Homework 1

Subsampling and Interpolation

2020311381 Sanghyeok Choi

**Goal of Project**: Understanding Linear Transformations

Developing the subsampling/interpolation scheme operated in frequency domain.

From this project, we want to know how we can subsample/interpolate the image in

frequency domain (compressed domain), by multiplying the proper frequency version

of transformation matrix in frequency domain directly.

**Tools :** Subsampling method, Interpolation method, Discrete Fourier Transform

for m, n = 0,1,2,3,.....,N-1,

1. Subsampling method : Simple averaging method, that is

In this case, f is 8 x 8 matrix. Lets set .

Then subsample matrix g is 4 x 4 matrix that

2. Interpolation method : Simple linear interpolation method, that is,

In this case, f is 4 x 4 matrix. Let’s set

f = . Then subsample matrix g is 4 x 4 matrix that

g =

Last row and last column can use previous one.

3. For frequency domain transformation, use Discrete Fourier Transform.

Use Unitary Matrix that U\*UH = I(Identity matrix). UH mean conjugate after transpose U matrix.

UT means transpose U matrix. DFT Image = Unitary Matrix \* Original Image \* Unitary MatrixT

DFT matrix is symmetry, DFT Image = Unitary Matrix \* Original Image \* Unitary Matrix

For example, let’s do DFT in 4 x 4 matrix.

Then Unitary Matrix = = .

When Original matrix is A, DFT matrix is Unitary Matrix \* Original Image \* Unitary Matrix = .

The reason why using unitary matrix is consider inverse DFT. Because of UUH = I, you can do inverse DFT by UH \* (Frequency Matrix) \* UH.H

**Process**

In this project, I use picture of Lenna. And I convert gray scale



This picture is 256 x 256 size, so it needs to divide in small patches. In this experiment I use 8 x 8 patch size that mean 256 x 256 image divided in 1024 8 x 8 patches.

In recommend method, you can subsample 8 x 8 image by S \* Image \* ST.

I get S matrix that

Also you can interpolate 4 x 4 image by I \* Image \* IT. I get matrix that

I = . Consider last row and column is same as previous one.

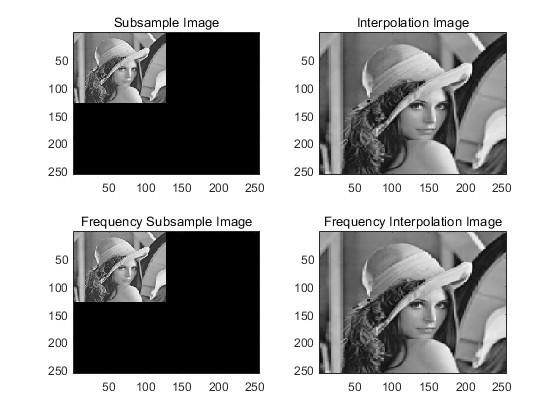
When you use Sf =D4SD8H, you can subsample in frequency domain by Sf\*Frequency matrix\*SfT.

Let’s apply 8 x 8 matrix F8 that DFT applied. Consider D4 and D8 is symmetric.

Then Sf \* F8 \* SfT = D4SD8HF8(D8H)TSTD4T = D4SD8HF8D8HSTD4.

D8HF8D8H mean inverse DFT, so you can get original image I8, D4SD8HF8D8HSTD4 = D4SI8STD4. Also SI8ST is subsampled image I4, D4SI8STD4 = D4I4D4. Now D4I4D4 is apply DFT in subsampled image. As a result, you can subsample in frequency domain by matrix product.

You can do interpolation same way If = D8ID4H. I do this experiment by Matlab tool.



I consider size difference, fill 0 in low resolution figure’s background. Also I compute difference between each similar figures, error is too low that you can ignore.

**Discussion** :

I never thought I can subsample or interpolation in frequency domain. Now I can use this method. S\*Image\*ST ‘s DFT is D4SD8HF8D8HSTD4. I know when product functions in time domain, they convolution in frequency domain. Also when convolution in time domain, they product in frequency domain. I wonder S\*Image\*ST is matrix product, however they do matrix product in frequency domain not convolution. I admire this method that I can apply matrix product in frequency domain.

**Code** :

%% Data clear

clear all; close all; clc % Clear everything

%% Prob 1

Original\_image = imread('Lenna.png');

% Read Original image "Lenna.png"

Original\_image = double(rgb2gray(Original\_image));

% Original image is color, so it need to convert gray image

partition = sub\_images(Original\_image,[256,256],[8,8]);

% Original image is 256 x 256 image, so divide it small patches

%% Prob 2

for i=1:size(partition,3)

% Do DFT in every original image patches and get 8 x 8 DFT unitary matrix

[F8s(:,:,i),D8] = DFT(partition(:,:,i));

end

%% Prob 3

% S is satisfied matrix that f4 = Sf8(S^T).

% f4 is subsampled image that 4 x 4 matrix,

% f8 is original image that 8 x 8 matrix.

S = [0.5, 0.5, 0, 0, 0, 0, 0, 0;...

0, 0, 0.5, 0.5, 0, 0, 0, 0;...

0, 0, 0, 0, 0.5, 0.5, 0, 0;...

0, 0, 0, 0, 0, 0, 0.5, 0.5];

% Get subsampled image in every patches.

% Do DFT in every subsampled image. Also get 4 x 4 DFT unitary matrix.

for i=1:size(partition,3)

Subsampling\_image(:,:,i) = S\*partition(:,:,i)\*S';

[F4s(:,:,i),D4] = DFT(Subsampling\_image(:,:,i));

end

%% Prob 4

% ^H is hermitian matrix that conjugate after transpose original matrix

% Get subsample matrix in frequency version Sf = D4\*S\*(D8^H).

S\_f = D4\*S\*D8';

for i=1:size(partition,3)

% Apply subsample matrix in frequency version in every original patches

Subsample\_frequency\_version(:,:,i) = S\_f\*F8s(:,:,i)\*S\_f.';

end

%% Prob 5

% I is satisfied matrix that f8 = Sf4(Sf^T).

% f4 is original image that 4 x 4 matrix,

% f8 is interpolation image that 8 x 8 matrix.

I = [1, 0, 0, 0;...

1/2, 1/2, 0, 0;...

0, 1, 0, 0;...

0, 1/2, 1/2, 0;...

0, 0, 1, 0;...

0, 0, 1/2, 1/2;...

0, 0, 0, 1;...

0 0 0 1];

% Get interpolation image in every patches.

for i=1:size(partition,3)

Interpolation\_image(:,:,i) = I\*Subsampling\_image(:,:,i)\*I';

end

%% Prob 6

% Get interpolation matrix in frequency version If = D8\*I\*(D4^H).

I\_f = D8\*I\*D4';

for i=1:size(partition,3)

Interpolate\_frequency\_version(:,:,i) = I\_f\*F4s(:,:,i)\*I\_f.';

end

%% Prob 7

% Apply inverse DFT all matrix that get from Prob4 and Prob 6

for i=1:size(partition,3)

Subsample\_image\_by\_frequency(:,:,i) = D4'\* Subsample\_frequency\_version(:,:,i) \* D4';

Interpolation\_image\_by\_frequency(:,:,i) = D8'\* Interpolate\_frequency\_version(:,:,i) \* D8';

end

% Connect each patches for show image

Subsampling\_image = paste(Subsampling\_image,[4,4],[128,128]);

Interpolation\_image = paste(Interpolation\_image,[8,8],[256,256]);

Subsample\_image\_by\_frequency = paste(Subsample\_image\_by\_frequency,[4,4],[128,128]);

Interpolation\_image\_by\_frequency = paste(Interpolation\_image\_by\_frequency,[8,8],[256,256]);

% Consider subsample image is 128 x 128 and others are 256 x 256, fit size

% by fill zeros in background.

Subsampling\_image(129:256,:) = 0;

Subsampling\_image(:,129:256) = 0;

Subsample\_image\_by\_frequency(129:256,:) = 0;

Subsample\_image\_by\_frequency(:,129:256) = 0;

% Show each image

figure(1)

colormap gray; % For show gray image

subplot(2,2,1); imagesc(real(Subsampling\_image));

title("Subsample Image");

subplot(2,2,2); imagesc(real(Interpolation\_image));

title("Interpolation Image");

subplot(2,2,3); imagesc(real(Subsample\_image\_by\_frequency));

title("Frequency Subsample Image");

subplot(2,2,4); imagesc(real(Interpolation\_image));

title("Frequency Interpolation Image");

%% Error between apply image domain and frequency domain

fprintf("Maximum error intensity in subsample method: %f\n\n",...

max(abs(Subsampling\_image(:)-Subsample\_image\_by\_frequency(:))))

fprintf("Maximum error intensity in interpolation method : %f\n\n",...

max(abs(Interpolation\_image(:)-Interpolation\_image(:))))

%% Functions that I made

% Function for make sub images

function partition = sub\_images(input\_image,input\_size,output\_size)

Patch\_x = input\_size(1)/output\_size(1); Patch\_y = input\_size(2)/output\_size(2);

% value for patch number

partition = zeros(output\_size(1),output\_size(2),Patch\_x\*Patch\_y);

% Make initial form for output

for x = 1:Patch\_x

for y = 1:Patch\_y

% Concat each patches in 3rd dimension

partition(:,:,(x-1)\*Patch\_y+y) = ...

input\_image(output\_size(1)\*(x-1)+1:output\_size(1)\*x,output\_size(2)\*(y-1)+1:output\_size(2)\*y);

end

end

end

% Function for paste sub images

function Image = paste(input\_image,input\_size,output\_size)

Patch\_x = output\_size(1)/input\_size(1); Patch\_y = output\_size(2)/input\_size(2);

% value for patch number

Image = zeros(output\_size(1),output\_size(2));

% Make initial form for output

for x = 1:Patch\_x

for y = 1:Patch\_y

% Connect each patches by 3rd dimension

Image(input\_size(1)\*(x-1)+1:input\_size(1)\*x,input\_size(2)\*(y-1)+1:input\_size(2)\*y)...

= input\_image(:,:,(x-1)\*Patch\_y+y);

end

end

end

% Function for DFT

function [Output\_Image,D] = DFT(Input\_Image)

% Check number of dimension. Consider Input\_image is square matrix

N = size(Input\_Image,1);

D = zeros(N,N);

for k=1:N

D(k,:) = (k-1)\*(0:1:N-1)/N;

end

D = exp(-1j\*2\*pi\*D)/sqrt(N);

% Consider DFT matrix is unitary matrix. So divide by sqrt(N)

Output\_Image = D\*Input\_Image\*D;

end

**Reference** :

<https://en.wikipedia.org/wiki/Unitary_matrix>

<https://en.wikipedia.org/wiki/DFT_matrix>

“[Chapter2 of TextbookFile](https://yscec.yonsei.ac.kr/mod/resource/view.php?id=1586373)” in YSCEC

https://ko.wikipedia.org/wiki/레나\_(이미지)