A network flow algorithm for reconstructing binary images from continuous X-rays

Kees Joost Batenburg

University of Antwerp, Belgium
joost.batenburg@ua.ac.be

Abstract Tomography is a powerful technique to obtain accurate images of the interior of an object in a nondestructive way. Conventional reconstruction algorithms, such as filtered backprojection, require many projections to obtain high quality reconstructions. If the object of interest is known in advance to consist of only a few different materials, corresponding to known image intensities, the use of this prior knowledge in the reconstruction procedure can dramatically reduce the number of required projections.

In previous work we proposed a network flow algorithm for reconstructing a binary image defined on a lattice from its projections. In this paper we propose a new algorithm for the reconstruction of binary images that do not have an intrinsic lattice structure and are defined on a continuous domain, from a small number of their projections.

Our algorithm relies on the fact that the problem of reconstructing an image from only two projections can be formulated as a *network flow problem* in a graph. We derive this formulation for parallel beam and fan beam tomography and show how it can be used to compute binary reconstructions from more than two projections.

Our algorithm is capable of computing high quality reconstructions from very few projections. We evaluate its performance on both simulated and real experimental projection data and compare it to other reconstruction algorithms.

Keywords: Geometric tomography, Discrete tomography, Image reconstruction, Network flow problems

1 Introduction

Tomography deals with the reconstruction of the density distribution inside an object from a number of its projections [15, 16], which are also known as *X-rays*. In many applications, such as the reconstruction of medical CT images, a large number of different density values will occur. Typically, the number of projections that is required to obtain sufficiently accurate reconstructions is large in such cases (more than a hundred).

For certain applications, however, it is known in advance that only a few possible density values can occur. Many objects scanned in industry for nondestructive testing or reverse engineering purposes are made of one homogeneous material, resulting in two possible density values: the material and the surrounding air. Another example is medical digital subtraction angiography, where one obtains projections of the distribution of a contrast in the vascular system.

In this paper we focus on the reconstruction of *binary* images, i.e., images that contain only two densities (usually for the interior of on object and the surrounding background). The problem of reconstructing binary images from their X-rays is an important problem in the fields of *geometric tomography* and *discrete tomography*.

Geometric tomography deals with the reconstruction of geometric objects from data about its sections, its projections, or both. The book [8] gives an excellent overview of the field. Although much of the work on geometric tomography is concerned with the reconstruction of rather specific objects, such as convex or star-shaped objects, the domain of geometric objects can also be as general as the collection of measurable sets. The problem of reconstructing a binary image from its projections can be considered as reconstructing a subset of the plane, corresponding to the interior of the object. The approach of this paper does not aim at *exact* reconstruction: the reconstruction is computed on a pixel grid, at a finite resolution.

According to [11, 12], the field of *discrete tomography* deals with the reconstruction of images from a small number of projections, where the set of pixel values is known to have only a few discrete values. On the other hand, when the field of discrete tomography was founded by Larry Shepp in 1994, the main focus was on the reconstruction of (usually binary) images for which the *domain* is a discrete set, which seems to be more natural as a characteristic property of discrete tomography. The number of pixel values may be as small as two, but reconstruction problems for more values are also considered.

By using the prior information about the set of possible pixel values, it is possible in certain cases to dramatically reduce the amount of projection data that is required to obtain accurate

reconstructions. For some applications, such as the atomic scale reconstruction of crystals from electron tomographic projections, using few projections is a necessity as the scanning process damages the sample [14]. In industrial imaging a reduction of the scanning time may result in cost savings. In medical imaging, reducing the number of projections reduces the amount of radiation used.

Since the 1990s discrete tomography has received considerable interest in the mathematics and computer science communities. Both theoretical and practical aspects have been studied; see [11,12] for an overview. Most of the theory was developed for reconstructing binary images that are defined on a square lattice, i.e., a subset of $\mathbb{Z}^2$. In [9] it was shown that the binary reconstruction problem is NP-hard for more than two projections.

In [3] an algorithm was proposed for reconstructing binary images that are defined on a lattice, using some smoothness assumptions. This algorithm exploits the fact that the reconstruction problem for only two projections *can* be solved in polynomial time. The proposed reconstruction procedure is iterative: in each iteration a new reconstruction is computed using only two projections *and* the reconstruction from the previous iteration. The new reconstruction simultaneously resembles the image from the previous iteration and adheres to the two selected projections.

In this paper we describe a new iterative algorithm for reconstructing binary images that do *not* have an intrinsic lattice structure (i.e., more general subsets of the plane), which is based on ideas similar to those used in [3]. Our new algorithm fits well in the domain of geometric tomography, whereas the algorithm from [3] can be considered as a discrete tomography algorithm.

First, a start solution is computed using the SIRT algorithm (see Chapter 7 of [15]). Subsequently, a sequence of two-projection subproblems is solved. To solve the two-projection problems efficiently, a different pixel grid has to be used in each iteration, corresponding to the selected pair of projections. The reconstruction problem can then be solved as a special case of the *minimum cost network flow* problem in graphs, for which efficient polynomial time algorithms are available [1]. Special care has to be taken to handle noisy projection data. We mainly focus on parallel beam tomography, as the network flow approach is particularly well suited for the parallel beam geometry. Our algorithm can also be applied to fan beam data. A limitation of our algorithm is that it cannot be used if all projection angles are very close together.

Alternative approaches to the problem of reconstructing binary images that do not have a lattice structure that have been considered in the literature are, among others, adaptation of continuous algebraic reconstruction algorithms to the binary reconstruction problem [5], stochastic reconstruction using Gibbs priors (Chapter 8 of [11]), linear programming [6, 20, 21], and D.C. programming [19]. To the best of our knowledge, no reconstruction results have been reported in the literature for

images of size larger than 256×256 pixels (the paper [19] reports on some results for images of size 256×256). Our reconstruction results show that images of this size can be reconstructed effectively.

Section 2 contains the basic definitions and concepts that are necessary to define our algorithm. In Section 3 we consider the problem of reconstructing a binary image from two projections and formulate this problem as a minimum cost flow problem. The algorithm for reconstructing images from more than two projections, which is described in Section 4, builds upon the techniques from Section 3. In Section 5, results of several experiments with simulated projection data are presented, followed by reconstruction results based on real-world data. The results are compared to two alternative algorithms. Section 6 concludes the paper.

2 Preliminaries

In this section the tomographic scanning geometry is described and notation is introduced to describe the imaging process mathematically.

2.1 Data collection geometries

This paper deals with *transmission tomography*: a beam passes through an unknown object, which attenuates the beam. The intensity of the attenuated beam is measured at the other side of the object by a detector array. The measured intensity of a ray depends on the length of the intersection between the ray and the unknown object, as well as on the materials the object is made of.

We consider two different scanning geometries: parallel beam and fan beam. In parallel beam tomography, parallel rays pass through the object. The resulting intensities are measured by a parallel array of detectors. The basic setup is shown in Figure 1a. The source and detector array rotate around the object, acquiring a number of projections from different angles. Parallel beams are often used in electron tomography, where the projections of a nanosample are acquired by an electron microscope.

Fan beam tomography is often used in medical imaging. Figure 1b shows the imaging setup. Rays from a single source pass through the object of interest. The detector array covers a range of angles. The rays from the source cover the entire cross-section of the object. Both the source and the detector array rotate around the object, acquiring projections from several angles. Parallel beam tomography can be considered as a special case of fan beam tomography, where the point source is located at an infinite distance from the object.

In the remainder of this paper we use the term *X-ray source* to denote the source of any beam (i.e., X-rays, electrons) that is used to measure projections of the unknown object.

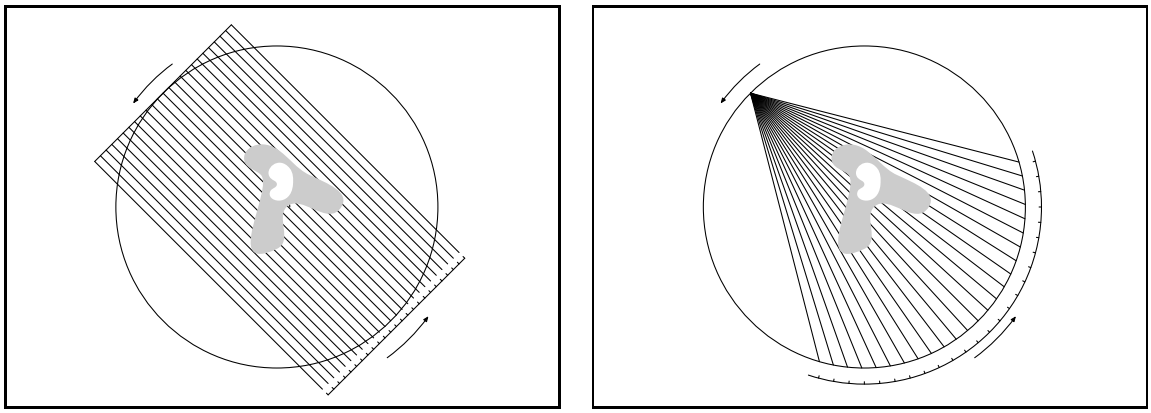


Figure 1: (a) Left: Parallel beam tomography setting (b) Right: Fan beam tomography setting

2.2 Definitions and notation

We denote the set $\{x \in \mathbb{R} : x \geq 0\}$ by $\mathbb{R}_{\geq 0}$. For any vector $v \in \mathbb{R}^n$, we denote the ℓ_1-norm (i.e., the sum-norm) of v by $|v|_1$ and the ℓ_2-norm by $|v|_2$. Let $R \in \mathbb{R}_{>0}$ be the *scanner radius*. Let $D = \{\theta_1, \ldots, \theta_d\}$ be a set of disjoint real numbers in the interval $[0, 2\pi)$, the *projection angles*. This is the set of angles at which the X-ray source is located during the measurements.

Let $T = \{t_0, \ldots, t_n\}$ be a set of real numbers in the interval $(-\frac{\pi R}{2}, \frac{\pi R}{2})$ satisfying $t_0 < t_1 < \ldots < t_n$. We call T the set of *fan parameters*. The fan parameters determine the set $F = \{\gamma_0, \ldots, \gamma_n\}$ of *fan angles* by the relation $\gamma_i = t_i/R$ for $0 \leq i \leq n$.

The scanner radius, the projection angles and the fan parameters jointly define the *data collection geometry* $S = (R, D, T)$. From this point on we assume that the data collection geometry is fixed. For $\tilde{t} \in (-\frac{\pi R}{2}, \frac{\pi R}{2})$, put $\gamma(\tilde{t}) = \tilde{t}/R$. Let $\theta \in [0, 2\pi)$, $t, t' \in (-\frac{\pi R}{2}, \frac{\pi R}{2})$. Define

$$l_\theta(x, y, t) = x \cos(\theta + \gamma(t)) + y \sin(\theta + \gamma(t)) - R \sin \gamma(t) \qquad x, y \in \mathbb{R}.$$

Define the *fan half line* $L_\theta(t)$ for projection angle θ and fan parameter t as

$$L_\theta(t) = \{(x, y) \in \mathbb{R}^2 : \quad l_\theta(x, y, t) = 0 \quad \text{and} \quad y \cos\theta - x \sin\theta < R\}$$

and the *fan set* $S_\theta(t, t')$ as

$$S_\theta(t, t') = \{\, (x, y) \in \mathbb{R}^2 : \quad l_\theta(x, y, t') \;\leq\; 0 \leq l_\theta(x, y, t) \,\}.$$

Figure 2b shows the geometric meaning of the last definitions. The half line $L_\theta(0)$ has angle θ with the y-axis, ends at $p = (-R\sin\theta, R\cos\theta)$ and passes through the origin O. Any half line $L_\theta(t)$ also ends in p and has angle $\gamma(t)$ with $L_\theta(0)$. The fan set $S_\theta(t,t')$ covers the area in between two such half lines.

The reason for using the *fan parameter* t instead of directly using the *fan angle* γ is that we can now consider the parallel beam geometry as a special case of the fan beam geometry. If we let $R \to \infty$, we obtain (using $\lim_{\gamma\to 0} \frac{\sin\gamma}{\gamma} = 1$):

$$S_\theta(t,t') = \{(x,y) \in \mathbb{R}^2 : \quad t \le x\cos\theta + y\sin\theta \le t'\}.$$

This is illustrated in Figure 2a. The line $x\cos\theta + y\sin\theta = t$ has an angle θ with the y-axis and has distance $|t|$ to the origin. By specifying the fan parameters instead of the fan angles, we can use the same model for both fan beam and parallel beam tomography.

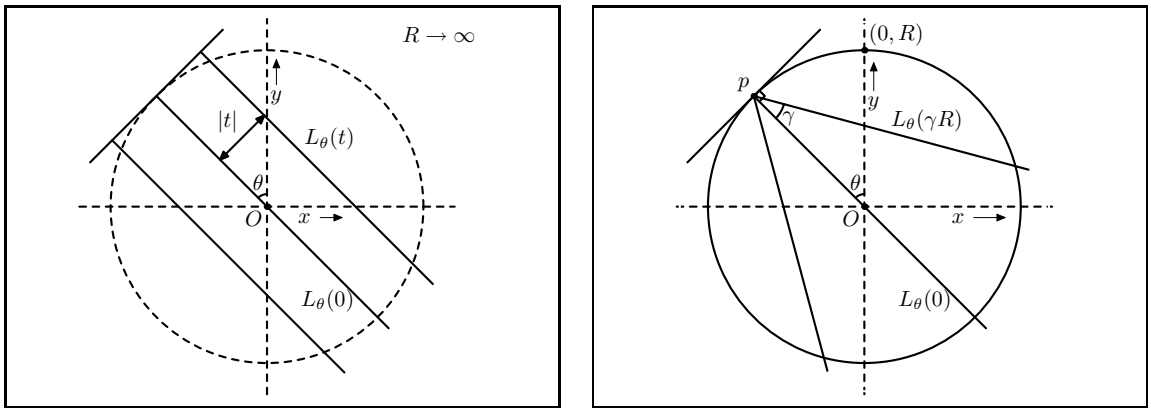


Figure 2: (a) Left: Parallel beam geometry (b) Right: Fan beam geometry

We call the set $S_{\theta_i}(t_0,t_n)$ the *field of view* for projection angle θ_i. Define the *imaging area* I as

$$I = \bigcap_{k=1}^{d} S_{\theta_k}(t_0,t_n).$$

The imaging area is the set of all points in $\mathbb{R}^2$ that are within the field of view for all projection angles.

3 Two projections

We first deal with the problem of reconstructing a gray level image or a binary image (i.e., black-and-white) from only two projections. We consider the unknown image as a mapping $f : I \to$

B, where B can be either the closed interval $[0,1]$ (gray value reconstruction) or the set $\{0,1\}$ (binary reconstruction). In this paper we assume that the projection data of the unknown object are measured as integrals of the form $\iint_{S_{\bar{\theta}_k}(t_{i-1},t_i)} f(x,y)\,\mathrm{d}x\,\mathrm{d}y$, which we call *strip projections*. This projection model represents a scanner device for which each of the X-ray detectors have a certain positive width. Another model that is often used in the tomography literature considers the projection data as line integrals, assuming a detector array where the individual detectors are located in single points.

Problem 1 *Let $\bar{\theta}_1, \bar{\theta}_2 \in D$ be two different projection angles. Let $p_1 = (p_{1,1} \dots p_{1,n})^T \in \mathbb{R}^n$, $p_2 = (p_{2,1} \dots p_{2,n})^T \in \mathbb{R}^n$ be two vectors of nonnegative real numbers (the measured strip projections for projection angles $\bar{\theta}_1$ and $\bar{\theta}_2$, respectively). Construct a function $f : I \to B$ such that*

$$\begin{aligned} \iint_{S_{\bar{\theta}_1}(t_{i-1},t_i)} f(x,y)\,\mathrm{d}x\,\mathrm{d}y &= p_{1,i} \qquad \text{for } i = 1,\dots,n \quad \text{and} \\ \iint_{S_{\bar{\theta}_2}(t_{i-1},t_i)} f(x,y)\,\mathrm{d}x\,\mathrm{d}y &= p_{2,i} \qquad \text{for } i = 1,\dots,n. \end{aligned}$$

Depending on whether B is the unit interval or the set $\{0,1\}$, we refer to the *gray value variant* and the *binary variant* of Problem 1 respectively.

To represent a mapping $f : I \to B$ in a computer we have to resort to an approximate representation. An image f is represented as a 2D array of pixels. Every measured strip projection then gives rise to a linear equation on the pixel values of f. Combining the linear equations for all measured strip projections yields a large system of equations. A real-valued solution of this system can be computed by methods from linear algebra, such as the ART algorithm (see Chapter 7 of [15]). However, it is likely that such a solution will not be *binary*. Note that the reconstruction problem from two projections is severely underdetermined, so it may have an infinite number of real-valued solutions.

Two projections jointly define a *two-projection grid*. The rows and columns of this grid correspond to the fan sets of the two projections. Define *grid cell* C_{ij} $(1 \le i,j \le n)$ as follows:

$$C_{ij} = S_{\bar{\theta}_1}(t_{i-1},t_i) \cap S_{\bar{\theta}_2}(t_{j-1},t_j).$$

Figure 3 shows examples of two-projection grids for the parallel and fan beam case.

For any polygon $V \subset \mathbb{R}^2$, denote its area by $A(V)$. As a shorthand notation we denote the area of grid cell C_{ij} by a_{ij}. In the parallel beam case all grid cells are parallelograms and have the same area.

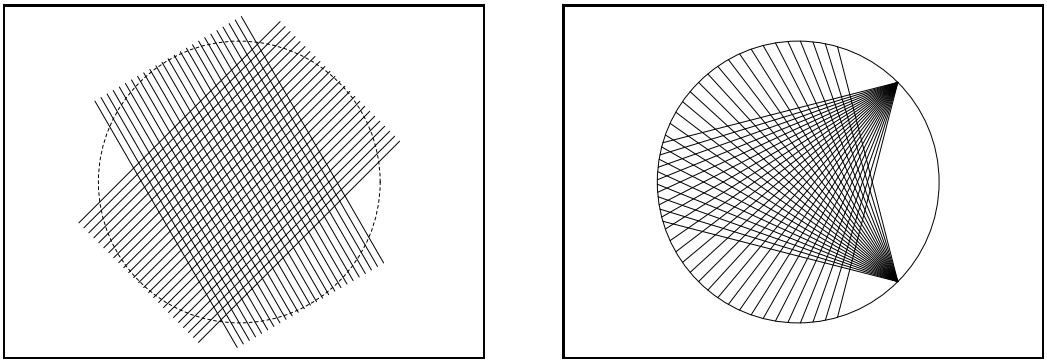


Figure 3: (a) (Left) Two parallel beam projections jointly define a two-projection grid. (b) (Right) Two-projection grid defined by two fan-beam projections.

For fan beam projections we have to make some additional assumptions on the pair of projections, to ensure that a two-projection grid can be constructed. We assume that

$$\begin{aligned} (-R\sin\bar{\theta}_1, R\cos\bar{\theta}_1) \quad &\notin \quad S_{\bar{\theta}_2}(t_0, t_n) \\ (-R\sin\bar{\theta}_2, R\cos\bar{\theta}_2) \quad &\notin \quad S_{\bar{\theta}_1}(t_0, t_n) \\ L_{\bar{\theta}_1}(t_i) \cap L_{\bar{\theta}_2}(t_j) \quad &\neq \quad \emptyset \qquad \text{for all} \quad 0 \le i, j \le n. \end{aligned} \tag{1}$$

The first two properties ensure that the source of the first projection is not within the fan range of the second projection and vice versa. The third property states that each fan half line of the first projection intersects each fan half line of the second projection.

A *two-projection image* is a mapping $\{1, \ldots, n\} \times \{1, \ldots, n\} \to [0, 1]$ which assigns a real value — the *gray value* — in the interval $[0, 1]$ to each grid cell of a two-projection grid. A *binary two-projection image* is a two-projection image for which all gray values are either 0 (black) or 1 (white). Figure 4 shows two examples of a binary two-projection image on the grids from Figure 3.

It is often convenient to consider a two-projection image X as a *matrix* (x_{ij}), where x_{ij} denotes the gray value of cell C_{ij}. It is straightforward to compute the strip projections of a two-projection image X for the projection angles $\bar{\theta}_1$ and $\bar{\theta}_2$, by summation of all entries in a row or column of X,

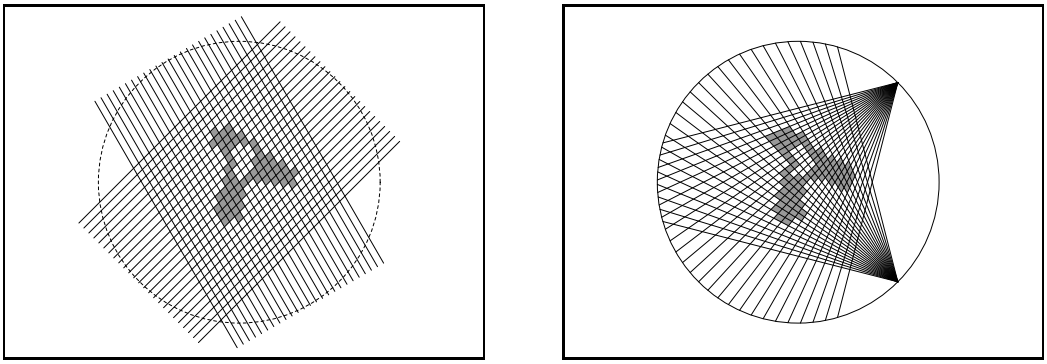


Figure 4: (a) Left: Binary two-projection image on the grid from Figure 3a (parallel beam case). (b) Right: Binary two-projection image on the grid from Figure 3b (fan beam case). Note that for the purpose of visualization, the background is shown as white, even though it usually corresponds to a gray value of 0.

weighted according to the cell areas. Define $P_1, P_2 : [0,1]^{n\times n} \to \mathbb{R}^n$ by

$$P_1(X) = \begin{pmatrix} \sum_{i=1}^n a_{i1}x_{i1} \\ \cdots \\ \sum_{i=1}^n a_{in}x_{in} \end{pmatrix} \tag{2}$$

and

$$P_2(X) = \begin{pmatrix} \sum_{j=1}^n a_{1j}x_{1j} \\ \cdots \\ \sum_{j=1}^n a_{nj}x_{nj} \end{pmatrix}. \tag{3}$$

We call P_1 and P_2 the *projections* of X for angles $\bar{\theta}_1$ and $\bar{\theta}_2$, respectively. Define the *total projection* $S : [0,1]^{n\times n} \to \mathbb{R}$ by

$$S(X) = \sum_{1\le i,j\le n} a_{ij}x_{ij}.$$

The representation of a binary image as a two-projection image is only an approximate representation. Possibly there does not exist a two-projection image for which the projections are exactly equal to the measured projections of Problem 1. In addition, if the measured projections have been obtained from a physical experiment they will usually contain errors. If no perfect solution of the reconstruction problem exists, we would like to find a reconstruction that adheres to the measured projections as well as possible. We now define a reconstruction problem for two-projection images that allows for such errors.

Problem 2 *Let $\bar{\theta}_1, \bar{\theta}_2 \in D$ be two different projection angles. Let $p_1 = (p_{1,1} \dots p_{1,n})^T \in \mathbb{R}^n$, $p_2 = (p_{2,1} \dots p_{2,n})^T \in \mathbb{R}^n$ be two vectors of nonnegative real numbers. Let $\bar{T} \in \mathbb{R}$ (the* prescribed

total projection*). Construct a matrix* $X \in B^{n \times n}$ *such that* $S(X) = \bar{T}$ *and*

$$|P_1(X) - p_1|_1 + |P_2(X) - p_2|_1$$

is minimal.

In any instance of Problem 2, $S(X)$ is considered to be a fixed parameter. A good value for the parameter $\bar{T}$ can be computed from the measured projection data. For a binary image, $S(X)$ represents the total area of the white pixels in X. For any image X we have $S(X) = |P_1(X)|_1 = |P_2(X)|_1$. A sensible value for $\bar{T}$ if the projection data is noisy is given by $(|p_1|_1 + |p_2|_1)/2$. For the parallel beam case, Problem 2 can only have a binary solution if $\bar{T}$ is an integral multiple of the pixel area a. A good value for $\bar{T}$ can be found by rounding $(|p_1|_1 + |p_2|_1)/2$ to the nearest multiple of a. For fan beam projections, such a rounding step cannot be applied, as the cell area is not constant. In the fan beam case we only deal with the gray value variant of Problem 2, as the binary variant cannot be solved efficiently within our proposed model.

We will now show that Problem 2 can be formulated as a *mininimum cost flow problem* in a particular graph. In fact, this network flow approach can be used to solve a generalization of Problem 2. This generalization offers the possibility of incorporating prior knowledge about the unknown image in the reconstruction, by specifying a *weight* w_{ij} for each pixel (i, j):

Problem 3 *Let* $\bar{\theta}_1, \bar{\theta}_2, p_1, p_2, \bar{T}$ *be as in Problem 2. Let* $W = (w_{ij}) \in \mathbb{R}^{n \times n}$ *and* $\alpha \in \mathbb{R}$*. Construct a matrix* $X \in B^{n \times n}$ *such that* $S(X) = \bar{T}$ *and*

$$\alpha(|P_1(X) - p_1|_1 + |P_2(X) - p_2|_1) - \sum_{1 \le i,j \le n} w_{ij} a_{ij} x_{ij}$$

is minimal.

Problem 3 is a generalization of Problem 2. Setting $\alpha = 1$ and $w_{ij} = 0$ for $1 \le i, j \le n$ yields Problem 2. We call the matrix W the *weight map*. The weight map is used extensively in the algorithm for reconstructing a binary images from more than two projections that we describe in the next section.

The basic idea of using network flow methods for the reconstruction of binary images from two projections was first described by Gale in 1957 [7], in the context of reconstructing binary *matrices* from their row and column sums.

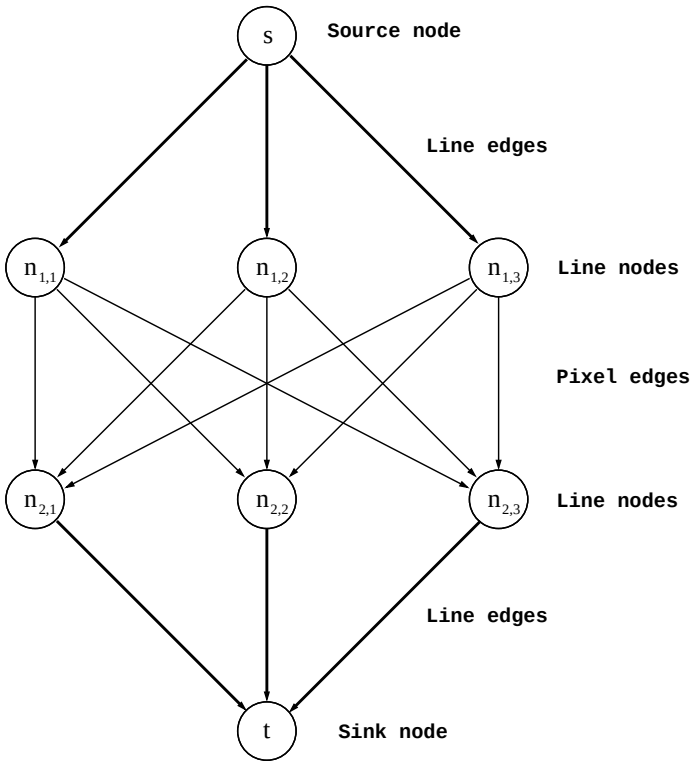


Figure 5: Basic structure of the associated graph

With a pair of projection angles $(\bar{\theta}_1, \bar{\theta}_2)$ and their respective measured projections (p_1, p_2), we associate a *directed* graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. We call G the *associated graph*; see Figure 5. The set V contains a node s (the *source*), a node t (the *sink*), one node for each strip of projection angle $\bar{\theta}_1$ and one node for each strip of projection angle $\bar{\theta}_2$. The node that corresponds to $S_{\bar{\theta}_k}(t_{i-1}, t_i)$ has label $n_{k,i}$. We call the nodes $n_{k,i}$ *line nodes*. Every pair $(n_{1,i},\ n_{2,j})$ of nodes is connected by a (directed) edge. We call these edges *pixel edges* and denote the set of all pixel edges by $E_p \subset E$. Besides the point edges the set E contains the subsets $E_1 = \{(s, n_{1,i}) : i = 1, \ldots, n\}$ and $E_2 = \{(n_{2,j}, t) : j = 1, \ldots, n\}$ of directed edges. We call the elements of E_1 and E_2 the *line edges of* G. The complete set of edges of G is given by $E = E_p \cup E_1 \cup E_2$. In the remainder of this section we assume that the reader is familiar with the basic concepts of *network flows*. The book [1] provides an excellent introduction to this subject.

To each edge $e \in E$ we assign a real-valued capacity, according to the following *capacity function* $U : E \to \mathbb{R}_{\geq 0}$:

$$\begin{aligned} U((n_{1,i}, n_{2,j})) &= a_{ij} \qquad \text{for} \quad 1 \le i, j \le n, \\ U((s, n_{1,i})) &= \sum_{j=1}^{n} a_{ij} \qquad \text{for} \quad 1 \le i \le n, \\ U((n_{2,j}, t)) &= \sum_{i=1}^{n} a_{ij} \qquad \text{for} \quad 1 \le j \le n. \end{aligned}$$

A *flow* in G is a mapping $Y : E \to \mathbb{R}_{\ge 0}$ such that $Y(e) \le U(e)$ for all $e \in E$ and such that for all $v \in V \backslash \{s, t\}$:

$$\sum_{w:\,(w,v)\in E} Y((w, v)) = \sum_{w:\,(v,w)\in E} Y((v, w)).$$

Define the *size* S(Y) of a flow Y as

$$S(Y) = \sum_{i=1}^{n} Y((s, n_{1,i})) = \sum_{j=1}^{n} Y((n_{2,j}, t)) = \sum_{1\le i,j\le n} Y((n_{1,i}, n_{2,j})).$$

Note that if a flow Y is specified on the pixel edges, its value for all line edges can be computed using the flow conservation constraint. A flow Y is called an *integral flow* if $Y(e) \in \mathbb{N}_0$ for all $e \in E$, where $\mathbb{N}_0 = \{n \in \mathbb{Z} : n \ge 0\}$.

For any matrix $X \in [0, 1]^{n\times n}$ we define its *corresponding flow* Y_X by

$$Y_X((n_{1,i}, n_{2,j})) = a_{ij} x_{ij} \qquad \text{for} \quad 1 \le i, j \le n.$$

Note that Y_X is a mapping $E \to \mathbb{R}_{\ge 0}$, which we define by specifying its values on a subset of E. The value of Y_X on the line edges of G then follows from the flow conservation constraints:

$$Y_X((s, n_{1,i})) \;=\; [P_1(X)]_i \quad \text{for} \quad 1 \le i \le n \tag{4}$$

and

$$Y_X((n_{2,j}, t)) \;=\; [P_2(X)]_j \quad \text{for} \quad 1 \le j \le n. \tag{5}$$

With each edge $e \in E$ we assign a *cost function*, which determines the cost of sending a certain amount x of flow through that edge. The cost function C_e depends on the weight map W, the constant α and the measured projections p_1 and p_2 and is defined by

$$
\begin{aligned}
C_{(n_{1,i},n_{2,j})}(x) &= -w_{ij}x \qquad \text{for} \quad i,j = 1,\ldots,n,\\
C_{(s,n_{1,i})}(x) &= 2\alpha \max(x - p_{1,i}, 0) \quad \text{for} \quad i = 1,\ldots,n,\\
C_{(n_{2,j},t)}(x) &= 2\alpha \max(x - p_{2,j}, 0) \quad \text{for} \quad j = 1,\ldots,n.
\end{aligned}
$$

Define the *total cost* $C(Y)$ of a flow Y as

$$C(Y) = \sum_{e \in E} C_e(Y(e)). \tag{6}$$

Lemma 1 *There is a 1-1 correspondence between the solutions of the gray value variant of Problem 3 and the flows Y of size $\bar{T}$ in G for which $C(Y)$ is minimal among all flows of size $\bar{T}$.*

Proof. For any real-valued vector $p \in \mathbb{R}^n$, define $|p|^+ = \sum_{i=1}^n \max(p_i, 0)$. Combining (4), (5) and (6), we find that

$$C(Y_X) = 2\alpha(|P_1(X) - p_1|^+ + |P_2(X) - p_2|^+) - \sum_{1 \le i,j \le n} w_{ij} a_{ij} x_{ij}.$$

Using the identities

$$S(X) = |P_k(X)|_1 = |p_k|_1 + |P_k(X) - p_k|^+ - |p_k - P_k(X)|^+ \quad \text{for} \quad k = 1,2$$

and

$$|P_k(X) - p_k|_1 = |P_k(X) - p_k|^+ + |p_k - P_k(X)|^+,$$

we find that

$$
\begin{aligned}
C(Y_X) = \alpha(|P_1(X) - p_1|_1 + |P_2(X) - p_2|_1) - \sum_{1 \le i,j \le n} w_{ij} a_{ij} x_{ij} \; + \\
\alpha(2S(X) - |p_1|_1 - |p_2|_1).
\end{aligned} \tag{7}
$$

The mapping $X \to Y_X$ yields a 1-1 correspondence between the flows in G of size $\bar{T}$ and the two-projection images X for which $S(X) = \bar{T}$. If we fix $S(X)$, as in Problem 3, the term $\alpha(2S(X) - |p_1|_1 - |p_2|_1)$ in (7) is constant. We conclude that the problem of finding a flow Y of size $\bar{T}$ in G for which $C(Y)$ is minimal is equivalent to finding a solution of the gray value variant of Problem 3.

The minimum cost flow problem in graphs is a well known problem in combinatorial optimization for which efficient, polynomial time algorithms exist (see [1] for a thorough description or [18] for an overview). However, most algorithms for this problem deal with *linear cost functions* for the edges, i.e., cost functions of the form $C_e(x) = cx$, where c is a constant. The cost function for a line edge can be formulated as a linear cost function by replacing the line edge with *two* parallel edges. For a line edge (s, n_{1i}) in the original graph G, the first new edge has a capacity of $p_{1,i}$ and a cost function of 0. The second parallel edge has a capacity of $(\sum_{j=1}^{n} a_{ij}) - p_{1,i}$ and a cost function $C(x) = 2\alpha x$; see Figure 6a. For any minimum cost flow, the second parallel edge will only carry a nonzero flow if the first edge is completely filled, because of the cost of using the second edge.

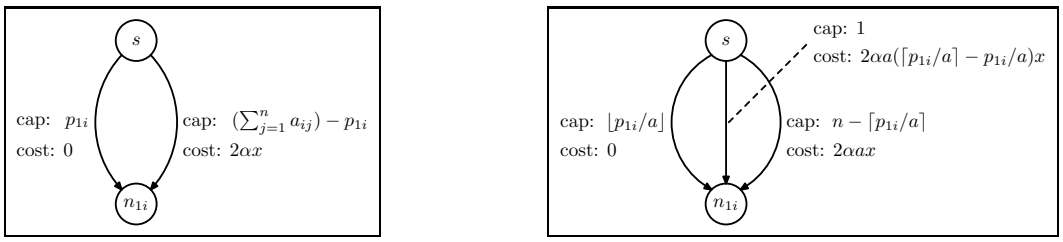


Figure 6: (a) (Left) To obtain linear cost functions for the edges, each line edge is replaced with two parallel edges that have linear cost functions. (b) (right) In the parallel beam case, the line edges are replaced by three parallel edges that each have integral capacity.

We first focus on the parallel beam case. For the parallel beam geometry, a solution of the *binary* version of Problem 3 can be computed efficiently (i.e., in polynomial time) using the mininimum cost flow formulation. It is a well known fact that if a network has only *integral* edge capacities and edge costs, there is always an integral flow among all flows for which the total cost is minimal (see, e.g., Theorem 6.5 of [1]). Such an integral flow can be computed in polynomial time. Note that for the parallel beam case, the pixel area is constant. We first divide all edge capacities in the associated graph by a, the pixel area. The total flow $\bar{T}$ and the coefficients of the linear cost functions for the edges have to be scaled accordingly. If $\bar{T}/a$ is not an integral value, it is rounded to the nearest integer; otherwise it would clearly be impossible to find an integral flow of size $\bar{T}/a$. We denote the graph that is obtained by this scaling step by G'. All pixel edges in G' have a capacity of 1, but the capacities of the line edges (one pair for each projected strip, as in Figure 6a) may still not be integral. If a pair of parallel edges that represents a line edge does not have integral capacities, we add a third parallel edge, as in Figure 6b. The figure also indicates the capacity and the cost function for each of the three parallel edges. Note that the derivatives of the cost functions are increasing

constants for the three parallel edges, from left to right. Therefore, in any minimum cost flow the second edge will only be used if the first one is completely filled and the third edge will only be used if the second one is full. It is easy to verify that for any *integral* amount x of flow from s to n_{1i} the total cost for the three parallel edges equals $2\alpha \max(x - p_{1,i}/a, 0)$ and all three edge capacities are integral. To obtain integral coefficients in all cost functions, note that the solution of the minimum cost flow problem in a graph does not change if we multiply all costs by the same constant (only the *total cost* of the solution changes). For each (linear) cost function $C_e = cx$, we multiply c by a constant K and round the result. For $x \in \mathbb{R}$, put $\text{round}(x) = \max(\{i \in \mathbb{Z} : i \leq x + \frac{1}{2}\})$. As $\text{round}(Kcx)/Kcx \to 1$ for large values of K, the effect of this rounding step can be made negligible by choosing K to be large. For the parallel beam case, Problem 3 can now be solved by computing an integral solution of the minimum cost flow problem in G'. As each pixel edge of G' has a capacity of 1, the resulting flow through each pixel edge must be either 0 or 1 (as the flow is integral), which directly provides a binary solution of the corresponding tomographic reconstruction problem.

We now move on to the fan beam case. Contrary to the parallel beam case, the minimum cost flow problem in the associated graph G for the fan beam geometry cannot be reduced to a flow problem in a graph G' where all pixel edges have a capacity of 1. As the cell area $A(C_{ij})$ is no longer constant in the fan beam case (see Figure 3b), the capacity of the pixel edges will no longer be constant. To obtain a network with integral edge capacities, we multiply all edge capacities, as well as the total flow, by a large constant L and round the resulting capacities to the nearest integer, obtaining a new network G''. In this new network, the minimum cost flow problem can be solved efficiently. Note, however, that the pixel edges in G'' will typically have integral capacities that are greater than 1. If the flow through a pixel edge in G'' is between (but not equal to) 0 and the edge capacity, the corresponding pixel value will be between (but not equal to) 0 and 1. Therefore, there is no guarantee that we obtain a binary solution in this way. Still, minimum cost flow algorithms can be used to compute a *gray value* image. As we will show in Section 5, this gray value image can still be very useful for computing a binary reconstruction, as most pixel values will typically be very close to 0 or 1.

If we set α to a very large value in Problem 3, the solution(s) of the reconstruction problem will correspond to the measured projections as well as possible, according to the sum-norm. For the experiments in Section 5 we used $\alpha = 10000$. In those experiments the pixel weights w_{ij} are always between -2 and 2.

4 More than two projections

In the previous section we showed that thetwo-projection reconstruction problem can be formulated as a network flow problem. For the case of parallel beam projections, a binary solution can be found efficiently. For the fan beam case the network flow approach can also be used, but it does not guarantee a binary solution. Unfortunately, there is no straightforward generalization of the network flow approach to the case of more than two projections. In this section we study the following reconstruction problem:

Problem 4 *Let* $p_1 = (p_{1,1} \dots p_{1,n})^T, \dots, p_d = (p_{d,1} \dots p_{d,n})^T \in \mathbb{R}^n$ *be vectors of nonnegative real numbers (the measured strip projections for projection angles* $\theta_1, \dots, \theta_d \in D$ *respectively). Construct a function* $f : I \to \{0, 1\}$ *such that*

$$\iint_{S_{\theta_k}(t_{i-1},t_i)} f(x, y)\, \mathrm{d}x\, \mathrm{d}y \quad = \quad p_{k,i} \qquad \text{for } i = 1, \dots, n,\ k = 1, \dots, d.$$

We propose an iterative algorithm, which makes use of the fact that the two-projection problem can be solved efficiently. The algorithm computes a reconstruction from more than two projections by solving a series of two-projection subproblems, each using two projection angles from the set D. The algorithm uses the concept of a weight map, as defined in Section 3. In each iteration a new pair of projection angles is selected. An instance of Problem 3 is then solved on the two-projection grid that corresponds to those two angles. The weight map is computed using the reconstruction from the previous iteration, in such a way that solutions are preferred which resemble the reconstruction from the previous iteration. The previous reconstruction was computed using a different pair of projections, which are thus incorporated into the new reconstruction. Repeating this argument, projections from earlier iterations are also incorporated.

To display a reconstruction that has been computed on a two-projection grid, it has to be converted to an image on the standard square pixel grid, aligned to the horizontal and vertical axes. Figure 7 shows the standard pixel grid, superimposed on a two-projection image. The value of each pixel in the standard pixel grid is computed by averaging the gray values of the overlapping pixels in the two-projection image, weighted by the overlap area. We will describe the details of this computation in Section 4.2.

Figure 8 shows the basic steps of the algorithm. First, a start solution is computed, using all projections simultaneously. The start solution should provide a good first approximation of the

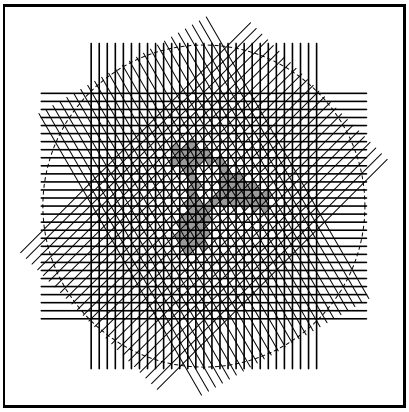


Figure 7: To display a reconstruction on a two-projection grid, it has to be converted to the standard pixel grid.

unknown image, while being easy to compute. The start solution can be computed on the standard $n \times n$ square pixel grid. For the experiments in Section 5 we used the SIRT (Simultaneous Iterative Reconstruction Technique, see Chapter 7 of [15]) to compute the start solution, which yields a gray value reconstruction.

Subsequently, the "total area of the white pixels in the unknown object" $\bar{T}$ is computed as $\bar{T} := (\sum_{k=1}^{d} |p_k|)/dn$. The value of $\bar{T}$ is used during the main loop of the algorithm, to determine the total flow in each of the network flow problems, corresponding to two-projection subproblems. Note that in the case of parallel beam tomography, $\bar{T}$ has to be rounded to the nearest integral multiple of the cell area, to be able to find *binary* solutions for the two-projection problems.

Next, the algorithm enters the main loop. In each iteration τ of the main loop a new pair $(\bar{\theta}_1^\tau, \bar{\theta}_2^\tau)$ of projection angles is first selected, which determines the two-projection grid for iteration τ. We refer to the cell (i, j) of this grid as C_{ij}^τ. Section 4.1 describes several possible angle selection schemes.

Subsequently the weight map $W^\tau = (w_{ij}^\tau)$ is computed from the previous reconstruction $X^{\tau-1}$. We describe the details of this computation in Section 4.3. The weight map W, the total flow $\bar{T}$ and the projections for angles $(\bar{\theta}_1^\tau, \bar{\theta}_2^\tau)$ define an instance of Problem 3. Solving this problem by the network flow approach yields the new reconstruction X^τ.

The criterion for termination of the main loop is described in Section 4.4, which also describes how the final reconstruction X^* is computed.

Compute the start solution X^0 on the standard pixel grid;
$\bar{T} := (\sum_{k=1}^{d} |p_k|)/dn$;
$\tau := 0$;
while (stop criterion is not met) **do**
begin
  $\tau := \tau + 1$;
  Select a new pair of projection angles $\bar{\theta}_1^\tau, \bar{\theta}_2^\tau \in D$;
  Compute the weight map $W^\tau = (w_{ij}^\tau)$ for the two-projection grid corresponding to $(\bar{\theta}_1^\tau, \bar{\theta}_2^\tau)$, using the previous reconstruction $X^{\tau-1}$;
  Compute a solution X^τ with $S(X^\tau) = \bar{T}$ of Problem 3 on the two-projection grid for angles $(\bar{\theta}_1^\tau, \bar{\theta}_2^\tau)$, using the weight map W^τ;
end
Return the final reconstruction X^* (see Section 4.4);

Figure 8: Basic steps of the algorithm for plane sets.

4.1 Choosing the pair of directions

Not every pair of projection angles is suitable for solving a two-projection problem. At least such a pair should satisfy the conditions in (1), but it is also important that both projection angles are sufficiently orthogonal. Figure 9 shows the grid cell shape for two projection angles that are almost orthogonal, versus the shape for two projection angles that have a small angular difference, resulting in a flat, stretched pixel shape. Such pixels are not well suited for accurate image representation, as the pixel diameter (i.e., the farthest distance between any two points in the pixel) is too large. Therefore the angular difference between $\bar{\theta}_1^\tau$ and $\bar{\theta}_2^\tau$ should always be larger than a certain minimum value, $\delta_{\min}$. We call a pair of projection angles *valid* if it satisfies all requirements to be used for solving a two-projection problem. We used $\delta_{\min} = \pi/4$ for all experiments (including the fan-beam experiments) in Section 5.

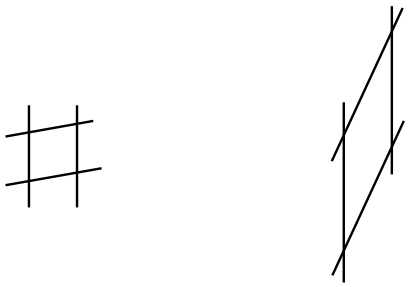


Figure 9: Pixel shape for two different two-projection grids in the parallel beam case. (a) Left: Two projection angles that are almost orthogonal. (b) Right: Small angular difference between the projection angles.

One can think of several criteria for choosing the projection pair at the start of a new iteration. Let X be a two-projection image for projection angles $\bar{\theta}_1, \bar{\theta}_2$. Define the *projection* $P_k(X)$ for projection angle $\theta_k \in D$ as follows:

$$P_k(X) = \begin{pmatrix} \sum_{1 \le i,j \le n} A(C_{ij} \cap S_{\theta_k}(t_0, t_1))x_{ij} \\ \cdots \\ \sum_{1 \le i,j \le n} A(C_{ij} \cap S_{\theta_k}(t_{n-1}, t_n))x_{ij} \end{pmatrix}$$

Recall that $A(V)$ denotes the area of a subset $V \in \mathbb{R}^2$. Note that for the two angles $\bar{\theta}_1$ and $\bar{\theta}_2$ this definition is equivalent to (2) and (3).

A good choice for the new pair of projection angles $\theta_i, \theta_j \in D$ in each iteration of the main loop is to choose the pair of valid directions for which the summed projection distance

$$|P_i(X) - p_i|_2 + |P_j(X) - p_j|_2$$

is largest. We used this criterion for the experiments in Section 5.

Under certain conditions it may be better to use a scheme that guarantees that all projections are used equally often. If the number of valid pairs of projection angles is small (e.g., $d = 4, 5$), it is likely that *cycling* occurs between two or three projection pairs when the total projection distance is used to select a new pair of projections. The other pairs will not be used, which may degrade the reconstruction quality. Also, when one projection contains significantly more noise than other projections, it will be selected over and over again, as it is more likely to have a higher projection error than the other projections. Possible selection schemes that avoid these problems are to select a random valid pair in each iteration, to select all pairs sequentially in a fixed order, or to select one random angle first and then choose the second angle such that the summed projection distance is largest.

4.2 Converting between different grids

Let $X = (x_{ij})$ be a two-projection image and let C_{ij} denote grid cell (i, j) in the corresponding two-projection grid. To display the image X, we need to convert it to an image $X' = (x'_{i'j'})$ on the square pixel grid, aligned with the horizontal and vertical axes. Denote the grid cells of this new grid by $C'_{i'j'}$ $(1 \le i', j' \le n)$. As X cannot be represented perfectly on the grid of X', we have to resort to an approximation. For each cell $C'_{i'j'}$ the intersection is computed with all cells of the

"old" grid and the gray-value is averaged, weighted by the intersection area:

$$x'_{i'j'} = \sum_{1 \le i,j \le n} \frac{A(\, C_{ij} \cap C'_{i'j'} \,)}{A(\, C'_{i'j'} \,)} x_{ij}.$$

The coefficients $A(\, C_{ij} \cap C'_{i'j'} \,)/A(\, C'_{i'j'} \,)$ can be precomputed for each pair of two-projection grids. For a fixed pixel (i', j') in the new grid, the number of pixels (i, j) in the old grid for which $A(\, C_{ij} \cap C'_{i'j'} \,)$ is nonzero is typically very small, because only valid projection pairs are used (see Section 4.1). By using a sparse representation that stores only the nonzero coefficients, the conversion of an image from one two-projection grid to another two-projection grid can be performed in linear time.

Another operation that is used by our algorithm, to obtain the weight map, is the computation of the *average gray value* in a small *neighbourhood* of a point $(x, y) \in \mathbb{R}^2$. Most common pixel neighbourhood definitions that are used in image processing, such as the 4-neighbourhood and 8-neighbourhood, are very suitable for square pixel grids. However, as our algorithm deals with many different pixel grids for which the pixel sizes and shapes may vary, we have to use a more general neighbourhood definition.

Define the *r-neighbourhood* $N(x, y, r)$ of (x, y), as

$$N(x, y, r) = \{(\tilde{x}, \tilde{y}) \in \mathbb{R}^2 : \sqrt{(x - \tilde{x})^2 + (y - \tilde{y})^2} \le r\}.$$

Let X be a gray value image and denote the corresponding grid cells by C_{ij}. The *r-neighbourhood gray value* $\Gamma_X(x, y, r)$ of a point (x, y) can now be computed as

$$\Gamma_X(x, y, r) = \sum_{1 \le i,j \le n} \frac{A(\, N(x, y, r) \cap C_{ij} \,)}{A(\, N(x, y, r) \,)} x_{ij}. \tag{8}$$

Although an intersection area of the form $A(\, N(x, y, r) \cap C_{ij} \,)$ can in principle be computed exactly (up to floating point errors), a sufficiently accurate approximation can be obtained by representing $N(x, y, r)$ as a regular polygon with $\tilde{n}$ vertices, where $\tilde{n}$ is a large integer, e.g., $\tilde{n} = 100$. The intersection area of two convex polygons of $\tilde{m}$ and $\tilde{n}$ vertices respectively can be computed in $O(\tilde{m} + \tilde{n})$ time; see Section 7.6 of [17].

4.3 Computing the weight map

The weight map W^τ is computed from the previous reconstruction $X^{\tau-1}$ and the measured projections for angles $(\bar{\theta}_1^\tau, \bar{\theta}_2^\tau)$. We denote the grid cells of the two-projection grid from the previous iteration by $C'(i'j')$ and the grid cells of the new two-projection grid by $C(i,j)$.

Define the *cell center* m_{ij} of cell C_{ij} as the intersection point of two fan half lines,

$$m_{ij} = (x_{m_{ij}}, y_{m_{ij}}) := L_{\bar{\theta}_1^\tau}((t_{i-1} + t_i)/2) \cap L_{\bar{\theta}_2^\tau}((t_{j-1} + t_j)/2).$$

The pixel weight w_{ij}^τ of pixel (i,j) depends directly on the r-neighbourhood gray value of the cell center m_{ij}, computed using the previous reconstruction $X^{\tau-1}$. In all experiments of Section 5 the parameter r is a constant. For each cell center m_{ij}, the r-neighbourhood gray value $\Gamma_{X^{\tau-1}}(x_{m_{ij}}, y_{m_{ij}}, r)$ of m_{ij} in the image $X^{\tau-1}$ is computed according to (8).

Let $g : [-1,1] \to \mathbb{R}_{>0}$ be an odd function (i.e., $g(-v) = -g(v)$), which is increasing on the interval $[0,1]$. We call g the *local weight function*. Together with the neighbourhood radius r, the local weight function determines the preference for locally smooth regions.

The pixel weight w_{ij}^τ is computed as

$$w_{ij}^\tau = g(2(\Gamma_{X^{\tau-1}}(x_{m_{ij}}, y_{m_{ij}}, r) - \frac{1}{2}))$$

Note that $0 \leq \Gamma_{X^{\tau-1}}(x_{m_{ij}}, y_{m_{ij}}, r) \leq 1$.

The basic motivation of the local weight function is that, as the neighbourhood of a pixel becomes more white, the preference to make this pixel white in the next iteration increases. The same holds for black neighbourhoods. If a pixel neighbourhood consists of 50% black and 50% white pixels, no preference is expressed as the pixel weight is zero. Which local weight function works best for a certain set of projection data depends on local properties of the (unknown) original image. If the original image contains large regions that are either completely black or completely white, it makes sense to use $g(v) = \text{sign}(v)v^2$: the pixel weight increases strongly near $v = 1$ (and decreases strongly near $v = -1$), which results in a strong preference for local neighbourhoods that are either completely white or completely black. On the other hand, if the image contains many fine details (i.e., local neighbourhoods containing both black and white pixels), it will be better to use $g(v) = \text{sign}(v)\sqrt{|v|}$, which results in a weaker preference for completely homogeneous

neighbourhoods. For the experiments in Section 5 we used the function

$$g(v) = \begin{cases} v & \text{if} \quad |v| \neq 1 \\ 2v & \text{if} \quad |v| = 1. \end{cases} \tag{9}$$

This function expresses a strong preference for pixels that are surrounded solely by pixels of the same value to retain their value in the next iteration. For pixel neighbourhoods that are not homogeneous, the pixel weight function is linear. The results in Section 5 show that this local weight function works well for a varied set of test images. Note that this particular local weight function has a strong discontinuity at $|v| = 1$, which raises the question of what happens in practice if the argument has rounding errors. In practical computations, we put $g(v) = 2v$ if $|v| \geq 1 - \epsilon$, where ϵ is chosen according to the machine precision. Cases where the local neighbourhood of a pixel is completely homogeneous and yet $|v|$ is not completely equal to 1 due to rounding errors, are avoided in this way.

The other parameter that determines the preference for smooth reconstructions is the neighbourhood radius r. If the neighbourhood radius is large, the local weight function expresses a preference for pixels to be assigned the same value as the majority of their neighbouring pixels in a large surrounding area, which corresponds to some form of "blurring". Fine details cannot be reconstructed in this way, but on the other hand the influence of noise is reduced. If the neighbourhood radius is small, the local weight function expresses a preference for pixels to retain their value from the previous reconstruction.

The interplay of the local weight function and the neighbourhood radius r is quite complex and difficult to analyze, due to the wide range of possible local weight functions. Therefore, the results in Section 5, which use only a single local weight function, are not necessarily the best possible results for our algorithm. A different choice for the local weight function and neighbourhood radius may lead to better results.

4.4 Termination and final output

Define the *total projection distance* $\Delta(X)$ as

$$\Delta(X) = \sum_{k=1}^{d} |P_k(X) - p_k|_2$$

The total projection distance provides an indication of the quality of the reconstruction X without

having knowledge of the unknown original image. Therefore it can be used to define a termination criterion for the main loop of the algorithm. The algorithm terminates if no improvement has been made in the total projection distance (i.e., by finding a new reconstruction that has a smaller projection distance than the best one so far) during T iterations, where T is a constant. We used $T = 30$ for all experiments in Section 5.

The result of one iteration of the algorithm is always a reconstruction that adheres well to the two projection angles that were used in that iteration. If the projection data contain noise, errors caused by the noise in those two projections will always be present in the last reconstructions. To reduce this effect, the final reconstruction that is returned by the reconstruction algorithm is computed as an *average* image over the last T' iterations, where T' is a constant. The reconstructions from the last T' iterations, each computed on a two-projection grid, are converted to the standard square pixel grid. On this grid, the pixel values are averaged and thresholded at 0.5 to obtain a binary reconstruction. We used $T' = 15$ for all experiments in Section 5.

4.5 Time complexity

The time complexity of our algorithm is largely determined by the complexity of the algorithm that is used to solve the min cost flow problems for each of the projection pairs.

We focus here on the parallel beam case. The associated graph for each pair of projections has $2n+2$ nodes and n^2+4n edges. The problem of finding a minimum cost flow of fixed size in a graph that has N nodes, M edges and integral edge costs that are bounded in absolute value by K, can be solved in $O(nm \log(n^2/m) \log(nK))$ time [10]. This yields a time complexity of $O(n^3 \log(nK))$ for solving the minimum cost flow problem in the associated graph. In all experiments in Section 5 we first computed the pixel weights using the local weight function, then multiplied the real-valued pixel-weights by $K = 1000$ and rounded the results to integral values.

Setting K anywhere in the interval $[100, 10000]$ did not result in any significant changes in both the reconstruction quality, the total number of iterations and the total reconstruction time.

5 Experimental results

In this section we present reconstruction results of our algorithm. First, we present reconstruction results for a set of four phantom images (i.e., synthetic images). For each of the phantoms we computed the measured projection data by simulation. In this way the noise level can be controlled, which is difficult for datasets obtained by real measurements. Subsequently, we provide reconstruc-

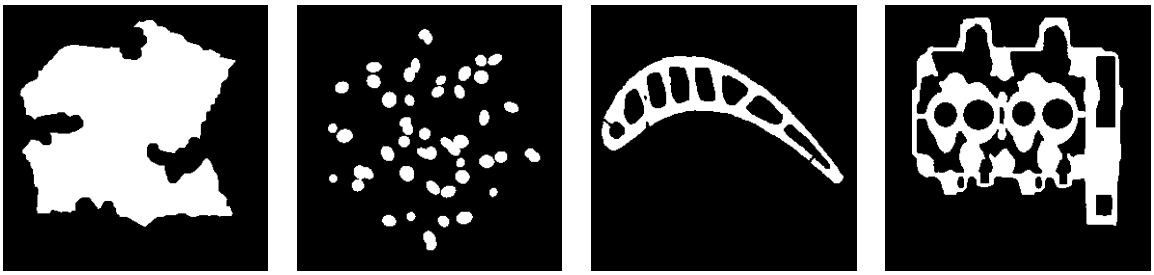
tion results for a real measured dataset, obtained by scanning a raw diamond using a micro-CT scanner.

We implemented the iterative network flow algorithm in C++, using the *RelaxIV* library [4] to solve the minimum cost flow problems. All experiments were performed on a 2.4GHz PentiumIV PC.

As noted in Section 4.2, a large part of the computations that are involved in converting an image from one grid to a different grid, or computing the neighbourhood gray value, can be precomputed. In the running times that are reported in this section we assume that these computations have already been carried out, as the precomputing step is independent from the measured projection data and has to be performed only once for each data collection geometry.

5.1 Parallel beam projections

The four phantom images that we use to evaluate the reconstruction time and the reconstruction quality of our algorithm are shown in Figure 10.



a. single object 300×300 | b. 50 ellipses 256×256 | c. turbine blade 276×276 | d. cylinder head 276×276

Figure 10: Four phantom images

The phantom in Figure 10a represents a single object, which is not convex. In contrast to the first phantom, the phantom in Figure 10b represents many small separate objects. The image contains 50 ellipses of various sizes. The phantom in Figure 10c represents a cross section of a turbine blade from a jet engine. Checking turbine blades for cracks and other defects is crucial, as defects in one of the blades may lead to malfunction of the engine. Note that the turbine blade phantom contains several small gaps and cracks, which we would like to see in the reconstruction. The fourth phantom, in Figure 10d represents a cross section of a cylinder head in a motorcycle engine. Cylinder heads, which are often made of aluminium, are occasionally scanned in industry for reverse engineering, using X-rays.

First, we compare our reconstruction results from perfect parallel beam projections to two al-

ternative approaches. It is well known that continuous tomography algorithms such as Filtered Backprojection require a large number of projections. Algebraic reconstruction methods, such as ART and SIRT (see Chapter 7 of [15]) typically perform much better than backprojection algorithms if the number of projections is very small. Our algorithm uses the SIRT algorithm to compute a start solution. As a first comparison, we compare the final reconstruction computed by our algorithm to the continuous SIRT reconstruction. In [20, 21], Weber et al. describe a linear programming approach to binary tomography which incorporates a smoothness prior. We implemented the *R-BIF* approach which is described in [21] using the ILOG CPLEX interior point solver [13]. The real-valued pixel values are rounded to binary values as a postprocessing step, which was also done in [20]. Besides the projection data, the linear program depends on a parameter α, which determines the preference for smoothness. For our set of four phantoms we found that $\alpha = 0.2$ works well.

For our network flow algorithm we use (9) as the local weight function. as the local weight function. The neighbourhood radius is set to 1.5 times the diameter of a pixel (in the phantom image).

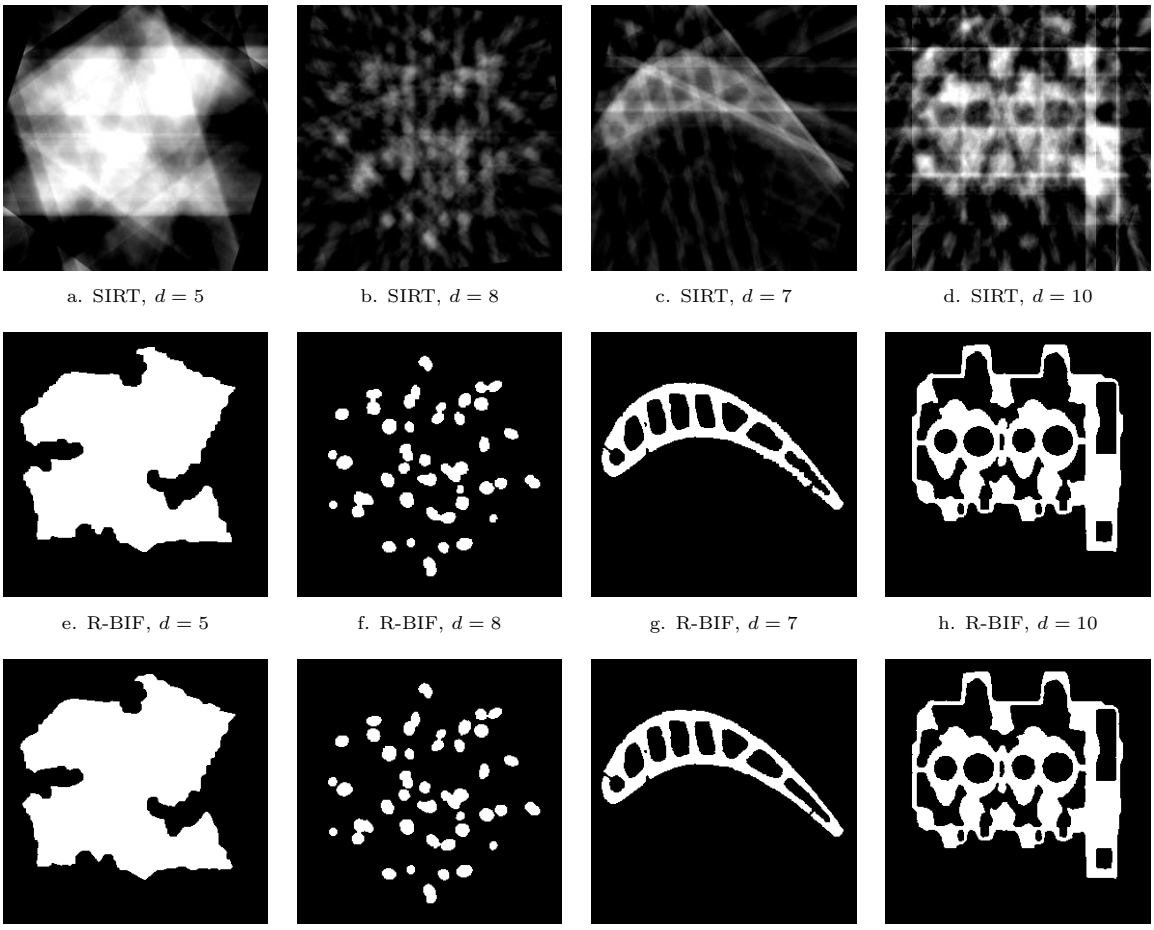
Figure 11 shows the reconstruction results for the four phantoms from parallel beam projection data. For each of the phantoms, the number of (equally spaced) strips in a projection equals the width, in pixels, of the phantom. Quantitative results for the four phantoms are shown in Table 1.

phantom	size	#proj.	R-BIF		Network Flow	
			#errors	time(min)	#errors	time(min)
single object	300×300	5	220	9.2	58	2.0
50 ellipses	256×256	8	216	3.5	152	1.5
turbine blade	276×276	7	214	3.2	108	1.8
cylinder head	276×276	10	375	10	665	1.9

Table 1: Quantitative comparison between the R-BIF reconstruction and the reconstruction computed by our network flow algorithm. The table shows the reconstruction time in minutes and the total number of pixel differences between the reconstruction and the original phantom.

The number d of projections is chosen for each phantom as the minimum number for which our algorithm computes an accurate reconstruction from the projection data. The projection angles are equally spaced between 0 and 180 degrees. Every projection consists of n strip projections, that each have a width equal to the phantom image pixel width.

Our algorithm is capable of computing an accurate reconstruction of each of the four phantom images from a small number of projections. The reconstruction quality of the R-BIF algorithm appears to be similar, although there are some small differences between the reconstructions by both methods. The reconstruction quality for both these algorithms is much better than for the



a. SIRT, $d = 5$ b. SIRT, $d = 8$ c. SIRT, $d = 7$ d. SIRT, $d = 10$

e. R-BIF, $d = 5$ f. R-BIF, $d = 8$ g. R-BIF, $d = 7$ h. R-BIF, $d = 10$

i. NF, $d = 5$ j. NF, $d = 8$ k. NF, $d = 7$ l. NF, $d = 10$

Figure 11: Reconstruction results for the four phantoms from parallel beam projections, using SIRT (top), R-BIF from [21] (middle) and our network flow algorithm (NF, bottom). The figure captions show the number d of projections that was used.

continuous SIRT algorithm, which is immediate by visual inspection of the resulting images. We observed that the number of projections that is required to compute an accurate reconstruction is the same for the R-BIF algorithm as for our algorithm, for each of the four phantoms. As there are major differences between both algorithms, this suggests that this minimal number of projections is an intrinsic property of the images themselves. It may be very difficult or even impossible to compute accurate reconstructions from fewer projections by using a different algorithm.

The reconstructions that result from using too few projections may be quite different between both algorithms. Figure 12 shows a reconstruction of the turbine blade phantom from 6 projections for both algorithms.

The minimal number of projections that is required to reconstruct a given (unknown) image depends strongly on characteristics of the unknown image, as can be seen by the varying number of

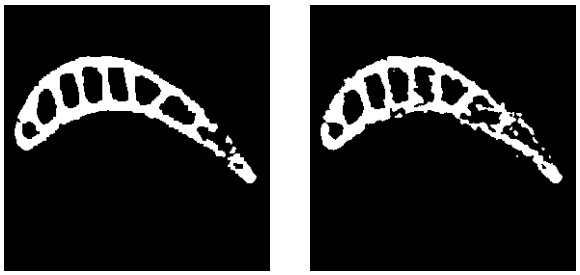


Figure 12: (a) (Left) R-FIB reconstruction from 6 projections. (b) (Right) Network flow reconstruction from 6 projections.

projections that is required for the four phantom images.

If more projections are available than the minimum number that is required, using a few additional projections reduces the number of iterations of our algorithm, resulting in a shorter running time.

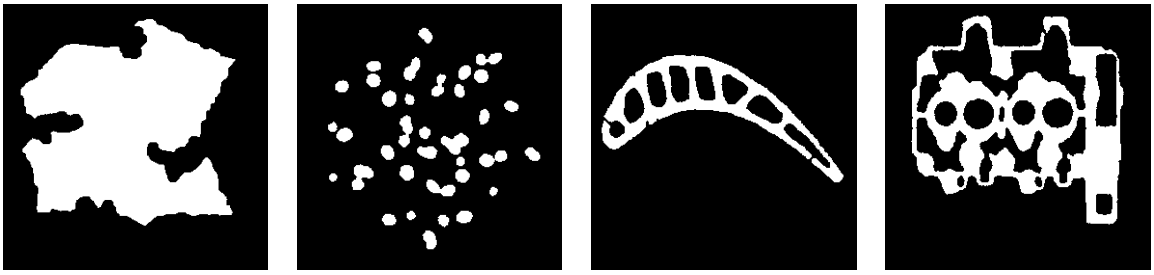
Our algorithm has several advantages over the R-BIF linear programming approach. First, the network flow algorithm is faster than general linear programming as used in [21], even though we used a high performance interior point solver for solving the linear program. It was demonstrated in [2] for the case of lattice images that the network flow approach for 2D reconstruction can be extended to a highly efficient algorithm for 3D reconstruction. Within each iteration a series of 2D reconstruction problems is solved, instead of one big 3D problem. The 2D reconstructions can be computed fast (because each subproblem is relatively small) and in parallel. A similar extension is possible for our new algorithm. Dealing with large volumes seems to be very difficult when using general linear programming, as the number of variables becomes huge for large 3D volumes.

Another advantage of our algorithm is that it can deal with noise effectively, as we will show in the next subsections. The linear programming approach from [20, 21] is not capable of handling noisy data. It can be extended to the case of noisy data, as described in [22]. However, the presented algorithm for dealing with noisy data solves a *series* of linear programs, which results in far longer reconstruction times.

More recently, Schüle et al. developed a different reconstruction algorithm based on *D.C. programming* (Difference of Convex functions) [19]. The results of this algorithm seem to be very promising and it does not have some of the drawbacks of the linear programming approach. We intend to perform a comparison between a wider set of reconstruction algorithms in the future.

5.2 Fan beam projections

The main focus of our reconstruction results is on parallel beam projection data. For parallel beam data the network flow method guarantees a *binary* solution of each two-projection reconstruction problem. For fan beam data, the solution of the two-projection problems is not necessarily binary. Figure 13 shows reconstruction results from fan beam projections. For all four phantoms, the center of rotation of the beam source is the center of the image and the distance from the source to the center (the detector radius R) equals the width (and height) of the image. The fan parameters $t_0, \ldots, t_n$ are equally spaced.



a. $d = 5$, 2.8min. b. $d = 8$, 3.3min. c. $d = 7$, 3.2min. d. $d = 10$, 8.1min.

Figure 13: Reconstruction results of the network flow algorithm from fan beam projections. For each phantom, the distance of the X-ray source to the center of the image equals the width (and height) of the image. The figure captions show the number d of projections and the reconstruction time in minutes.

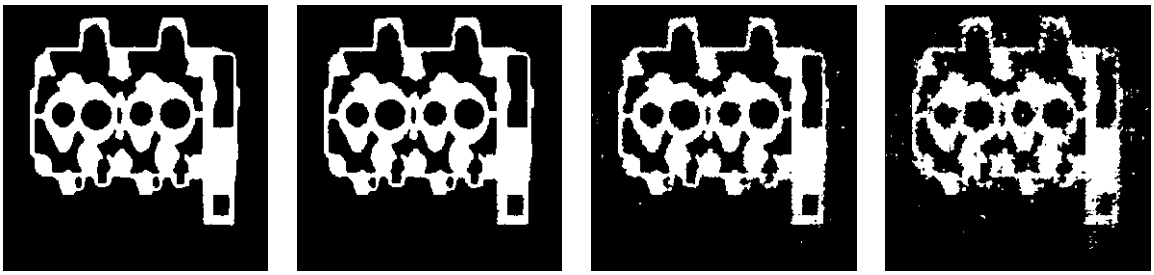
The results show that even though the solutions of the two-projection subproblems are not necessarily binary, the algorithm is still capable of computing a binary reconstruction. The reconstructions are somewhat less accurate than for the parallel beam case. In particular, there is a small gap visible in the left part of the turbine blade phantom, which is not visible in the reconstruction. The cylinder head reconstruction contains a small crack at the top, which is not present in the original phantom. Adding two more projections for these phantoms results in reconstructions without these errors. As the pixel sizes and shapes vary significantly between pixel grids in different iterations of the algorithm, we expect the loss of accuracy caused by conversions between different grids to be larger for fan beam tomography than for parallel beam tomography.

5.3 Noisy projection data

So far all experiments were carried out with perfect projection data. We now focus on the reconstruction of images from noisy projection data, which is practically more realistic. We also assume that the noise is independent for each projected strip. To be more precise, we assume that the noise is additive and that it follows a Gaussian distribution with $\mu = 0$ and σ constant over all projected

strips. The standard deviation σ is expressed as $\sigma = vy$, where y denotes the average measured strip projection over all projected strips. For example, taking $v = 0.1$ results in independently distributed additive noise for each strip projection, following the $\mathcal{N}(0, 0.1y)$ distribution.

Figure 14 shows reconstruction results for the cylinder head phantom, using 10 parallel beam projections and varying noise levels. The reconstruction quality decreases gradually as the noise level is increased. Up to $v = 0.02$, the noise hardly has any visible influence on the reconstruction quality.



a. $v = 0.01$ b. $v = 0.02$ c. $v = 0.05$ d. $v = 0.10$

Figure 14: Reconstruction results of the network flow algorithm from 10 parallel beam projections for increasing noise levels.

It is clear from Figure 14d that for high noise levels the reconstruction is not very smooth at all. The smoothness of the reconstruction can be increased by increasing the size r of the local neighbourhood used in the computation of the weight map. However, increasing the neighbourhood size reduces the ability to reconstruct fine details. If such details are not present, such as in the "single object" phantom, or if they are not important, increasing the neighbourhood size results in better reconstructions for high noise levels. Figure 15 compares reconstructions of the single object phantom for neighbourhood sizes of 1.5 and 6 times the diameter of a pixel in the phantom image. Increasing the neighbourhood size results in far better reconstructions. The ability to compute reconstructions under noisy conditions is very important in practical applications.

a. $v = 0.02, r = 1.5$ b. $v = 0.02, r = 6$ c. $v = 0.05, r = 1.5$ d. $v = 0.05, r = 6$

Figure 15: Reconstruction results of the network flow algorithm for the single object phantom with two noise levels ($v = 0.02$ and $v = 0.05$) and two local neighbourhood sizes ($r = 1.5$ and $r = 6$).

5.4 Real-world data

Besides the experiments with simulated projection data, we also performed a reconstruction experiment with real X-ray CT data. Figure 16a shows a *sinogram* obtained by X-ray tomography of a slice of raw diamond. The sinogram shows the measured parallel beam projections of the slice for 500 consecutive projection angles, equally spaced between 0 and 180 degrees. Each horizontal line of the sinogram corresponds to a measured projection. If the diamond is not polluted by other materials, it can be considered as a homogeneous object, which leads to a binary reconstruction problem.

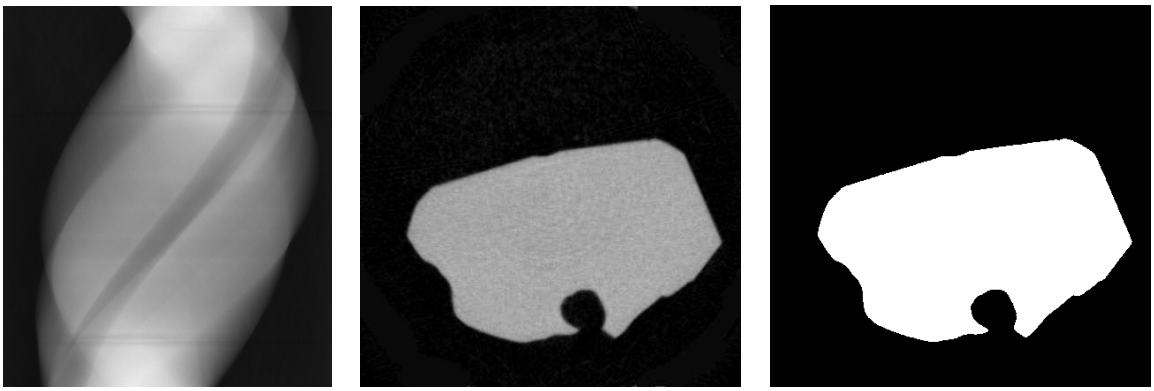


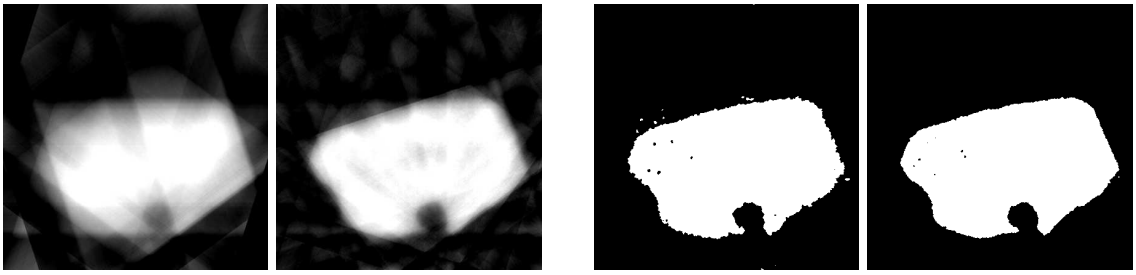
Figure 16: (a) (Left) Parallel beam sinogram of a raw diamond slice. (b) (Middle) SIRT reconstruction from 125 projections. (c) (Right) Thresholded SIRT reconstruction: a binary image.

Figure 16b shows a gray value reconstruction computed by the SIRT algorithm from the sinogram in Figure 16a, using 125 of the available 500 projections. The continuous reconstruction was thresholded to obtain a binary reconstruction, which is shown in Figure 16c. The sinogram in Figure 16a contains several artifacts, caused by some problems with the scanning device at the time of the experiment. In particular there are several very faint projections (dark horizontal lines) visible. When using continuous tomography, the influence of such errors becomes smaller as the number of projections is increased. A second source of errors is that the sinogram is an 8-bit gray scale image, which limits the accuracy of the measured projection values. The errors make the task of reconstructing the diamond slice from few projections more challenging.

Figure 17 shows reconstruction results from 5 and 10 projections for the SIRT algorithm and our network flow algorithm. Although the basic shape of the diamond slice can be distinguished in the SIRT reconstruction from 10 projections, this reconstruction is of very poor quality. The reconstruction computed by our algorithm from 5 projections approximates the real shape quite well. Due to noise and other errors in the projection data there are still some errors in the reconstruction.

The reconstruction from 10 projections is much more accurate. In particular, the boundary of the diamond slice is reconstructed well.

This example shows that our algorithm is capable of handling real-world projection data, even if the data contains significant noise or other errors. The number of projections that is required to compute an accurate reconstruction is far lower than for continuous tomography, which is currently used in most applications.



a. SIRT, $d = 5$ b. SIRT, $d = 10$ a. NF, $d = 5$ b. NF, $d = 10$

Figure 17: Reconstruction results for the SIRT algorithm and our network flow algorithm (NF) from 5 and 10 equally spaced projections.

6 Conclusions

We have described a novel algorithm for the reconstruction of binary images from a small number of their projections. Our algorithm is iterative. In each iteration a reconstruction problem is solved that depends on two of the projections and the reconstruction from the previous iteration. We showed that the two-projection reconstruction problem is equivalent to the problem of finding a flow of minimal cost in the associated graph. This equivalence allows us to use network flow algorithms for solving the two-projection subproblems. The network flow approach is most effective for parallel beam projections, but it can also deal with fan beam projection data.

The reconstruction results show that the reconstruction quality of our algorithm is far better than for the SIRT algorithm from continuous tomography. A comparison with the R-BIF linear programming approach from [21], which uses a smoothness prior, shows that both algorithms yield comparable reconstruction quality. Our algorithm runs faster than the linear programming approach and can be easily extended to 3D reconstruction or reconstruction from noisy projections. These extensions are difficult to accomplish efficiently for the linear programming approach.

Our results for noisy projection data shows that the algorithm is capable of dealing with significant noise levels. The reconstruction quality decreases gradually as the noise level is increased. A smoother reconstruction can be obtained by increasing the local neighbourhood size that is used to

compute the weight map. For images that do not contain fine details it is likely that increasing the smoothness results in better reconstructions.

By providing reconstruction results of a raw diamond slice from real X-ray scanner data, we showed that our algorithm is capable of computing high quality reconstructions from real-world data.

In future research we intend to perform more extensive comparisons between different algorithms for reconstructing binary images from few projections. Such a comparison is often difficult, as each algorithm makes different assumptions on the class of images, the detector setting, etc. Generalization of our algorithm to 3D reconstruction is straightforward, following the same approach as in [2]. Even for 3D objects that could be reconstructed as a series of 2D slices, an algorithm that takes the 3D connectivity of the object into account (using a 3D local neighbourhood) could possibly improve the reconstruction quality.

References

[1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: *Network flows: theory, algorithms and applications*. Prentice-Hall (1993).

[2] Batenburg, K.J.: A new algorithm for 3D binary tomography. *Electron. Notes Discrete Math.* **20**, 247–261 (2005).

[3] Batenburg, K.J.: A network flow algorithm for reconstructing binary images from discrete X-rays. *J. Math. Im. Vision*, to appear (already available online) (2006).

[4] Bertsekas, D.P., Tseng, P.: RELAX-IV: a faster version of the RELAX code for solving minimum cost flow problems. *LIDS Technical Report LIDS-P-2276*, MIT (1994).

[5] Censor, Y.: Binary steering in discrete tomography reconstruction with sequential and simultaneous iterative algorithms. *Linear Algebra Appl.*, **339**, 111–124 (2001).

[6] Fishburn, P., Schwander, P., Shepp, L., Vanderbei, R.: The discrete Radon transform and its approximate inversion via linear programming. *Discrete Appl. Math.* **75**, 39–61 (1997).

[7] Gale, D.: A theorem on flows in networks. *Pacific J. Math.*, **7**, 1073–1082 (1957).

[8] Gardner, R.J.: *Geometric Tomography*. Cambridge University Press, New York, 2nd edition (2006).

[9] Gardner, R.J., Gritzmann, P., Prangenberg, D.: On the computational complexity of reconstructing lattice sets from their X-rays. *Discrete Math.*, **202**, 45–71 (1999).

[10] Goldberg, A.V., Tarjan, R.E.: Finding minimum-cost circulations by successive approximations. *Math. Oper. Res.*, **15**, 430–466 (1990).

[11] Herman, G.T., Kuba, A., eds.: *Discrete tomography: foundations, algorithms and applications.* Birkhäuser, Basel (1999).

[12] Herman, G.T., Kuba, A., eds.: *Advances in discrete tomography and its applications.* Birkhäuser, Basel, to appear (2007).

[13] *ILOG CPLEX*, `http://www.ilog.com/products/cplex/`

[14] Jinschek, J.R., Calderon, H.A., Batenburg, K.J., Radmilovic, V., Kisielowski, C.: Discrete tomography of Ga and InGa particles from HREM image simulation and exit wave reconstruction. *MRS Proc.*, **839**, 4.5.1–4.5.6 (2004).

[15] Kak, A.C., Slaney, M.: *Principles of computerized tomographic imaging.* SIAM (2001).

[16] Natterer, F.: *The mathematics of computerized tomography*, SIAM (2001).

[17] O'Rourke, J.: *Computational geometry in C*, Cambridge University Press (2001).

[18] Schrijver, A.: *Combinatorial optimization: polyhedra and efficiency.* Algorithms and Combinatorics series, **24**, Springer, Heidelberg (2003).

[19] Schüle, T., Schnörr, C., Weber, S., Hornegger, J.: Discrete Tomography by Convex-Concave Regularization and D.C. Programming. *Discrete Appl. Math.*, **151**, 229–243 (2005).

[20] Weber, S., Schnörr, C., Hornegger, J.: A linear programming relaxation for binary tomography with smoothness priors. *Electron. Notes Discrete Math.*, **12** (2003).

[21] Weber, S., Schüle, T., Schnörr, C., Hornegger, J.: A Linear Programming Approach to Limited Angle 3D Reconstruction from DSA Projections. *Methods Inf. Medicine*, **4**, Schattauer Verlag, 320-326 (2004).

[22] Weber, S., Schüle, T., Hornegger, J., Schnörr, C.: Binary Tomography by Iterating Linear Programs from Noisy Projections. *Proc. of IWCIA 2004*, Lecture Notes in Computer Science **3322**, 38–51 (2004).