Lab 6

The Discrete Cosine Transform

0. Abstract

A discrete cosine transform (DCT) is defined and an algorithm to compute it using the fast Fourier transform is developed. It has been widely used in processing the image compressing. In this lab, we will try to use the DCT to process the sequence and image and find how the coefficients will affect the quality of the reconstruction of the figure.

1. Sections

1.1. Plot the basis functions and N = 8

For the 1-D DCT, and N = 8, the code for the eight figures are as follows:

```
function exe1_1()
N = 8;
k = linspace(0 , N -1, N)
for n = 0 : N-1
sum = cos((2 * n + 1) * k * pi./(2 * N))
subplot(4,2,n+1),plot(k,sum)
end

end
```

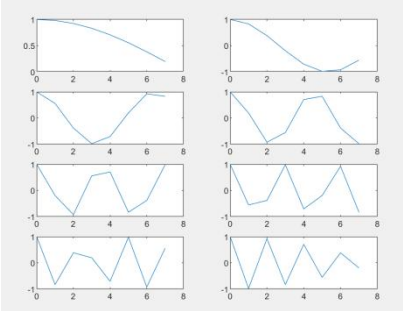

Figure 1: The basis function when N = 8

1.2. **Find and plot the DCT and DFT versus k**

The sample sequence we use is:

```
(0.8 ^ n) * cos(n * pi ./ 4) * (heaviside(n) -heaviside(n-8))
```

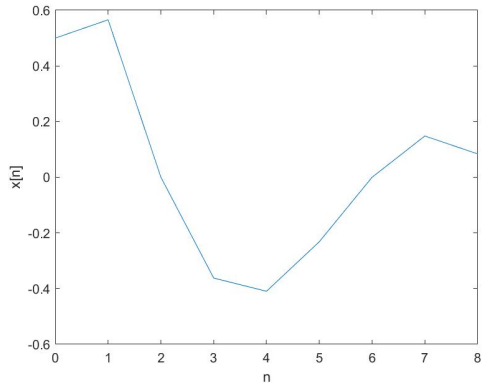

Figure 2: x[n] versus n

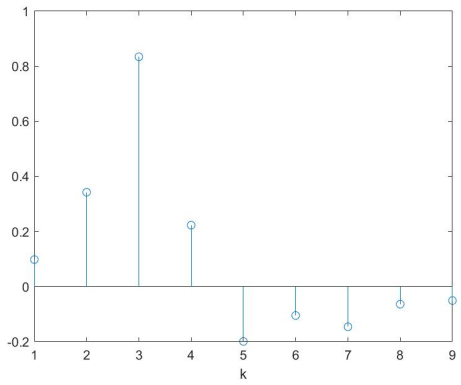

Figure 3: The DCT of the signal versus k

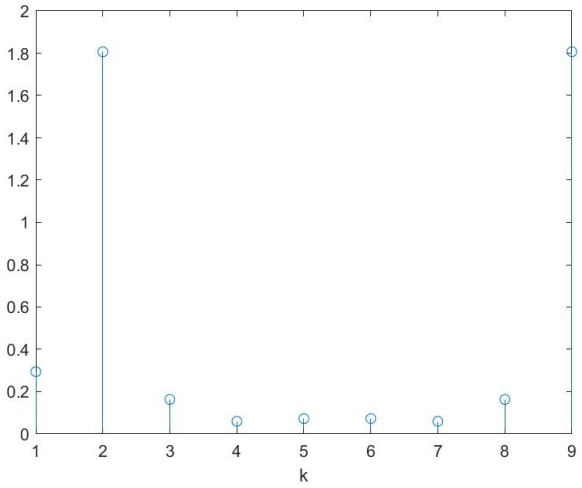

Figure 3: The DFT of the signal versus k

1.3. Answer to the question

The energy is concentrated in DCT, when the k = 2, 3, 4 and the value of the energy turn to negative when k is bigger than 5.Each DCT output bin is a (weighted) correlation against a cosine function of a certain frequency. A negative value would represent a negative correlation. The energy distribution of DFT is highly concentrated when k = 2. It is more concentrated than the energy in DCT.

1.4. Each of basis function scaled by the DCT values

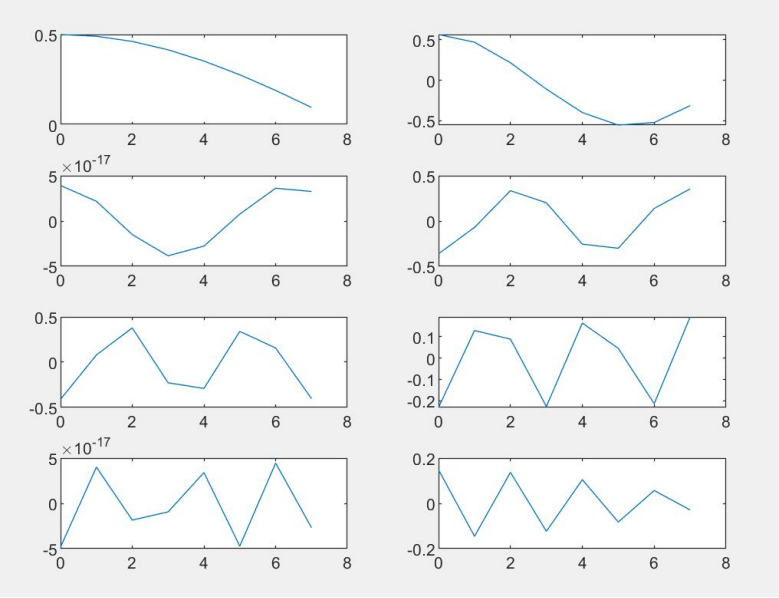

Figure 4: The 1-D DCT basis functions scaled by the DCT values found in 1.2

1.5. Find the mean squared error(MSE)

In statistics, the mean squared error (MSE)or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. The value(mse) turned out to be 0.0466. The code to calculate MSE is as follows:

```
function exe1_4()
N = 8;
for n = 0 : 7
x(n + 1) = (0.8 ^ n) * cos(n * pi ./ 4) * (heaviside(n) -heaviside(n-8))
end

a = dct(x)
for n = 0 : 7
sum = 0;
for k = 0 : 3
sum = sum + a(k+1) * cos((2 * n + 1) * k * pi ./ (2 * N))
end
y(n + 1) = w_k(n,N) * sum
end
sum_n = 0;
squared_norm = 0;
for n = 0 : 7
sum_n = sum_n + (abs(x(n + 1)-y(n + 1))) ^ 2;
squared_norm = squared_norm + (x(n + 1)) ^ 2;
end
mse = sum_n ./ squared_norm
```

```
end
```

2.1.For M = 8, N = 8,plot the 64 2-D basis functions

The code for 2-D basis function is as follows:

```
function exe2_1()
M = 8; N = 8;
n = [0:N-1]; m = [0:M-1];
for k=0:M-1
for l=0:N-1
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N) = ...
repmat(cos((2*m+1)*k*pi/(2*M))',1,N).* ...
repmat(cos((2*n+1)*l*pi/(2*M)),M,1);
if k==0
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N) = ...
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N)./sqrt(2);
end
if l==0
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N) = ...
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N)./sqrt(2);
end
end
end
basis = basis.*(sqrt(2/N)*sqrt(2/M));
basis = basis .* 50
basis = uint8(basis);
imagesc(basis)
colormap(gray(256))
end
```

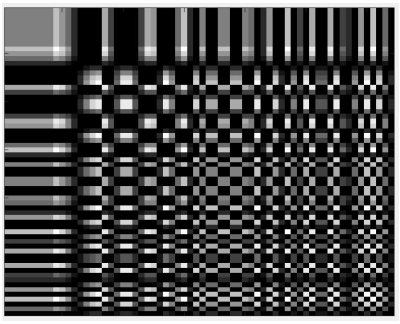

Figure 5: The 2-D basis functions

2.2 Process the image

The code and comments for each line are as follow:

```
function exe2_2()
M = 8; N = 8; %define N and M
n = [0:N-1]; m = [0:M-1];
for k=0:M-1
for l=0:N-1
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N) = ...%working as the equation(4)
repmat(cos((2*m+1)*k*pi/(2*M))',1,N).* ...%treat w[k] as sqrt(2./N) first
repmat(cos((2*n+1)*l*pi/(2*M)),M,1);%treat w[l] as sqrt(2./N) first
if k==0
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N) = ... %w[0] = sqrt(1 ./ N)
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N)./sqrt(2);
end
if l==0
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N) = ...%w[0] = sqrt(1./N)
basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N)./sqrt(2);
end
end
end
basis = basis.*(sqrt(2/N)*sqrt(2/M)); % get the basis function
x = double(imread('barbaraLarge.gif')); %read the gif
[r,c] = size(x); % get the size of gif
X = zeros(r,c); % create a zero matrix of the same size
for i=0:r/M-1 %acting as i loop
for j=0:c/N-1 %acting as j loop
xb = x(i*M+1:(i+1)*M, j*N+1:(j+1)*N);% assign the matrix to xb
for k=0:M-1
for l=0:N-1
X(i+k*(r/M)+1,j+l*(c/N)+1) = ...
sum(sum(xb.*basis(k*M+1:(k+1)*M,l*N+1:(l+1)*N)));% cimpute the xb with basis
function
end
end
end
end
XdB = 20*log10(abs(X)./max(max(abs(X))));% get the dB expression
XdBt = XdB + 60;
XdBt = max(XdBt,0); %find the max in XdBt
f1 = figure('Position',[500 300 600 600],'Units','Normalized'); %create a
figure
set(f1,'PaperPosition',[0.25 1.5 8 8]); % set the graphic object property
axes('Position',[0.09 0.09 0.88 0.88]);% Create Cartesian axes
imagesc([0:c-1],[0:r-1],XdBt); %Display image with scaled
cmap = colormap('gray');%View and set current colormap to gray
colormap(flipud(cmap)); %Flip array up to down
set(gca,'XTick',[0:c/N:c],'YTick',[0:r/M:r]); % set the graphic object
property
set(gca,'XTickLabel','','YTickLabel',''); %set the graphic object property
grid on; set(gca,'GridLineStyle','-');%Display or hide axes grid lines
axis equal;
for k=0:M-1
text(-40,k*r/M+r/M/2,['k = ' num2str(k)],'FontSize',14);% Add text
descriptions to data points using the size of 14
end
```

```
for l=0:M-1
text(l*c/M+c/N/4,r+15,['l = ' num2str(l)],'FontSize',14); %dd text
descriptions to data points using the size of 14
end
end
```

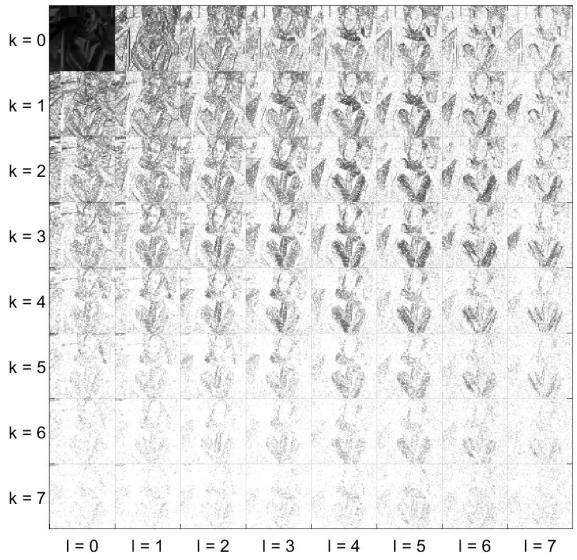

Figure 5: The DCT of the image by using 8 * 8 basis functions

2.3Interpretation of the 2-D DCT

From the graph, we can find that the energy is concentrated when k and l are small, the energy concentration is the highest when k=l=0. As the k and l increase the energy concentration decrease. The energy should be the same in horizontal and vertical stripes since the formula that applied to it is actually the same.

2.4 Approximate the image using only the DC term

The commented code is as follows:

```
kv = [0 1 2];% create a array for loop for row
lv = [0 1 2];% crete a array for loop for column
xr = zeros(fix(2 * M * r ./ ((M - 1) * 2)),fix(2 * M * c ./ ((M - 1)*2))); %
create a zero array
for k=kv
for l=lv
Xb = X(k*r/M+1:(k+1)*r/M, l*c/N+1:(l+1)*c/N); % assign the vlue to xb
for i=0:r/M-1
for j=0:c/N-1
xr(i*M+1:i*M+M,j*N+1:j*N+N) = ...% compute the value and assig the value to xr
xr(i*M+1:i*M+M,j*N+1:j*N+N) + ...
```

```
Xb(i+1,j+1)*basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N);
end
end
```

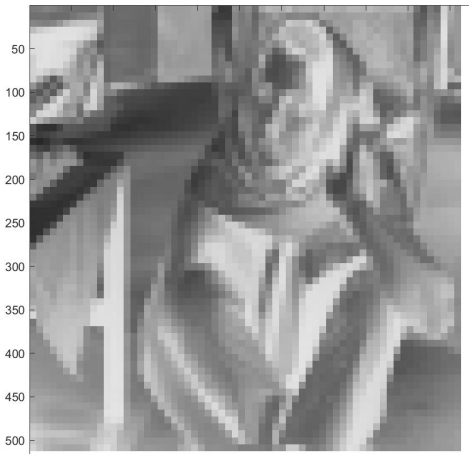

Figure 6: Approximate the image using only the DC term

2.5 Approximate the signal using every DCT term

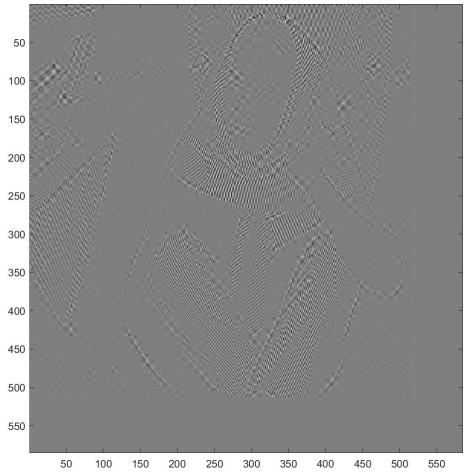

Figure 6: Barbara Large approximated using everything but the DC term of the DCT. All the details can be seen

2.6 Add the result in 2.4 and 2.5

The figure produced has no discrepancy between it and the original image. The whole part of the code is as follows:

```
kv = [0 1 2 3 4 5 6 7 ];% create a array for loop for row
lv = [0 1 2 3 4 5 6 7 ];% crete a array for loop for column
```

```
xr = zeros(fix(2 * M * r ./ ((M - 1) * 2)),fix(2 * M * c ./ ((M - 1)*2))); %
create a zero array
for k=kv
for l=lv
Xb = X(k*r/M+1:(k+1)*r/M, l*c/N+1:(l+1)*c/N); % assign the vlue to xb
for i=0:r/M-1
for j=0:c/N-1
xr(i*M+1:i*M+M,j*N+1:j*N+N) = ...% compute the value and assig the value to xr
xr(i*M+1:i*M+M,j*N+1:j*N+N) + ...
Xb(i+1,j+1)*basis(k*M+1:(k+1)*M, l*N+1:(l+1)*N);
end
end
end
end
imagesc(xr)
```

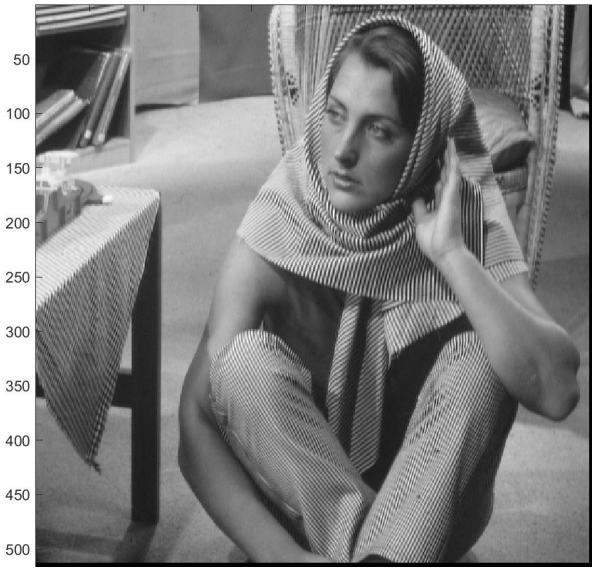

Figure 7: The results that adding the 2.4 and 2.5

3.1Choose one of the gif to compute

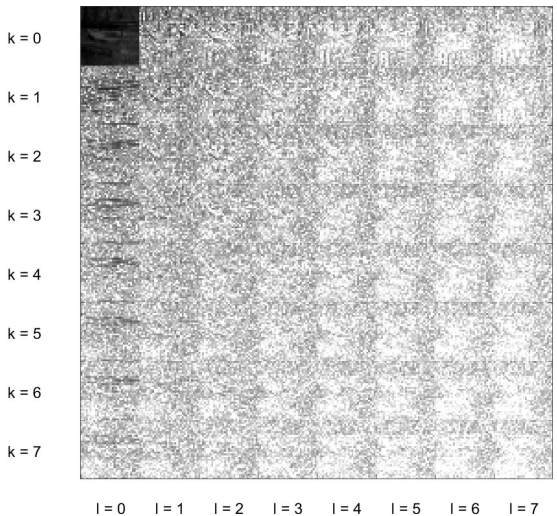

Figure 7: The resulting DCT of “bridge.gif”

3.2 Interpret the DCT in 3.1

From the graph created in 3.1, we can find that the energy is concentrated when k and l are small, the energy concentration is the highest when k=l=0 and it has the most of the energy in the whole image. As the k and l increase the energy concentration decrease. When k and l approach 7, there is only very little energy left, so sometimes it is acceptable when we just abandon the high-frequency coefficients. If we use every DCT term except for the DC term, we will find the image of the bridge is still vague.

3.3 For 0<=k<=3,0<=l<=3

Figure 8: The reconstruction of the image by using one quarter of all the DCT values

3.4 Reconstruct the signal with only 6.25% of the DCT coefficients around the DC

Figure 8: Reconstructed using 6.25% of the coefficients

3.5 Propose a method

When doing the reconstruction in 3.3 and 3.4,an 8 x 8 DCT is performed on each block and the resulting transform coefficients are quantized and entropy coded. This independent processing of blocks does not take into account the between-block pixel correlations. Therefore, at low bit rates, such an encoding scheme typically leads to blocking artifacts, which manifest themselves as artificial discontinuities between adjacent blocks.In this case, when we making the blocks form a between-block pixel correlations, it will help alleviate these artifacts without increasing the number of DCT coefficients used in the reconstruction.

2. Conclusion

In lab6, we have create a 1-D DCT basis function to process the sequence and make a comparison between DCT and DFT. And we extend the 1-D to 2-D DCT to use the DCT to process image and show how the percentage of coefficient will affect the quality of reconstruction. And at last, we try to explore a method to alleviate the artifacts without increasing the number of DCT coefficients used in the reconstruction.

3. Acknowledgments

Thanks for the TA gives us the illustration explain the basic principle of DCT.