A Fast Algorithm to Calculate the Exact Radiological Path through a Pixel or Voxel Space

Filip Jacobs, Erik Sundermann, Bjorn De Sutter, Mark Christiaens and Ignace Lemahieu

Sint–Pietersnieuwstraat 41, Gent, Belgium

Calculating the exact radiological path through a pixel or voxel space is a frequently encountered problem in medical image reconstruction from projections and greatly influences the reconstruction time. Currently, one of the fastest algorithms designed for this purpose was published in 1985 by Robert L. Siddon [1]. In this paper, we propose an improved version of Siddon's algorithm, resulting in a considerable speedup.

Keywords: reconstruction, projection, radiological path

Introduction

In image reconstruction from projections we calculate an image from a given set of measurements which are related to line-integrals over the image [3, 4]. Examples of such images can be found in PET, CT and MRI studies where the images represent the distribution of an administered radio-active tracer, the distribution of the linear attenuation coefficients of tissue and the distribution of protons, respectively, in cross-sections of a patient.

In order to simplify the notations, we will restrict ourselves to a 2D-description of the algorithms. The theory can easily be extended to rays lying in a 3D space. The results presented in the last section of this paper, however, are based upon both a 2D- and a 3D-implementation of the algorithms.

Prior to the reconstruction, the image is discretized into pixels and the line-integrals into weighted sums. The weighting factor for the value $\rho(i,j)$ of pixel (i,j) is denoted by $l(i,j)$ and equals the intersection length of the ray with pixel (i,j). Hence, the line-integral from point $p(p_{1x},p_{1y})$ to $p(p_{2x},p_{2y})$, over the discretized image, can be approximated by the following weighted sum:

$$d_{12} = \sum_{(i,j)} l(i,j)\rho(i,j). \tag{1}$$

Because of the huge amount of measurements given by a medical scanner and the large number of pixels, it is impossible to store all weighting factors $l(i,j)$ in a file prior to the reconstruction. Therefore, they have to be calculated on the fly, which greatly limits the reconstruction time. A fast algorithm to evaluate equation (1) is a necessity to obtain acceptable reconstruction times.

Currently, one of the fastest algorithms designed for this purpose was published in 1985 by Robert L. Siddon [1]. We have improved his algorithm in a way that the time spent in the inner loop is reduced considerably, and the reconstruction time accordingly (also see [2]). In the first section we introduce the used notations. In the following section, we review Siddon's algorithm for rays lying in a 2D plane and give the basic formulas needed to explain the improved algorithm in the subsequent section. In the final section, we compare the two algorithms for rays lying in a 2D plane as well as for rays lying in a 3D space by comparing the obtained reconstruction times for 3D (i.e. a stack of several 2D planes) and fully 3D (i.e. oblique raysums are also available) PET images.

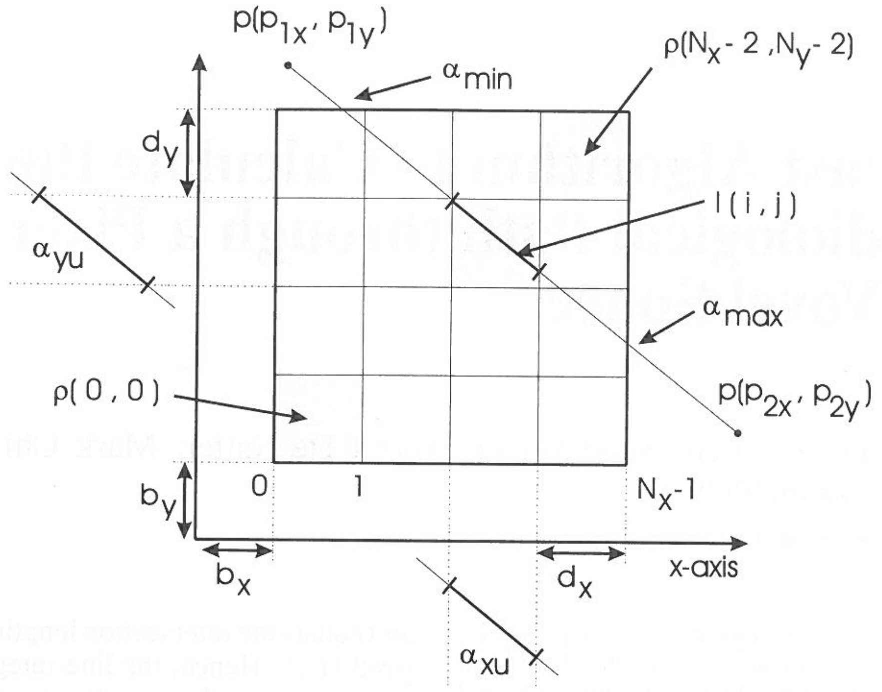

Fig. 1. A schematic overview of the used notations. The pixel values are denoted by $\rho(i,j)$ and a point in the 2D plane is referred to as $p(x,y)$. The α-variable holds the relative distance of a point on the line through $p(p_{1x}, p_{1y})$ and $p(p_{2x}, p_{2y})$ and the point $p(p_{1x}, p_{1y})$. The α-variable equals 1 for the point $p(p_{2x}, p_{2y})$ and any value between 0 and 1 for points between $p(p_{1x}, p_{1y})$ and $p(p_{2x}, p_{2y})$. The other variables hold real distances.

Notations

The pixel space is determined by the intersection of two sets of equally spaced parallel planes, perpendicular to an x- and an y-axis (see Figure 1). The x- and y-axis are perpendicular to each other. The two sets will be referred to as the x- and y-planes, respectively. The number of x-planes equals N_x and the distance between them is denoted by d_x. The x-planes are numbered from 0 to $N_x - 1$. Similar notations hold for the y-planes. Hence, the pixel values $\rho(i,j)$ have indices running from $(0,0)$ to $(N_x - 2, N_y - 2)$. The lowest left corner of the pixel space, i.e. the intersection point of x-plane 0 and y-plane 0, has co-ordinates (b_x, b_y). Each ray goes from a point denoted by $\mathbf{p}_1 = p(p_{1x}, p_{1y})$ to another point denoted by $\mathbf{p}_2 = p(p_{2x}, p_{2y})$.

Siddon's Algorithm

We could evaluate equation (1) by summing over all (i,j). This would be very inefficient, as pointed out by Siddon, because most $l(i,j)$ are zero. It is more efficient to follow the ray through the pixel space. Therefore, we use a parametrical representation of the ray,

$$\mathbf{p}_{12} = \left\{ \begin{array}{lcl} \mathbf{p}_x(\alpha) & = & p_{1x} + \alpha(p_{2x} - p_{1x}) \\ \mathbf{p}_y(\alpha) & = & p_{1y} + \alpha(p_{2y} - p_{1y}) \end{array} \right. \quad (2)$$

with $\alpha \in [0,1]$ for points between $\mathbf{p}_1$ and $\mathbf{p}_2$ and $\alpha \notin [0,1]$ for all other points. In what follows, we will assume that the ray is generic, i.e. that $p_{1x} \neq p_{2x}$ and $p_{1y} \neq p_{2y}$. Non-generic rays are trivial to handle and will not be discussed further. Following Siddon's algorithm, we first determine the entry point ($\alpha = \alpha_{min}$) and exit point ($\alpha = \alpha_{max}$) of the ray (see Figure 1). Equation (9) calculates the α parameter corresponding to the intersection of the i-th x-plane and the line going through $p(p_{1x}, p_{1y})$ and $p(p_{2x}, p_{2y})$, hence, these values are not restricted to the interval $[0,1]$.

$$\alpha_{min} = \max(\alpha_{xmin}, \alpha_{ymin}) \quad (3)$$
$$\alpha_{max} = \min(\alpha_{xmax}, \alpha_{ymax}) \quad (4)$$

with

$$\alpha_{xmin} = \min(\alpha_x(0), \alpha_x(N_x - 1)) \quad (5)$$
$$\alpha_{xmax} = \max(\alpha_x(0), \alpha_x(N_x - 1)) \quad (6)$$
$$\alpha_{ymin} = \min(\alpha_y(0), \alpha_y(N_y - 1)) \quad (7)$$
$$\alpha_{ymax} = \max(\alpha_y(0), \alpha_y(N_y - 1)) \quad (8)$$

and

$$\alpha_x(i) = \frac{(b_x + id_x) - p_{1x}}{p_{2x} - p_{1x}} \quad (9)$$

$$\alpha_y(j) = \frac{(b_y + jd_y) - p_{1y}}{p_{2y} - p_{1y}} \quad (10)$$

Given that the ray does intersect the pixel space, i.e. $\alpha_{min} < \alpha_{max}$, we calculate the number of the first intersected x-plane i_f after the ray entered the pixel space and the number of the last intersected x-plane i_l including the outer plane. We will use the variables $i_{min} = \min(i_f, i_l)$ and $i_{max} = \max(i_f, i_l)$ to simplify the following formulas. Similar definitions hold for j_{min} and j_{max} concerning the y-planes. Whenever $p_{1x} < p_{2x}$ we calculate i_{min} and i_{max} with equations (11)-(14) and otherwise with equations (15)-(18). The definition of $\varphi_x(\alpha)$ is given by equation (19). Similar formulas hold for j_{min} and j_{max}.

$$\alpha_{min}=\alpha_{xmin} \rightarrow i_{min}=1 \quad (11)$$
$$\alpha_{min}\neq\alpha_{xmin} \rightarrow i_{min}=\lceil\varphi_x(\alpha_{min})\rceil \quad (12)$$
$$\alpha_{max}=\alpha_{xmax} \rightarrow i_{max}=N_x-1 \quad (13)$$
$$\alpha_{max}\neq\alpha_{xmax} \rightarrow i_{max}=\lfloor\varphi_x(\alpha_{max})\rfloor \quad (14)$$

$$\alpha_{min}=\alpha_{xmin} \rightarrow i_{max}=N_x-2 \quad (15)$$
$$\alpha_{min}\neq\alpha_{xmin} \rightarrow i_{max}=\lfloor\varphi_x(\alpha_{min})\rfloor \quad (16)$$
$$\alpha_{max}=\alpha_{xmax} \rightarrow i_{min}=0 \quad (17)$$
$$\alpha_{max}\neq\alpha_{xmax} \rightarrow i_{min}=\lceil\varphi_x(\alpha_{max})\rceil \quad (18)$$

$$\varphi_x(\alpha) = \frac{p_x(\alpha) - b_x}{d_x} \quad (19)$$

Further, we calculate two arrays $\alpha_x[.]$ and $\alpha_y[.]$ holding the parametric values of the intersection points of the ray with the x- resp. y-planes, after the ray entered the pixel space. If $p_{1x} < p_{2x}$ the first array is given by equation (20) and otherwise by equation (21). Similar formulas are used to calculate the second array.

$$\alpha_x[i_{min}\cdots i_{max}] = (\alpha_x(i_{min}), \alpha_x(i_{min}+1), \cdots, \alpha_x(i_{max})) \quad (20)$$

$$\alpha_x[i_{max}\cdots i_{min}] = (\alpha_x(i_{max}), \alpha_x(i_{max}-1), \cdots, \alpha_x(i_{min})) \quad (21)$$

Subsequently, we sort the elements of (α_{min}, $\alpha_x[.]$, $\alpha_y[.]$) in an ascending order and replace all the values that occur twice by one copy of the value, resulting in the array $\alpha_{xy}[0\cdots N_v]$ holding the parametric values of all intersected points. The occurence of dual α-values is due to the simultaneous intersection of an x-plane, a y-plane and the ray. Given the $\alpha_{xy}[.]$ array, we calculate the co-ordinates (i_m, j_m) of the intersected pixels with equations (22)-(23) and their intersection lengths $l(i_m, j_m)$ with equation (24) for all $m \in [1\cdots N_v]$. The variable d_{conv} equals the Euclidean distance between the points $\mathbf{p}_1$ and $\mathbf{p}_2$. The pixels (i,j) which do not correspond to a certain (i_m, j_m) are not intersected.

$$i_m = \left\lfloor \varphi_x\left(\frac{\alpha_{xy}[m]+\alpha_{xy}[m-1]}{2}\right)\right\rfloor \quad (22)$$

$$j_m = \left\lfloor \varphi_y\left(\frac{\alpha_{xy}[m]+\alpha_{xy}[m-1]}{2}\right)\right\rfloor \quad (23)$$

$$l(i_m, j_m) = (\alpha_{xy}[m] - \alpha_{xy}[m-1])d_{conv} \quad (24)$$

After implementing and profiling Siddon's algorithm, we found that its speed is greatly limited by the frequent use of equations (22) and (23) where floating point values are converted into integer values. In the following section we present an altered algorithm, based on Siddon's algorithm, which restricts the use of these equations to once for each ray.

Improved algorithm

As pointed out in the above section, frequent use of equations (22) and (23) limits the speed of Siddon's algorithm. In this section we propose an improved algorithm which restricts the use of these equations to once for each ray. It also obviates the need to allocate memory for the different α-arrays.

We follow Siddon's approach until the values of α_{min} and α_{max} are calculated. Starting from here, our approach differs from the one used by Siddon. Instead of calculating the arrays $\alpha_x[.]$ and $\alpha_y[.]$ we only calculate the values $\alpha_x = \alpha_x[0]$ and $\alpha_y = \alpha_y[0]$, i.e. the parametric value of the first intersection point of the ray with the x- resp. y-planes, after the ray entered the pixel space.

We also calculate the values i_{min} and i_{max} given by equations (11)-(18) and j_{min} and j_{max} given by similar equations. These values are used to calculate the number of planes N_p crossed by the ray when it runs through the pixel space, after it has entered the pixel space, i.e.

$$N_p = (i_{max} - i_{min} + 1) + (j_{max} - j_{min} + 1) \quad (25)$$

Note that $N_p \geq N_v$ because the simultaneous crossing of an x-plane, a y-plane and the ray is subdivided into two separate events, i.e. the crossing of an x-plane and then the crossing of a y-plane, or vice versa. We only use equations (22) and (23) to calculate the indices (i,j) of the first intersected pixel, i.e.

$$i = \left\lfloor \varphi_x \left(\frac{\min(\alpha_x, \alpha_y) + \alpha_{min}}{2} \right) \right\rfloor \quad (26)$$

$$j = \left\lfloor \varphi_y \left(\frac{\min(\alpha_x, \alpha_y) + \alpha_{min}}{2} \right) \right\rfloor \quad (27)$$

Following the ray through the pixel space, we have to update the values of α_x, α_y, i and j according to whether we cross an x- or y-plane. Whenever $\alpha_x < \alpha_y$ the next intersected plane is an x-plane and we increase α_x and i with α_{xu} resp. i_u, given by equations (28) and (29). Similar equations hold for updating α_y and j when the ray crosses a y-plane, i.e. $\alpha_y < \alpha_x$. If $\alpha_x \equiv \alpha_y$, then we can use either case to update the variables.

$$\alpha_{xu} = \frac{d_x}{|p_{2x} - p_{1x}|} \quad (28)$$

$$i_u = \begin{cases} 1 & \text{if } p_{1x} < p_{2x} \\ \text{-1} & \text{else} \end{cases} \quad (29)$$

Finally, after initialising d_{12} to 0 and α_c to α_{min}, we are able to calculate the radiological path by running N_v times through the following algorithm: if $\alpha_x < \alpha_y$ then calculate $l(i,j)$ with (30) and update d_{12}, i, α_c and α_x with equations (31)-(34), else calculate $l(i,j)$ with (35) and update d_{12}, j, α_c and α_y with equations (36)-(39).

$$\begin{aligned} l(i,j) &= (\alpha_x - \alpha_c)d_{conv} && (30)\\ d_{12} &= d_{12} + l(i,j)\rho(i,j) && (31)\\ i &= i + i_u && (32)\\ \alpha_c &= \alpha_x && (33)\\ \alpha_x &= \alpha_x + \alpha_{xu} && (34) \end{aligned}$$

$$\begin{aligned} l(i,j) &= (\alpha_y - \alpha_c)d_{conv} && (35)\\ d_{12} &= d_{12} + l(i,j)\rho(i,j) && (36)\\ j &= j + j_u && (37)\\ \alpha_c &= \alpha_y && (38)\\ \alpha_y &= \alpha_y + \alpha_{yu} && (39) \end{aligned}$$

Besides the calculation of raysums, some algorithms also need exact intersection lengths $l(i,j)$ to calculate something else, e.g. the backprojection of an image. It is for these algorithms that we formulated (30–39). Algorithms which do not need explicit calculation of the intersection lengths should incorporate (30) and (35) into (31) resp. (36) without the multiplication with d_{conv}. The multiplication of d_{12} with d_{conv} can be done afterwards.

Evaluation and Results

In order to compare the two algorithms in realistic situations, we chose the reconstruction of 3D and fully 3D Positron Emission Tomography (PET) images with the Maximum Likelihood Expectation Maximization (MLEM) algorithm [5].

3D PET data consists of a set of sinograms. Each sinogram corresponds to a spatial plane through the patient. Each element of a sinogram corresponds to a raysum of a ray through the spatial plane and is determined by its angle with respect to the x-axis of a 2D Cartesian xy-co-ordinate system and its distance to the origin. Because each sinogram corresponds to a 2D image, 3D PET is actually a 2D problem. For the evaluation, we used a data set obtained with an ECAT 951 PET-scanner, consisting of 31 sinograms of 256 angles and 192 distances each. The 31 reconstructed images have dimensions 192 by 192.

Fully 3D PET data consists of a set of data planes. Each data plane corresponds to a spatial plane through the origin of a 3D Cartesian xyz-co-ordinate system and is determined by its tilt with respect to the xy-plane and its angle with respect to the xz-plane. Each element of a data plane corresponds to a raysum of a ray perpendicular to the spatial plane and is determined by two Cartesian co-ordinates. Because raysums are available for rays with different tilts, fully 3D PET is a real 3D problem. For the evaluation we used a data set calculated by the software package `eval3dpet` [6, 7]. The data planes in the data set correspond to 15 tilts and

	Siddon	Improved	Speedup	
3D (ray calculation)	75	10	7.5	5 iterations
3D (reconstruction)	84	17	5.0	5 iterations
fully 3D (reconstruction)	52	15	3.5	1 iteration

Table 1. A comparison of the algorithm of Siddon and the improved algorithm by comparing the reconstruction times (in min.) of a 3D PET image after 5 iterations and a fully 3D PET image after 1 iteration.

96 angles and have dimensions of 90 by 128. The reconstructed image has dimensions 64 by 128 by 128.

The MLEM algorithms have been implemented in C on a Sun Ultra 2 Creator with 2 Ultra-SPARC processors. In Table 1 we compare the reconstruction times for 5 iterations for the 3D PET case and 1 iteration for the fully 3D PET case. We observe that speedups between 3 and 5 are obtained. The speedup for fully 3D PET is smaller than the one for 3D PET because only a smaller fraction of the total reconstruction time is used to calculate the radiological paths, i.e. 37% instead of 57%.

We found that the improvement of Siddon's algorithm resulted in a speedup of 7.5 for the calculation of radiological paths and in a speedup of 5.0 for the total reconstruction time in the case of 3D. The time used to calculate the raysums was reduced from 89% to 57% which emphasises the importance of reducing the time spent on calculating the raysums and/or the intersection lengths.

Acknowledgment The authors would like to thank "*Het Vlaams Instituut voor de Bevordering van het Wetenschappelijk-Technologisch onderzoek in de industrie*" (IWT, Brussels, Belgium) and the "*Fund for Scientific Research - Flanders*" for financially supporting their work (NFWO, Brussels, Belgium).

References

[1] R. L. SIDDON "Fast calculation of the exact radiological path for a three-dimensional CT array", *Medical Physics*, vol. 12, pp. 252–255, March 1985.

[2] M. CHRISTIAENS, B. DE SUTTER, K. DE BOSSCHERE, J. VAN CAMPENHOUT, I. LEMAHIEU, "A Fast, Cache-Aware Algorithm for the Calculation of Radiological Paths Exploiting Subword Parallelism", *Journal of Systems Architecture, Special Issue on Parallel Image Processing*, 1998, (to appear).

[3] Y. CENSOR, "Finite Series-Expansion Reconstruction Methods", *Proceedings of the IEEE*, vol. 71, pp. 409–419, March 1983.

[4] R. M. LEWITT, "Reconstruction Algorithms: Transform Methods", *Proceedings of the IEEE*, vol. 71, pp. 390–408, March 1983.

[5] L. A. SHEPP, AND Y. VARDI, "Maximum likelihood reconstruction for emission tomography", *Trasanctions on Medical Imaging*, vol. 1, pp. 113–122, October 1982.

[6] S. S. FURUIE, G. T. HERMAN, T. K. NARAYAN, P. E. KINAHAN, J. S. KARP, R. M. LEWITT AND S. MATEJ, "A methodology for testing for statistically significant differences between fully 3D PET reconstruction algorithms", *Phys. Med. Biol.*, vol. 39, pp. 341–354, 1994.

[7] S. MATEJ, G. T. HERMAN, T. K. NARAYAN, S. S. FURUIE, R. M. LEWITT AND P. E. KINAHAN, "Evaluation of task-oriented performance of several fully 3D PET reconstruction algorithms", *Phys. Med. Biol.*, vol. 39, pp. 355–367, 1994.

Received: June, 1997
Accepted: March 1998

Contact address:
Filip Jacobs (correspondent)
Sint–Pietersnieuwstraat 41
9000 Gent
Belgium
tel: 32-9-264.66.18
fax: 32-9-264.35.94
E-mail: jacobs@petultra.rug.ac.be

Erik Sundermann, Mark Christiaens,
Bjorn De Sutter and Ignace Lemahieu
Sint–Pietersnieuwstraat 41
9000 Gent
Belgium

FILIP JACOBS was born in Wilrijk (Belgium) in 1970. He graduated in electrical engineering from the University of Ghent in 1994, and obtained a degree in biomedical and clinical engineering techniques in 1996 from the same university. Currently, he works as a Ph.D. student at the ELIS Department of the University of Ghent.

ERIK SUNDERMANN was born in Antwerp (Belgium) in 1968. He obtained an engineering degree in physics from the University of Ghent in 1992. In 1993, he was a trainee with the Siemens Company in Munich, Germany. Currently, he works as a Ph.D. student at the ELIS department of the University of Ghent.

BJORN DE SUTTER was born in Gent, Belgium, in 1974. He received his degree in computing science from the University of Gent, Belgium in 1997. He is currently researching whole-program optimization at link-time at the Electronics and Information Systems Department of the Faculty of Applied Sciences at the University of Gent.

MARK CHRISTIAENS was born in Tielt, Belgium, in 1974. He received his degree in computing science from the University of Gent, Belgium in 1997. He is currently researching the debugging of distributed programs at the Electronics and Information Systems Department of the Faculty of Applied Sciences at the University of Gent.

IGNACE LEMAHIEU was born in Varsenare (Belgium) in 1961. He graduated in physics at the University of Ghent in 1983, and obtained his doctoral degree in physics in 1988 from the same university. He is now a professor of Medical Image Processing and head of the MEDISIP research group at the ELIS department of the University of Ghent.