Digital Audio and Video Fundamentals

12^{th} week Project

Prepared By

Mohamed Gouda Ismail

Salma Ibrahim soliman

Ibrahim Ahmed Mahmoud

Nada Ahmed Zakaria

Under supervision

Dr. Wesam Atia Askar

Eng. Mohamed Abdalnaby Abdalfatah Alsayed

12th week Project

1) Objective

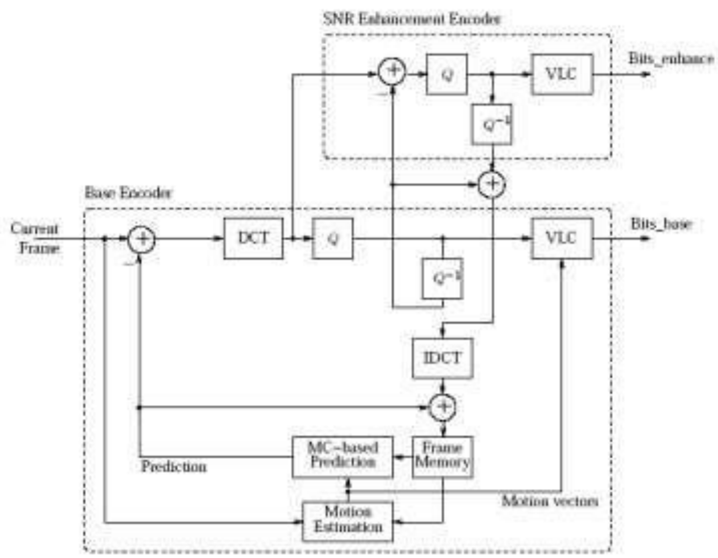
Implement a Matlab project that encode and decode a file or live stream video based on motion compensation approaches. You may choose the H.261, MPEG-1 or MPEG-2 compression standards. Also, at least one of the search methods taken in lectures (sequential, 2D logarithmic or hierarchal) should be used to detect the best match macroblocks and to compute the motion vectors.

2) Project Idea

We read a video in AVI file format. The MATLAB function videoReader will read the entire video. This project uses one base layer and one enhancement layer. Also, this project codes the luma component only. The first picture is intracoded and stored in the buffer. Subsequent pictures are predictive coded with a quantization step size of 16 and a quantizer scale of 4 for all the differential DCT coefficients. This forms the base layer. The difference between the actual DCT and the quantized/dequantized DCT in the base layer is then quantized with a quantization step of 4 for all the coefficients. This additional data is the enhancement layer. Both base and enhancement layers are variable length coded and transmitted or stored. For lower quality video, only the base layer is decoded. To obtain a higher quality video, both layers are decoded. It does not calculate the bit rate nor does it generate the VLC codes. The aim is to illustrate the idea of SNR scalability. However, the codes follow the P-picture coding and quantization rules of MPEG-2.

3) MPEG-2 SNR Scalability (Encoder)

The input video is in full resolution. It is lowpass filtered and down sampled to the base resolution and then encoded using MPEG2 scheme. The difference between the full resolution input video and the base layer decoded (locally) video is coded using the DCT transform. The two bit streams are transmitted or stored

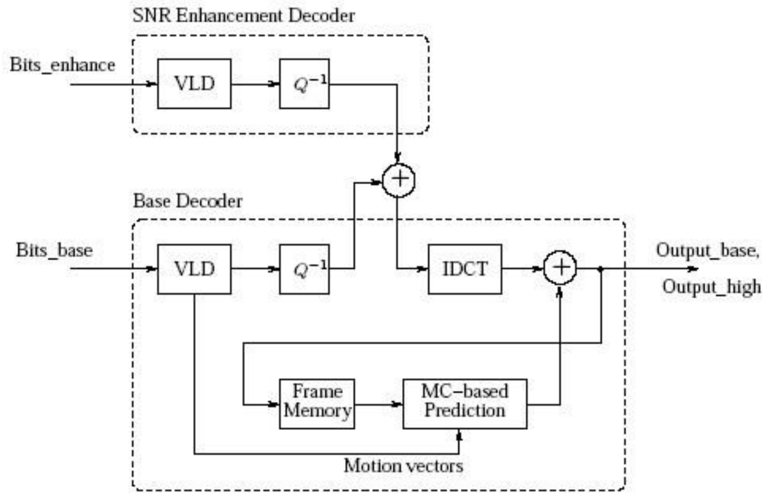


(a) Encoder

Fig 11.8 (a): MPEG-2 SNR Scalability (Encoder).

4) MPEG-2 SNR Scalability (Decoder)

On the decoder side (Figure 2), only the base layer is decoded to obtain the base resolution video. To obtain the full resolution video, both base and enhanced layers are decoded, the decoded base resolution pictures are upsampled and filtered, and the two are added. The quantizer scale may be the same in both layers and as a result the decompressed videos may have essentially the same quality but at different resolutions. It must be pointed out that spatial scalability can also be accomplished in the wavelet domain



(b) Decoder

Fig 11.8 (b): MPEG-2 SNR Scalability (Decoder).

5) Matlab Code for MPEG-2 Scalability

```
clear all; close
all;
N = 8;% block size is N x N pixels
N2 = 2*N;
W = 16; % search window size isWxW pixels
quantizer_scale = 4; % used only for the base layer
[chosenfile, chosenpath] = uigetfile('*.avi', 'Select a video');
if ~ischar(chosenfile)      return;    %user canceled dialog
end
  filename = fullfile(chosenpath, chosenfile);

frames = VideoReader(filename);
vidHeight=frames.Height; vidWidth=frames.Width;

F=read(frames,[1 20]);
M = struct('cdata',zeros(vidHeight,vidWidth,3,'uint8'),'colormap',[]);
k = 1; figure; subplot(2,2,1)
%hold on, title('original video'); for
k=1:15
M(k).cdata = F(:,:,:,k);
image(M(k).cdata);
```

```
pause(1/frames.FrameRate);
end

[X,Y,Z] = size(M(1).cdata); if
mod(X,8)~=0
    Height = floor(X/8)*8;
else      Height = X; end
if mod(Y,8)~=0     Width
= floor(Y/8)*8; else
Width = Y; end Depth =
Z; clear X Y Z %
if Depth == 3
   A = rgb2ycbcr(M(1).cdata);% Convert RGB to YCbCr & retain only Y
y_ref = A(:,:,1); else
A     = M(1).cdata;
y_ref = A; end
% pad the reference frame left & right and top & bottom y_ref
= double(padarray(y_ref,[W/2 W/2],'replicate'));
% arrays to store SNR and PSNR values
Base_snr = zeros(1,k-2); Enhanced_snr = zeros(1,k-2);
Base_psnr = zeros(1,k-2); Enhanced_psnr = zeros(1,k-2);
% Encode the monochrome video using MPC
for f = 2:k-1     if Depth == 3
B     =
rgb2ycbcr(M(f).cdata);
y_current = B(:,:,1);
else
      y_current = M(f).cdata;
end
   y_current = double(padarray(y_current,[W/2 W/2],'replicate'));
for r = N:N:Height       rblk = floor(r/N);       for c =
N:N:Width          cblk = floor(c/N);
         D = 1.0e+10;% initial city block distance          for u = -N:N
for v = -N:N              d=y_current(r+1:r+N,c+1:c+N)-
y_ref(r+u+1:r+u+N,c+v+1:c+v+N);              d = sum(abs(d(:)));% city
block distancebetween pixels              if d < D                D =
d;
                x1 = v; y1 = u; % motion vector
end
            end
end
% MC compensated difference coding          temp = y_current(r+1:r+N,c+1:c+N)-
y_ref(r+1+y1:r+y1+N,c+1+x1:c+x1+N);         TemP = dct2(temp); % DCT of
difference         s = sign(TemP); % extract the coefficient sign
TemP1 = s .*
round(abs(TemP)/(16*quantizer_scale))*(16*quantizer_scale); %
quantize/dequantize DCT
        temp = idct2(TemP1); % IDCT
        Base(r-N+1:r,c-N+1:c) = y_ref(r+1+y1:r+y1+N,c+1+x1:c+x1+N)+temp; %
reconstructed block - base quality
        delta_DCT = TemP - TemP1; % incremental DCT
        s1 = sign(delta_DCT); % extract the sign of incremental DCT
delta_DCT = s1 .* round(abs(delta_DCT)/4)*4;
        temp1 = idct2(TemP1 + delta_DCT);
        Enhanced(r-N+1:r,c-N+1:c) = y_ref(r+1+y1:r+y1+N,c+1+x1:c+x1+N)
+temp1;
end   end
```

```
% Calculate the respective SNRs and PSNRs
    Base_snr(f-1) =
20*log10(std2(y_current(N+1:Height+N,N+1:Width+N))/std2(y_current(N+1:Height+
N,N+1:Width+N)-Base));
    Enhanced_snr(f-1) =
20*log10(std2(y_current(N+1:Height+N,N+1:Width+N))/std2(y_current(N+1:Height+
N,N+1:Width+N)-Enhanced));
    Base_psnr(f-1) = 20*log10(255/std2(y_current(N+1:Height+N,N+1:Width+N)-
Base));
    Enhanced_psnr(f-1) =
20*log10(255/std2(y_current(N+1:Height+N,N+1:Width+N)-Enhanced));
% replace previous frames by the currently reconstructed frames
y_ref = Base;      y_ref = double(padarray(y_ref,[W/2
W/2],'replicate')); end
FNO = int16(1:k); subplot(2,2,3),plot(FNO(2:end-
1),Base_snr,'k*','LineWidth',1), hold on %figure,plot(FNO(2:end-
1),Base_snr,'k*','LineWidth',1), hold on plot(FNO(2:end-
1),Enhanced_snr,'kd','LineWidth',2), title('SNR (dB)') % axis([M(2)
M(end) min(Base_snr)-2 max(Enhanced_snr)+2]) % for Rhinos sequence
legend('Base Quality','Enhanced Quality',0) xlabel('Frame
#'), ylabel('SNR (dB)'), hold off
  subplot(2,2,4),plot(FNO(2:end-1),Base_psnr,'k*','LineWidth',1), hold
on % figure,plot(FNO(2:end-1),Base_psnr,'k*','LineWidth',1), hold on
plot(FNO(2:end-1),Enhanced_psnr,'kd','LineWidth',2), title('PSNR (dB)')
% axis([F(2) F(end) min(Base_psnr)-2 max(Enhanced_psnr)+2]) % for Rhinos
sequence
legend('Base Quality','Enhanced Quality',0)
xlabel('Frame #'), ylabel('PSNR (dB)'), hold off
```

6) Results and Discussion

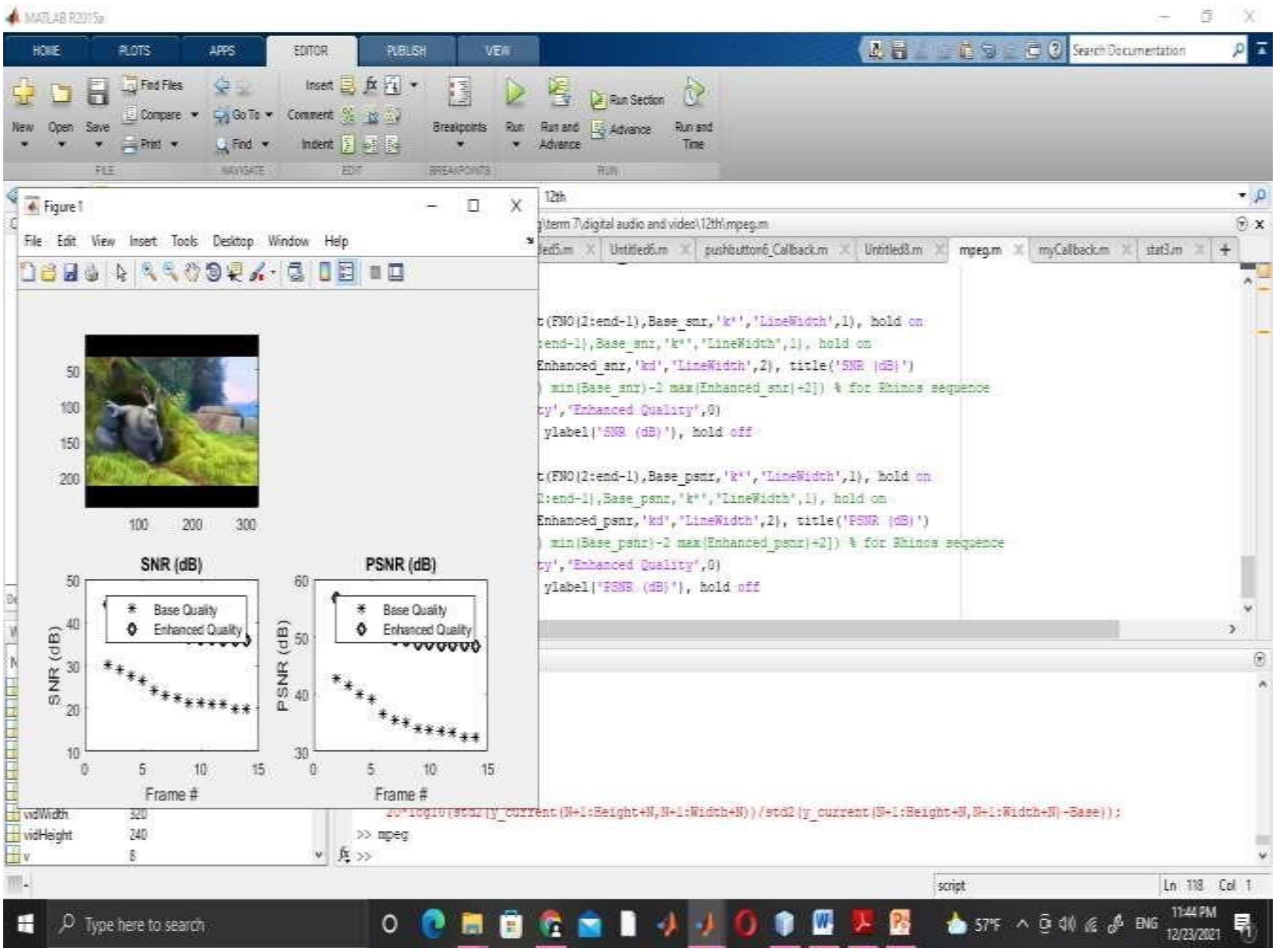


Figure 1 SNR and PSNR values for a video sequence using spatial scalable MPEG coding: (a) video sequence consists of 15 frames (b) SNR in dB and (c) PSNR in dB. The frames used are 2 through 15, inclusive.

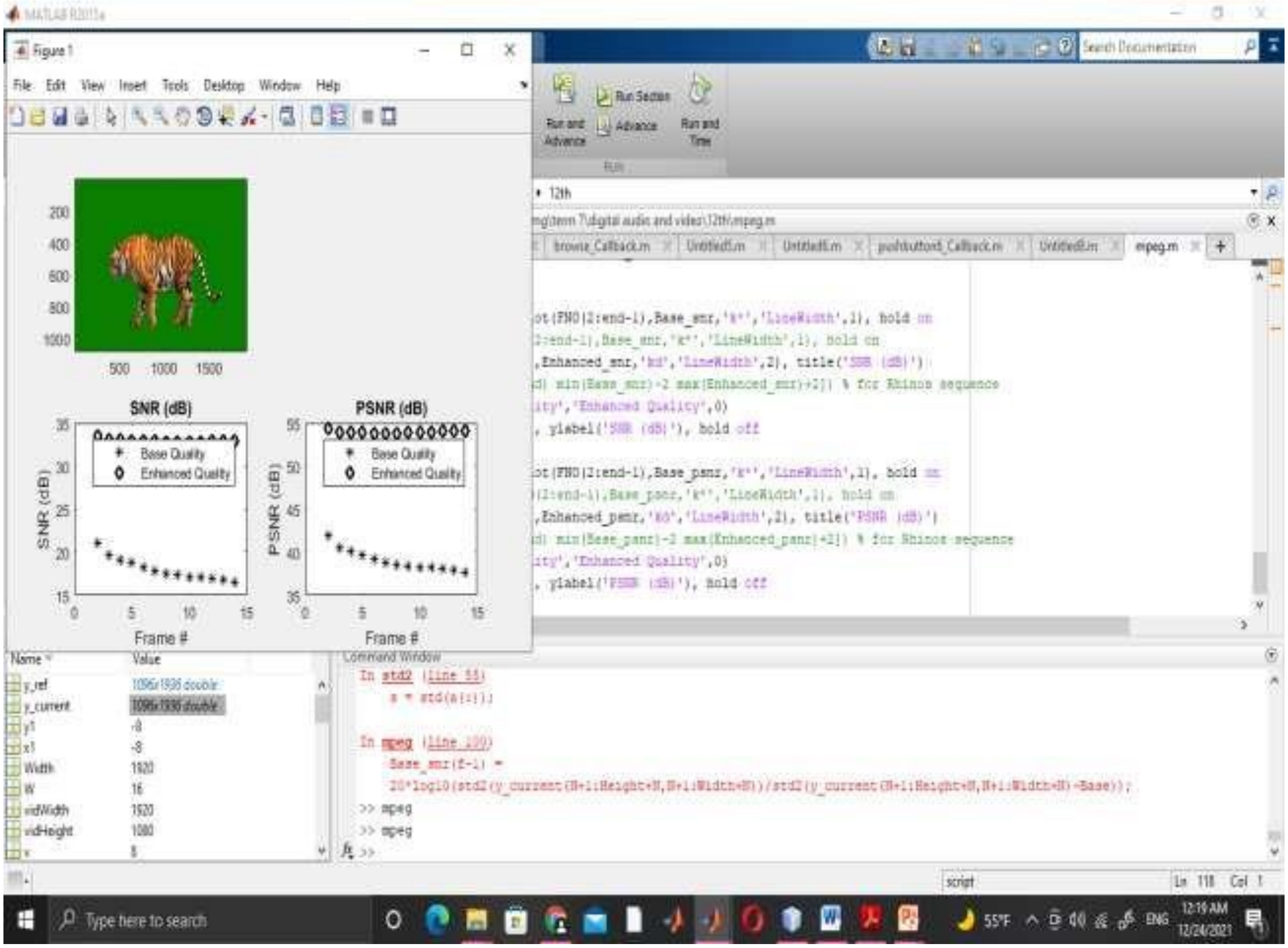


Figure 2 SNR and PSNR values for a tiger video sequence using spatial scalable MPEG coding: (a) video sequence consists of 15 frames (b) SNR in dB and (c) PSNR in dB. The frames used are 2 through 15, inclusive.

7) References

[1] Introduction to Digital Audio and Video Fundamentals, lecture Notes, Dr. Wesam askar, 2021

[2] Robert M. Goodman, Patrick McGrath "Editing Digital Video: The Complete Creative and Technical Guide" 2007

[3] K. S. Thyagarajan, STILL IMAGE AND VIDEO COMPRESSION WITH MATLAB, John Wiley & Sons, Inc, 2011.