EE368B – Image and Video Compression

Solution to Homework Set #3

Prepared by Markus Flierl

1 Blockwise 8×8 DCT

A DCT-II of blocksize 8×8 is given by the 8×8 transform matrix A. The 8×8 transform coefficients are given by $y = AxA^T$.

$$A = \begin{pmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{pmatrix}$$

2 Uniform Quantizer

A uniform mid-tread quantizer has a zero representative level.

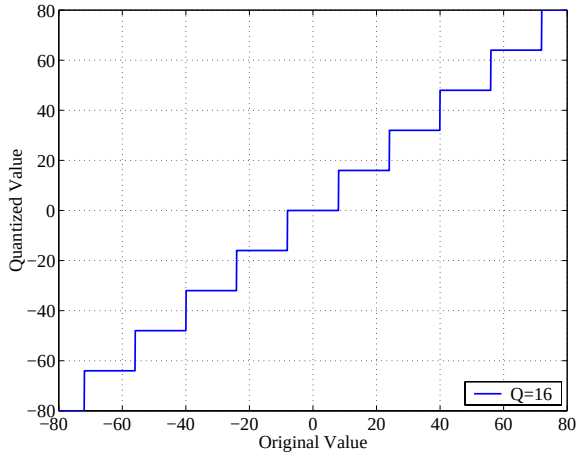

Figure 1: Uniform mid-tread quantizer without threshold characteristic.

3 Distortion and Bit-Rate Estimation

We selected the the same quantizer step-size for all coefficients. When the quantization step-size becomes very large, we get an one-level quantizer in effect, for which the bit-rate is 0 bits/pixel.

Figure 2: Transform coder with DCT.

The MSE measured between the original signal $\mathbf{x}$ and the reconstructed signal $\mathbf{x}'$ can also be measured between the original DCT coefficients $\mathbf{y}$ and the quantized DCT coefficients $\mathbf{y}'$. Representing the signals and coefficients as vectors, the transformation is given by $\mathbf{y} = \mathbf{A}\mathbf{x}$ and $\mathbf{x} = \mathbf{A}^T\mathbf{y}$.

$$\begin{aligned} \text{MSE} &= \frac{1}{64}(\mathbf{x}-\mathbf{x}')^T(\mathbf{x}-\mathbf{x}') \\ &= \frac{1}{64}(\mathbf{y}-\mathbf{y}')^T\mathbf{A}\mathbf{A}^T(\mathbf{y}-\mathbf{y}') \\ &= \frac{1}{64}(\mathbf{y}-\mathbf{y}')^T(\mathbf{y}-\mathbf{y}') \end{aligned}$$

This relation holds because the DCT is an orthonormal transform. Therefore, the computation of the IDCT is not necessary when determining the MSE.

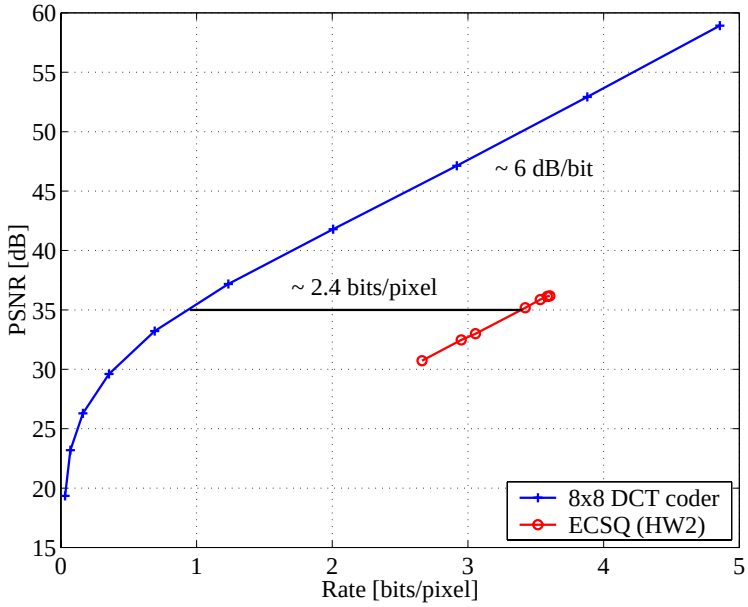

Figure 3: Rate-distortion function for the transform coder. The data set consists of the images *boats, harbour,* and *peppers*. The performance of the ECSQ in homework set #2 is also plotted. We gain about 2.4 bits/pixel by exploiting the spatial redundancy. A slope of about 6 dB/bit is observed for large bit-rates.

4 Judging Image Quality (Bonus Exercise)

Ideally one subject would score a set of original and quantized image pairs for a number of times and average the resulting impairment scores. However, just for demonstration here, we only had one person perform the scoring once. A plot of impairment vs. PSNR as well as a plot of impairment vs. rate are shown below for each of the three images. The PSNR and bit-rate are calculated for each individual image with the VLC tables from Problem 3. We selected the quantizer step-size from the range $2^0, 2^1, 2^2, ..., 2^9$.

R [bits/pixel]	PSNR [dB]	DSIS
4.4369	58.9228	5
3.4676	52.9013	5
2.5141	47.0896	5
1.6270	42.3085	5
0.9854	38.3311	4
0.5649	34.3918	3
0.2967	30.6197	2
0.1420	27.1124	1
0.0630	23.8465	1
0.0312	19.4061	1

R [bits/pixel]	PSNR [dB]	DSIS
5.3410	58.9101	5
4.3672	52.9853	5
3.4029	47.4152	5
2.5074	42.0144	5
1.7059	36.7875	4
1.0099	31.9782	3
0.4972	27.9587	2
0.2022	24.6780	1
0.0714	21.9205	1
0.0258	18.7366	1

R [bits/pixel]	PSNR [dB]	DSIS
4.7916	58.9292	5
3.8040	52.8926	5
2.8357	46.9119	5
1.8866	41.1493	5
1.0124	36.6316	5
0.5003	33.6576	4
0.2707	30.8607	3
0.1433	27.7092	2
0.0743	24.1765	1
0.0357	19.9890	1

Table 1: PSNR, bit-rate, and Double Stimulus Impairment Scale for the images *boats* (left), *harbour* (middle), and *peppers* (right).

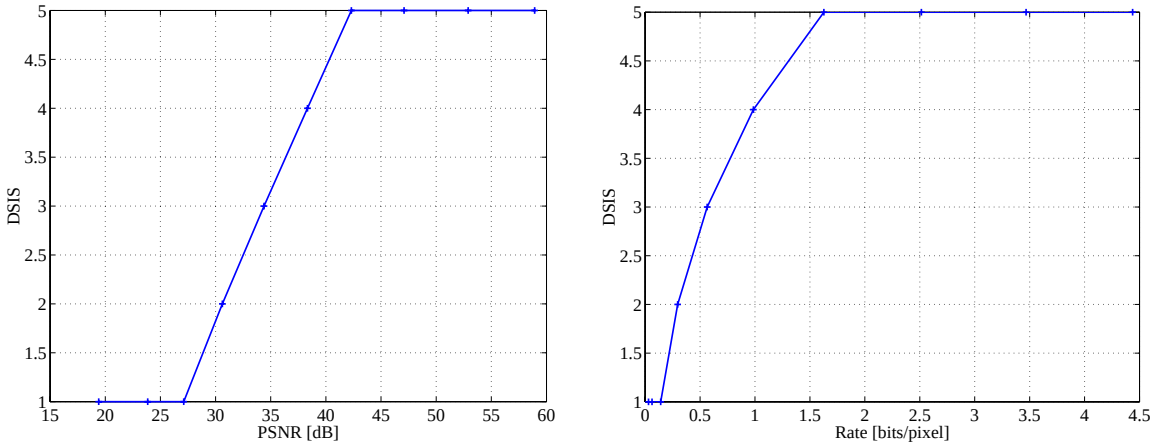

Figure 4: DSIS vs. PSNR and bit-rate for the image *boats*.

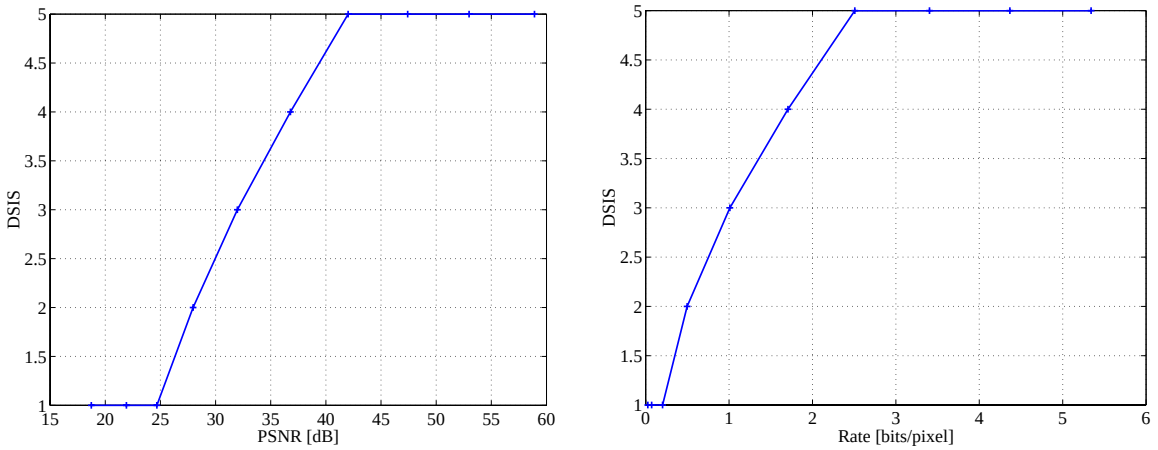

Figure 5: DSIS vs. PSNR and bit-rate for the image *harbour*.

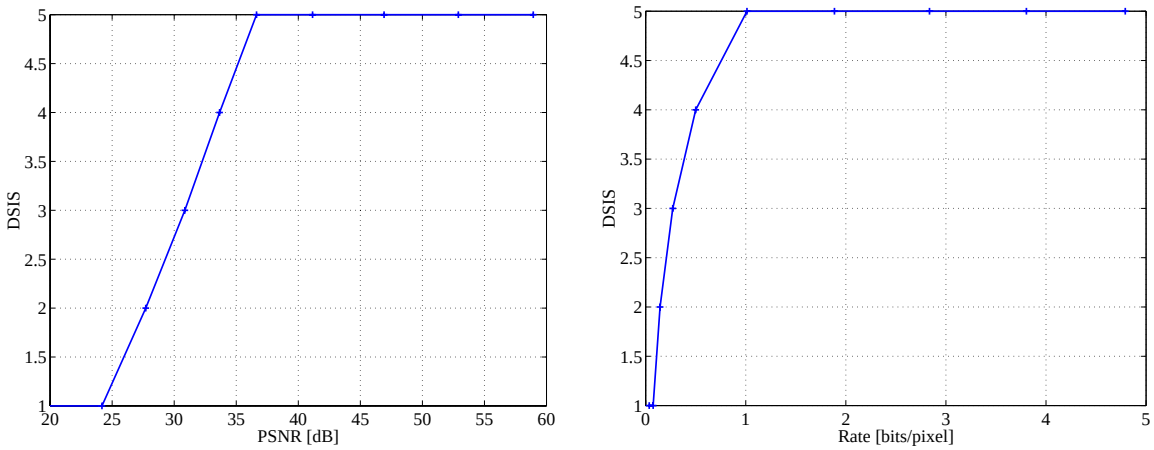

Figure 6: DSIS vs. PSNR and bit-rate for the image *peppers*.

A Problem 1: Function matrixDCT.m

```
function A = matrixDCT(M)
%matrixDCT Determines MxM matrix for DCT-II
%
%  A : transfrom matrix
%  y = A*x*A'
%
%  EE368B - Image and Video Compression
%  Markus Flierl, 2000-10-17

% row index i
i = (0:M-1)'*ones(1,M);

% column index k
k = ones(M,1)*(0:M-1);

% scaling factor
a = sqrt(2/M)*ones(M,1);
a(1) = sqrt(1/M);
alpha = a*ones(1,M);

% transform matrix
A = alpha.*cos((2*k+1).*i*pi/(2*M));
```

B Problem 2: Function uniformquant.m

```
function y = uniformquant(x,q)
%UNIFORMQUANT Uniform mid-tread quantizer with step-size q
%  y = uniformquant(x,q)
%
%  x : input
%  y : ouput
%  q : quantizer step-size
%
%  EE368B - Image and Video Compression
%  Markus Flierl, 2000-10-17

y = round(x/q)*q;
```

C Script for Problem 3

```
%  Distortion and Bit-Rate Estimation
%
%  EE368B - Image and Video Compression
%  Markus Flierl, 2000-10-17

% block size
M = 8;

% generate transform matrix
A = matrixDCT(M);

% image set
noi   = 3;
name1 = 'Images/boats512x512.tif';
name2 = 'Images/harbour512x512.tif';
name3 = 'Images/peppers512x512.tif';

img = zeros(512,512, noi);
for i=1:noi
  file = eval(['name' num2str(i)]);
  img(:,:,i) = imread(file, 'tiff');
end;

% convert to sequence of blocks
blkSeq = imgseq2blkseq(img, M);

% DCT
sz = size(blkSeq);
no = sz(3);
for n = 1:no
  if ~mod(n,1000) disp(n); end;
  % x(:,:,n) = A*blkSeq(:,:,n)*A';
  x(:,:,n) = dct2(blkSeq(:,:,n));
end;

% rate-distortion curve
q = 2.^(0:9);
D = zeros(1, length(q));
R = zeros(1, length(q));
for cnt = 1:length(q)
  % quantization
  y = uniformquant(x, q(cnt));

  % mse
  D(cnt) = sum(sum(sum( (x-y).^2 )))/M/M/no;

  % entropy
  v = zeros(1,no);
  H = zeros(M,M);
  for i = 1:M
    for j = 1:M
      v = y(i,j,:);
      bins = min(v):q(cnt):max(v);
      bSave{i}{j}{cnt} = bins;
      if(length(bins)>1)
p = hist(v,bins);
p = p/sum(p);
lSave{i}{j}{cnt} = -log2(p+eps);
H(i,j) = -sum(p.*log2(p+eps));
      else
lSave{i}{j}{cnt} = 0.0;
H(i,j) = 0;
      end;
    end;
  end;
  R(cnt) = mean2(H);
end;

% save VLC tables
save vlc bSave lSave
```

```
% plot graph
PSNR = 10*log10(255*255./D);
plot(R, PSNR, '+-', 'linewidth', 1.5);
grid;
set(gca, 'fontname', 'Times');
set(gca, 'fontsize', 16);
xlabel('Rate [bits/pixel]');
ylabel('PSNR [dB]');
```

D Function imgseq2blkseq.m

```
function blkseq = imgseq2blkseq(imgseq, M);
%IMGSEQ2BLKSEQ Convert a sequence of images to a sequence of blocks
%  of size MxM
%
%  blkseq = imgseq2blkseq(imgseq, M);
%
%  EE368B - Image and Video Compression
%  Markus Flierl, 2000-10-17

sz = size(imgseq);
if length(sz)<3 sz(3) = 1; end;
noBlk = (sz(1)/M)*(sz(2)/M)*sz(3);
blkseq = zeros(M,M,noBlk);
n = 0;
for k = 1:sz(3)
  for i = 1:M:sz(1)
    for j = 1:M:sz(2)
      n = n + 1;
      blkseq(:,:,n) = imgseq(i:i+M-1, j:j+M-1, k);
    end;
  end;
end;
```

E Function blkseq2imgseq.m

```
function imgseq = blkseq2imgseq(blkseq, N);
%BLKSEQ2IMGSEQ Convert a sequence of blocks to a sequence of images
%  of size NxN
%
%  imgseq = blkseq2imgseq(blkseq, N);
%
%  EE368B - Image and Video Compression
%  Markus Flierl, 2000-10-17

sz = size(blkseq);

noImg = sz(3)/(N/sz(1)*N/sz(2));
imgseq = zeros(N,N,noImg);
n = 0;
for k = 1:noImg
  for i = 1:sz(1):N
    for j = 1:sz(2):N
      n = n + 1;
      imgseq(i:i+sz(1)-1, j:j+sz(2)-1, k) = blkseq(:,:,n);
    end;
  end;
end;
```

F Script for Problem 4

```
%  Judging Image Quality
%
%  EE368B - Image and Video Compression
%  Markus Flierl, 2000-10-17

clear;

% block size
M = 8;

% generate transform matrix
A = matrixDCT(M);

% image set
name1 = 'Images/boats512x512.tif';
name2 = 'Images/harbour512x512.tif';
name3 = 'Images/peppers512x512.tif';

imgSeq = zeros(512,512,1);
imgSeq(:,:,1) = imread(name1, 'tiff');

% convert to sequence of blocks
blkSeq = imgseq2blkseq(imgSeq, M);

% load VLC tables
load vlc

% DCT
sz = size(blkSeq);
no = sz(3);
for n = 1:no
  if ~mod(n,1000) disp(n); end;
  % x(:,:,n) = A*blkSeq(:,:,n)*A';
  x(:,:,n) = dct2(blkSeq(:,:,n));
end;

q = 2.^(0:9);
D = zeros(1, length(q));
R = zeros(1, length(q));
DSIS = zeros(1,length(q));
for cnt = 1:length(q)
  % quantization
  y = uniformquant(x, q(cnt));

  % mse
  D(cnt) = sum(sum(sum( (x-y).^2 )))/M/M/no;

  % bit-rate
  v = zeros(1,no);
  L = zeros(M,M);
  for i = 1:M
    for j = 1:M
      v = y(i,j,:);
      bins = bSave{i}{j}{cnt};
      if(length(bins)>1)
        p = hist(v,bins);
        p = p/sum(p);
        L(i,j) = sum(p.*lSave{i}{j}{cnt});
      else
        L(i,j) = 0.0;
      end;
    end;
  end;
  R(cnt) = mean2(L);

  % IDCT
  for n = 1:no
    if ~mod(n,1000) disp(n); end;
    % r(:,:,n) = A'*y(:,:,n)*A;
    r(:,:,n) = idct2(y(:,:,n));
```

```
  end;

  % convert to sequence of images
  imgSeqRec = blkseq2imgseq(r, 512);

  % reference image
  figure;
  h=imagesc(imgSeq(:,:,1));
  colormap('gray');
  axis('equal');

  % degraded image
  figure;
  h=imagesc(imgSeqRec(:,:,1));
  colormap('gray');
  axis('equal');

  DSIS(cnt) = input('Your score (1-5):');
  close all;
end;

% plot PSNR vs. impairment
figure;
PSNR = 10*log10(255*255./D);
plot(PSNR, DSIS, '+-', 'linewidth', 1.5);
grid;
set(gca, 'fontname', 'Times');
set(gca, 'fontsize', 16);
xlabel('PSNR [dB]')
ylabel('DSIS');

% plot bit-rate vs. impairment
figure;
plot(R, DSIS, '+-', 'linewidth', 1.5);
grid;
set(gca, 'fontname', 'Times');
set(gca, 'fontsize', 16);
xlabel('Rate [bits/pixel]')
ylabel('DSIS');
```