ISSN 1975-8359(Print) / ISSN 2287-4364(Online)
The Transactions of the Korean Institute of Electrical Engineers Vol. 66, No. 6, pp. 942~948, 2017
http://doi.org/10.5370/KIEE.2017.66.6.942

Ray와 Voxel 교차 길이 반복성 기반 고속 Projection 영상 생성 기법

Fast Computation of Projection Image Based on the Repeated Patterns of Intersection between Ray and Voxel

이 현 정* · 김 정 태†
(Hyunjeong Lee · Jeongtae Kim)

Abstract - Ray-tracing based method for computing projection image calculates the exact amounts of the intersection between voxels and a ray. Among several different implementations of the ray tracing based methods, Siddon's method is the earliest one. Later faster implementation such as Jacobs's method, Zhao's method, were investigated. To our knowledge, Zhao's method is the fastest one among these. We improve the speed of the Zhao's method by predicting the number of the same intersection length between voxel and a ray. In our experiment, the proposed method showed significantly faster computation speed than Zhao's method.

Key Words : Low dose CT, Projection, Ray-tracing, Iterative image reconstruction

1. 서 론

방사선 피폭량을 줄일 수 있는 저선량 CT(Computed Tomography)의 고성능 영상 복원을 위해서는 반복적 영상 복원이 필수적이다 [1]. 반복적 영상복원 기법은 FBP (Filtered Back Projection) 기법에 비해 영상 복원의 성능은 우수하지만 연산시간이 많이 소요되는 문제가 있다. 특히 프로젝션(projection)과 백프로젝션(back projection) 연산이 반복적으로 실행되므로 이 연산의 고속화는 매우 중요한 연구 주제이다[5][6].

프로젝션 영상 생성을 위한 방법으로는 ray-tracing[7], pixel-driven[2], ray-driven[3], distance-driven[4], separable footprint [5]와 같은 기법들이 있다. 다른 기법들과 비교해 볼 때, ray-tracing 방법은 detector의 면적을 고려하지 못하기 때문에 백프로젝션을 수행할 때 성능 저하를 일으키는 문제점을 가지고 있다. 그러나, 다른 방법들에 비해서 ray와 voxel의 교차 길이를 근사 값이 아닌 정확하게 구할 수 있는 장점이 있어서 detector의 면적이 작은 경우에는 우수한 성능을 보일 수 있다. 특히 dental CT의 경우에는 detector의 resolution 이 높기 때문에, 프로젝션 및 백프로젝션 계산 시 유용하게 사용될 수 있다 [6]. Ray-tracing을 사용하여 프로젝션을 정확히 계산하는 방법으로는 Siddon 방법이 [7] 최초로 제안되었으며, 이를 기반으로 속도를 개선한 Jacobs 방법[8], Zhao 방법[9] 등이 연구되어 왔다. Siddon 방법은 ray와 각 voxel의 교차 길이를 정확히 구한 후 다시 ray가 지나는 픽셀의 index 를 구해서 voxel의 값을 곱한 값을 누적하는 과정을 통해 프로젝션 영상을 생성한다[7]. Siddon 방법은 최초의 연구로서 큰 의미를 지니고 있으나 voxel의 index 를 구하는 과정이 교차 길이를 구하는 과정이후에 다시 반복되는 redundancy 를 가지고 있다[8].

계산을 간단하게 하는 Jacobs 방법 [8]이 제안되었으며, 이를 개선한 Zhao 방법[9]이 다시 연구되었다. Zhao 방법은 ray가 통과할 때 동일한 x좌표, y좌표 혹은 z좌표를 가지는 voxel 들과 교차길이가 대부분 일정한 값을 가지는 점을 이용하여 길이 및 voxel index 를 계산하여 계산 속도를 향상시켰다. 그러나 이 방법도 ray가 통과하는 voxel의 index 변화를 매번 비교 연산을 통해 확인하는 필요하지 않은 연산을 반복 수행하는 부분이 있다.

본 논문에서는 Zhao 방법에서 불필요한 비교 연산을 제거함으로써 계산 시간을 더 단축할 수 있는 알고리즘을 제안한다. 본 논문의 구성은 다음과 같다. 먼저, 2장에서는 선행연구들의 분석과 Zhao 방법을 기반으로 하여 비교 연산을 줄여 계산 시간을 단축하는 알고리즘을 제안한다. 3장에서는 MATLAB의 Shepp-Logan phantom 영상에 제안하는 알고리즘을 적용하여 프로젝션 영상을 생성한 계산 시간 결과와 Zhao 방법을 적용하여 나온 계산 시간 결과를 비교하고, 4장에서는 결론을 도출한다.

† Corresponding Author : Department of Electronics Engineering, Ewha Womans University, Seoul, Korea
E-mail: jtkim@ewha.ac.kr
* Department of Electronics Engineering, Ewha Womans University, Seoul, Korea
Received : August 9, 2017; Accepted : March 29, 2017

2. 제안하는 알고리즘

2.1 선행 연구

프로젝션 영상을 계산하는 ray-tracing 방법은 픽셀과 ray의

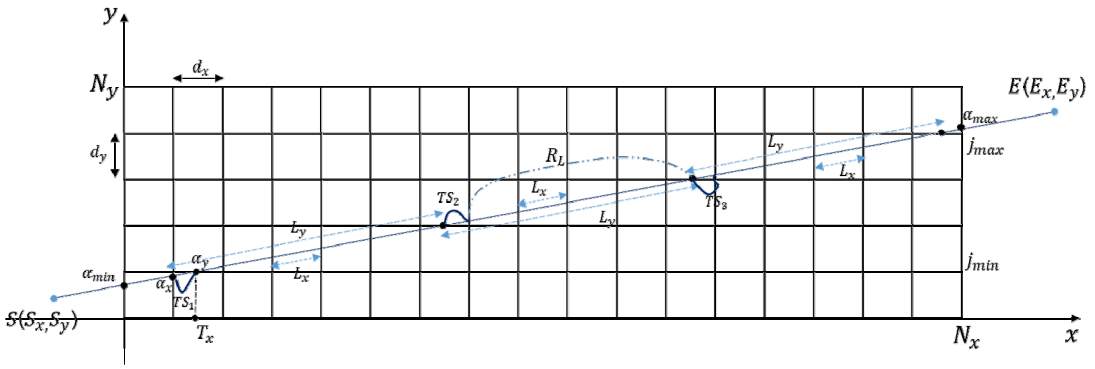

그림 1 Ray-tracing을 이용한 프로젝션 geometry
Fig. 1 Geometry for ray-tracing projection

교차하는 길이와 픽셀을 곱하여 합하는 방법이다. 이를 위하여 ray와 픽셀의 교차하는 길이를 정확히 구하는 최초의 연구는 Siddon 기법이다[7]. 그림 1은 2 차원 영상에서 선적분 계산이 수행되는 과정의 일부를 나타낸다. 이 그림은 영상의 크기가 $N_x \times N_y$ 일 때, ray의 기울기가 $|d_y/d_x|$ 보다 작은 경우를 나타낸다. 이때 d_x, d_y는 x 축과 y 축 간의 간격을 의미한다. 본 논문에서는 수식을 간결하게 설명하기 위하여 2차원 영상을 기준으로 알고리즘을 설명한다. 설명하는 알고리즘은 3 차원 영상에 대해서도 차원을 하나 추가하여 단순하게 확장 가능하다.

Siddon 기법은 ray와 영상의 픽셀을 구성하는 경계선들과 교차점을 계산하여 길이를 구하는 방법이다. 이를 위해서 source인 점 S에서부터 detector 상의 한 점 E까지 연결하는 ray 를 식 (1)과 식 (2)와 같이 직선으로 모델링한다.

$$X(\alpha) = S_x + \alpha(E_x - S_x) \tag{1}$$

$$Y(\alpha) = S_y + \alpha(E_y - S_y) \tag{2}$$

여기서 α는 점 S에서 0의 값을, 점 E에서 1의 값을 가지고 α가 0과 1 사이에서 변경됨으로써 직선상의 임의의 점을 나타낼 수 있다. 영상과 교차하는 ray의 범위는 그림 1의 geometry에서 ray와 영상이 처음 만나는 $\alpha_{\min}$과 마지막으로 만나는 $\alpha_{\max}$를 이용하여 구한다. 이 값을 구하기 위해 ray가 $x=0$, $x=N_x$ 축과 만날 때의 α 값 $\alpha_x(0), \alpha_x(N_x)$을 식 (3)과 식 (4)와 같이 구해진다.

$$\alpha_x(0) = (-S_x)/(E_x - S_x) \tag{3}$$

$$\alpha_x(N_x) = (N_x - S_x)/(E_x - S_x) \tag{4}$$

동일한 방법으로 $\alpha_y(0), \alpha_y(N_y)$을 구하고 이를 이용하여 $\alpha_{\min}, \alpha_{\max}$을 식 (5)와 식 (6)으로 구한다[6].

$$\alpha_{\min} = \max[\alpha_x(0), \alpha_y(0)] \tag{5}$$

$$\alpha_{\max} = \min[\alpha_x(N_x), \alpha_y(N_y)] \tag{6}$$

이때 ray 선상에서 $\alpha_{\min}, \alpha_{\max}$일 때의 픽셀의 x축 index는 다음 식 (7)과 식 (8)로 구할 수 있다.

$$i_{\min} = N_x - [X_{plane}(N_x) - \alpha_{\min}(E_x - S_x) - S_x]/d_x \tag{7}$$

$$i_{\max} = 1 + [X_1 + \alpha_{\max}(E_x - S_x) - X_{plane}(1)]/d_x \tag{8}$$

여기서 $X_{plane}(N_x)$는 $x = N_x$인 plane을 의미하고 동일한 방법으로 y 축의 index인 $j_{\min}, j_{\max}$을 구한다. 결정된 픽셀들의 index 범위를 이용하면 모든 교점에서의 α 값들을 식 (9)와 식 (10)으로 정리할 수 있다.

$$\alpha_x = \{\alpha_x(i_{\min}), \alpha_x(i_{\min}+1), ..., \alpha_x(i_{\max})\} \tag{9}$$

$$\alpha_y = \{\alpha_y(j_{\min}), \alpha_y(j_{\min}+1), ..., \alpha_y(j_{\max})\} \tag{10}$$

영상과 ray가 교차하는 모든 픽셀의 교차점들의 α 값을 구하고 이 값들을 $\alpha_{\min}$부터 $\alpha_{\max}$ 까지 오름차순으로 정렬하고 나면 이웃하는 두 개의 α 값의 차이를 이용하여 교차하는 길이를 구할 수 있다.

교차하는 픽셀들의 index 를 구하기 위해 정렬한 α 를 이용하여 ray가 m 번째 픽셀을 지난다고 할 때, 픽셀의 index는 다음 식 (11)과 식 (12)로 구할 수 있다[6].

$$i(m) = 1 + [X_1 + ((\alpha(m) + \alpha(m-1))/2)(E_x - S_x) - X_{plane}(1)]/d_x \tag{11}$$

$$j(m) = 1 + [Y_1 + ((\alpha(m) + \alpha(m-1))/2)(E_y - S_y) - Y_{plane}(1)]/d_y \tag{12}$$

앞서 구한 ray와 교차하는 픽셀들의 교차길이를 식 (11)과 식 (12)로 구한 픽셀들의 index와 곱하여 누적하는 과정을 수행하면 Siddon 방법을 이용한 프로젝션이 완성된다. 하지만 이 두 가지 과정을 따로 수행하는 것보다 교차된 ray의 길이를 구하면서 그 픽셀의 index 를 바로 곱하여 누적하며 진행한다면 픽셀과 교차된 α 값들을 한번만 구해도 되기 때문에 더 빠른 프로젝션이 가

능하게 된다[8][9].

이에 기반한 Jacobs 기법은 길이를 구하면서 index 를 구해서 바로 곱하는 방법으로 Siddon 기법을 발전시켰다. Jacobs 방법은 α_x와 α_y의 비교 연산을 통해 index 를 업데이트 하며 ray와 픽셀의 교차길이를 구하고 동시에 픽셀 값을 곱하여 누적하는 과정을 통해 계산시간을 단축하는 것이다[8].

Zhao 기법은 ray와 픽셀의 교차된 길이를 구할 때 영상의 동일 행이나 열에서의 길이가 동일하다는 사실을 이용하여 연산 시간을 Jacobs 기법보다 단축시켰다. 그림 1에서 볼 수 있듯이 동일 행에서의 ray의 길이를 L_y 라 하고, 동일 열에서의 ray의 길이를 L_x 라 하면 L_x, L_y와 남은 길이 RL 과의 비교 연산을 통해 교차하는 길이와 픽셀의 index 를 구할 수 있다.

그림 1에서 ray가 픽셀을 통과한 길이의 합이 L_y가 될 때 j index가 바뀌게 된다. 따라서 L_y에서 ray와 픽셀의 교차한 길이를 빼고 남은 길이 RL과 L_x와 대소 비교를 하면 ray가 영상의 어느 픽셀과 교차하고 있는지를 알 수 있게 된다. 남은 길이 RL이 길이 L_x 보다 크다면 ray가 이웃한 픽셀과 교차하고 있는 것이고, L_x 보다 작다면 동일 행이 아닌 위의 행의 픽셀과 교차하고 있음을 알 수 있다. 이를 수식으로 설명하면 다음과 같다. 먼저 y 축 index가 $j_{\min}$인 픽셀 중에 ray와 처음 교차하는 픽셀의 x 축 index i는 식 (13)으로 결정된다.

$$i = \lfloor T_x / d_x \rfloor \tag{13}$$

처음 교차하는 픽셀과 ray의 교차 길이를 TS_1 이라 하면 남은 길이 RL은 식 (14)와 식 (15)로 구해진다.

$$l(i, j_{\min}) = TS_1 \tag{14}$$

$$RL = L_y - TS_1 \tag{15}$$

남은 길이 RL이 길이 L_x보다 크다면 ray와 교차된 픽셀의 길이는 L_x이고, 작다면 ray와 교차된 픽셀의 길이는 RL이 된다. 그리고 교차하는 픽셀의 index는 픽셀의 x 값만 바뀌고 y 값은 $j_{\min}$으로 동일한 값을 갖는다.

$$l(i+m, j_{\min}) = \begin{cases} L_x, & RL > L_x \\ RL, & RL \le L_x \end{cases} \tag{16}$$

길이가 RL이 되면 ray와 교차하는 다음 픽셀은 동일 열, 다른 행에 있는 픽셀이고 그때의 교차 길이는 식 (17)로 구할 수 있고, 행이 바뀌었으므로 픽셀의 index가 식 (18)과 같다.

$$TS_2 = L_x - RL \tag{17}$$

$$l(i, j_{\min}+1) = TS_2 \tag{18}$$

그 후의 과정은 픽셀의 j index가 $j_{\max}$가 될 때까지 동일한 방법으로 반복 진행한다.

위의 Zhao 방법을 다음 pseudo 코드로 나타낼 수 있다.

```
1:  procedure INTERSECTION PIXEL(l)
2:      Determine j_min , j_max   ▷ For the range
3:              j = j_min
4:              l(i,j) = TS       ▷ The intersection of the ray with first pixel
5:          RL = L_y − TS         ▷ The remaining uncalculated length
6:      While ( j ≤ j_max )
7:          if ( RL ≥ L_x )
8:                  i = i + 1     ▷ update i index
9:              l(i,j) = L_x ;    ▷ update l and index
10:                 RL = RL − L_x ▷ update first length of next pixel
11:         else ( RL < L_x )
12:                 i = i + 1     ▷ update i index
13:             l(i,j) = RL       ▷ update l and index
14:                 RL = L_x − RL ▷ update first length of next pixel
15:                 j = j + 1     ▷ update j index
16:         end if
17:     end while
18: end procedure
```

그림 2 Zhao 기법의 pseudo 코드
Fig. 2 Pseudo code of Zhao method

Zhao 기법은 남은 길이 RL과 일정한 값 L_x 나 L_y의 비교만으로 ray와 픽셀의 교차하는 길이와 index 를 구할 수 있기 때문에 Jacobs 방법보다 더 빠르게 프로젝션 영상 생성이 가능하다. 하지만 길이 RL의 값을 계속 구하면서 일정한 값과 비교 연산하는 과정은 영상과 source와 detector의 geometry을 안다면 줄일 수 있는 과정이다. 본 논문에서는 이러한 점에 착안하여 Zhao 기법의 계산 속도를 더욱 개선하는 방법을 제안한다.

2.2 제안하는 방법

제안하는 알고리즘은 Siddon 방법과 같이 우선 ray가 지나가는 픽셀의 범위를 결정한다. 그림 1을 예로 들면 ray가 영상과 교차하는 첫 번째 행의 index인 $j_{\min}$과 마지막으로 만나는 행의 index $j_{\max}$ 를 다음 식 (19)와 식 (20)으로 구할 수 있다[8].

$$j_{\min} = N_y - \lfloor (N_y d_y - (S_y + \alpha_{\min}(E_y - S_y))) / d_y \rfloor \tag{19}$$

$$j_{\max} = \lfloor (S_y + \alpha_{\max}(E_y - S_y)) / d_y \rfloor \tag{20}$$

여기서 $\lfloor \ \rfloor$는 floor 연산 기호이다.

이때 ray의 전체 길이인 L_{SE} , 이웃한 x 축 사이 길이 L_x와 이웃한 y 축 사이 길이 L_y는 식 (21)부터 식 (23)으로 구해진다[8].

$$L_{SE} = \sqrt{(E_x - S_x)^2 + (E_y - S_y)^2} \tag{21}$$

$$L_x = L_{SE} d_x / (E_x - S_x) \tag{22}$$

$$L_y = L_{SE} d_y \tag{23}$$

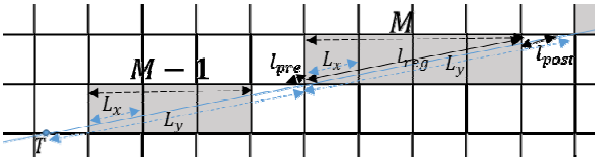

그림 3 규칙적 패턴
Fig. 3 Regular pattern

그림 1에서 볼 수 있듯이 픽셀의 index가 $j_{\min}$부터 $j_{\max}$까지일 때는 이웃한 축 사이에서 일정한 교차 길이가 반복되는 패턴을 가진다. Zhao 기법은 각 픽셀마다 길이 RL과 길이 L_x 와의 비교 연산을 통해 교차된 길이와 해당픽셀의 index를 결정한다. 그러나 프로젝션 geometry를 통해 길이 L_x을 가지는 픽셀의 개수를 미리 계산할 수 있으므로 이 사실을 활용하면 연산시간을 더욱 줄일 수 있다.

2.2.1 Regular pattern

그림 3은 그림 1에서 교차길이가 반복되는 부분을 확대한 그림이다. 영상의 동일 행에서 L_x 보다 작은 처음 길이를 l_{pre}, 픽셀과 ray의 교차하는 길이가 L_x인 픽셀들의 교차한 길이를 누적한 값을 l_{reg}, L_x보다 작은 l_{reg}의 다음 길이를 l_{post}라고 정의한다. 이 경우 식 (24)의 관계식이 성립한다.

$$L_y = l_{pre} + kL_x + l_{post} \tag{24}$$

이때 $M = \lfloor L_y / L_x \rfloor$라 정의하면 식 (25)의 조건이 성립한다.

$$0 \le l_{pre} + l_{post} < 2L_x \tag{25}$$

따라서 식 (24)의 k는 M 혹은 $M-1$의 값을 가질 수 있고 l_{reg}는 식 (26)과 같이 나타난다.

$$l_{reg} = \begin{cases} (M-1)L_x, & l_{pre} + l_{post} > L_x \\ ML_x, & l_{pre} + l_{post} \le L_x \end{cases} \tag{26}$$

따라서 $M-1$개의 픽셀에서는 ray와 교차된 길이가 L_x이기 때문에 교차된 길이를 따로 계산하지 않아도 된다. 그리고 M번째 픽셀이 길이 L_x인지 l_{post}인지를 판단하기 위해 $M-1$번째 까지 계산하고 ray의 남은 길이 R_L과 L_x와 비교 연산을 진행한다.

길이 R_L이 L_x보다 크거나 같다면 M 번째 픽셀은 교차한 길이가 L_x이고, 작다면 교차한 길이는 l_{post}이다.

$$l_M(i_M, j) = \begin{cases} L_x, & RL \ge L_x \\ l_{post}, & RL < L_x \end{cases} \tag{27}$$

만약 M 번째 픽셀의 교차한 길이가 L_x 였다면 $M+1$ 번째 픽셀은 식 (28)과 같다.

$$l_{M+1}(i_{M+1}, j) = l_{post} \tag{28}$$

길이 l_{reg}을 구한 후 길이 l_{pre}와 l_{reg}을 이용하여 l_{post} 값을 식 (29)와 같이 구할 수 있다.

$$l_{post} = L_y - l_{pre} - l_{reg} \tag{29}$$

그림 3에서 영상의 픽셀의 y 축 index가 $j_{\min}$인 픽셀 중에 ray와 처음 교차하는 픽셀의 ray 길이인 l_{pre}는 식 (30)과 같이 결정된다.

$$l_{pre}(i, j_{\min}) = \begin{cases} (d_x - T_x - I_x d_x)L_{SE}/(E_x - S_x), & T_x > I_x d_x \\ L_x, & T_x = I_x d_x \end{cases} \tag{30}$$

여기서 I_x는 $\lfloor T_x / d_x \rfloor$을 의미한다. 구간 M을 지난 후 다시 시작되는 l_{pre} 길이는 길이 L_x와 l_{post} 를 이용하여 식 (31)로 구할 수 있다. 길이 L_x는 동일 열에서의 교차하는 ray의 전체 길이로 식 (22)로 구하고 길이 l_{post}는 식 (29)로 구한다.

$$l_{pre}(i, j) = L_x - l_{post}(i, j) \tag{31}$$

이처럼 regular pattern에서의 ray와 픽셀의 교차하는 길이는 픽셀의 y 축 index가 $j_{\min}$ 부터 $j_{\max}$까지 변할 때 $l_{pre}, l_{reg}, l_{post}$ 길이를 반복적으로 구하면서 동시에 픽셀의 index 를 계산하면 regular pattern 안에서의 프로젝션이 완성된다.

2.2.2 Pre-pattern and post-pattern

그림 1에서 볼 수 있듯이 regular pattern의 전후에 일반적으로 pre-pattern과 post-pattern이 존재한다. 이 구간은 ray와 교차된 픽셀들의 길이의 총합이 L_x나 L_y보다 작아 regular pattern에서처럼 규칙성이 없는 구간이다.

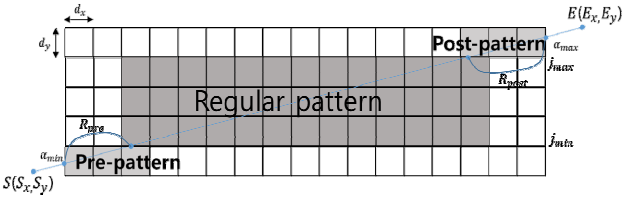

그림 4 패턴 geometry
Fig. 4 Pattern geometry

그림 4는 전체 영상을 세 가지의 패턴으로 나타낸 영상이다. 그림에서 pre-pattern은 ray와 교차하는 픽셀의 y 축 index가

$j_{\min}-1$을 가지는 픽셀들만 있는 구간이고, post-pattern은 ray와 교차하는 픽셀의 y 축 index가 $j_{\max}+1$을 가지는 픽셀들만 있는 구간이다. 앞서 설명한 바와 같이 regular pattern은 영상과 ray가 교차하는 픽셀의 길이가 L_x인 개수를 미리 알 수 있는 점을 이용하여 반복되는 방법으로 빠른 길이 계산이 가능한 구간이다.

먼저 pre-pattern을 지나는 ray와 교차하는 픽셀들의 ray 전체 길이를 R_{pre}라 하면, 이 길이는 다음 식 (32)와 같이 구해진다.

$$R_{pre} = L_{SE}\left(\frac{j_{\min}d_y - S_y}{E_y - S_y} - \alpha_{\min}\right) \tag{32}$$

여기서 길이 L_{SE}는 식 (21)에서 구한 값으로 source와 detector까지 ray의 전체 길이를 의미한다. 길이 R_{pre}와 길이 L_x의 대소 비교를 통해서 ray와 교차하는 픽셀의 개수를 알 수 있고 길이 R_{pre}가 길이 L_x 보다 작아질 때까지 반복하여 진행하면 된다.

유사한 방법으로 post-pattern을 지나는 ray와 교차하는 픽셀들의 ray 전체 길이 R_{post}는 식 (33)과 같다.

$$R_{post} = L_{SE}\left(\alpha_{\max} - \frac{(j_{\max}+1)d_y - S_y}{E_y - S_y}\right) \tag{33}$$

또한, 길이 R_{post}와 길이 L_x의 대소 비교를 통해서 ray와 교차하는 픽셀의 개수를 알 수 있고 길이 R_{post}가 길이 L_x 보다 작아질 때 까지 반복하여 진행하면 된다.

앞 절에서 설명한 regular pattern 부분에서의 프로젝션 연산과 더불어 이어 pre-pattern, post-pattern 부분의 프로젝션 연산을 수행하면 전체 프로젝션 연산이 완성된다. 그림 5는 제안하는 알고리즘을 정리한 pseudo 코드이다.

```
1:  procedure INTERSECTION PIXEL(l)
2:      Determine  j_min , j_max          ▷ For the range
3:      Determine M − 1                   ▷ For the pre-computation length
4:          j = j_min
5:      While  (j ≤ j_max)
6:          count = count + 1             ▷ Check for repeating
7:          Process  pre-length
8:          Process  Regular-pattern
9:              if  (count < M)
10:                 Process  regular-length
11:             else if (count = M)
12:                 Process  regular-length
13:                 Process  post-length
14:                 count reset
15:             else
16:                 Process post-length
17:                 count reset
18:             end if
19:     end while
20: end procedure
```

그림 5 제안한 기법의 pseudo 코드

Fig. 5 Pseudo code of the proposed method

3. 실험 결과

본 논문에서 제안하는 알고리즘의 성능을 기존 방법과 비교 평가하기 위하여 실험을 수행하였다. 실험에 사용된 컴퓨터 사양은 프로세서 2.7 GHz Intel Core i5, 메모리 8 GB 1600 MHz DDR3, Max OS X(EI Capitan 버전 10.11.6) 이고, MATLAB R2016a에서 수행되었다.

그림 6은 MATLAB에서 제공하는 Shepp-Logan phantom 256×256 영상과 제안하는 방법으로 이 phantom 영상의 sinogram을 구한 그림이다.

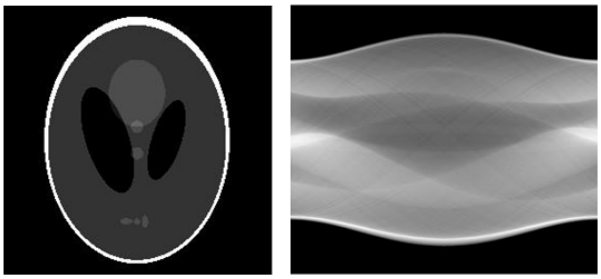

그림 6 Shepp-Logan 팬텀과 제안한 방법을 사용한 프로젝션의 결과 sinogram

Fig. 6 Shepp_Logan phantom and its sinogram using the proposed method

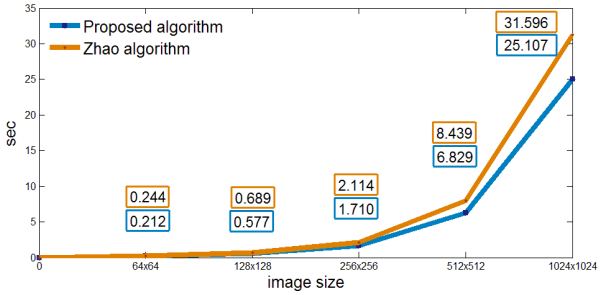

그림 7 영상 크기에 따라 제안한 기법과 Zhao 기법을 이용한 프로젝션 시간 비교 (Shepp-Logan 팬텀)

Fig. 7 Computation time of the proposed method and Zhao method (Shepp-Logan phantom)

그림 7은 제안한 방법과 Zhao 방법을 phantom 영상에 적용한 프로젝션 계산 시간 비교 그래프이다. 영상의 크기는 64×64 부터 1024×1024로 5개의 다른 크기를 가진 영상에 적용하였다. 영상의 크기가 제일 작은 64×64에서 는 제안하는 방법이 Zhao 방법에 비해 약 13% 정도 시간이 단축되었고, 영상의 크기가 커질수록 계산시간의 단축정도가 커졌다. 영상이 1024×1024 일 때는 약 20% 정도 시간이 단축됨을 확인할 수 있다. 따라서 영상의 크기가 클수록 제안하는 방법이 Zhao 방법보다 프로젝션 연

산에 더 효율적인 방법이다.

그림 8은 듀크 대학에서 제공하는 XCAT 팬텀(512×512)과 제안하는 방법을 이 팬텀에 적용하여 얻은 sinogram이다[10].

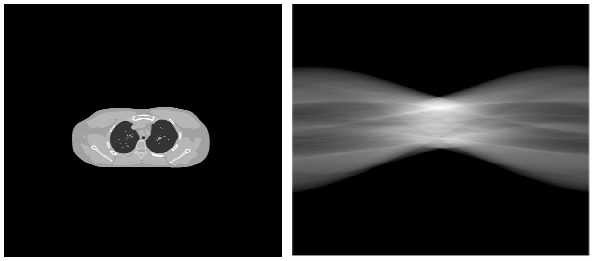

그림 8 XCAT 팬텀 영상과 프로젝션 결과인 sinogram
Fig. 8 XCAT phantom and its sinogram using the proposed method

앞선 실험과 다른 환경에서 제안하는 방법과 기존 방법의 차이를 확인하기 위해 그림 8의 XCAT 팬텀을 사용하여 다른 조건에서 프로젝션 영상을 생성하는 실험을 진행하였다. 그림 7에서 진행한 실험보다 검출기의 해상도를 2배 더 높이고 각도의 분해능 역시 2배 더 높여서 더 많은 연산이 필요한 경우에 대해서 실험을 수행하였다. 연산량이 많아질수록 제안하는 방법이 기존 방법보다 효과적인 것을 확인할 수 있다. 표 1에 이 결과를 영상의 크기에 따라 정리하였다.

표 1 영상 크기에 따라 제안한 기법과 Zhao 기법을 이용한 프로젝션 시간 비교(XCAT 팬텀)
Table 1 Computation time of the proposed method and Zhao method (XCAT phantom)

XCAT 팬텀	Image 256x256	Image 512x512	Image 1024x1024
Proposed method	6.95 sec	23.82 sec	91.48 sec
Zhao method	8.32 sec	30.48 sec	116.04 sec

4. 결 론

본 논문은 ray-tracing 방법을 이용한 프로젝션을 빠른 시간 내에 수행하기 위한 알고리즘을 제안하였다. 제안하는 알고리즘은, 프로젝션 geometry 로부터 영상을 통과하는 ray의 교차 길이가 같은 픽셀의 개수를 미리 알 수 있고, 이를 이용하면 교차하는 길이 계산의 생략으로 계산 속도를 빠르게 할 수 있다는 사실에 기반 한다. 시뮬레이션에서 제안하는 방법은 현재 알려진 가장 빠른 방법인 Zhao 방법보다 최소 10% 이상 많게는 20% 정도의 속도 향상을 보였다.

감사의 글

이 논문은 2014년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임. (NRF-2014R1A2A1A11052512).

References

[1] Hara, A. K., Paden, R. G., Silva, A. C., Kujak, J. L., Lawder, H. J., and Pavlicek, W, "Iterative reconstruction technique for reducing body radiation dose at CT: feasibility study." American Journal of Roentgenology 193.3 (2009): 764-771.

[2] T. Peters, "Algorithms for fast back- and re-projection in computed tomography," IEEE Trans. Nucl. Sci. 28 3641-7, 1981.

[3] G. Zeng, and G. Gullberg, "A ray-driven backprojector for backprojection filtering and filtered backprojection algorithms," IEEE Nuclear Science Symp. Medical Imaging Conf. (San Francisco) pp 1199-201, 1993

[4] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," Phys. Med. Biol., vol. 49, no. 11, pp. 2463-2475, 2004

[5] Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for X-ray CT using separable footprints," IEEE Trans. Med. Imag.,vol. 29, no. 11, pp. 1839-1850, Nov. 2010.

[6] V.-G. Nguyen, J. Jeong, and S.-J. Lee, "GPU-accelerated iterative 3D CT reconstruction using exact ray-tracing method for both projection and backprojection," in Proc. IEEE NSS-MIC, 2013, pp. 1-4.

[7] R. L. Siddon, "Fast calculation of the exact radiological path for a threedimensional CT array," Med. Phys., vol. 12, no. 2, pp. 252-255, 1985.

[8] F. Jacobs, E. Sundermann, B. De Sutter, M. Christiaens, and I. Lemahieu, "A fast algorithm to calculate the exact radiological path through a pixel or voxel space", Journal of computing and information technology , Vol. 6, No. 1, pp. 89-94, March 1998.

[9] H. Zhao and A. J. Reader, "Fast ray-tracing technique to calculate line integral paths in voxel arrays," in Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf., 2003, pp. 2808-12.

[10] Segars, W. P., & Tsui, B. M. (2009). MCAT to XCAT: The evolution of 4-D computerized phantoms for imaging research. Proceedings of the IEEE, 97(12), 1954-1968.

저 자 소 개

이 현 정 (Lee Hyun Jeong)

2014년 이화여자대학교 전자공학과 졸업.
2015년 9월~현재 동 대학원 석사과정 재학중.
Tel : +82-2-3277-4236
E-mail : hjlee918@ewhain.net

김 정 태 (Kim Jeong Tae)

1989년 서울대학교 제어계측공학과 졸업.
1991년 동 대학원 석사과정 졸업. 1991년~1998년 삼성전자 디지털미디어연구소 책임연구원. 2004년 미시간대학교 전기공학과 졸업(Ph.D), 2004년-현재 이화여자대학교 전자공학과 교수.
Tel : +82-2-3277-4084
Fax : +82-2-3277-3494
E-mail : jtkim@ewha.ac.kr