Department of Signals & Systems
Chalmers University

Laboratory Exercise 4

Multimedia Communications Over IP Networks

May 21, 2009

Submitted By

Group 43

Rayyan Ali Qureshi

Department of Signals & Systems

Chalmers University

rayyan@student.chalmers.se

Sohaib Maalik

Department of Signals & Systems

Chalmers University

sohaib@student.chalmers.se

Submitted To

Irene Y. H. Gu

Department of Signals & Systems

Chalmers University

irenegu@student.chalmers.se

Table of Contents

List of Figures

1. Abstract

This lab focuses on the issue of multimedia communication through IP networks. The multimedia data considered is a 2D raw image. Simplest communication system has been built and studied by taking into consideration the erasure channel noise on reconstructed multimedia data at the receiver side [1].

2. Tests on the Noise impact on Image Communications

2.1. Test a – CASE I

Test Parameters for case 1 are as follows;

- Channel Noise = Symbol Errors
- t = 60 symbol errors for each code word
- No use of Interleaver & De-interleaver
- No. of Code words = 260

Original Image

Figure 1: Plot of Original Image

Reconstructed Image

Figure 2: Plot of Reconstructed Image

It can be seen clearly that original and reconstructed images are almost same as introduced symbol errors of 60 per code word is within the error handling capability of RS decoder i.e. 64 symbols per code word. The minor difference between the images is due to compression only.

2.2. Test b – CASE II

Test Parameters for case 2 are as follows;

- Channel Noise = Symbol Errors
- ***t = 100*** symbol errors for each code word
- No use of Interleaver & De-interleaver
- No. of Code words = 260

Original Image

Figure 3: Plot of Original Image

Reconstructed Image

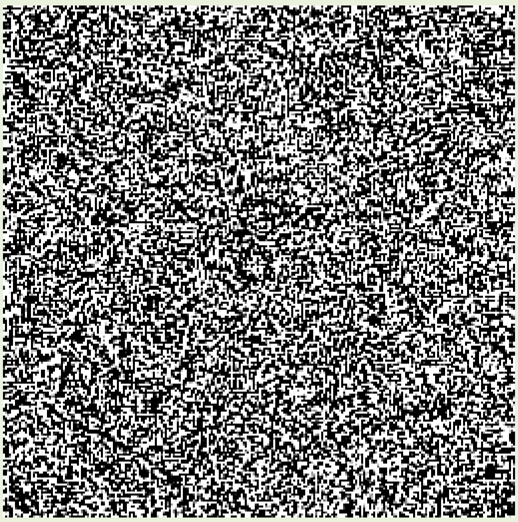

Figure 4: Plot of Reconstructed Image

As it is mentioned above, 100 symbol errors in each code have been introduced. Now it is evident from figure 4 that the reconstructed image is very much distorted and not recognizable. This is due to the fact that code word contains more errors than the error correcting capability of the RS decoder which is of $\left[\left(n-\frac{k}{2}\right)\right] = 64$ errors per codeword.

2.3. Test c – CASE III

Test Parameters for case 3 are as follows;

- Channel Noise = Packet Losses
- Packet Loss rate = 2%
- No use of Interleaver & De-interleaver

Original Image

Figure 5: Plot of Original Image

Reconstructed Image

Figure 6: Plot of Reconstructed Image

The packet loss rate in the above reconstructed image is 2%. This means that out of 260, 5 packets will be lost. The blocks will be affected depending upon how the lost packet was bundled. Furthermore, RS decoder can correct up to $\left[\left(n-\frac{k}{2}\right)\right]=64$ errors per code word. But in this case total number of symbol errors is around 127*5=635 symbols and the entire code word has been lost and it is impossible to recover using channel encoding & decoding techniques.

2.4. Test d – CASE IV

Test Parameters for case 4 are as follows;

- Channel Noise = Packet Losses
- Packet Loss rate = 2%
- ***Interleaver & De-interleaver is used***

Original Image

Figure 7: Plot of Original Image

Reconstructed Image

Figure 8: Plot of Reconstructed Image

In figure 6 and 7, original vs. reconstructed image has been shown. It can be seen clearly that reconstructed image which is at the receiver side is as good as the original image with minor differences which is at transmitter side when an interleaver and a de-interleaver is used. The reason behind this is simple. What interleaver does is it spreads the bits of each packet randomly and the new packet formed contains bits from different code words. Now during the transmission, data is affected and sometimes lost due to channel errors. At the receiver side, data is de-interleaved and original data containing fewer errors is obtained. These errors can be easily corrected using RS decoder and hence we get a complete error free multimedia data. The visual difference which can be observed in the original and reconstructed image is due to the data compression only.

3. Summary & Comments

This lab focuses on the issue of multimedia communication through IP networks. The multimedia data considered is a 2D raw image. Simplest communication system has been built and studied by taking into consideration the erasure channel noise on reconstructed multimedia data at the receiver side. To limit the scope of this lab only wired IP networks have been considered [1].

After building up the complete IP system, different tests were performed on the multimedia data communication using different parameters. In the first test a comparison was done between the original transmitted image and the reconstructed image after reception. Used parameters have already been mentioned earlier. It was observed that the reconstructed image was as good as the original image. The reason behind this was that the symbols errors which were introduced into each code word were 60 and was less than the error correcting capability of this code which is $\left[\left(n-\frac{k}{2}\right)\right]=64$ errors per code word. The only difference in the original and reconstructed image exists due to compression.

In second test unlike the first one, 100 symbol errors in each code were introduced. It can be seen form figure 4 that the reconstructed is very much distorted and is not recognizable. The reason behind this is more errors have been introduced in each codeword than the error correcting capability of the RS decoder.

The third test was performed by setting noise as network “packet losses”. Other parameters used have already been defined above. It was seen that entire code word was effected/lost in this case and it was impossible to recover using channel encoding & decoding techniques. So the figure 5 of the reconstructed image contained some distorted blocks. Further details have already been discussed above under this test.

In the last test an interleaver and de-interleaver was introduced in the system and its effect was studied. Other parameters used have already been described above. It was observed that unlike the third test, in which interleaving was not included, results got better and we got as good image at the receiver side as we have on the transmitter side. This was because the interleaver spread the symbols of one codeword over many code words. In this way interleaver makes data more efficient against burst errors. Thus, if a single packet gets in error we do not lose the whole codeword as this packet contains symbols from different code words. In this way only one symbol from each code word gets corrupted and on the receiver side with the help of de-interleaver the whole data can be recovered correctly. Interleaving is very useful in fading channels and makes the data more robust against burst errors [2].

References

[1] Irene Y. H. Gu, Laboratory Exercise 4: *Multimedia Communications Over IP Networks,* Chalmers University, 2009

[2] http://en.wikipedia.org/wiki/Interleaving

Appendix – MATLAB Code

```
Close all[1];
clear all;
clc;
%% Block 1
%step 4.1
orgimg=imread('lena.bmp');                          %reading image
orgimg=mat2gray(orgimg);                            %converting image format to intensity image
figure;imshow(orgimg);title('Original Image')      %Displaying image
%step 4.2
Rcompression=0.5;                                   %Setting the compression ratio
%step 4.3(Block1)
%step 4.3.1
[Z_c CImage Z]=DCT(orgimg);                         %function call
%% Task1
% %Step 1.1
% t=(0:0.1:10*pi);                          %generating synthetic signal
% signal=signal(t);               %function call
%Step 1.2(Block 2)
[Index,quantized,codebook]=quantization(Z_c);  %function call
%% Task2
%step 2.1(Block3)
packet=packetizer(Index);       %function call
%% Task3
%step 3.1(Block4)
packet=packet.';                        %Each row representing 1 source packet of size
127symbols
[codes,msgwords]=RSencoder(packet);     %function call
%% Task7
% %step7.1(Block 5)
% codeswrds=codes.x;
% intrlvr=matintrlv(codes,15,17);                        %matrix interleaving
Nrows = 260;                                             % Number of interleved rows
Ncols = 255;                                             % Number of interleaved columns
data = codes;
intrlvd = reshape(data,Ncols,Nrows).';
%% Task5
% %block 7
[code_noisy]=channelmodel(intrlvd);             %function call
%step7.2(Block 9)
deintrlvd = reshape(code_noisy.',Nrows,Ncols);
%deintrlvr=matdeintrlv(intrlvd,15,17);            %matrix deinterleaving

%step 3.2(Block9)
[decoded,cnumerr]=RSdecoder(deintrlvd);     %function call
%isequal(decoded,msgwords)              %To verify encoded and decoded msg
%step 2.2(Block10)
depacket=depacketizer(decoded);           %function call
%step 1.3(Block 11)
quantizedv=block11(depacket,codebook);      %function call
%step 4.4(block12)
%step4.4.1
invimg=IDCT(quantizedv,orgimg);                        %function call
figure;imshow(invimg);title('Reconstructed image')



function[2] [code_noisy]=channelmodel(codes)
m=8;                                                      %Bits per symbol
n=2^m-1;                                                  %codeword length
k=127;                                                    %Message length
% t=floor(n-k/2);                                         %Number of correctable errors
t=100;
channel=2;                                                %for selecting channel 1 or 2
nw=260;                                                  %number of codewords
switch channel
    case 1
     noise=(1+randint(nw,n,n)).*randerr(nw,n,t);          %'t'errors per row,for'codes'
codewords
```

[1] Start of Main file

[2] Functions are not necessarily in the order of their call in the main file

```
     cnoisy=codes+noise;                                  %add noise to the code
     code_noisy = cnoisy;
    case 2
     PL=floor(0.02*nw);                                   %packet loss(10% of total packets)
     PL_in=round(nw*rand(1,PL));                          %randomly generated indices
     for i=1:PL
     e_packet = zeros(1, n);                              % generate a codeword with zero
value, n is codeword size
     errorpacket= gf(e_packet,m);                         % generate an error packet in
Matlab class gf
     codes(PL_in(i), :) = errorpacket;                    % replace the ith codeword by a
packet with zero values
     end
     code_noisy=codes;
end
```

```
function [Z_c CImage Z2]=DCT(orgimg)
[R C]=size(orgimg);
Blocksize=16;
RB=R/Blocksize;                                  %TO make 16*16 blocks
CB=C/Blocksize;
%setting for compression
Rcompression = 0.5;                              %compression ratio
N=16*16;                                         %Block size
N1=round(Rcompression*N);                        %coefficients to remove from each block
Nc=N-N1;                                         %coefficients to retain in each block
Z_c =[];
Z2=[];
%step 4.3.2
for i=1:RB
    for j=1:CB
       temp=orgimg((i-1)*Blocksize+1:i*Blocksize,(j-1)*Blocksize+1:j*Blocksize);
       tempDCT = dct2(temp);                      %applying 2d dct to all 16*16 blocks and
saving it in bbi
       CImage((i-1)*Blocksize+1:i*Blocksize,(j-1)*Blocksize+1:j*Blocksize) = tempDCT;
       Z=zigzag2dto1d(tempDCT);                   %function call(zigzag scanning)
       Z1 = Z(1:Nc);
       Z_c = [Z_c Z1];                             %Zigzag scanned DCT coefficients
       Z2=[Z2 Z];
    end
end
figure;imshow (CImage);title('2D block based transformed DCT image');
```

```
function [Index,quantized,codebook]=quantization(signal)
Max=max(signal);
Min=min(signal);
m=8;                            %bits per sample
L=2^m;                          %No of quantization levels
Delta=(Max-Min)/(L-1);          %Step size
codebook=Min:Delta:Max;         %Reconstruction levels
partition=(codebook-Delta/2);   %Decision thresholds
partition=partition(2:end);     %Decision threshold vector
[Index,quantized]=quantiz(signal,partition,codebook);    %scalar quantization
```

```
function invimg=IDCT(quantizedv,orgimg)
L =length(orgimg);
% %setting for compression
Rcompression = 0.5;                              %compression ratio
N=16*16;                                         %Block size
N1=round(Rcompression*N);                        %coefficients to remove from each block
Nc=N-N1;                                         %coefficients to retain in each block
N =16;
i =1;
for j = 0:N:L-N
    for l = 0:N:L-N
        iscan = zeros(1,N*N);
        iscan(1:Nc) = quantizedv((i-1)*Nc+1:i*Nc);
        iscan1(j+1:j+N,l+1:l+N) = dezigzag1dto2d(iscan);
```

```
        i =i+1;
    end
end

back =zeros(L,L);
for inc2 = 0:N:L-N
for inc = 0:N:L-N
for i=1+inc2:N+inc2
    for j =1+inc:N+inc
        back(i,j) = iscan1(i,j);            % Creating 16x16 Blocks for inverse transform
    end
end
    invimg(inc2+1:inc2+N,inc+1:inc+N) = idct2(back(inc2+1:inc2+N,inc+1:inc+N));      % Taking
Inverse Transform
end
end



function A=dezigzag1dto2d(Z)

ind = zigzag4(sqrt(length(Z)));

A=[];
for k=1:length(Z)
    A( ind(k,1),ind(k,2) )=Z(k);
end



function Index=depacketizer(packet)
packet1=[];
for i=1:length(packet)
    x=packet(i,:);
    packet1=[packet1 x];
end
Index=packet1(1:32768);              %Removing zero padding done in packetizer function



function packet=packetizer(Index)
 k=127;                         %127 symbols per packet(size of source packet)
zIndex=[Index zeros(1,252)];      %Zero padding
 packetnumber=length(zIndex)/k;   %dividing in 3 packets
 packet=reshape(zIndex,k,packetnumber);  %Array of packets.No of rows  is equal to no of
packets


function quantizedv=block11(Index,codebook)

quantizedv=codebook(Index+1);                                   %Quantized Levels(quantization
values)

%figure;plot(quantizedv),title('QSW at output of block11');        %plot of quantized sine
wave



function [decoded,cnumerr]=RSdecoder(codes)
m=8;                                    %Bits per symbol
n=2^m-1;                                %codeword length
k=127;                                  %Message length
%dec_msg=gf(codes,m);                     %represent encoded msg using glaois array
[decoded,cnumerr]=rsdec(codes,n,k);  %RS decoding
decoded=decoded.x;


clc, close all, clear all
N=5;
A=randint(N,N,[2,10]) %N-by-N, random array of integers
Z=zigzag2dto1d(A)   % zig-zag
B=dezigzag1dto2d(Z) % de zig-zag
isequal(A,B)
```

```
function Z = zigzag4(N)

if ~exist('N')
  N = 8;
end

Z = zeros(N*N,2);
u = 0; v = 2; inc = 1;

for n = 1:N*N
  v = v - inc; u = u + inc;
  if (u > N)
    v = v + 2; u = N; inc = -1;
  elseif (v == 0)
    v = 1; inc = -1;
  elseif (v > N)
    u = u + 2; v = N; inc = 1;
  elseif (u == 0)
    u = 1; inc = 1;
  end
  Z(n,:) = [v u];
end
```

```
function Z=zigzag2dto1d(A)

[r,c]=size(A);

if r~=c
    error('input array should have equal number of rows and columns')
end

ind = zigzag4(r);
Z=[];
for k=1:size(ind,1)
    Z=horzcat(Z, A(ind(k,1),ind(k,2)) ) ;
end
```

```
function [codes,msgwords]=RSencoder(packet)
m=8;                                    %Bits per symbol
n=2^m-1;                                %codeword length
k=127;                                  %Message length
msgwords=gf(packet,m);                  %represent packets using glaois array
codes=rsenc(msgwords,n,k);              %RS encoding
msgwords=msgwords.x;
%codewords=codes.x;                     %Extract rows of codewords from GF array
```