Radon Transform in Positron Emission Tomography

Jarkko Johansson

12th May 2004

Turku PET Centre Image Processing Report Series Report
TPCIMG0002

Contents

1 Preface **3**

2 Data Acquisition Using a PET Scanner **3**

3 Computational Data Acquisition **6**

3.1 Definition of a Digital Image 6

3.2 Approximating the Radon transform 9

3.3 Properties of The Family of Coincidence Lines 14

3.3.1 Definition of The Family of Coincidence Lines 15

3.3.2 Symmetries of the Family of Coincidence Lines 16

3.4 Definition of a Virtual Scanner 21

4 Algorithms **22**

4.1 Determining The Set Of Intersected Pixels and The Weighting Coefficients 22

4.2 Applying The Symmetries Of A Square 27

A Source Codes **29**

1 Preface

This work has been done in Turku PET Centre...

Who should read this document? This document is intented to everyone who is interested in modelling the two-dimensional [1] *data acquisition* in *positron emission tomography* (PET).

Document is divided up into relatively independent sections that can be read separately. Although this document is most enjoyable as a complex, for many readers the most interesting sections are in the beginning. We start by briefly introducing the data acquisition techniques with a PET scanner, after that we can start to build up an algorithm which gives us a model of the real data acquisition. The main focus of this work is to define and implement an efficient algorithm for computational data acquisition.

Computational data acquisition can be utilized for data simulations and in iterative image reconstruction. Examples of these both are presented in this work.

2 Data Acquisition Using a PET Scanner

Emission tomography exploits the fact that certain chemical compounds containing radioactive nuclei have a tendency to affix themselves to specific areas of the body, such as bone, blood, tumors and the like. In PET, the radioactive nuclei used are such that positrons are emitted during decay. When a positron comes in contact with an electron, the two particles annihilate turning the mass of the two particles into two 511KeV γ-rays that are emitted in nearly opposite directions. These two rays easily escape from the human body and can be recorded by external detectors. Event of annihilation is detected if detectors **A** and **B** (see Fig.1) receive γ-rays simultaneously. A line connecting the centers of the detectors **A** and **B** is called a coincidence line or a line of response.

A PET scanner has a limited number of detectors [2] and therefore only a limited number of detector pairs can be formed. In Figure 2 some pairs of detectors are connected with coincidence lines.

[1] Although data acquisition is often done in three-dimensions we will only consider the two-dimensional case in this work.

[2] which are not necessarily arranged on a ring

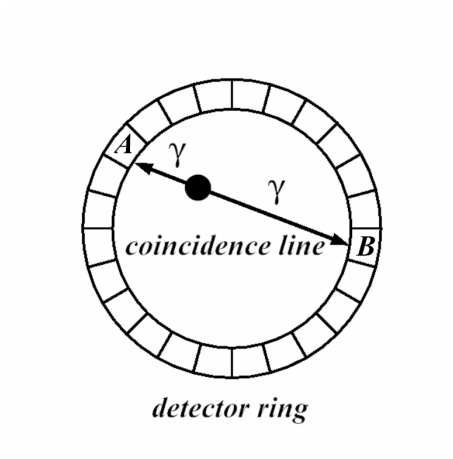

Figure 1: Detection of two γ-rays with a detector ring. Detectors work in pairs registering coincident γ-rays.

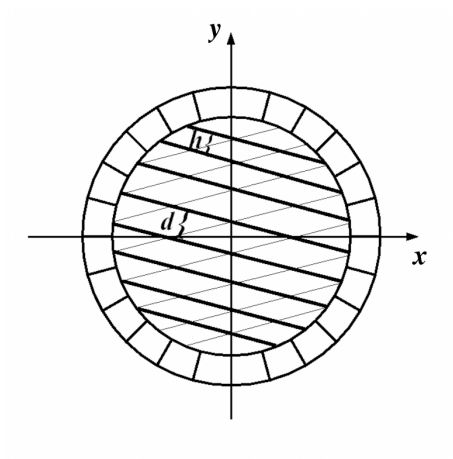

Figure 2: Some pairs of detectors connected with coincidence lines.

The set of connected detectors, that is the set of coincidence lines, is defined by the *scanner geometrics*. Scanner geometrics is a triple (S, Θ, d), where Θ is the number of angles in which the coincidence lines can be drawn, S is the number of coincidence lines in every angle and d is the distance between two neighbouring lines.

If detectors are arranged in a ring as shown in Figure 2 distances d and h between coincidence lines are not equal. For this reason a geometrical correction must be done before we can illustrate the arrangement of the coincidence lines as shown

in Figure 3.

After geometrical correction distance between any two neighbouring coincidence lines is constant defined by the width of a detector in a PET scanner. Distance d between two neighbouring coincident lines is called the *sample distance*.

Figure 3 also introduces the recording of the registered events originating from the radioactive point source P. The recording obtained by registering the events during time period Δt is called a *sinogram*. Coincidence lines are uniquely defined by angle θ from the y-axis and distance s from the origin. Events registered in coincidence lines are stored in the sinogram according to coordinates (s, θ), as shown in Figure 3 for one angle θ.

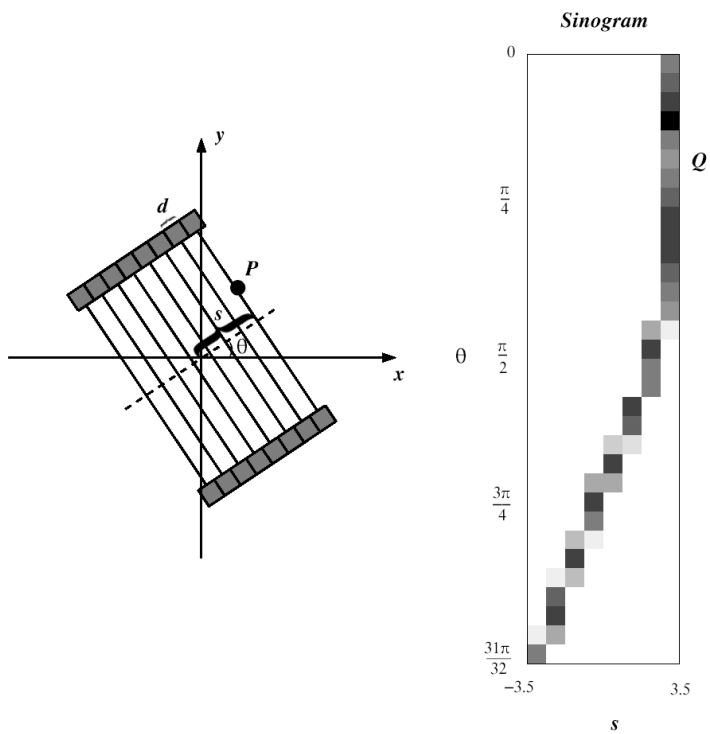

Figure 3: Example of data acquisition. Coincidence lines are drawn for one angle θ ($\frac{3\pi}{32}$) as they are arranged after geometrical correction. Point Q in the sinogram corresponds to coincidence line going through point P in angle θ. Other coincidence lines in angle θ do not register any events of annihilation and therefore only point Q is non zero in the sinogram.

Figure 4 shows an example of a sinogram acquired using ECAT931 (CTI/Siemens)

PET scanner. Sinogram is shown in inverse colormap and colorscaling is shown in the colorbar above the sinogram. ECAT931 PET scanner geometrics is defined by triple $(192, 256, 3.1mm)$, thus events are registered in 49152 coincidence lines.

Figure 4: Example of a sinogram acquired using PET scanner ECAT931.

3 Computational Data Acquisition

Computational data acquisition in two-dimensional PET imaging is utilized to simulate the acquisition made by a PET scanner computationally. We will define our algorithms for computational data acquisition using simple geometrical rules originating from the arrangement of coincidence lines descriped in section 2.

3.1 Definition of a Digital Image

We will start by giving a definition for a digital image to be utilized throughout this work. Digital images are utilized in the place of a patient when performing the computational data acquisition. That is to say, a digital image is utilized as a map of the spatial distribution of radioactivity concentrations, a digital image

in this context is called an *input* image. Our goal is to generate a sinogram from this map, which is as similar as possible with the one acquired with a PET scanner.

Definition 3.1(*Digital image*)**.** A digital image is assumed to have two components:
(i) the *image region* which is a square whose center is at the origin of the coordinate system, and which is subdivided into n^2 equal squares (pixels) by an n^2-element *grid*;
(ii) an *image function* f of two variables x and y, whose value in the interior of any pixel of an n^2-element grid is uniform.

Sometimes when this leads to no confusion, the function $f(x, y)$ in (ii) is called the digital or also the *discrete* image. The value of the image at the point (x, y) is often referred to as the *density* at (x, y).

In this work we will denote the image region, subdivided by an n^2-element grid, by $\mathbf{R}^2$, where

$$\mathbf{R} = \{-n/2, -n/2+1, -n/2+2, \ldots, 0, \ldots, n/2-2, n/2-1\} \ ,$$

in addition the center pixel $(0, 0)$ has its south-west corner at the origin of the coordinate system. That is, the pixel coordinates concur with the integral points of the plane, which is often not the case in computer science applications. Notice that the above denotation requires that n is even, which does not cause any loss of generality as we can always add zero rows and columns in the original image. Figure 5 illustrates an 8×8-element grid.

In principle the values of the function $f(x, y)$ are real numbers (gray-scale images) or arrays of real numbers (color images). Gray-scale images can be illustrated as shown in Figure 6.

In PET imaging the object of interest is the spatial distribution of radioactivity concentrations, that is, every pixel in the image represents the number of events of annihilation in that position. Because each tomographic slice has a certain thickness, a pixel is actually a volumetric box, a voxel. We will, however consider image elements as pixels.

Figure 7 shows an example of a so-called phantom image which are utilized in the place of a patient in the computational data acquisition. This particular phantom is often called the Shepp-Logan phantom. Phantom was generated

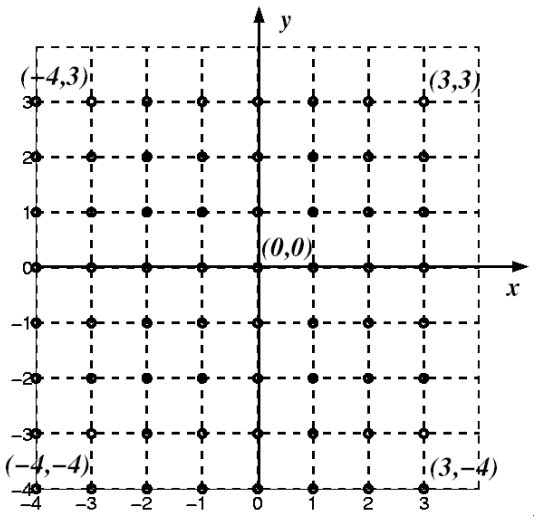

Figure 5: An 8×8 image grid.

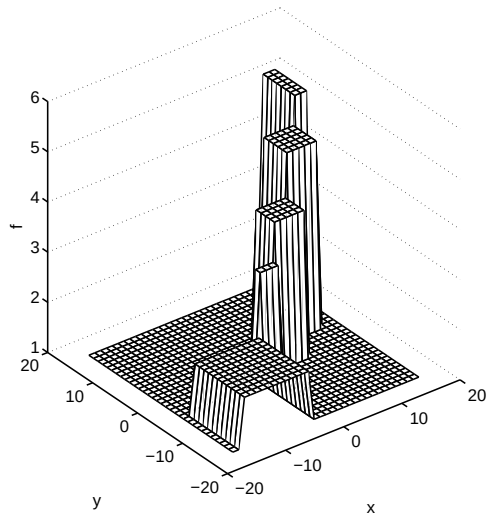

Figure 6: Illustration of a digital image function $f(x, y)$ on a 32×32 image grid.

with `MATLAB` function `phantom` on 64×64 image grid, image is displayed with 7 color intensities.

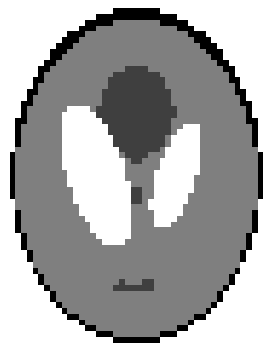

Figure 7: Shepp-Logan phantom defined on a 64×64 image grid.

3.2 Approximating the Radon transform

Computational data acquisition, modelling the data acquisition with a PET scanner, is performed by approximating the line integrals of an image function $f(x, y)$ in coincidence lines defined by the scanner geometrics. A line integral of $f(x, y)$ in a certain direction is called a *projection* of function $f(x, y)$ in that direction, or also the *Radon transform* of the function $f(x, y)$. For example, the line integral of $f(x, y)$ in the vertical direction is the projection of $f(x, y)$ onto the x-axis, and the line integral of $f(x, y)$ in the horizontal direction is the projection of $f(x, y)$ onto the y-axis. Figure 8 shows vertical and horizontal projections for a simple image function $f(x, y)$.

Analytically the Radon transform of an image function $f(x, y)$ along a line inclined at an angle θ from the y-axis and at distance s from the origin, denoted as $g(s, \theta)$, is written as

$$g(s, \theta) \triangleq \mathcal{R}f = \iint\limits_{\mathbf{R}\times\mathbf{R}} f(x, y)\delta(x\cos\theta + y\sin\theta - s)\, dx\, dy. \tag{1}$$

Where for s and θ; $-\infty < s < \infty$, $0 \leq \theta < \pi$.

In this work, our intention is to approximate the Radon transform of a discrete image function $f(x, y)$ by a discrete line integral. In our approach the discrete line integral will be calculated via a sum over the pixels intersected by a line. We define that a line intersects a pixel iff the line and the pixel have at least *two* common points. To make notations more simple we will use the following notation for the set of pixels inside the image region $\mathbf{R}^2$, intersected by the line

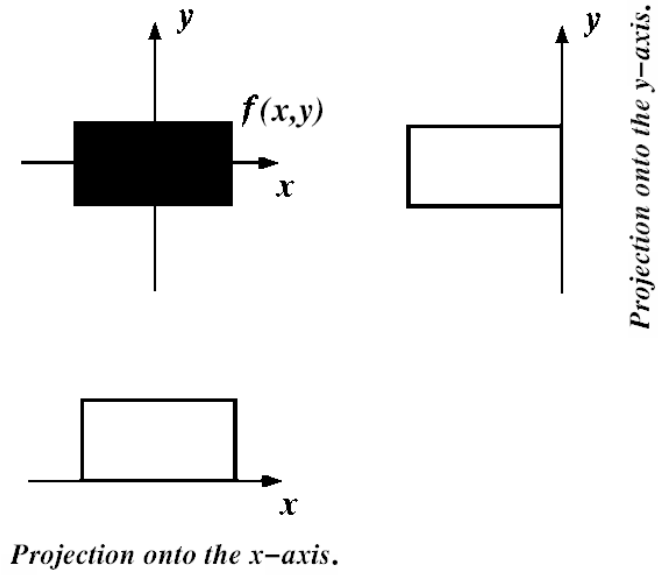

Figure 8: Vertical and horizontal projections of a simple image function $f(x, y)$.

defined by angle θ from the y-axis and distance s from the origin.

$$P_{(s,\theta)} = \{(x, y) \in \mathbb{Z}^2 \mid (x, y) \in \mathbf{R}^2 \text{ , pixel in place } (x, y) \text{ is intersected.}\}$$

Pixels in the set $P_{(s,\theta)}$ for the line in distance s from the origin and angle θ from the y-axis are filled in the Figure 9.

We now define the approximation formula for the line integral of a discrete image function $f(x, y)$ to be utilized throughout this work.

Definition 3.2(*Discrete line integral*)**.** Assume that the set $P_{(s,\theta)}$ is known in image region $[-\frac{n}{2}, \frac{n}{2} - 1] \times [-\frac{n}{2}, \frac{n}{2} - 1]$. Then the line integral of a discrete image function $f(x, y)$ in line defined by distance s and angle θ, is approximated by a weighted sum over the pixels inside the image region. Let us denote the sum by $\hat{g}(s, \theta)$.

$$\hat{g}(s, \theta) \triangleq \sum_{i=-\frac{n}{2}}^{\frac{n}{2}-1} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}-1} \varepsilon_{(s,\theta)}(i, j) f(i, j) \ , \tag{2}$$

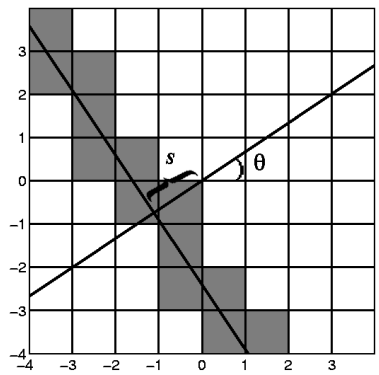

Figure 9: Pixels inside an 8×8 image grid intersected by the line (s, θ).

where

$$\varepsilon_{(s,\theta)}(i,j) = \begin{cases} k(i,j) \neq 0 & \text{if } (i,j) \in P_{(s,\theta)}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The non zero weighting coefficients $k(i,j)$ in (3) can be defined in three ways:
(i) $k(i,j) = 1$ for all $(i,j) \in P_{(s,\theta)}$,
(ii) $k(i,j) = \frac{l}{\sqrt{2}}$ for all $(i,j) \in P_{(s,\theta)}$, where l is the *length of intersection* shown in Fig.10 and $\sqrt{2}$ is the maximum length of intersection i.e. the diagonal of a square.

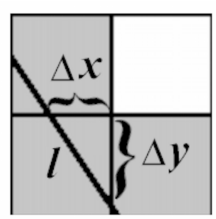

Figure 10: Length of intersection: $l = \sqrt{\Delta x^2 + \Delta y^2}$.

Third definition of the coefficients $k(i,j)$ is motivated by the structure of a PET scanner. Indeed, although events registered by a pair of detectors are all located to a coincidence line the event may have originated from a point P which does not belong to the line. Therefore coincidence lines could also be regarded as coincidence *stripes* as shown in Figure 11. Width of a coincidence stripe is d the distance between two neighbouring coincidence lines, and the corresponding

coincidence line bisects the coincidence stripe.

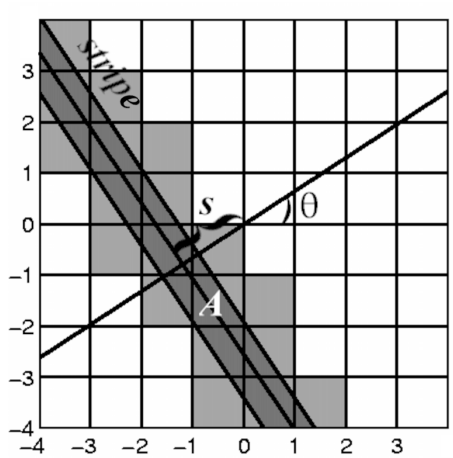

Figure 11: Corresponding coincidence stripe of the coincidence line in angle θ and distance s.

With notations from Figure 11:
(iii) $k(i,j) = A$, where A is the area of a stripe overlapping a pixel in place (i,j).

Usually the set $P_{(s,\theta)}$ and the set of pixels intersected by the corresponding coincidence stripe are not identical. Compare Figures 9 and 11 for example. Nevertheless we will use the same notations for both sets, as it can be concluded from the context which one is meant.

Mathematically, the computational data acquisition for scanner geometrics (S, Θ, d) is written as a set of equations,

$$\begin{cases} \hat{g}(s_0,\theta_0) = \varepsilon_{(s_0,\theta_0)}(1,1)f(1,1) + \varepsilon_{(s_0,\theta_0)}(1,2)f(1,2) + \ldots + \varepsilon_{(s_0,\theta_0)}(n,n)f(n,n) \\ \hat{g}(s_0,\theta_1) = \varepsilon_{(s_0,\theta_1)}(1,1)f(1,1) + \varepsilon_{(s_0,\theta_1)}(1,2)f(1,2) + \ldots + \varepsilon_{(s_0,\theta_1)}(n,n)f(n,n) \\ \ldots \\ \hat{g}(s_{S-2},\theta_{\Theta-1}) = \varepsilon_{(s_{S-2},\theta_{\Theta-1})}(1,1)f(1,1) + \ldots + \varepsilon_{(s_{S-2},\theta_{\Theta-1})}(n,n)f(n,n) \\ \hat{g}(s_{S-1},\theta_{\Theta-1}) = \varepsilon_{(s_{S-1},\theta_{\Theta-1})}(1,1)f(1,1) + \ldots + \varepsilon_{(s_{S-1},\theta_{\Theta-1})}(n,n)f(n,n) \end{cases}, \tag{4}$$

where $f(i,j)$ and $\varepsilon_{(s,\theta)}(i,j)$ are renumbered as follows,

$$\begin{cases} f(1,1) \triangleq f(-n/2, n/2+1) \\ f(1,2) \triangleq f(-n/2+1, n/2+1) \\ \dots \\ f(n,n) \triangleq f(n/2-1, -n/2) \\ \text{and} \\ \varepsilon_{(s,\theta)}(1,1) \triangleq \varepsilon_{(s,\theta)}(-n/2, n/2+1) \\ \varepsilon_{(s,\theta)}(1,2) \triangleq \varepsilon_{(s,\theta)}(-n/2+1, n/2+1) \\ \dots \\ \varepsilon_{(s,\theta)}(n,n) \triangleq \varepsilon_{(s,\theta)}(n/2-1, -n/2) \ . \end{cases}$$

In matrix language the computational data acquisition for scanner geometrics (S, Θ, d) can be expressed as

$$\mathbf{G} = \mathbf{EF} \ , \tag{5}$$

where $\mathbf{G}$ is the lefthand side of the equation (4), that is, $\mathbf{G}_0 = g(s_0, \theta_0)$ etc. $\mathbf{E}$ is an $S\Theta \times n^2$ *projection matrix*, that is $\mathbf{E}_{s\theta,ij} = \varepsilon_{(s,\theta)}(i,j)$ in (4), and $\mathbf{F}$ is an $n^2 \times 1$ *image vector*, where $\mathbf{F}_0 = f(1,1)$ in (4) etc.

In our terminology, the column vector $\mathbf{G}$ in (5) is the sinogram and the image vector $\mathbf{F}$ is the input image. In computational data acquisition the sinogram is obtained by multiplying the input image from the left by the projection matrix, which contains the weighting coefficients for every pixel inside the image region.

Assosiated with the Radon transform is the *back-projection operator* $\mathcal{B}$, which in continuos space is defined as

$$b(x,y) \triangleq \mathcal{B}g = \int_0^{\pi} g(xcos\theta + ysin\theta, \theta)\, d\theta. \tag{6}$$

The back-projection operator $\mathcal{B}$ maps a function of (s,θ) coordinates into a function of spatial coordinates (x,y). Figure 12 shows back-projections of simple functions $g(s,0)$ and $g(s,\pi/2)$, which were obtained by taking horizontal and vertical projections of function $f(x,y)$ in Fig. 8.

Assume that the projection matrix is computed in scanner geometrics (S, Θ, d). Consider a sinogram, denoted by $\mathbf{G}$ as in (5), containing projections in lines belonging to scanner geometrics (S, Θ, d). Then the back-projection of the sinogram is approximated by multiplying the sinogram by the *transpose* of the

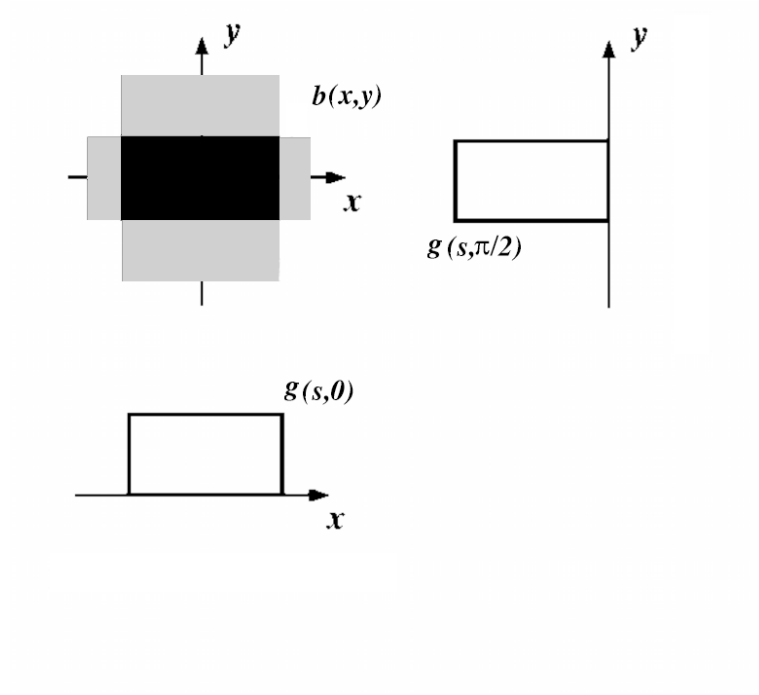

Figure 12: Back-projections of functions $g(s,0)$ and $g(s,\pi/2)$.

projection matrix from right. That is,

$$\mathbf{B} = \mathbf{E}^{\top}\mathbf{G} \ , \tag{7}$$

where $\mathbf{B}$ is an $n^2 \times 1$ image vector.

3.3 Properties of The Family of Coincidence Lines

The computational data acquisition is perfomed by approximating the line integrals of an image function $f(x,y)$ in coincidence lines defined by scanner geometrics (S,Θ,d). Evaluating the approximation formula in Definition 3.2 for one line requires approximately $10n$ [3] algebraic operations, and the number of coincidence lines $S \times \Theta$ is in the class $O(10^5)$, thus the computational cost of performing the computational data acquisition is in the class $O(n10^6)$. Although the computational data acquisition can be performed in linear time, we are still able to downsize the factor 10^6 by exploiting the special properties of the family of coincidence lines.

[3] n is the dimension of the image grid

In what follows we will first mathematically descripe the family of coincidence lines according to scanner geometrics (S, Θ, d). We will then define explicitly the setting of a virtual detector ring and an input image, modelling a PET scanner and a patient. And then finally show how, by utilizing the special properties of the family of coincidence lines the factor 10^6 can be reduced to 10^5.

3.3.1 Definition of The Family of Coincidence Lines

Let us begin by establishing the range of angles θ. The angles θ from the y-axis are limited in the range $[0, \pi)$ as the coincidence line in angle θ and distance s concurs with the line in angle $\theta + \pi$ and distance $-s$. With the approximation formula $\hat{g}$ we can expres this fact by writing

$$\hat{g}(s, \theta) = \hat{g}(-s, \theta \pm \pi).$$

Let the number of angles be Θ, then the set of angles θ is

$$\{0, \pi/\Theta, 2\pi/\Theta, \ldots, (\Theta - 1)\pi/\Theta\} \ , \tag{8}$$

that is, the range of angles $[0, \pi)$ is divided into Θ equally spaced angles.

After the geometrical correction the coincidence lines belonging in the same angle θ from the y-axis are S parallel, equally spaced lines with spacing d.

Let us denote the family of coincidence lines, defined by scanner geometrics (S, Θ, d), by $C_{(S,\Theta,d)}$.

$$C_{(S,\Theta,d)} = \{(s, \theta) \mid s = pd \ , \ \theta = r\pi/\Theta\} \ , \tag{9}$$

where

$$p = \begin{cases} -\frac{S+1}{2}, -\frac{S+1}{2} + 1, -\frac{S+1}{2} + 2, \ldots, -\frac{1}{2}, \frac{1}{2}, \ldots, \frac{S+1}{2} & \text{if } S \text{ is even,} \\ -\frac{S-1}{2}, -\frac{S-1}{2} + 1, -\frac{S-1}{2} + 2, \ldots, -1, 0, 1, \ldots, \frac{S-1}{2} & \text{if } S \text{ is odd,} \end{cases} \tag{10}$$

and

$$r = 0, 1, 2, \ldots, \Theta - 1 \ . \tag{11}$$

And the line denoted by coordinates (s, θ) is the set of points:

$$(s, \theta) = \{(x, y) \mid y = \frac{s - x\cos\theta}{\sin\theta} \ , \ x \in \mathbb{R}\} \tag{12}$$

or equivalently,

$$(s, \theta) = \{(x, y) \mid x = \frac{s - y\sin\theta}{\cos\theta} \ , \ y \in \mathbb{R}\} \ . \tag{13}$$

3.3.2 Symmetries of the Family of Coincidence Lines

Let us now introduce the concepts of isometry and symmetry. Later we will see how the symmetries of the family of coincidence lines are utilised to obtain a more efficient algorithm for performing the computational data acquisition.

A plane isometry is any function $\alpha : \mathbb{R}^2 \to \mathbb{R}^2$ that preserves distance:

$$\forall (x_1, y_1), (x_2, y_2) \in \mathbb{R}^2 : d(\alpha(x_1, y_1), \alpha(x_2, y_2)) = d((x_1, y_1), (x_2, y_2))$$

where distance $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^2$ is the usual Euclidean distance.

If we denote the points in the plane by capital letters the isometry property can also be written as

$$\forall P, Q \in \mathbb{R}^2 : d(\alpha(P), \alpha Q) = d(P, Q).$$

It can be shown that an isometry maps every line into a line, and that an isometry preserves parallelism of lines. It can also be shown that there exists only four types of isometries of the plane. Namely, *translations, rotations, reflections and glide reflections.*

A translation by vector $A = (a, b)$ shifts every point (x, y) into position $(x + a, y + b)$. We denote a translation by vector A as τ_A.

Let $C \in \mathbb{R}^2$ be a point, and $\psi \in \mathbb{R}$ an angle.The rotation $\rho_{C,\psi}$ by the directed angle ψ about point C is the isometry that fixes point C, and otherwise takes point $P \neq C$ into the point P' where $d(C, P) = d(C, P')$ and ψ is the directed angle from CP to CP'. Point C is called the center of the rotation. If $\psi = \pi$ we get a special case of the rotation called the halfturn about point C. In terms of analytic geometry a rotation by the directed angle ψ about point $C = (c_x, c_y)$ can be expressed as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} x - c_x \\ y - c_y \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}. \tag{14}$$

The reflection σ_m in line m is the mapping that does not move the points of line m, but any point P outside line m is moved to the point P' such that line m is the perpendicular bisector of segment PP'.

A glide reflection is a composition of a translation and reflection in line m that is parallel with the direction of translation.

Let $\mathcal{S} \subseteq \mathbb{R}^2$ be a set of points. We say that isometry α is a *symmetry* of set $\mathcal{S}$ if and only if $\alpha(\mathcal{S}) = \mathcal{S}$. Extension of an isometry to apply on a set of points $\mathcal{S}$

is defined by,

$$\alpha(\mathcal{S}) = \bigcup_{P \in \mathcal{S}} \alpha(P).$$

We will now show that the symmetries of set $\mathcal{S}$ form a subgroup of the group of isometries.

Theorem 3.1. Let $\mathcal{S} \subset \mathbb{R}^2$ be an arbitrary set of points. The symmetries of $\mathcal{S}$ form a *subgroup* of $\mathcal{I}$, the *group* of isometries.

Proof. Every set has at least one symmetry, namely the trivial isometry ι. If $\alpha(\mathcal{S}) = \mathcal{S}$ the $\alpha^{-1}(\mathcal{S}) = \alpha^{-1}(\alpha(\mathcal{S})) = \mathcal{S}$, so the inverse of each symmetry of $\mathcal{S}$ is also a symmetry of $\mathcal{S}$. Let α and β be two symmetries of $\mathcal{S}$. Then $\alpha\beta(\mathcal{S}) = \alpha(\mathcal{S}) = \mathcal{S}$, so the product $\alpha\beta$ is also a symmetry of $\mathcal{S}$.

□

To downsize the number of operations required in the computational data acquisition we can use the symmetries of the image region. As the image region was defined to be a square we have exactly eight symmetries to observe. Indeed, in the square we may map the corner A into any of the four corners, after which corner B has still two possible images. Then the images of C and D are uniquely determined. The symmetries of a square $ABCD$ are presented in Figure 13.

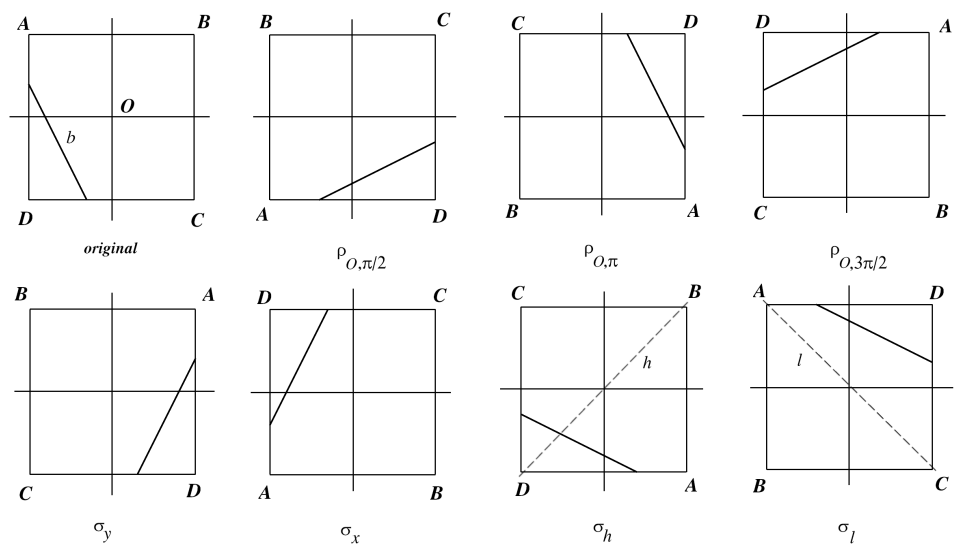

Figure 13: Symmetries of a square.

The symmetry group of a square consists of four rotations about the same center O, top line, and in addition reflections in four lines that meet at O at angles that are multiples of $\pi/4$, bottom line. Group is called the dihedral group, and denoted by D_4.

Let us consider line b in Figure 13 as a coincidence line defined by angle θ from the y-axis and distance s from the origin. We can see that we obtain seven new coincidence lines by applying the symmetries presented in Figure 13 to line b. We will now prove that the new coincidence lines are contained in the set $C_{(S,\Theta,d)}$ if the number of angles Θ is divisible by four.

Theorem 3.2. Let $\mathcal{S} \subset C_{(S,\Theta,d)}$, containing all the coincidence lines in angles $0 \leq \theta \leq \pi/4$ from the y-axis and in distances $-s_M \leq s \leq 0$ from the origin, where s_M is the maximal distance from the origin in scanner geometrics (S, Θ, d). Let scanner geometrics (S, Θ, d) be such that four divides Θ. Then the family of coincidence lines $C_{(S,\Theta,d)} = \bigcup_{\alpha \in D_4} \alpha(\mathcal{S})$, where D_4 is the group of symmetries of a square.

Proof. Recall the denotation of the coincidence lines by a pair (s, θ), where s is the distance from the origin and θ the angle from the y-axis, that is,

$$(s, \theta) = \{(x, y) \mid y = \frac{s - x\cos\theta}{\sin\theta} \ , \ x \in \mathbb{R}\}$$

or equivalently,

$$(s, \theta) = \{(x, y) \mid x = \frac{s - y\sin\theta}{\cos\theta} \ , \ y \in \mathbb{R}\} \ .$$

Proof of the theorem is divided into several cases according to the symmetry $\alpha \in D_4$.

(i) Assume that $\alpha = \rho_{O,\pi}$, the halfturn about the origin.

Our claim is that for $(s, \theta) \in \mathcal{S}$, $\alpha((s, \theta)) = (-s, \theta)$. Indeed,

$$\begin{aligned}
\alpha((s, \theta)) &= \{\alpha((x, y)) \mid y = \frac{s - x\cos\theta}{\sin\theta} \ , \ x \in \mathbb{R}\} \\
&= \{(-x, -y) \mid y = \frac{s - x\cos\theta}{\sin\theta} \ , \ x \in \mathbb{R}\} \\
&= \{(x, y) \mid \ -y = \frac{s + x\cos\theta}{\sin\theta} \ , \ x \in \mathbb{R}\} \\
&= (-s, \theta)
\end{aligned}$$

Above we used the fact that the halfturn about the origin maps a point (x, y) of the plane to its conjugate $(-x, -y)$, see Eq.14.

(ii) Assume that $\alpha = \rho_{O,\pi/2}$.

Our claim is that for $(s,\theta) \in \mathcal{S}$, $\alpha((s,\theta)) = (s, \theta + \pi/2)$.

$$\begin{aligned}
\alpha((s,\theta)) &= \{\alpha((x,y)) \mid y = \frac{s - x\cos\theta}{\sin\theta}\ ,\ x \in \mathbb{R}\} \\
&= \{(-y,x) \mid y = \frac{s - x\cos\theta}{\sin\theta}\ ,\ x \in \mathbb{R}\} \\
&= \{(x,y) \mid -x = \frac{s - y\cos\theta}{\sin\theta}\ ,\ y \in \mathbb{R}\} \\
&= \{(x,y) \mid -x = \frac{s - y\sin(\theta+\pi/2)}{-\cos(\theta+\pi/2)}\ ,\ y \in \mathbb{R}\} \\
&= (s, \theta + \pi/2).
\end{aligned}$$

(iii) Assume that $\alpha = \sigma_y$.
Our claim is that for $(s,\theta) \in \mathcal{S} \setminus (s,0)$, $\alpha((s,\theta)) = (s, \pi - \theta)$.

$$\begin{aligned}
\alpha((s,\theta)) &= \{\alpha((x,y)) \mid y = \frac{s - x\cos\theta}{\sin\theta}\ ,\ x \in \mathbb{R}\} \\
&= \{(-x,y) \mid y = \frac{s - x\cos\theta}{\sin\theta}\ ,\ x \in \mathbb{R}\} \\
&= \{(x,y) \mid y = \frac{s + x\cos\theta}{\sin\theta}\ ,\ x \in \mathbb{R}\} \\
&= \{(x,y) \mid y = \frac{s - x\cos(\pi-\theta)}{\sin(\pi-\theta)}\ ,\ x \in \mathbb{R}\} \\
&= (s, \pi - \theta).
\end{aligned}$$

(iv) Assume that $\alpha = \sigma_h$.
Our claim is that for $(s,\theta) \in \mathcal{S}$, $\alpha((s,\theta)) = (s, \pi/2 - \theta)$.

$$\begin{aligned}
\alpha((s,\theta)) &= \{\alpha((x,y)) \mid y = \frac{s - x\cos\theta}{\sin\theta}\ ,\ x \in \mathbb{R}\} \\
&= \{(y,x) \mid y = \frac{s - x\cos\theta}{\sin\theta}\ ,\ x \in \mathbb{R}\} \\
&= \{(x,y) \mid x = \frac{s - y\cos\theta}{\sin\theta}\ ,\ y \in \mathbb{R}\} \\
&= \{(x,y) \mid x = \frac{s - y\sin(\pi/2-\theta)}{\cos(\pi/2-\theta)}\ ,\ y \in \mathbb{R}\} \\
&= (s, \pi/2 - \theta).
\end{aligned}$$

For the rest of the cases we will apply the fact that the group D_4 is generated by the symmetries observed above, that is, $D_4 =< \rho_{O,\pi}, \rho_{O,\pi/2}, \sigma_y, \sigma_h >$.

Assume that $\alpha = \rho_{O,\pi}$, the halfturn about the origin. Then we obtain the rest of the coincidence lines in $C_{(S,\Theta,d)}$ by applying compositions $\alpha\beta$ to lines $(s,\theta) \in \mathcal{S}$, where $\beta \in \{\rho_{O,\pi/2}, \sigma_y, \sigma_h\}$,

$$\rho_{O,3\pi/2}(s,\theta) = \alpha\rho_{O,\pi/2}(s,\theta) = \alpha(s, \theta + \pi/2) = (-s, \theta + \pi/2),$$

$$\sigma_x(s,\theta) = \alpha\sigma_y(s,\theta) = \alpha(s,\pi-\theta) = (-s,\pi-\theta),$$

and

$$\sigma_l(s,\theta) = \alpha\sigma_h(s,\theta) = \alpha(s,\pi/2-\theta) = (-s,\pi/2-\theta).$$

We must then show that the lines obtained by applying the symmetries in the group D_4 to the set $\mathcal{S}$, are exactly the coincidence lines in the family $C_{(S,\Theta,d)}$. Recall the assumption that the number of angles Θ is divisible by four, that is, $\exists\Phi \in \mathbb{N} : \Theta = 4\Phi$.

The new lines obtained by applying the halfturn belong to the family $C_{(S,\Theta,d)}$ for all scanner geometrics. Indeed, as defined the distances s are pd, where

$$p = \begin{cases} -\frac{S+1}{2}, -\frac{S+1}{2}+1, -\frac{S+1}{2}+2, \ldots, -\frac{1}{2}, \frac{1}{2}, \ldots, \frac{S+1}{2} & \text{if } S \text{ is even,} \\ -\frac{S-1}{2}, -\frac{S-1}{2}+1, -\frac{S-1}{2}+2, \ldots, -1, 0, 1, \ldots, \frac{S-1}{2} & \text{if } S \text{ is odd,} \end{cases} \tag{15}$$

or equivalently the distances s are $\pm pd$, where

$$p = \begin{cases} \frac{S+1}{2}, \frac{S+1}{2}+1, \frac{S+1}{2}+2, \ldots, \frac{1}{2} & \text{if } S \text{ is even,} \\ \frac{S-1}{2}, \frac{S-1}{2}+1, \frac{S-1}{2}+2, \ldots, 1, 0 & \text{if } S \text{ is odd.} \end{cases} \tag{16}$$

Let us now consider the lines obtained by applying the symmetries $\rho_{O,\pi/2}, \sigma_y$ and σ_h. The new lines are in angles $\theta+\pi/2$, $\pi-\theta$ and $\pi/2-\theta$, where $0 \le \theta \le \pi/4$, respectively. The set of angles for such scanner geometrics where Θ is divisible by four, can be subdivided into four subsets containing the angles in the ranges $[0,\pi/4]$, $(\pi/4,\pi/2]$, $(\pi/2,3\pi/4]$ and $(3\pi/4,\pi)$. Let us denote the set of angles by $\mathcal{A}$;

$$\begin{aligned} \mathcal{A} = \{0, \frac{\pi}{4\Phi}, \frac{2\pi}{4\Phi}, \ldots, \frac{\Phi\pi}{4\Phi}\} \\ \bigcup \{\frac{(\Phi+1)\pi}{4\Phi}, \frac{(\Phi+2)\pi}{4\Phi}, \ldots, \frac{2\Phi\pi}{4\Phi}\} \\ \bigcup \{\frac{(2\Phi+1)\pi}{4\Phi}, \frac{(2\Phi+2)\pi}{4\Phi}, \ldots, \frac{3\Phi\pi}{4\Phi}\} \\ \bigcup \{\frac{(3\Phi+1)\pi}{4\Phi}, \frac{(3\Phi+2)\pi}{4\Phi}, \ldots, \frac{(4\Phi-1)\pi}{4\Phi}\} \end{aligned}$$

for some $\Phi \in \mathbb{N}$. Then we have,

$$\begin{aligned} \mathcal{A} = \{0, \frac{\pi}{4\Phi}, \frac{2\pi}{4\Phi}, \ldots, \frac{\Phi\pi}{4\Phi}\} \\ \bigcup \{\frac{\pi}{2} - \frac{(\Phi-1)\pi}{4\Phi}, \frac{\pi}{2} - \frac{(\Phi-2)\pi}{4\Phi}, \ldots, \frac{\pi}{2}\} \\ \bigcup \{\frac{\pi}{2} + \frac{\pi}{4\Phi}, \frac{\pi}{2} + \frac{2\pi}{4\Phi}, \ldots, \frac{3\pi}{4}\} \\ \bigcup \{\pi - \frac{(\Phi-1)\pi}{4\Phi}, \pi - \frac{(\Phi-2)\pi}{4\Phi}, \ldots, \pi - \frac{\pi}{4\Phi}\}\ , \end{aligned}$$

where the first angles of the partition are in the set $\mathcal{S}$, angles in the second part are obtained by applying the symmetry σ_h to the set $\mathcal{S}$, angles in the third part are obtained by applying the symmetry $\rho_{O,\pi/2}$ to the set $\mathcal{S}$ and finally angles in the last part are obtained by applying the symmetry σ_y to the set $\mathcal{S}$.

$$\therefore\ C_{(S,\Theta,d)} = \bigcup_{\alpha \in D_4} \alpha(\mathcal{S}).$$

□

3.4 Definition of a Virtual Scanner

Before giving the algorithms for performing the computational data acquisition we still have to decide how to attach the input image to the detector ring. Figure 14 shows a setting where the origin of the input image is set in the center of the detector ring, and the input image is scaled to cover exactly the *field of view* defined by the scanner geometrics. The field of view is determined by the outermost coincidence lines in angles 0 and $\pi/2$. These four lines form a square, and the field of view is the maximal circle drawn inside the square.

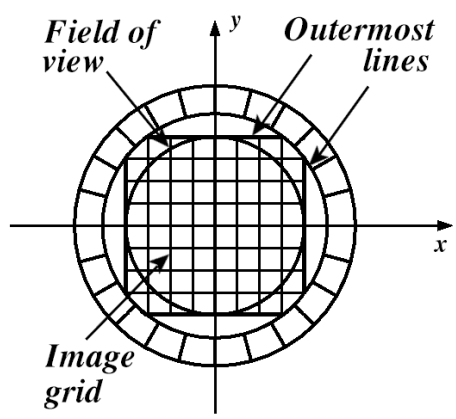

Figure 14: The input image is fixed in the center of the detector ring.

4 Algorithms

In this section we will give the required algorithms for performing the computational data acquisition. We will utilise the family

$$\mathcal{S} = \{(s, \theta) \mid -s_M \leq s \leq 0 \ , \ 0 \leq \theta \leq \pi/4\}$$

introduced in Theorem 3.2 as a base set of coincidence lines. All the rest of the set $C_{(S,\Theta,d)}$ is obtained by applying the symmetries of a square to the base set $\mathcal{S}$.

The computational data acquisition can be divided into three procedures:

1. Determine the set $P_{(s,\theta)}$ for coincidence lines in the family $\mathcal{S}$.
2. Compute the weighting coefficients $k(i,j)$ $\forall (i,j) \in P_{(s,\theta)}$.
3. Apply the symmetries of a square to determine the set $P_{(s,\theta)}$ and the corresponding weighting coefficients for coincidence lines in the family $C_{(S,\Theta,d)}$.

In the following sections the algorithms performing these three procedures are presented. The first two procedures will be combined to obtain more efficient implementation, and they are discussed together. Let us begin with the first two.

4.1 Determining The Set Of Intersected Pixels and The Weighting Coefficients

In this section we will give an algorithm for determining the set

$$P_{(s,\theta)} = \{(x, y) \in \mathbb{Z}^2 \mid (x, y) \in \mathbf{R}^2 \ , \text{pixel in place } (x, y) \text{ is intersected}\} \ ,$$

where $\mathbf{R}^2$ is the image region, s the distance from the origin and θ the angle from the y-axis, for coincidence lines in the family

$$\mathcal{S} = \{(s, \theta) \mid -s_M \leq s \leq 0 \ , \ 0 \leq \theta \leq \pi/4\} \ .$$

And the corresponding weighting coefficients.

It is easily seen that the angular coefficients of the lines in the family $\mathcal{S}$ are in the range $[-1, 0]$, that is, all the lines in the family $\mathcal{S}$ are strictly decreasing. This fact helps us in defining the algorithm for determining the set $P_{(s,\theta)}$, since it limits the possible ways a line can hit a pixel. Recall also, that a pixel is

intersected by the line (s, θ) if and only if the pixel and the line have at least *two* common points, that is, a pixel hit from only one corner does not belong to the set $P_{(s,\theta)}$. These observations limit the possible cuttings to six different cases. Five different cuttings are presented in the Figure 15.

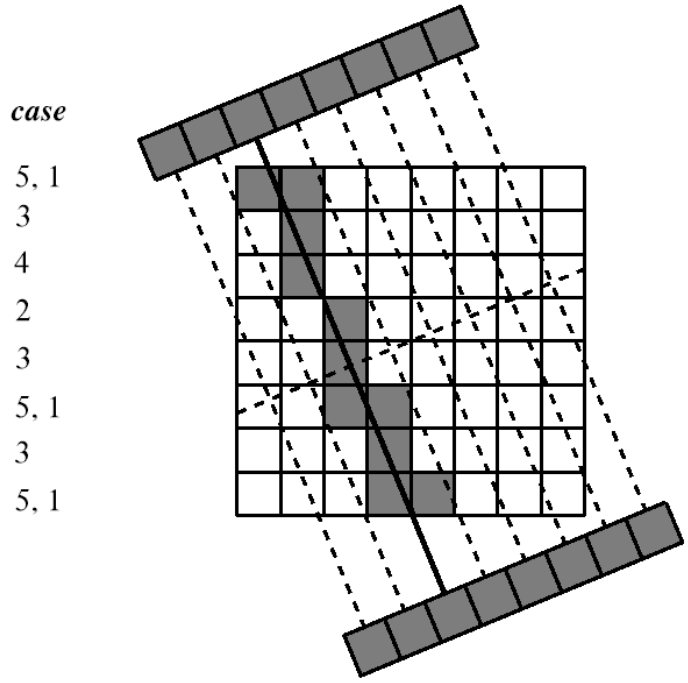

Figure 15: Pixels hit by the third line from the left are filled, and different cuttings are given a number from one to five. The sixth cutting, namely the diagonal of a pixel, is not shown here, since it only occurs for lines (s, θ) where $\theta = \pi/4$.

Our algorithm takes two arrays of coordinates $x \in \mathbb{R}$ as input and gives a list of coordinate pairs of intersected pixels as output. The input arrays, $\mathcal{A}_y$ and $\mathcal{A}_x$, are defined as follows,

$$\mathcal{A}_y = \{y \mid x \in \mathbf{R} \ , \ y = \frac{s - x\cos\theta}{\sin\theta}\}$$

and

$$\mathcal{A}_x = \{x \mid y \in \mathbf{R} \ , \ x = \frac{s - y\sin\theta}{\cos\theta}\} \ .$$

That is, array $\mathcal{A}_y$ contains the y-coordinates of points $(x, y) \in (s, \theta)$ $\forall x \in \mathbf{R}$, and array $\mathcal{A}_x$ contains the x-coordinates of points $(x, y) \in (s, \theta)$ $\forall y \in \mathbf{R}$.

Coordinates in arrays $\mathcal{A}_y$ and $\mathcal{A}_x$ are computed in points $\mathbf{R} = \{-n/2, -n/2 + 1, \ldots, n/2 - 2, n/2 - 1\}$, in this order.

Idea of our algorithm is to search for pixel coordinates in two parts. The first part is a horizontal search, where we go through the entries in the array $\mathcal{A}_y$ and search for pixels hit to the *right* edge, and the second part is a vertical search, where we go through the entries in the array $\mathcal{A}_x$ and search for pixels hit to the *bottom* edge. Defining the search in this way ensures that all the pixels hit by a decreasing line are found exactly once.

Figure 16 shows the pixels found in the two parts of the algorithm.

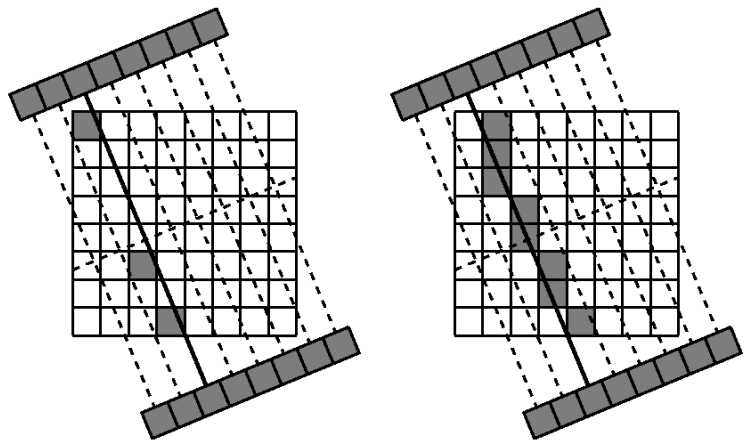

Figure 16: In the left the pixels found in the horizontal search, and in the right pixels found in the vertical search.

The non-zero coefficients $k(i,j)$ to be computed have three different definitions in Def.3.2, which all give different discretisation for the continuous line integral. In the first discretisation scheme the non-zero factors are all set to one, in which case we only need to find the intersected pixels. In the second discretisation scheme we must calculate the length of intersection for every intersected pixel. And finally in the thirth case we calculate the exact area of the stripe overlapping a pixel.

Let us first observe the second case where the weighting coefficient $k(i,j)$ is defined as follows,

$$k(i,j) = \frac{l}{\sqrt{2}} \qquad \forall (i,j) \in P_{(s,\theta)} \ ,$$

where l is the length of intersection shown in Fig.17 and $\sqrt{2}$ is the maximum length of intersection i.e. the diagonal of a square.

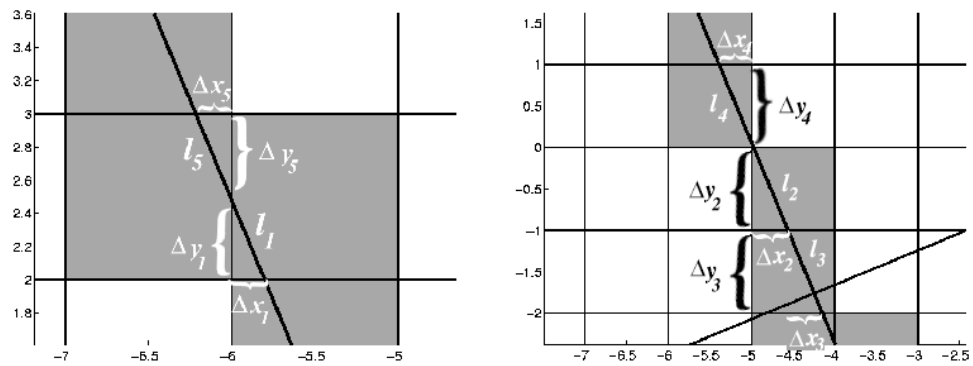

Figure 17: Computing the length of intersection for different cuttings (cases 1-5).

With denotations from Fig.17 the length of intersection l is

$$l_i = \sqrt{\Delta x_i^2 + \Delta y_i^2} \qquad \text{for} \qquad i = 1, 2, \ldots, 5 \ ,$$

where indices i also correspond to the cases shown in Fig.15.

The `MATLAB` code in algorithm 4.1 takes the image dimension n, angle $\theta \in [0, \pi/4]$ from the y-axis and distance s from the origin as input and returns a matrix containing the x-coordinates of the hit pixels in the first column, the y-coordinates of the hit pixels in the second column and the corresponding weighting coefficients in the last column.

Algorithm 4.1 (*Get coordinates and weighting coefficients*). File: `getweigths.m`

Let us now consider the thirth discretisation type. A strictly decreasing stripe can cut a pixel in twelve different ways, which are shown in Figure 18.

Our second algorithm takes arrays $\mathcal{A}_y^{(1)}$ and $\mathcal{A}_x^{(1)}$ for line (s_1, θ) and arrays $\mathcal{A}_y^{(2)}$ and $\mathcal{A}_x^{(2)}$ for line (s_2, θ) as input, and returns an array of weighting coefficients. Line (s_1, θ) is the left border and (s_2, θ) the rigth border of a stripe whose bisector is in distance s from the origin and in angle θ from the y-axis.

In the first part of the algorithm we again go through the entries in the array $\mathcal{A}_y^{(1)}$. Cases 1 and 11 in Figure 18 present the possible cuttings found in the first part, as they are hit to the right edge. Others are determined in the last part. In the last part we are simuntaneously computing weighting coefficients for two neighbouring pixels. If for example a pixel in place (x, y) is cut as shown in the

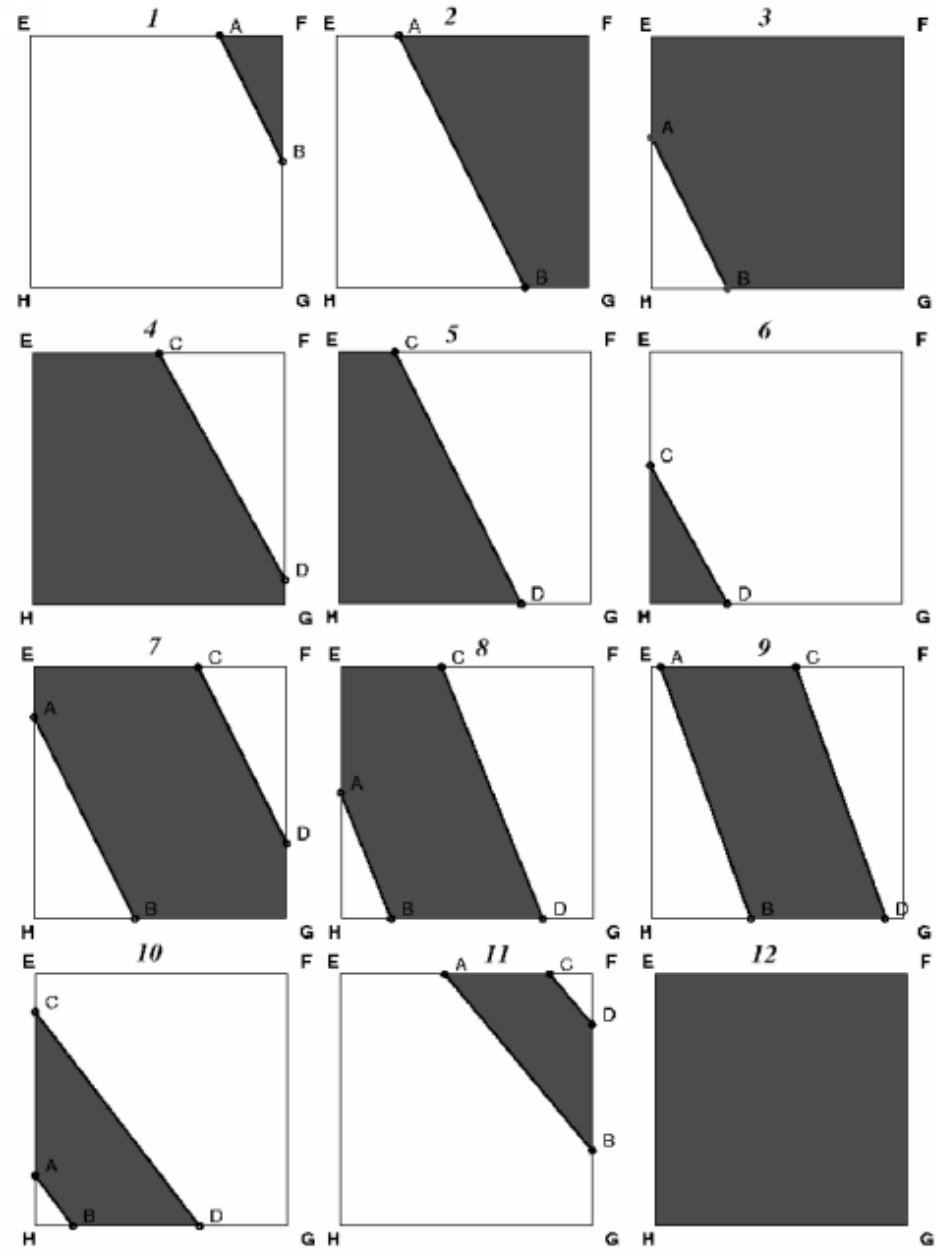

Figure 18: Different cuttings with a stripe.

case 2 in Fig.16, we notice that the pixel in place $(x+1, y)$ is cut as shown in the case 4, 5 or 12, and we must compute the weighting coefficient also for that pixel. The following table lists the connections between different cuttings.

cutting on pixel (x, y)	possible cuttings on pixel $(x+1, y)$
1	(7 and 8)
2 and 3	4, 5 and 12
4	6
5 and 6	-
7	6
8, 9 and 10	-
11	(10)
12	(4 and 5)

In the first part we only compute the weighting coefficient for pixel in place (x, y) as the pixel in place $(x+1, y)$ is also found in the last part of the algorithm. Cuttings shown in the cases 4, 5, 6 and 12 do not occur for pixel in place (x, y) as the left border line does not intersect the pixel. Thus only cuttings where we must compute the weigthing coefficient also for the pixel in place $(x+1, y)$ are shown in the cases 2, 3, 7 and 10 in Figure 18.

The `MATLAB` code in algorithm 4.2 takes the image dimension n, angle $\theta \in [0, \pi/4]$ from the y-axis and distance s from the origin as input and returns a matrix containing the x-coordinates of the hit pixels in the first column, the y-coordinates of the hit pixels in the second column and the corresponding weighting coefficients in the last column.

Algorithm 4.2(*Get coordinates and weighting coefficients*). File: `getweigths_ea.m`

4.2 Applying The Symmetries Of A Square

References

[1] G. T. Herman: Image Reconstruction From Projections - The Fundamentals of Computerized Tomography, *Academic Press*, 1980.

[2] Anil K. Jain: Fundamentals of Digital Image Processing, *Prentice Hall*, 1989.

A Source Codes

Printed by Jarkko Johansson

May 05, 04 14:41 **getweights.m** Page 1/1

```
function C = getweights(N,THETA,S)
% GETWEIGHTS determine pixel coordinates and weighting coefficients.
%   The GETWEIGHTS function returns pixel coordinates of
%   the pixels in N x N image grid hit by a line defined
%   by angle THETA from the y-axis and distanse S from the
%   origin, and the weighting coefficient.
%
%   [X,Y,W] = GETWEIGHTS(N,THETA,S) returns the pixel coordinates
%   in N x N image grid hit by line (S,THETA), and the
%   weighting coefficients.
%

% Initialise the result matrix.
% Result matrix will contain the coordinates of the hit pixels
% in row and column indices and the weighting coefficients.
C = [];

% If angle is zero (0) find column only.
if (THETA == 0)
     x = floor(S + N/2 +1);
     C = [x*ones(N,1), (1:N)', ones(N,1)];

else
  % Set integer points in cartesian coordinate system.
  P = -N/2:1:N/2;

  % Set cosine and sine for this angle.
  COSINUS = cos(THETA);
  SINUS = sin(THETA);

  % Calculate x and y in integer points for this angle.
  X = (S - P*SINUS)/COSINUS;
  Y = (S - P*COSINUS)/SINUS;

  % Calculate left and rigth limit for indices in the horizontal search.
  % Notice that indices are positive integers.
  LEFT = floor(X(N+1) + N/2) + 1;
  RIGTH = floor(X(1) + N/2) + 1;

  if LEFT <= 0 LEFT=1; end
  if RIGTH > N+1 RIGTH=N+1; end

  % Search for pixels hit to the right side.
  for I = LEFT:RIGTH
    x = I - 1;
    y = N/2 - floor(Y(I));
    % Check that pixel is inside the image region
    if x > 0 & x <= N & y > 0 & y <= N
       % Compute delta x and y.
       DX = P(I) - X(N + 1 - (y-1));
       DY = 1 - (Y(I) - floor(Y(I)));
       if DX > 1 | DX < 0 DX = 1; end
       if DY > 1 | DY < 0 DY = 1; end
       LOI = sqrt(DX^2 + DY^2);

       % Add coordinates and the coefficient in the result vector.
       C = [C ; x, y, LOI];
    end
  end

  % Calculate top and bottom limit for indices in the vertical search.
  TOP = floor(Y(1) + N/2) + 1;
  BOTTOM = floor(Y(N+1) + N/2) + 1;

  if TOP > N+1 TOP = N+1; end
  if BOTTOM < 1 BOTTOM = 1; end

  % Search for pixels hit to the bottom.
  for I = TOP:-1:BOTTOM
    x = floor(X(I)) + N/2 + 1;
    y = N + 1 - I;
    % Check that pixel is inside the image region
    if x > 0 & x <= N & y > 0 & y <= N
       % Compute delta x and y.
       DX = X(I) - floor(X(I));
       DY = Y(x) - P(I);
       if DX > 1 | DX < 0 DX = 1; end
       if DY > 1 | DY < 0 DY = 1; end
       LOI = sqrt(DX^2 + DY^2);

       % Add coordinates and the coefficient in the result vector.
       C = [C ; x, y, LOI];
    end
  end
end

% Normalize the weighting coefficients.
if(length(C) > 0)
 C(:,3) = C(:,3)./sqrt(2);
end

return
```

Wednesday May 05, 2004 1/1