Project 4 : CT Image Reconstruction

생체의공학과
2016110543
김상훈

Abstract

이번 Project에선 medical image를 처리하는 syetem 중 CT의 작동양상 및 image reconstruction에 대해 알아본다. 실제 CT는 안에 대상은 가만히 있고 기계가 돌아가며 image를 얻어내지만, 이번에는 그렇게 할 수 없으니 나만의 image를 회전시켜 data를 얻는다. Image rotating을 위한 function을 개별적으로 만들어 이를 이용한 angle step당 projection data를 얻어 1차적으로 sinogram을 얻어낸다. 그 후 얻어낸 sinogram을 MATLAB의 iradon function을 이용해 back-projectio하여 image를 reconstruction하고, 이 때 filter의 유무와 더 나아가 filter의 종류가 image quality에 어떠한 영향을 미치는지 알아본다. 또한 interpolation의 종류가 image quality에 미치는 영향도 알아본다. 이러한 simulation을 통해 CT image reconstruction을 진행해본다.

Table of Contents

List of Figures and Tables

1. Introduction

CT는 X-ray를 이용하여 우리 몸을 진단하는 medical imaging system이다. CT의 구조와 원리를 이해하고 image 처리 방식을 알아본다. CT image를 얻기 위해선 projection data를 기반으로 한 sinogram을 얻어야 하는데, 이를 위해 image를 회전시켜줄 function을 직접 구현한다. 이때 image rotating을 진행하기 전에 image size를 256×256으로 변환해준다.

Sinogram을 얻은 후에는 image reconstruction을 위해 iradon이라는 function을 이용하여 back-projection을 진행하면서 filtered-back-projection과의 image quality 차이를 확인한다.

또한 5종류의 filter를 각각 적용하여 back-projection을 진행하고 original image와의 MSE 값을 비교하여 내가 선택한 image를 reconstruction 할 때 가장 적합한 filter를 찾아내고 다양한 interpolation 방법에 의한 image quality 차이도 알아본다. 마지막으로 image를 회전할 때 angle step마다 reconstruction 되는 image가 얼마나 차이나는지 확인한다.

2. Methods

CT 스캐너는 인체 내부의 3D image를 재현한다. 스캐너는 서로 마주보는 위치에 있는 X-ray 튜브와 검출기로 구성되며 환자 주위를 돌며 360° image 스캔을 한다. 환자가 스캐너 안으로 이동하는 동안 튜브에서 방출된 X-ray가 신체를 통과하며 검출기 array는 반대편에서 감쇠된 X-ray 광자를 캡처하여 여러 각도에서 데이터를 수집한다.

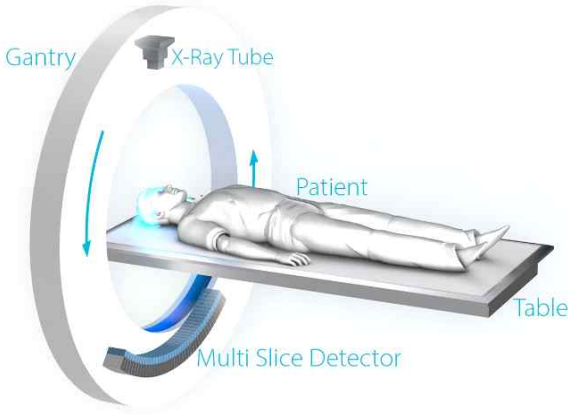

Figure 1.1 CT 스캐너의 구조[1]

그러나 이번 Project에서의 CT image를 모델링하기위해서는 정해놓은 image를 회전시키며 데이터를 수집하기 때문에 image를 회전시키기 위한 function이 필요하다. image rotation에 관한 matrix는 Eq. (1.1)과 같다.

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_o \\ y_o \end{bmatrix} \tag{1.1}$$

where, x_o : original image's x coordinate

x_n : image's x coordinate after rotation

y_o : original image's y coordinate

y_n : image's y coordinate after rotation.

회전을 하면 sin 및 cos 함수 때문에 좌표가 integer에서 floating point number로 바뀌기 때문에 변환 후의 좌표를 integer로 구하기 위해선 역행렬을 사용해 수식을 Eq. (1.2)처럼 다시 정리할 수 있다.

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_o \\ y_o \end{bmatrix} \tag{1.2}$$

Figure 1.2처럼 Eq. (1.2)에 대입하고 역 추정을 이용하여 원래 좌표를 mapping하는 방식으로 진행한다.

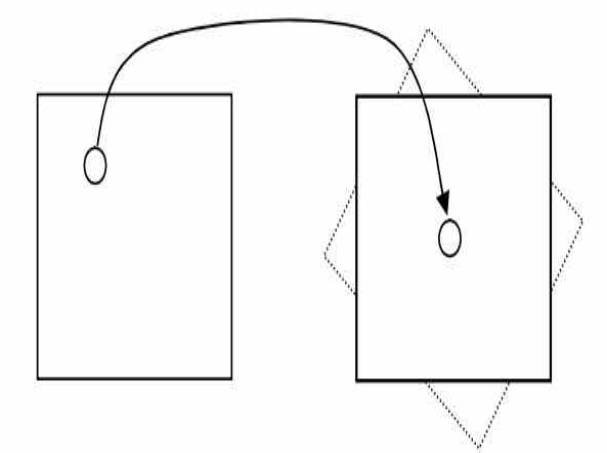

Figure 1.2 Mapping of pixel intensities

이 때 Figure 1.2와 같은 방식으로 역 추정을 이용하여 mapping해서 얻은 값은 마찬가지로 integer이 아니기 때문에 그 값을 효율적으로 찾기 위해 Figure 1.3과 같은 Bilinear Interpolation 기법을 이용한다.

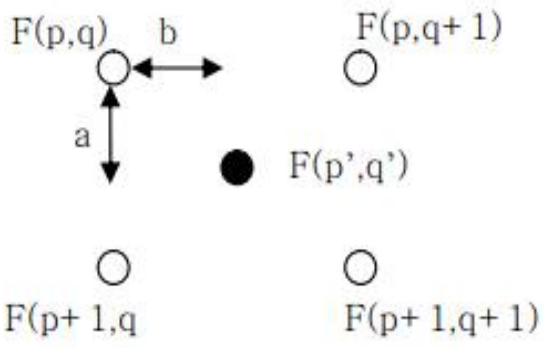

Figure 1.3 Bilinear Interpolation

보간법(interpolation)이란 알려진 값을 가진 두 점 사이 어느 지점의 값이 얼마인지 추정하는 기법을 의미한다. Image rotation에 사용되는 Bilinear Interpolation은 가운데 위치한 회전한 image의 원래 좌표를 주변에 가까운 4개 integer의 평균값으로

추정하는 방식으로써 다음과 같은 수식으로 나타낼 수 있다.

$$F(p',q') = (1-a)[(1-b)F(p,q)+bF(p,q+1)]+a[(1-b)F(p+1,q)+bF(p+1,q+1)] \quad (1.3)$$

where, p : 역 추정된 유리수와 가장 가까운 정수 값
p' : 역 추정된 실제 유리수 값
q : 역 추정된 유리수와 가장 가까운 정수 값
q' : 역 추정된 실제 유리수 값

이러한 방식으로 image를 0°에서 180°에서 데이터를 얻으면 물체의 투과정도에 따라 데이터의 누적량이 다르기에 각 각도에 따라 쌓아서 한 장의 단면도로 나타내면 Figure 1.4와 같은 sinogram을 얻을 수 있다.

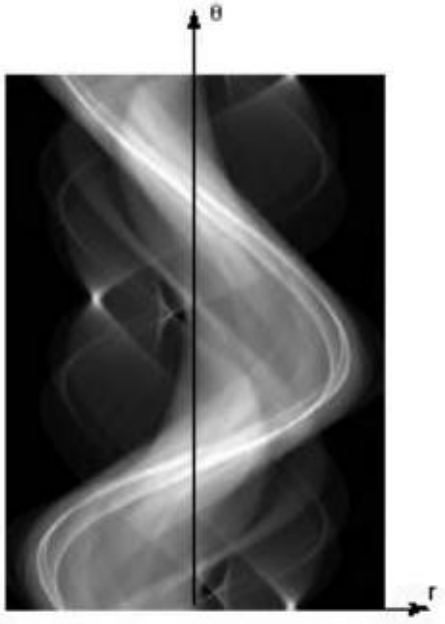

Figure 1.4 Sinogram

Sinogram을 얻으면 image reconstruction을 위해 back-projection을 진행해야 한다. Back-projection이란 얻은 projection data들을 역으로 다시 쏘아주는 것인데, 이 때 reconstruction image는 실제 image보다 크기가 크며 단점으론 blurring(star-like artifacts)가 발생하며 Figure 1.5가 이를 보여준다. 즉, 원래의 신호가 0이었던 부분도 back projection을 진행함으로써 data 값을 가지게 된다. 이러한 현상 때문에 reconstruction image의 spatial resolution이 크게 감소하게 된다.

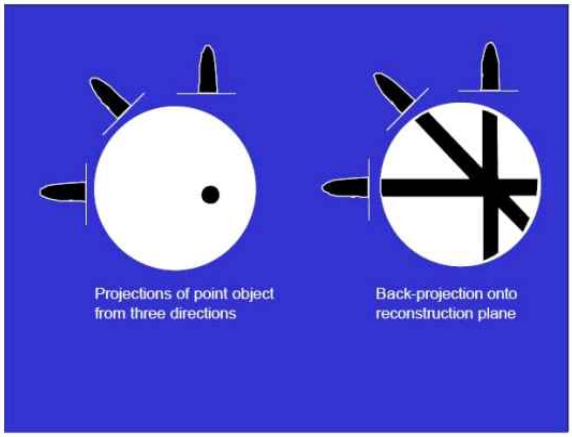

Figure 1.5 Non-filtered-back-projection

이를 보완하며 image quality를 높여줄 수 있는 방식이 바로 filtered-back-projection이다. 아래 Figure 1.6을 보면 filter를 사용하여 reconstruction한 image가 훨씬 선명하게 나온 것을 볼 수 있다. 이는 High-Pass-Filter를 이용하여 저주파 대역의 신호를 제거하여 blurring을 없애주고 image의 edge를 더욱 선명하게 보여주는 것이다.

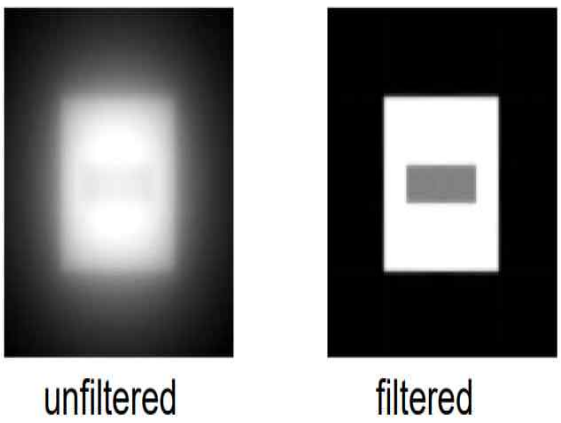

Figure 1.6 Filter의 유무에 따른 image quality의 비교

즉, background에서 (+)ripple과 (-)ripple이 만나면서 상쇄작용이 일어나 reconstruction image에서 edge가 살아나게 된다.

3. Results & Discussion

Exercise 1

Create projection data with $\Delta\theta = 1^\circ$ from 0 to 180 degree. Display the projection data (i.e., sinogram) as an image. In simulation, the projections from 0 to 180 degrees will do, but not in real situations, why? Discuss

Exercise 1에서는 본인이 정한 image를 0°에서 180°까지 1°마다 회전시켜 projection data를 기반으로 한 sinogram을 얻는 활동이다. Image를 회전 시킬 때는

$$x_n = x_o\cos\theta - y_o\sin\theta$$
$$y_n = x_o\sin\theta + y_o\cos\theta \qquad (2.1)$$

Eq. (2.1)을 기반으로 각각의 각도에서 얻은 projection data는 sinogram array에 저장한다. Figure 2.1은 크기가 512×512인 원본 image를 보여준 것이며 직사각형 형태를 가지고 있다. 아래 Figure 2.2는 원본 image를 256×256으로 변환해주고 rotate function에 의해 최종적으로 180° 회전된 image이다.

Figure 2.1 512×512 sized original image

Figure 2.2 원본 image를 256×256 size로 변환하고 흑백으로 설정한 뒤 180° 회전시킨 image

Figure 2.3은 sinogram을 보여주는 그림인데, 256개의 projection data를 0°에서 180°까지 회전시키며 181번 얻어온 것이기 때문에 크기가 256×181이 된다.

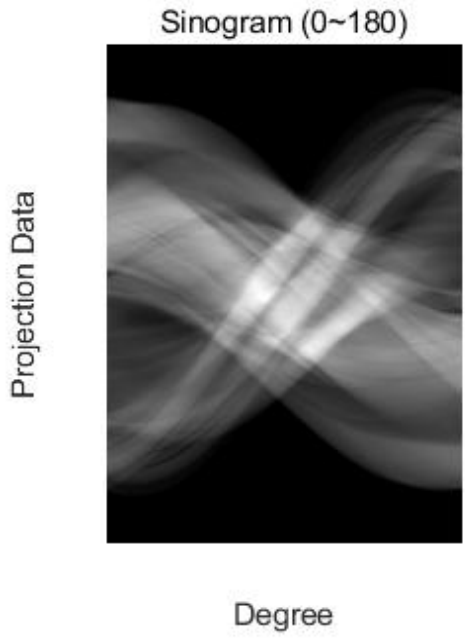

Figure 2.3 Sinogram of original image

Figure 2.3을 보면 왜 회전을 360°가 아닌 180°로 하는 것이다. Sinogram 행렬의 행은 각 단위 각도마다 돌았을 때 projection data 181개를 나타내며, 열은 0°부터 180°를 나타낸다. 360°로 회전시키게 되면 중복된 data가 생기기 때문에 360° rotation이 아닌 180° rotation을 사용한다.
위의 Figure 2.3에서 확인할 수 있듯이, 원본 image의 sinogram에서 좌우는 정상적으로 회전이 되는 것을 관찰할 수 있지만, 상하는 image가 잘린 느낌이 난다. 위의 Figure 2.2에서 원본 image를 256×256으로 변환하였지만 좌우는 회전할 때 공간이 있어서 안정적으로 회전하는 반면, 상하는 그림이 빼곡이 차있으므로 공간이 부족하기에 잘리게 된다.

Exercise 2

Reconstruct your image using the backprojection method only. Display the original image and reconstructed image, Compare the mean square error. Discuss on the reconstructed image and its quality and errors. Hint: use iradon.m for reconstruction, but you will need to disable the digital filter.

Exercise 2는 Exercise 1에서 얻은 sinogram으로 image를 reconstruction 하기 위해 back-projection을 진행하는 활동이다. Back-projection을 진행하면 sinogram의 data들이 projection시에 attenuation된 부분만 복원이 되는 것이 아니라 모든 지점에 복원이 되기 때문에 blurring이 일어난 image로 복원이 된다.
아래 Figure 3.1과 Figure 3.2를 비교해보면 그 차이를 확실하게 알 수 있다. Figure 3.1에선 image의 경계가 명확한 사람 모양을 나타내고 있는 반면, Figure 3.2에선 image의 경계가 확실히 보이지 않고 blurring 현상이 일어나서 알아볼 수 없게 왜곡되어 있다.

Figure 3.1 512×512 sized original image

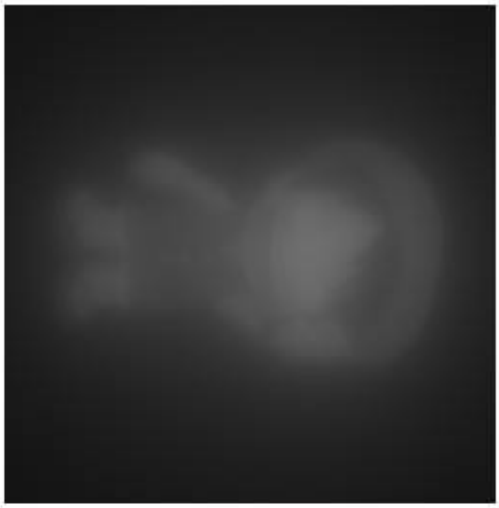

Figure 3.2 Non-filtered-back-projection image

Table 1.1 Original image와 back-projection image 사이의 MSE 값

Name	Value
MSE_Back-Projection	101.4704

위의 Table 1.1에서 확인할 수 있듯이, 이러한 점을 수치적으로 확인하기 위해 original image와 back-projection image 사이의 MSE 값을 구한 결과 약 101.4704이라는 매우 큰 값이 나왔고, 이는 원본 image가 제대로 복원되지 않았음을 뜻한다.

Exercise 3

You are going to reconstruct your original image using iradon.m (i.e., filtered back-projection), but before reconstructing the image

A. iradon.m has 5 filter options. Plot their filter responses, and compare their characteristics.

B. Choose the best filter for your image based on the mean square error. Plot one projection before and after filtering in one figure. What seems to be an effect of filtering? Discuss about the effect of digital filtering.

Exercise 3에서는 Exercise 2와는 달리 non-filtered-back-projection이 아니라 iradon.m에서 구현할 수 있는 5가지의 filter를 사용하여 back-projection을 진행하였다. Iradon function에서는 H1부터 H5까지의 filter를 구현할 수 있는데, 각각의 filter는 아래의 Table 2.1과 같다.

Table 2.1 Filter name of H1~H5

Iradon Function	Filter Name
H1	Ram-Lak filter
H2	Shepp-Logan filter
H3	Hamming filter
H4	Hann filter
H5	Cosine filter

아래 Figure 4.1부터 Figure 4.10은 각각의 filter들을 주파수 대역에 따라 그래프와 적용햇을 때의 image를 나타낸 것이다. 여기서 x축의 frequency는 원본의 image를 복원하기 위한 최소 비율은 Nyquist Ratio를 기준으로 하고 있고 y축은 filter의 gain 값을 나타낸다.

모든 filter들은 250Hz를 기준으로 대칭적이고 High-Pass-Filter와 비슷한 역할을 한다. Ram-Lak filter는 sharp한 특성을 띠고 있으며 spatial resolution이 좋다는 장점이 있다, 그러나 단점으로는 noise가 심한데 이를 보완한 filter가 Shepp-Logan filter로 Ram-Lak filter와는 다르게 smooth하다. 하지만 이 filter 또한 Gibb's phenomenon을 유발할 수 있다는 단점이 있다. Hamming filter의 경우 Shepp-Logan filter와 같이 smooth하지만 spatial resolution이 낮다는 단점이 있다. Hann filter와 Cosine filter 경우 Hamming filter와 유사하지만 중간 대역의 주파수를 어떻게 깎아주는가에 따라 나뉘는 것을 알 수 있다.

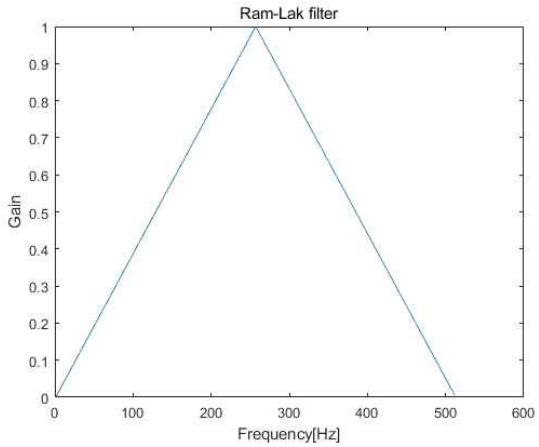

Figure 4.1 Ram-Lak filter

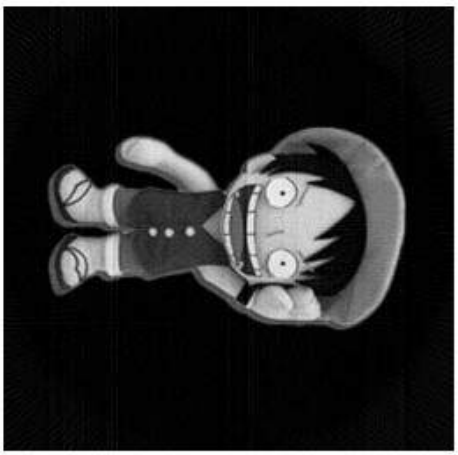

Figure 4.2 Ram-Lak filter를 이용한 filtered-back-projection image

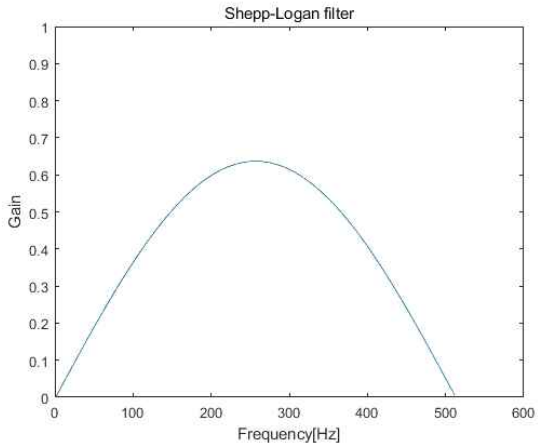

Figure 4.3 Shepp-Logan filter

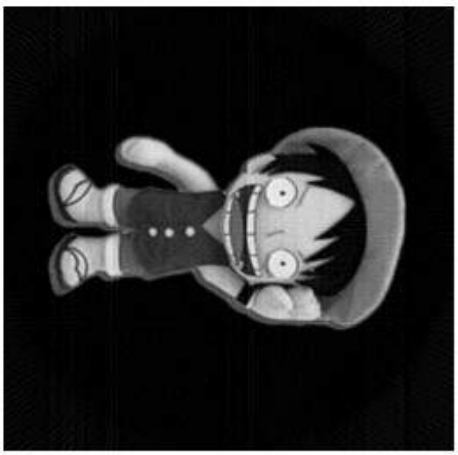

Figure 4.4 Shepp-Logan filter를 이용한 filtered-back-projection image

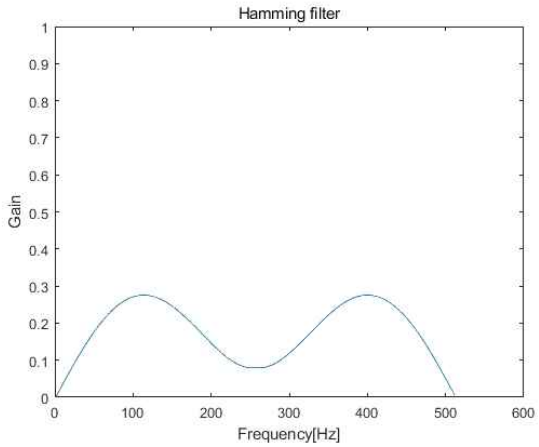

Figure 4.5 Hamming filter

Figure 4.6 Hamming filter를 이용한 filtered-back-projection image

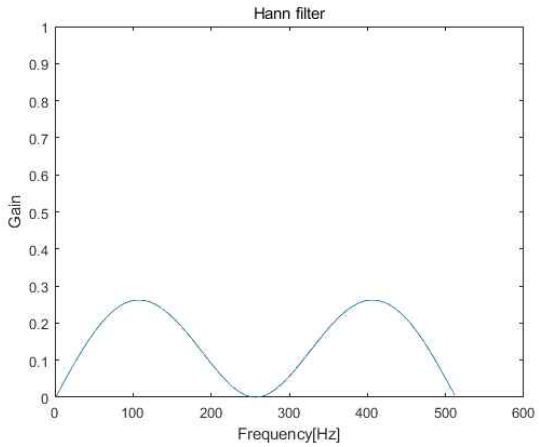

Figure 4.7 Hann filter

Figure 4.8 Hann filter를 이용한 filtered-back-projection image

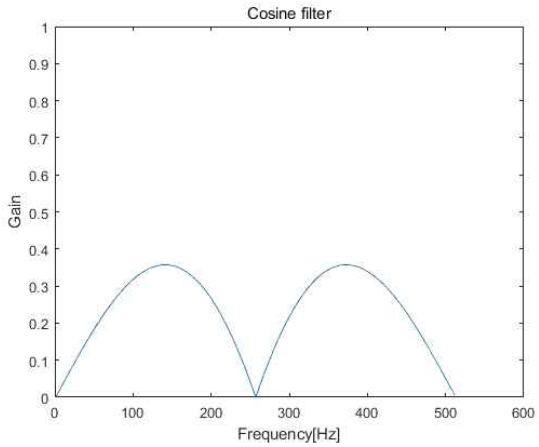

Figure 4.9 Cosine filter

Figure 4.10 Cosine filter를 이용한 filtered-back-projection image

Filtered-back-projection을 진행했을 때 Exercise 2보다 훨씬 선명하게 spatial resolution이 향상된 것을 확인할 수 있다. 하지만 optimal한 filter를 찾기에는 육안으로 큰 차이를 구별할 수 없다. 따라서 아래의 Table 2.2를 보면 가장 optimal한 filter를 찾기 위해 MSE 값을 구한 결과를 확인할 수 있다. Filter를 적용하여 얻은 image와 원본 image를 비교하여 구한 MSE 값이 가장 작은 것은 Hann filter를 적용했을 때로 그 값은 약 100.0304이고 가장 큰 값은 Ram-Lak filter로 약 204.2605이다. 따라서 image에 가장 optimal한 filter는 Hann filter라는 것을 알 수 있다.

Table 2.2 Filter마다 back-projection을 진행했을 때 원본 image와의 MSE 값

Filter Name	MSE Value
Ram-Lak filter	204.2605
Shepp-Logan filter	171.8804
Hamming filter	105.8004
Hann filter	100.0304
Cosine filter	123.7304

Exercise 4

Reconstruct your original image with the sinogram from Exercise 1. Try the best digital filter and all interpolation options of iradon.m in reconstruction and find the best interpolation option again in the mean square error sense. What is the effect of the interpolation options for reconstruction. (you should be careful with the size of the reconstructed image and the original image. Check the image size.)

Exercise 4는 interpolation(보간법) 방식의 차이에 따른 image quality의 변화를 알아보는 활동이다. Exercise 3에서 보았듯이 가장 optimal한 filter는 Hann filter이기 때문에 filter를 고정한 상태에서 각각 'nearest', 'linear', 'spline', 'pchip', 'cubic', 'v5cubic'의 6가지 interpolation 방식을 이용하여 iradon function을 진행하였다. 주의해야할 점은 size가 256×256인 원본 image와는 달리 interpolation을 바꾸면서 iradon을 진행할 경우 reconstruction image의 size는 180×180이 되기 때문에 iradon function을 이용할 때 size를 256으로 설정해주었다. 각각의 interpolation을 하였을 때 image quality를 수치적으로 확인하기 위해 MSE 값을 구한 결과는 아래 Table 3.1과 같다.

Table 3.1 Interpolation 별 MSE 값

Interpolation Option	MSE Value
nearest	101.1004
linear	100.0003
spline	100.8404
pchip	100.4804
cubic	100.3807
v5cubic	100.7704

가장 작은 MSE 값을 보이는 interpolation option은 linear으로써 그 값은 약 100.0003이고, 가장 큰 MSE 값을 보이는 interpolation option은 nearest로써 그 값은 약 101.1004이다. 결과적으로 내 image quality를 가장 좋게 할 수 있는 interpolation option은 linear임을 알 수 있으며 filter의 종류만 바꿨을 때보다 interpolation option까지 같이 조정했을 때 MSE가 모두 감소하였다.

Exercise 5

Create projection data with THREE different values of the rotation angle. Reconstruct the image. What is the effect of the number of projections? Discuss the effect of increasing the number of projections. What would you say on the effect of increasing the projections (i.e., more angular views)? You should relate to Fourier slice theorem.

Exercise 5에서는 image를 회전시킬 때 angle step을 다르게 하여 reconstruction을 진행하는 활동이다. Exercise 4까지는 angle step을 1°로 고정시키고 181개의 projection data를 기반으로 reconstruction을 진행하였지만 이번 Exercise 5에선 angle step을 1° 말고도 5°, 10°로 바꾸어 Hann filter, linear interpolation을 진행하였다. 아래의 Figure 5.1에서 Figure 5.3까지를 보면 angle step이 커질수록 image quality가 점점 낮아지는 것을 확인할 수 있다. 이는 angle step이 커질수록 projection data가 181개에서 37개, 19개로 점점 줄어들고 이로 인한 reconstruction 또한 적은 data 양을 기반으로 진행되기 때문에 spatial resolution이 낮아져 image가 왜곡되어 결과적으로 quality가 낮아지게 되는 것이다.

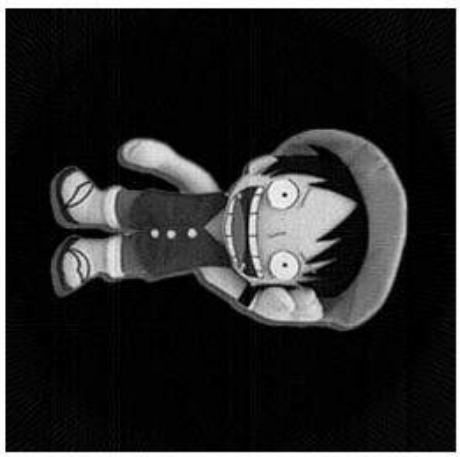

Figure 5.1 Reconstruction image when angle step is 1°

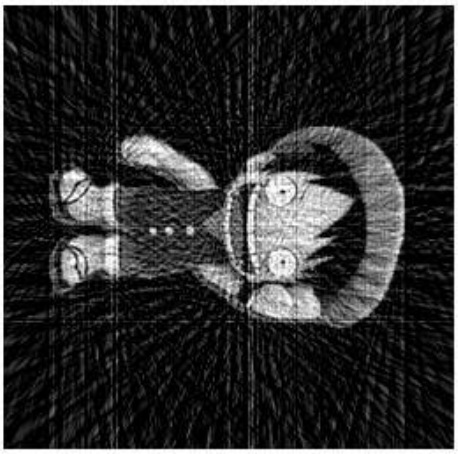

Figure 5.2 Reconstruction image when angle step is 5°

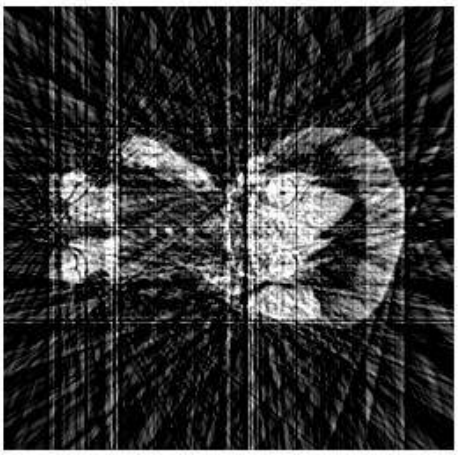

Figure 5.3 Reconstruction image when angle step is 10°

영상을 찍을 때, angle step을 너무 촘촘하게 하면 처리해야하는 data가 많아지기 때문에 image quality가 너무 나빠지지 않는 선에서 적당한 angle step을 찾아 회전해야 가장 효율적으로 reconstruction image를 얻을 수 있을 것이다.

Exercise 6

For the best reconstruction options out of Exercises 3 and 4, try image reconstruction by iradon on GPU. Compare the gains of performing image reconstruction between CPU vs. GPU (i.e., w/ GPU vs. w/o GPU).

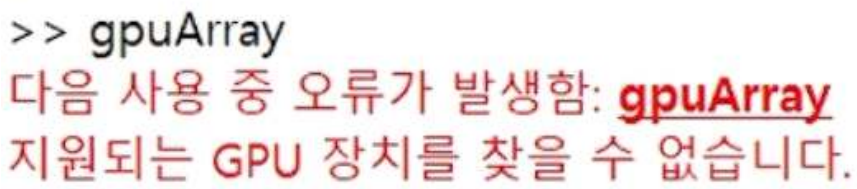

Figure 6.1 Error

4. Conclusion

이번 Project에서는 X-ray가 물체를 통과할 때 생기는 linear attenuation coefficient를 이용하여 우리 몸을 진단하는 의료영상기기인 CT를 modeling하고 simulation 하였다. 대상은 가만히 있고 기계가 돌아가며 image를 얻어내는 실제 CT와는 다르게, 내가 선택한 image를 직접 회전시켜 projection data를 얻어냈다.

Image를 회전시키기 위한 function을 구현하여 projection data의 집합인 sinogram을 얻어낼 수 있었다. Sinogram을 얻은 후 다시 back-projection하는 과정에서 image의 size가 감소하는데 이는 iradon function의 option으로 원래의 크기로 맞추어 줄 수 있었다. 또한, back-projection 시에 filter의 유무에 따른 image quality 차이를 알아보고, filter의 종류와 interpolation option에 따른 MSE 값의 차이를 기반으로 내가 선택한 image의 spatial resolution을 가장 좋게 만들 수 있었다. 마지막으로 image를 회전할 때 angle step의 차이에 따라 image가 얼마나 왜곡되는가에 대해서도 직접 확인하였다.

Image를 회전시키며 직접 projection data를 얻고 그것을 reconstruction함으로써 원래의 image를 다시 만들어내는 과정에서 CT의 작동 원리에 대해 더욱더 이해할 수 있었다.

References

[1] https://ams.com/ko/how-a-ct-works, 2012.

[2] http://www.mathworks.co.kr, 2022.

[3] PM_6_Biomedical Systems Modeling lecture

Appendices

Exercise 1

```
img_original = imread('0429.jpg');
image = imresize(img_original,[256,256]);
image = rgb2gray(image);
image_convert = im2double(image);
figure(1), imshow(image_convert);
step=1;
sinogram = [];
for theta = 0:step:180
    radian = (theta*pi)/180;
    image_rotation = rotate(image_convert,radian);
    projection_data = sum(image_rotation,2);
    sinogram = uint8([projection_data sinogram]);
end
figure(2), imshow(image_rotation);
figure(3), imshow(sinogram);
```

Exercise 2

```
theta2=0:1:180;
Back_projection = iradon(im2double(sinogram),theta2,'linear','none',256);
figure(4), imshow(Back_projection);
for theta3 = 0:step:270
    radian = (theta3*pi)/180;
    image_rotation2 = rotate(image_convert,radian);
end
ssq_back_projection=0;
for i=1:256
    for j=1:256
ssq_back_projection=ssq_back_projection+(image_rotation2(i,j)-Back_projection(i,j))^2;
    end
end
```

Exercise 3A

```
[filtered_Back_projection_Ram_Lak,H1] =
iradon(sinogram,theta2,'linear','Ram-Lak',256);
figure(6), imshow(filtered_Back_projection_Ram_Lak);
[filtered_Back_projection_Shepp_Logan,H2] =
iradon(sinogram,theta2,'linear','Shepp-Logan',256);
```

Appendices

```
figure(7), imshow(filtered_Back_projection_Shepp_Logan);
[filtered_Back_projection_Hamming,H3] =
iradon(sinogram,theta2,'linear','Hamming',256);
figure(8), imshow(filtered_Back_projection_Hamming);
[filtered_Back_projection_Hann,H4] = iradon(sinogram,theta2,'linear','Hann',256);
figure(9), imshow(filtered_Back_projection_Hann);
[filtered_Back_projection_Cosine,H5] = iradon(sinogram,theta2,'linear','Cosine',256);
figure(10), imshow(filtered_Back_projection_Cosine);
figure(11), plot(H1), xlabel('Frequency[Hz]'), ylabel('Gain');
axis([0, 600, 0, 1]);
title('Ram-Lak filter');
figure(12), plot(H2), xlabel('Frequency[Hz]'), ylabel('Gain');
axis([0, 600, 0, 1]);
title('Shepp-Logan filter');
figure(13), plot(H3), xlabel('Frequency[Hz]'), ylabel('Gain');
axis([0, 600, 0, 1]);
title('Hamming filter');
figure(14), plot(H4), xlabel('Frequency[Hz]'), ylabel('Gain');
axis([0, 600, 0, 1]);
title('Hann filter');
figure(15), plot(H5), xlabel('Frequency[Hz]'), ylabel('Gain');
axis([0, 600, 0, 1]);
title('Cosine filter');
```

Exercise 3B

```
SSQ_filter_difference = zeros(5,1);
for i = 1:256
    for  j = 1:256
       SSQ_filter_difference(1,1) = SSQ_filter_difference(1,1)+ (image_rotation2(i,j) -
filtered_Back_projection_Ram_Lak(i,j))^2;
    end
end
for i = 1:256
    for  j = 1:256
       SSQ_filter_difference(2,1) = SSQ_filter_difference(2,1)+ (image_rotation2(i,j) -
filtered_Back_projection_Shepp_Logan(i,j))^2;
    end
end
```

Appendices

```
for i = 1:256
    for  j = 1:256
        SSQ_filter_difference(3,1) = SSQ_filter_difference(3,1)+ (image_rotation2(i,j) -
filtered_Back_projection_Hamming(i,j))^2;
    end
end
for i = 1:256
    for  j = 1:256
        SSQ_filter_difference(4,1) = SSQ_filter_difference(4,1)+ (image_rotation2(i,j) -
filtered_Back_projection_Hann(i,j))^2;
    end
end
for i = 1:256
    for  j = 1:256
        SSQ_filter_difference(5,1) = SSQ_filter_difference(5,1)+ (image_rotation2(i,j) -
filtered_Back_projection_Cosine(i,j))^2;
    end
end
```

Exercise 4

```
interpol_near = iradon(sinogram,theta2,'nearest','Ram-Lak',256);
interpol_lin = iradon(sinogram,theta2,'linear','Ram-Lak',256);
interpol_spl = iradon(sinogram,theta2,'spline','Ram-Lak',256);
interpol_pch = iradon(sinogram,theta2,'pchip','Ram-Lak',256);
interpol_cub = iradon(sinogram,theta2,'cubic','Ram-Lak',256);
interpol_v5cu = iradon(sinogram,theta2,'v5cubic','Ram-Lak',256);
SSQ_interpolation_difference = zeros(6,1);
for i = 1:256
    for  j = 1:256
        SSQ_interpolation_difference(1,1)      =      SSQ_interpolation_difference(1,1)+
(image_rotation2(i,j) - interpol_near(i,j))^2;
    end
end
for i = 1:256
    for  j = 1:256
        SSQ_interpolation_difference(2,1)      =      SSQ_interpolation_difference(2,1)+
(image_rotation2(i,j) - interpol_lin(i,j))^2;
    end
```

```
end
for i = 1:256
    for  j = 1:256
        SSQ_interpolation_difference(3,1)      =      SSQ_interpolation_difference(3,1)+
(image_rotation2(i,j) - interpol_spl(i,j))^2;
    end
end
for i = 1:256
    for  j = 1:256
        SSQ_interpolation_difference(4,1)      =      SSQ_interpolation_difference(4,1)+
(image_rotation2(i,j) - interpol_pch(i,j))^2;
    end
end
for i = 1:256
    for  j = 1:256
        SSQ_interpolation_difference(5,1)      =      SSQ_interpolation_difference(5,1)+
(image_rotation2(i,j) - interpol_cub(i,j))^2;
    end
end
for i = 1:256
    for  j = 1:256
        SSQ_interpolation_difference(6,1)      =      SSQ_interpolation_difference(6,1)+
(image_rotation2(i,j) - interpol_v5cu(i,j))^2;
    end
end
```

Exercise 5

```
img_original = imread('0429.jpg');
image = imresize(img_original,[256,256]);
image = rgb2gray(image);
image_convert = im2double(image);
sinogram = [];
step = 1;
 for theta = 0:step:180
             radian = (theta*pi)/180;
             image_rotation = rotate(image_convert,radian);
             projection_data = sum(image_rotation,2);
             sinogram = uint8([projection_data sinogram]);
```

Appendices

```
  end
theta2=0:step:180;
[filtered_Back_projection_Ram_Lak1,H1]                                                    =
iradon(sinogram,theta2,'spline','Ram-Lak',256);
figure(), imshow(filtered_Back_projection_Ram_Lak1);
```

#rotate

```
function [ rotate_image ] = rotate( original_image, radian )
 [x0,y0] = size(original_image);
rotate_image = [];
 for i=1:x0
     for j=1:y0
          xn = (i-x0/2)*cos(radian) - (j-y0/2)*sin(radian) +x0/2;
          yn = (i-x0/2)*sin(radian) + (j-y0/2)*cos(radian) +y0/2;

          if(xn<1 || yn<1 || xn>x0-1 || yn>y0-1)
               rotate_image(j,i) = 0;
          else
               p=floor(xn);
               q=floor(yn);
               a=xn-p;
               b=yn-q;

rotate_image(j,i)=(1-b)*((1-a)*original_image(q,p)+a*original_image(q,p+1))+b*((1-a)*ori
ginal_image(q+1,p)+a*original_image(q+1,p+1));
          end
     end
end
```

#angle

```
clear all;
close all;
 img_original = imread('111.jpg');
image = imresize(img_original,[256,256]);
image = rgb2gray(image);
image_convert = im2double(image);
  sinogram = [];
step = 15;
```

Appendices

```
for theta = 0:step:180
            radian = (theta*pi)/180;
            image_rotation = rotate(image_convert,radian);
            projection_data = sum(image_rotation,2);
            sinogram = uint8([projection_data sinogram]);
  end
theta2=0:step:180;
 [filtered_Back_projection_Hann,H1] = iradon(sinogram,theta2,'spline','Hann',256);
figure(16), imshow(filtered_Back_projection_Hann)
```