Project 3 – Motion Estimation

1. Reading a Video file in MATLAB

```
video_frames = aviread('foreman_qcif.avi');
```

Alternatively you may try "help VideoReader" in newest Matlab version.

2. Accessing the component from the Frame

```
frame_ref = video_frames (<frame_number>).cdata(:,:,1);   //gives y component of #frame_number
```

Note: We perform motion estimation on the 1st 5 frames of the video file. Thus the trick is to consider the motion estimation only for the 1st 2 frames at first and then put the whole code into a 'for' loop reading the 5 frames and progressively making the current frame as the ref frame for the next step.

Hint1: Here the <frame_number> actually will be 1 to 5, so it would be included in the 'for' loop.
Also <u>do not</u> start coding by accessing the frames with numerical values like 1 and 2..
i.e.,

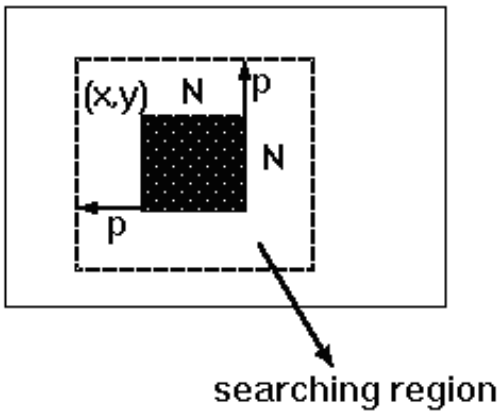
```
video_frames(1).cdata…   giving you the ref frame and
video_frames(2).cdata…  giving you the current frame…
```

instead do this..

```
for n=1:1
video_frames(n).cdata…   giving you the ref frame and
video_frames(n+1).cdata…  giving you the current frame…
end
```

and when u are done with the 1st 2 frames, change it to for n=1:4.

Block Search Tips:



Remember: The big box represents the search region (search window of 32 x 32) from the reference (i.e., previous) frame and the shaded box (16 x 16) represents the macroblock for the current frame. The basic principle of any search algorithm is to find the exact match of this 16 x 16 block in the 32 x 32 block of ref frame.
Very Important: Note the location of the search window. It should be such that it surrounds the 16x16 block from current frame equally. (see figure)

Hint2: We first fix the position of the 16x16 block in the current frame and then go 8 pixels away in the ref frame in each direction to get our window in which we have to search. This implies that in any search algorithm, we decide our window depending on where our block is in the current frame.

Hint3: Create a matrix for storing the residues

```
r_block=zeros(mb,mb);
```

And check the value that has the minimum residue value and proceed for further calculations.

Hint4: You should create a loop in steps of your macroblock size in a frame, and search the minimum MAD value block that matches in the frames; search area should be your window size.

Please see the figure carefully and read below.

Important: This window size cannot remain 32x32 when the 16x16 macroblock is on the edges of the frame. In that case just remember the basic principle to go 8 pixels in each **allowable** direction (within the video frame0 and search in that region.

Calculating the MADs

```
difference_matrix = macroblock in ref frame – macroblock in current frame;
%Computing the Mean Absolute Difference
diff(nn) = abs(sum(difference_matrix (:)));
```

Here diff(nn) is the array in which all the MADs for a given macroblock are stored. Please make a note of the way MAD is calculated: Abs(sum(…)

To find the minimum MAD, use the min command in Matlab.

The block having the minimum MAD is the best match and motion vector is calculated wrt to it.

Motion Vector Representation

The motion vector represents the displacement of the 1st pixel of current 16x16 macroblock with respect to the 1st pixel of 16x16 macroblock in the reference frame. *Remember Hint1 here too. It is very important that you follow it throughout the project*.

The 'quiver' command requires the start and end co-ordinates of the motion vector. Thus the trick is to find the motion vector for each 16x16 block searched and store it in 4 arrays such that we form an array each for x1, y1, x2, y2. Just pass these arrays into the quiver command and you get your motion vectors in a pictorial form.
Do look into the quiver command help.

Example:

```
figure()
quiver(searchwindow(:,2,1), searchwindow(:,1,1), searchwindow(:,2,2), searchwindow(:,1,2)); title(['motion vector for image ',num2str(frames),' and image ',num2str(frames+1)] );
```

Here search window(:,2,1), search window(:,1,1) , search window(:,2,2), and search window(:,1,2) correspond to x1,y1,x2,y2, respectively, in the above paragraph.

Note: 'quiver' command expects the **position** of the pixel and NOT its values…so store the positions correctly. The positions will be all unique values depending on where you currently are in the frame.