CompE 565: Multimedia Communication Systems

Project 3: Motion Estimation for Video Compression

Due Date: Saturday, April 12, 2014 at 11:00 pm

Learning Goals: Learn Motion Estimation techniques for reducing the temporal redundancy in video sequences.

Project Description:

- Design and implement the full search motion estimation and at least one fast motion estimation method (preferably 'conjugate directions search') for video compression.
- Compare the computational load as well as the quality of both the motion estimation algorithms.

Use a sequence of first 5 frames of the given motion video 'walk_qcif.avi.

Your report should include the following:
- A thorough discussion of your implementation, the simulation results, the error images, and the video images reconstructed by using motion compensated macroblocks.
- You should also provide a visual representation of motion vectors.
- The comparison of the full search motion estimation and a fast motion estimation method of your choice must include MAD or MSE values and computational load (number of multiplications, additions, comparison).

Hint: Matlab has a function that may be used to display motion vectors. In Matlab command, type "help quiver" to learn more about this command.

Motion Estimation

Motion estimation over a macroblock of pixels is a standard approach for estimating motion in a moving image sequence. Four important parameters in motion estimation are: the macroblock size, matching criteria, search technique, and prediction mode.

Macroblock size in most cases is assumed to be 16x16 although other block sizes could also be used. Macroblock size is very important in accuracy and speed of the motion estimation operation. A smaller macroblock size will increase the side information generated, whereas a larger macroblock size may result in lower accuracy.

Recall that for a given displacement $\mathbf{d} = (d_x, d_y)$, the Mean Square Error (MSE) distortion for a macroblock $B_k(n)$ is given by:

$$Distortion(B_k(n),d) \quad = \quad \sum_{\forall (x,y) \in B_k(n)} (U_k(x,y) - U_r(x+d_x, y+d_y))^2 \qquad \ldots\textbf{(1)}$$

For Mean Absolute Difference (MAD), the distortion for a macroblock $B_k(n)$ is given by:

$$Distortion(B_k(n),d) \quad = \quad \sum_{\forall (x,y) \in B_k(n)} \left| U_k(x,y) - U_r(x+d_x, y+d_y) \right| \qquad \ldots\textbf{(2)}$$

where, for both cases, $U_k(x,y)$ represents pixels in the current frame, and $U_r(x,y)$ represents pixels in the reference frame.

For a macroblock size of 8x8, MSE requires 1x64 = 64 multiplications and 3x64 = 192 additions. On the other hand, MAD only requires 2x64 = 128 additions, which is a substantial reduction as compared to MSE. Furthermore, MAD performs almost as well as MSE in terms of reducing entropy.

(a) Exhaustive Search Block Matching Algorithm:
In *Exhaustive Search* (ES), every possible displacement within a rectangular search window is attempted. The displacement that produces the minimum distortion is chosen as the motion vector. As shown in Figure 1, ff the maximum search range in either direction is *w* (assuming a square search), $(2w+1)^2$ possible values exist for the motion vector.

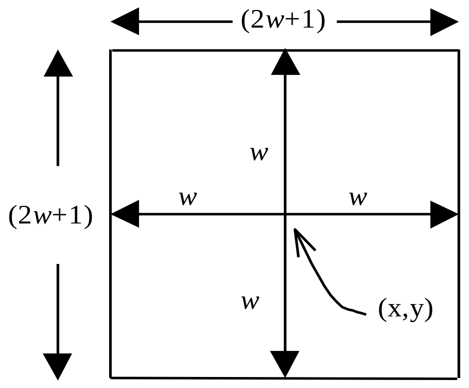

Figure 1 Motion Search Range

For ES, the distortion measure must therefore be calculated and compared $(2w+1)^2$ times. The resulting motion vector will be the one that minimizes the distortion within the search range. The cost of minimizing the distortion in ES is high computational intensity. For example, for a maximum displacement of w = 6, the matching criteria must be evaluated $(2\text{x}6+1)^2$ = 169 times. If MAD is chosen as the distortion criteria, and the macroblock size is 8x8, each macroblock requires 169x128 = 21,632 additions and 169 comparisons. For an image size of 352x288, the 1,584 macroblock motion vectors need to be calculated (assuming no motion detection). Therefore, a total of: 21,632x1,584 = 34,265,088 additions and 169x1,584 = 267,696 comparisons need to be performed for each frame.

If we assume a frame rate of 30 frames/second, there are over 1 G additions and 8 M comparisons per second, for a frame size of 352x288. The high computational requirements of ES make it unacceptable for many real time image sequence coding applications.

(b) Conjugate Directions Search:

An alternate algorithm referred to as Conjugate Directions Search (CDS) requires lesser computations than the full search algorithm. It is based on the assumption that the energy of the prediction error is monotonically decreasing towards the optimum motion vector in the search range, which is observed to be common in many fast search algorithms. Figure 2 shows an example of a Conjugate Directions Search. The pseudo-code of the CDS algorithm is shown in Figure 4.

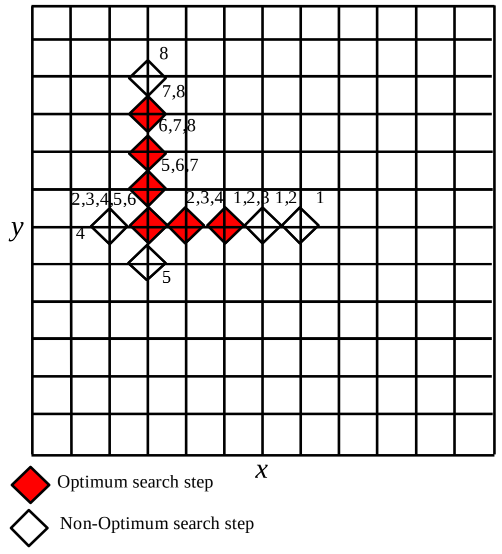

Figure 2 Conjugate Directions Search

CDS requires a maximum of 3+2*w* searches over the search range, where *w* is the maximum displacement. For a maximum displacement of w = 6, the matching criteria must be evaluated 3+2x6 = 15 times. For MAD as the distortion criteria and 8x8 block size, each macroblock requires 15x128 = 1,920 additions and 15 comparisons. This is significantly lower than the 21,632 additions and 169 comparisons required by the ES algorithm. For an image of size 352x288, the 1920x1584 = 3,041,280 additions and 15x1584 = 23,760 comparisons need to be performed for each frame.

If we assume a frame rate of 30 frames/second, we have over 91 M additions/second and fewer then 713 k comparisons/second.

While the CDS algorithm significantly reduces the computational complexity as compared to the ES algorithm, it does not always find the optimal motion vector within the search range - it may get stuck in a local minimum of the prediction error energy.

© Modified Logarithmic Search:

Another efficient fast search algorithm is the Modified Logarithmic Search (MLS). Figure 3 shows an example of a Modified Logarithmic Search. The pseudo-code for the MLS algorithm is given in Figure 5.

The MLS algorithm is very efficient; it has been shown that it requires a maximum of $2 + 7\log_2(w)$ searches, where w is again used as the search range. For a maximum displacement of w = 6, the matching criteria must be evaluated $2 + 7\log_2(6) = 20$ times.

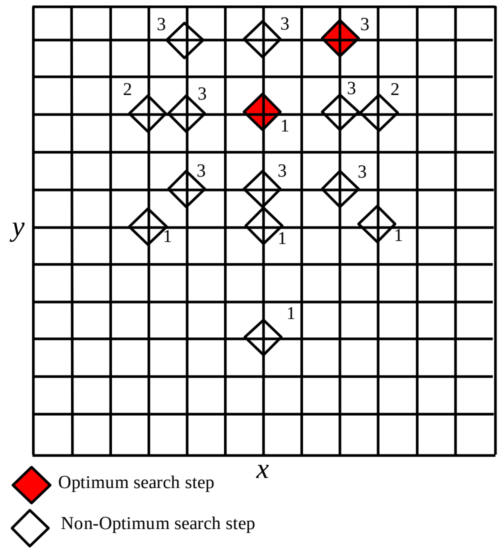

Figure 3 Modified Logarithmic Search

For MAD as the distortion criteria, each 8x8 block requires 20x128 = 2,560 additions and 20 comparisons. For an image of size 288x352, the 2,560x1,584 = 4,055,040 additions and 20x1,584 = 31,680 comparisons need to be performed for each frame. If we assume a frame rate of 30 frames/second, we have over 121M additions/second and 950K comparisons/second.

MLS is unable to search all of the locations at the boundaries of the search window, hence it does not always result in the optimum motion vector within the search window. However, its performance is very good for smaller displacements.

```
Conjugate Directions Search Algorithm

1.    Initialize $d_{cent} = (0,0)^t$;
2.    Initialize minmad = $MAD_k(B, d_{cent})$;
3.    Evaluate MAD at vectors $d_{cent}$ +/- $(1, 0)^t$;
4.    Set minmad to minimum MAD.
5.    If minmad is not achieved at $d_{cent};$
          set $d_{cent}$ to minimizing vector.
          go to 3.
6.    Evaluate MAD at vectors $d_{cent}$ +/- $(0, 1)^t$;
7.    Set minmad to minimum MAD.
8.    If minmad is not achieved at $d_{cent};$
          set $d_{cent}$ to minimizing vector.
          go to 6.

MAD is the mean absolute difference. d is the motion vector.B is the block.
```

Figure 4: Conjugate Directions Search Algorithm

```
Modified Logarithmic Search Algorithm

1.    Initialize    $d_{cent} = (0,0)^t$;
                    offset = $d_{MAX}/2$;
                    minmad = $MAD_k(B, d_{cent})$;

2.    Evaluate MAD at vectors $d_{cent}$ +/- $(offset, 0)^t$ and $d_{cent}$ +/- $(0, offset)^t$
3.    Set minmad to minimum MAD.
4.    If minmad is not achieved at $d_{cent}$
          Evaluate MAD at two of the four vectors $d_{cent} + (-/+offset, -/+offset)^t$
          which are closest to the minimizing vector.

5.    Set minmad to minimum MAD;
          set $d_{cent}$ to minimizing vector;
6.    offset = offset - 1.
7.    if offset > 0
          goto 2.

where:    MAD is the mean absolute difference.
          d is the motion vector.
          B is the block.
```

Figure 5: Modified Logarithmic Search Algorithm