1, 2, 3, and 4 with the predicted frame as a predictor was calculated using the DPCM code from lab 3.

Content Based Retrieval

Subsection (1)

1. 15 images were loaded into the Matlab workspace using the imread() built in function.

Subsection (2)

1. A function named quantHSV was created has 1 input argument imageRGB and 1 output argument outHSV.
2. the function first converts the input image to HSV color space using the built in function rgb2hsv()
3. The image was then split into its three distinct channels.
4. A function named quant30bin was created that takes in 1 input argument named inputArr and has 1 output argument outputArr.
5. The H channel is passed into this function which is then quantized into discreet 30 bins.
6. Another function named quant15bin was created which is similar two quant30bin but quantizes the input into 15 discreet bins instead.
7. The S and V channels were passed into this function.
8. Finally the channels are combined back together and a quantized HSV image is returned from the quantHSV function
9. Steps 1-8 were repeated for the remaining 14 images.
10. A function named histData was created which takes 1 input argument inputImage and has one output argument freqData.
11. This function creates a weighted full color histogram of the input image and stores it in the output variable freqData.
12. All 15 images are passed through the histData function and there full color weighted histograms are stored.

Subsection (3)

1. A function named cityBlock was created which calculates the Manhattan distance between two images.
2. Image 1 was then compared two each of the other images and an ordered list from best to worse was created.
3. A function named euclidean was created which calculates the euclidean distance between images.
4. Image 1 was then compared two each of the other images and an ordered list from best to worse was created.
5. A function named histIntersection was created which calculates the histogram intersection between two images.
6. Image 1 was then compared two each of the other images and an ordered list from best to worse was created.

Subsection (4)

1. A function named histDataBy8 was created which takes 1 input argument inputImage and has one output argument freqData.
2. The function splits the input image into 8 discreet equal blocks then creates a weighted full color histogram of each of the blocks and stores it in the output variable freqData.
3. All 15 images are passed through the histDataBy8 function and their full color weighted histograms are stored.
4. This data was then passed through the cityBlock, euclidean, and histIntersection functions and Image 1 was then compared two each of the other images and an ordered list from best to worse was created for each of those functions.

IV. Results

Motion Vector Computation

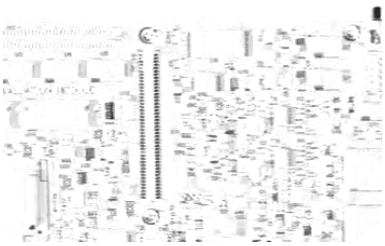

Figure 1: Direct Frame Difference

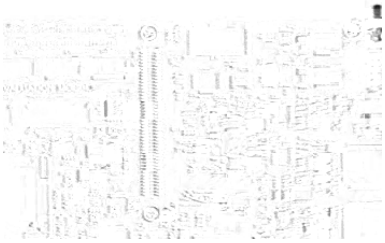

Figure 2: Predicted Frame Difference

Table 1: Entropy of Coding Methods using Predicted Frames

	Direct FD	Predicted FD	DPCM 1	DPCM 2	DPCM 3	DPCM 4
Entropy	3.4585	2.6917	2.8851	2.9943	3.1253	4.5525

Content Based Retrieval

Figure 3: Image 1

Figure 4: City Block and Euclidean, and Histogram Intersection Best Match

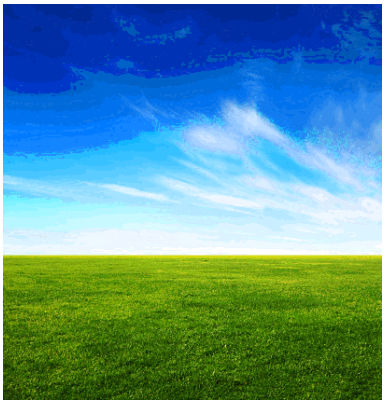

Figure 5: City Block, Euclidean and Histogram Intersection Best Match (8 Blocks)

V. Discussion

In part one, the benefits of using motion vectors to encode a video was explored. As can be seen in Table 1 every method to encode a video using a motion vector except for DPCM 4 performed better than a direct frame difference. The predicted frame difference performed the best overall. This can be attributed to the fact that the frame difference does not need to store the first row/column data like the other DPCM modes. Next mode one performed the best out of all of the DPCM modes. One reason for this is that the video in question simply pans from left two right in a smooth motion so there is a horizontal change between frames. Modes 2 and 3 also performed better than the direct frame difference. Mode 4 is the exception which actually performed worse. This result can be explain by the fact that the motion vectors on average were the most correct position for most pixels in a given block, so the adjustment made by using many pixels around the co-located pixel produced a worse than average result.

In part two, 15 images were quantized and converted to HSV color space. Images were compared to each other using the three distance metrics. In the case of Manhattan distance, Euclidean distance and Histogram Intersection a Same image flipped vertically performed the best. This was expected because the the Histogram would be exactly the same. For the case of the Histograms where the image is split into 8 discreet blocks and the same measures of distance another similar image seen in figure 5 actually performed better. In this case, the Histogram was computed more locally and because the images have similar colors located in similar areas of the image. The second field image performed better even if the prior full histogram over the entire image is the same.

VI. Conclusion

In this lab, the effects of motion vectors encoding with respect to entropy and CBIR with respect to finding similar images was observed. For the most part, encoding a video with motion vectors actually performed better than taking a direct frame difference. With CBIR do a more local search actually provided a more similar image because generally similar colors would be located in the same areas. This could be seen with the similarities between figures 3 and 5. Just because an image shared the same or similar color histogram over the entire image does not mean they look more similar because those colors could be placed in any way across the image.

References

[1] "Lab 4: Motion Compensation and CBIR," D2L, 2020.

[2] P. Ze-Nian Li and Mark S. Drew Fundamentals of Multimedia. Boston, MA: Course Technology Cengage Learning, 2014.