Lab Assignment
Motion Compensated Frame Interpolation
June 29th, 2008

I. Frame Down-rate Conversion:

Purpose: write the Matlab file downrate.m to down-rate the YUV video sequence from 30fps to 15fps to see the jerky artifacts.

I.1. Open the YUV file Football with SIF size 352x240, 250 frames using the following code:

```
clear all
close all

filename='fb';
Framestart=0;
Framestop=124;
Frame_height=240;
Frame_width=352;
% read the data from YUV file
fileId=fopen(strcat('databasefolder\',filename,'.yuv'));

% create the Original and Downrate YUV files
fileId1=fopen(strcat(' databasefolder \',filename,'Original.yuv'),'w');
fileId2=fopen(strcat(' databasefolder \',filename,'Downrate.yuv'),'w');
```

I.2. Downrate the video sequence:

```
% start reading from Frame No. Framestart
Frame_size=1.5*Frame_height*Frame_width; %YUV 4:2:0
fseek(fileId, Framestart*Frame_size, 'bof');

% process each frame
for Frame_index=Framestart:Framestop

  Frame_index
  subSampleMat = [1, 1; 1, 1];
  % read Y component
  Y = fread(fileId, Frame_width * Frame_height, 'uchar');
```

```
    % write Y component
    count = fwrite(fileId1, Y, 'uchar');

    % read U component
    U = fread(fileId, Frame_width / 2 * Frame_height / 2, 'uchar');
    % write U component
    count = fwrite(fileId1, U, 'uchar');

    % read V component
    V = fread(fileId, Frame_width / 2 * Frame_height / 2, 'uchar');
    % write V component
    count = fwrite(fileId1, V, 'uchar');

    % for even frame, write to downrate yuv file
    if mod(Frame_index,2)==0
      % write Y component
      count = fwrite(fileId2, Y, 'uchar');
      % write U component
      count = fwrite(fileId2, U, 'uchar');
      % write V component
      count = fwrite(fileId2, V, 'uchar');
    end
  end

  fclose(fileId);
  fclose(fileId1);
  fclose(fileId2);
```

I.3. Play the down-rated video sequences:

- Install the YUV File Player using the file YuvFilePlayer.exe
- Open the file fbOriginal.yuv file with Width=352, Height=240, Format 4:2:0, Frame rate 30, Header 0 bytes.
- Open the file fbDownrate.yuv file with Width=352, Height=240, Format 4:2:0, Frame rate 15, Header 0 bytes.
- Compare the jerkiness of these 2 YUV files.
-

II. Frame repetition and frame averaging:

Purpose: frame rate up-convert the YUV video sequence from 30fps to 60fps using frame repetition and frame averaging.

II.1. Open the file MCFI.m, this file includes the code to open the original video sequence. The main idea is that the output even frame of the frame up-converted sequence is always copied from the corresponding even frame YUV0 from the original sequence, while the output odd frame YUV1 will be the same as YUV0 (for frame repetition) or the average between YUV0 and its future frame YUV2.

For frame repetition:

$$YUV1(t)=\begin{cases} YUV0(t)=YUV\left(\frac{t}{2}\right) & if \quad \mathrm{mod}(t,2)=0 \\ YUV0(t)=YUV\left(\frac{t-1}{2}\right) & if \quad \mathrm{mod}(t,2)\neq 0 \end{cases}$$

For frame averaging:

$$YUV1(t)=\begin{cases} YUV0(t)=YUV\left(\frac{t}{2}\right) & if \quad \mathrm{mod}(t,2)=0 \\ 0.5\times\left(YUV0(t)+YUV2(t)\right)=0.5\times\left(YUV\left(\frac{t-1}{2}\right)+YUV\left(\frac{t+1}{2}\right)\right) & if \quad \mathrm{mod}(t,2)\neq 0 \end{cases}$$

where YUV is the input video sequence.

II.2. Uncomment the code for frame repetition and change the output frame rate up converted video sequence to fb_Uprate_Repetion.yuv. Run these code. Play the output YUV file using YUV File Player with frame rate 60fps. Is its quality the same with the original YUV file when played at frame rate 30fps?

II.3. Comment the code for frame repetition**,** uncomment the code for frame averaging and change the output frame rate up converted video sequence to fb_Uprate_Averaging.yuv. Run these code.

III. Bi-lateral MCFI:

Purpose: frame rate up-convert the YUV video sequence from 30fps to 60fps using bi-lateral MCFI and justify the effectiveness of MV correction and MV refinement.

III.1. Full-search ME: use the Matlab file MCFI.m.

- Comment the code for frame averaging and frame repetiotion**,** and change the output frame rate up converted video sequence to fb_Uprate_Bilateral.yuv. Set Framestart=3 and Framestop=3 so that the algorithm will only run for one frame.
- Find the MV by full search ME with the function ME_Fullsearch_Bilateral with NBlocksize=8 and search_range=8. Its syntax is:

[YUV1 v_y v_x SBAD_min]=ME_Fullsearch_Bilateral(YUV0,YUV2, Frame_height,Frame_width,NBlocksize,search_range);

- Show the MV field by using the following code.

```
%show the vector plane
[x,y] = meshgrid(1:Frame_width/NBlocksize,1:Frame_height/NBlocksize);
```

```
% flip the Y for imshow
for j=1:Frame_width/NBlocksize
   for i=1:Frame_height/NBlocksize
      v_y1(i,j)=v_y(Frame_height/NBlocksize+1-i,j);
      v_x1(i,j)=v_x(Frame_height/NBlocksize+1-i,j);
   end
end
quiver(x,y,10*v_x1,10*v_y1)
```

- Run the code to see the MV field. Does it show the true motion of the sequence?
- Use the tic and toc function to estimate the running time for ME.
- Change Framestart=0 and Framestop=50 so that the algorithm will run a longer sequence. Play the output YUV file. Does it have some artifacts?
- Change back the Framestart and Framestop value to 3.
- Correct the MV field using the function MV_Correction. Its syntax is:

[YUV1 v_y v_x SBAD_min]=MV_Correction(YUV0,YUV1,YUV2, v_y, v_x, SBAD_min,Frame_height,Frame_width,NBlocksize,search_range);

- Run and show the correct MV field. Does it more correctly show the true motion of the sequence?
- Refine the MV field using the function MV_Refine with syntax:

[YUV1]=MV_Refine(YUV0,YUV1,YUV2,v_y,v_x,Frame_height,Frame_width, NBlocksize,search_range);

- Change Framestart=0, Framestop=50 and the output frame rate up converted video sequence to fb_Uprate_Bilateral_MV_Processed.yuv. Run the algorithm. Play the output frame rate up-converted sequence and compare the quality between sequences with and without MV processing.

III.2. Three step search ME:

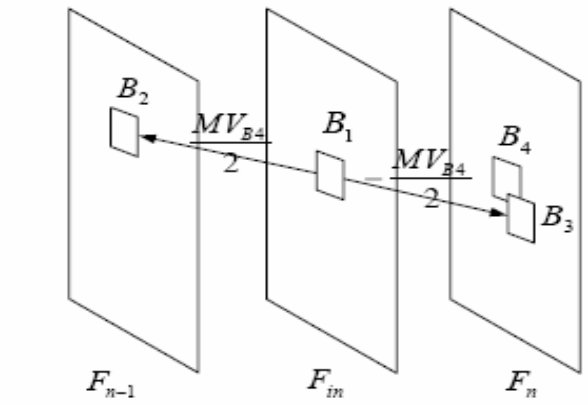
- Comment the code for function ME_Fullsearch_Bilateral**.** Set Framestart=3 and Framestop=3 so that the algorithm will only run for one frame.
- Apply the three step search ME by using the function ME_TSS_Bilateral with syntax:

[YUV1 v_y v_x SBAD_min]=ME_TSS_Bilateral(YUV0,YUV2,Frame_height, Frame_width,NBlocksize,search_range);

- Use the tic and toc function to estimate the running time of ME. Compare this running time with case of using full search. Show the MV field and compare to the resulted MV field in case of using full search. Which one more correctly show the true motion of the sequence?
- Use the tic and toc function to estimate the running time for ME, MV correction and MV refinement.

IV. Direct MCFI:

Purpose: use the available motion vectors from the coded bit stream to directly interpolate the intermediate frame using the direct MCFI algorithm:



$$F_{in}(B_1)=\frac{1}{2}(F_{n-1}(B_2)+F_n(B_3))=\frac{1}{2}(F_{n-1}(B_1-\frac{MV_{B_4}}{2})+F_n(B_1+\frac{MV_{B_4}}{2})$$

B1 is at the same location with B4 in current frame Fn.

IV.1 Direct MCFI without MV processing:

- Open the Matlab file MCFI_Direct.m. This file will at first read the MV field and MV type from the mv_0xx.bin and mp_type_0xx.bin file using the function Get_264MVF, then read the previous and current odd frame and finally using the function FRUC to frame interpolate the even frame. Set Video_length = 3 so that we can interpolate the 2nd frame from the 1st and 3rd frames.
- Use the following code to show the MV field read from the mv_0xx.bin file.

```
%show the vector plane
[x,y] = meshgrid(1:imx/8,1:imy/8);
% flip the Y for imshow
for j=1:imx/8
   for i=1:imy/8
      v_y1(i,j)=mvy8(imy/8+1-i,j);
      v_x1(i,j)=mvx8(imy/8+1-i,j);
   end
end

figure(1)
quiver(x,y,0.1*v_x1,0.1*v_y1)
```

- Use the imshow function to show the interpolated frame. Based on the MV field and its corresponding interpolated frame, point out the MVs which cause blocking artifacts.

IV.1 Direct MCFI with MV processing using Median Vector Filter:

- From the Matlab file MCFI_Direct.m, use the following code to process the MV field using the median vector filter:

```
%with MV processing by Vector Median Filter
    for i=2:blk_y-1                % 8x8
      for j=2:blk_x-1
        ntempx=newmx(i-1:i+1, j-1:j+1);
        ntempy=newmy(i-1:i+1, j-1:j+1);
        tempx=ntempx(:);
        tempy=ntempy(:);

        [nx, ny] = vmf(tempx, tempy);
        if nx~=100 || ny~=100
          new_vec(1,1) = nx;
          new_vec(1,2) = ny;
        end

        newmx(i,j)=new_vec(1,1);
        newmy(i,j)=new_vec(1,2);
      end
    end
```

- Then show the processed MV field. Save the interpolated frames in the folder \vmf. Compare with the un-processed MV field.
- Show both the interpolated frames using Direct MCFI with and without MV processing. Compare the visual quality of these 2 frames.
- Use the MV_Refine function to further reduce the blocking artifacts.