9

Network Flow Algorithms for Discrete Tomography

K.J. Batenburg

Summary. There exists an elegant correspondence between the problem of reconstructing a 0–1 lattice image from two of its projections and the problem of finding a maximum flow in a certain graph. In this chapter we describe how network flow algorithms can be used to solve a variety of problems from discrete tomography. First, we describe the network flow approach for two projections and several of its generalizations. Subsequently, we present an algorithm for reconstructing 0–1 images from more than two projections. The approach is extended to the reconstruction of 3D images and images that do not have an intrinsic lattice structure.

9.1 Introduction

The problem of reconstructing a 0–1 image from a small number of its projections has been studied extensively by many authors. Most results deal with images that are defined on a lattice, usually a subset of $\mathbb{Z}^2$. Already in 1957, Ryser studied the problem of reconstructing an $m \times n$ 0–1-matrix from its row and column sums [19, 20]. He also provided an algorithm for finding a reconstruction if it exists. Ryser's algorithm is extremely efficient. In fact, it can be implemented in linear time, $O(m + n)$, by using a compact representation for the output image [5].

The problem of reconstructing a 0–1 matrix from its row and column sums can also be modeled elegantly as a *network flow problem*. In 1957, Gale was the first to describe the two-projection reconstruction problem in the context of flows in networks, providing a completely different view from Ryser's approach [7]. In the latter work, there was no reference to the algorithmic techniques for solving network flow problems. In 1956, Ford and Fulkerson published their seminal paper on an algorithm for computing a maximum flow in a network [6], which can be used to solve the two-projection reconstruction problem. Using the network flow model, Anstee derived several mathematical properties of the reconstruction problem [2].

The reconstruction problem from two projections is usually severely underdetermined. The number of solutions can be exponential in the size of the

image. In practice, the goal of tomography is usually to obtain a reconstruction of an *unknown original image*, not just to find any solution that has the given projections. If only two projections are available, additional prior knowledge must be used. Certain types of prior knowledge can be incorporated efficiently into the network flow approach, by using the concept of *min cost flows*.

A drawback of the network flow approach is that it cannot be generalized to the case of more than two projections. The reconstruction problem is NP-hard for any set of more than two projections [8]. Recently, an iterative approach for reconstructing 0–1 images from more than two projections was proposed by Batenburg [3]. In each iteration a reconstruction is computed from only two projections, using the network flow approach. The reconstruction from the previous iteration, which was computed using a different pair of projections, is used as prior knowledge such that the new reconstruction resembles the previous one.

In this chapter the network flow approach will be described, starting with the basic two-projection case. Section 9.2 describes the basic network flow formulation. In Section 9.3, the model is extended to incorporate prior knowledge in the reconstruction procedure. Section 9.4 deals with how the network flow approach can be made tolerant to noise and other errors. The implementation of network flow algorithms for discrete tomography is discussed in Section 9.5. Several highly efficient implementations of network flow algorithms are available. This section also addresses the time complexity of the relevant network flow algorithms. The basic iterative algorithm for reconstructing from more than two projections is described in Section 9.6. This algorithm can be generalized to 3D reconstruction very efficiently, which is discussed in Section 9.7. So far, all sections deal with *lattice images*. In Section 9.8, we discuss how the algorithms from the previous sections can be adapted to the problem of reconstructing binary images that do not have a lattice structure.

9.2 Network Flow Formulation for Two Projections

The reconstruction problems of this paper can be posed in several different forms. We mainly consider the reconstruction of a *subset* F of $\mathbb{Z}^2$ from its projections, but one can also formulate this problem in the context of reconstructing *binary matrices* or *black-and-white images*. In the case of binary matrices, the set F is represented by the set of matrix entries that have a value of 1. If we want to display a set $F \subseteq \mathbb{Z}^2$ and F is contained in a large rectangle $A \subseteq \mathbb{Z}^2$ (e.g., 256^2 elements), it is convenient to represent F as a *black-and-white* image. The white pixels correspond to the elements of F; the black pixels correspond to the remaining elements of A. *Continuous tomography algorithms*, such as the algebraic reconstruction technique (see, e.g., Chapter 7 of [17]), represent the reconstruction as a *gray-level image*. At several points in this chapter, we discuss how to utilize algorithms for continuous tomography for solving the discrete reconstruction problems. In these cases

we use the black-and-white image representation of F, as this representation can easily be connected with the gray-level images from continuous tomography. Depending on the representation of the set F, points in A may also be called *entries* (in the context of binary matrices) or *pixels* (in the context of black-and-white images).

In this section we consider the problem of reconstructing a subset F of the lattice $\mathbb{Z}^2$ from its projections in *two* lattice directions, $v^{(1)}$ and $v^{(2)}$. This is a generalization of the problem of reconstructing a binary matrix from its row and column sums.

We assume that a *finite* set $A \subseteq \mathbb{Z}^2$ is given such that $F \subseteq A$. We call the set A the *reconstruction lattice*. As an illustration of the concept of the reconstruction lattice, consider the representation of F as a black-and-white image. The set A defines the *boundaries* of the image: All white pixels are known to be within these boundaries.

We denote the cardinality of any finite set F by $|F|$. Define $\mathbb{N}_0 = \{x \in \mathbb{Z} \mid x \geq 0\}$. Let $v^{(1)}, v^{(2)} \in \mathbb{Z}^2$. A *lattice line* is a line in $\mathbb{Z}^2$ parallel to either $v^{(1)}$ or $v^{(2)}$ that passes through at least one point in $\mathbb{Z}^2$. Any lattice line parallel to $v^{(k)}$ $(k = 1, 2)$ is a *set* of the form $\{nv^{(k)} + t \mid n \in \mathbb{Z}\}$ for $t \in \mathbb{Z}^2$. The sets $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$ denote the sets of lattice lines for directions $v^{(1)}$ and $v^{(2)}$ respectively. For $k = 1, 2$, put $L^{(k)} = \{\ell \in \mathcal{L}^{(k)} \mid \ell \cap A \neq \emptyset\}$. Note that $L^{(1)}$ and $L^{(2)}$ are finite sets. We denote the elements of $L^{(k)}$ by $\ell_{k,i}$ for $i = 1, \ldots, |L^{(k)}|$. As an example, Fig. 9.1 shows the reconstruction lattice for $A = \{1, 2, 3\} \times \{1, 2, 3\}$, $v^{(1)} = (1, 0)$, and $v^{(2)} = (1, 1)$. For this example, the sets $L^{(1)}$ and $L^{(2)}$ contain three and five lattice lines, respectively.

For any lattice set $F \subseteq \mathbb{Z}^2$, its projection $P_F^{(k)} : L^{(k)} \to \mathbb{N}_0$ in direction $v^{(k)}$ is defined as

$$P_F^{(k)}(\ell) = |F \cap \ell| = \sum_{x \in \ell} f(x) \,, \tag{9.1}$$

where f denotes the characteristic function of F. The reconstruction problem can now be formulated as follows:

Problem 1. Let $v^{(1)}$, $v^{(2)}$ be given distinct lattice directions, and let $A \subseteq \mathbb{Z}^2$ be a given lattice set. Let $p^{(1)} : L^{(1)} \to \mathbb{N}_0$ and $p^{(2)} : L^{(2)} \to \mathbb{N}_0$ be given functions. Construct a set $F \subseteq A$ such that $P_F^{(1)} = p^{(1)}$ and $P_F^{(2)} = p^{(2)}$.

Define $S^{(k)} = \sum_{\ell \in L^{(k)}} p^{(k)}(\ell)$. We call $S^{(k)}$ the *projection sum for direction* $v^{(k)}$. Note that if F is a solution of Problem 1, we have $S^{(k)} = |F|$ for $k = 1, 2$. In Section 9.4, a generalization of Problem 1 will be described for which the prescribed projections $p^{(1)}$ and $p^{(2)}$ may contain errors. In that case the projection sum for direction $v^{(1)}$ may be different from the projection sum for direction $v^{(2)}$.

With the triple $(A, v^{(1)}, v^{(2)})$, 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* of $(A, v^{(1)}, v^{(2)})$.

The set V contains a node s (the *source*), a node t (the *sink*), one node for each $\ell \in L^{(1)}$, and one node for each $\ell \in L^{(2)}$. The node that corresponds to $\ell_{k,i}$ has label $n_{k,i}$. We call the nodes $n_{k,i}$ *line nodes*.

Nodes $n_{1,i}$ and $n_{2,j}$ are connected by a (directed) edge $(n_{1,i}, n_{2,j})$ if, and only if, $\ell_{1,i}$ and $\ell_{2,j}$ intersect in A. We call these edges *point edges* and denote the set of all point edges by $E_p \subseteq E$. There is a bijective mapping $\Phi : E_p \to A$ that maps $(n_{1,i}, n_{2,j}) \in E_p$ to the intersection point of $\ell_{1,i}$ and $\ell_{2,j}$. We call Φ the *edge-to-point mapping* of G. For $e \in E_p$, we call $\Phi(e)$ the *corresponding point of* e and for $x \in A$, we call $\Phi^{-1}(x)$ the *corresponding edge of* x.

Besides the point edges, the set E contains the subsets $E_1 = \{(s, n_{1,i}) \mid i = 1, \ldots, |L^{(1)}|\}$ and $E_2 = \{(n_{2,j}, t) \mid j = 1, \ldots, |L^{(2)}|\}$ 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 $E = E_p \cup E_1 \cup E_2$. Figure 9.2 shows the associated graph for the triple $(A, v^{(1)}, v^{(2)})$ of Fig. 9.1.

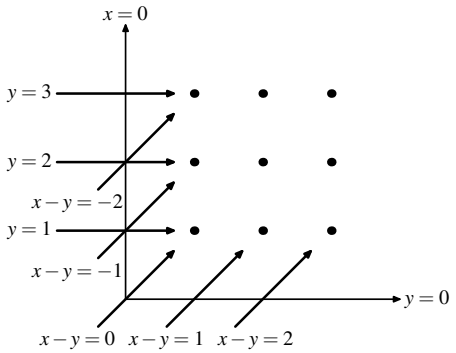


Fig. 9.1. Example lattice: $A = \{1, 2, 3\} \times \{1, 2, 3\}$, $v^{(1)} = (1, 0)$, $v^{(2)} = (1, 1)$.

Note that the structure of the associated graph is independent of the projections $p^{(1)}$ and $p^{(2)}$. To use the associated graph G for solving a particular instance of the reconstruction problem, we assign *capacities* to the edges of G. A *capacity function for* G is a mapping $E \to \mathbb{N}_0$. We use the following capacity function U:

$$\text{for } i = 1, \ldots, |L^{(1)}|;\ j = 1, \ldots, |L^{(2)}|; \tag{9.2}$$

$$U((n_{1,i}, n_{2,j})) = 1\,; \tag{9.3}$$

$$U((s, n_{1,i})) = p^{(1)}(\ell_{1,i})\,; \tag{9.4}$$

$$U((n_{2,j}, t)) = p^{(2)}(\ell_{2,j})\,. \tag{9.5}$$

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

$$\sum_{w:\,(w,v)\in E} Y((w, v)) = \sum_{w:\,(v,w)\in E} Y((v, w))\,. \tag{9.6}$$

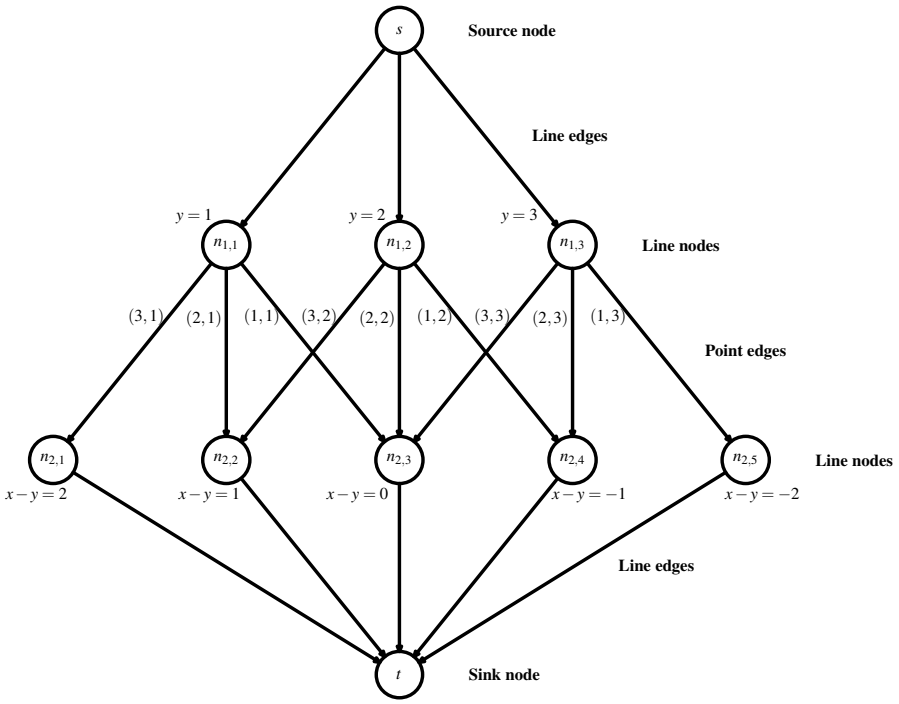


Fig. 9.2. Associated graph G for the triple $(A, v^{(1)}, v^{(2)})$ from Fig. 9.1.

The latter constraint is called the *flow conservation constraint*. Flows in graphs are also known as *network flows* in the literature. Let $\mathcal{Y}$ be the set of all flows in G. For a given flow $Y \in \mathcal{Y}$, the *size* $T(Y)$ of Y is given by $T(Y) = \sum_{(s,v)\in E} Y((s,v))$. If we consider G as a network of pipelines, carrying flow from s to t, the size of a flow is the net amount of flow that passes through the network. Due to the flow conservation constraint, we also have $T(Y) = \sum_{(v,t)\in E} Y((v,t))$. The associated graph G has a layered structure: All flow that leaves the source s must pass through the point edges. This yields the equality $T(Y) = \sum_{e\in E_p} Y(e)$. If $Y(e) \in \mathbb{N}_0$ for all $e \in E$, we call Y an *integral flow*. Note that for any integral flow Y in the associated graph G, we have $Y(e) \in \{0,1\}$ for all $e \in E_p$, as the capacity of all point edges is 1.

There is an elegant correspondence between the solutions of the reconstruction problem and the integral flows of maximal size (*max flows*) in the associated graph G:

Theorem 1. *Suppose that* $S^{(1)} = S^{(2)} =: \bar{T}$. *Problem 1 has a solution if, and only if, there exists an integral flow in* G *of size* $\bar{T}$. *Moreover, there is a 1-1*

correspondence between the solutions of Problem 1 and the integral flows of size $\bar{T}$ in G.

Proof. We show first that any integral flow in G of size $\bar{T}$ corresponds to a unique solution of Problem 1. Let Y be a flow in G of size $\bar{T}$. For each $e \in E_p$, we have $Y(e) \in \{0, 1\}$. Put $F_Y = \{\Phi(e) \mid e \in E_p \text{ and } Y(e) = 1\}$, where Φ is the edge-to-point mapping of G. The set F_Y contains all lattice points for which the corresponding point edge in G carries a flow of 1. We call F_Y the *corresponding point set of* Y. We claim that F_Y is a solution of Problem 1. We show that $P^{(1)}_{F_Y} = p^{(1)}$; the proof for direction $v^{(2)}$ is completely analogous.

From the capacity constraints on the line edges of G and the fact that $T(Y) = S^{(1)}$, it follows that all line edges of G must be filled completely by Y. Therefore, we have $Y((s, n_{1,i})) = p^{(1)}(\ell_{1,i})$ for all $i = 1, \ldots, |L^{(1)}|$. Because of the flow conservation constraint at the line nodes of G, we have

$$\sum_{j=1}^{|L^{(2)}|} Y((n_{1,i}, n_{2,j})) = p^{(1)}(\ell_{1,i}) \quad \text{for } i = 1, \ldots, |L^{(1)}| \tag{9.7}$$

and, therefore,

$$|\{\Phi((n_{1,i}, n_{2,j})) \mid Y((n_{1,i}, n_{2,j})) = 1\}| = p^{(1)}(\ell_{1,i}) \quad \text{for } i = 1, \ldots, |L^{(1)}| \,. \tag{9.8}$$

From the structure of G, it follows that

$$F_Y \cap \ell_{1,i} = \{\Phi((n_{1,i}, n_{2,j})) \mid Y((n_{1,i}, n_{2,j})) = 1\} \,, \tag{9.9}$$

which yields $P^{(1)}_{F_Y}(\ell_{1,i}) = p^{(1)}(\ell_{1,i})$ for $i = 1, \ldots, |L^{(1)}|$. To prove that every flow Y of size $\bar{T}$ in G corresponds to a *unique solution* of Problem 1, we note that Y is completely determined by its values on the point edges of G. Therefore, a flow $Y' \neq Y$ of size $\bar{T}$ must be different from Y at at least one of the point edges; hence, $F_{Y'} \neq F_Y$.

We will now show that the mapping from flows of size $\bar{T}$ in G to solutions of Problem 1 is surjective. For any solution F of Problem 1, define the *corresponding flow* Y_F:

$$Y_F((n_{1,i}, n_{2,j})) = \begin{cases} 1 & \text{if } \Phi((n_{1,i}, n_{2,j})) \in F \,, \\ 0 & \text{otherwise} \,. \end{cases} \tag{9.10}$$

Specifying Y_F on the point edges completely determines the flow through the remaining edges by the conservation of flow constraint. We call Y_F the *corresponding flow of* F. It is easy to verify that Y_F satisfies all edge capacity constraints. By definition, F is the corresponding point set of Y_F. We have $T(Y_F) = \sum_{(v,w) \in E_p} Y((v, w)) = |F|$, so Y_F is a flow of size $|F| = S^{(1)} = \bar{T}$. This shows that the mapping $Y \to F_Y$ is a bijection. □

The proof of Theorem 1 shows that we can find a solution of Problem 1 by finding an integral flow of size $\bar{T} = S^{(1)} = S^{(2)}$ in the associated graph. This flow is a *maximum flow* in G, because all line edges are completely saturated. Finding a maximum integral flow in a graph is an important problem in operations research, and efficient algorithms have been developed to solve this problem; see Section 9.5.

The equivalence between the reconstruction problem for two projections and the problem of finding a maximum flow in the associated graph was already described by Gale in 1957 [7] in the context of reconstructing binary matrices from their row and column sums. Theorem 1 generalizes this result to the case of any reconstruction lattice A and any pair of lattice directions $(v^{(1)}, v^{(2)})$.

In the next sections we will see that the network flow approach can be extended to solve more complex variants of the reconstruction problem and that it can be used as a building block for algorithms that compute a reconstruction from more than two projections.

9.3 Weighted Reconstruction

Problem 1 is usually severely underdetermined: The number of solutions can be exponential in the size of the reconstruction lattice A. In practical applications of tomography, the projection data are usually obtained by measuring the projections of an unknown object (the *original object*), and it is important that the reconstruction closely resembles this object. One way to achieve this is to use *prior knowledge* of the original object in the reconstruction algorithm. One of the first attempts to incorporate prior knowledge in the network flow approach was described in [22], in the context of medical image reconstruction.

In this section we consider a weighted version of Problem 1:

Problem 2. Let A, $v^{(1)}$, $v^{(2)}$, $p^{(1)}$, $p^{(2)}$ be given as in Problem 1. Let $W : A \to \mathbb{R}$ be a given mapping, the *weight map*. Construct a set $F \subseteq A$ such that $P_F^{(1)} = p^{(1)}$ and $P_F^{(2)} = p^{(2)}$ and the *total weight* $\sum_{x \in F} W(x)$ is maximal.

As a shorthand notation, we refer to the total weight of F as $W(F)$. Problem 2 is a generalization of Problem 1. Through the weight map, one can express a preference for a particular solution if the reconstruction problem has more than one solution. This preference is specified independently for each $x \in A$. The higher the weight $W(x)$, the stronger is the preference to include x in the reconstruction F. Note that a preference for image features that involve several pixels cannot be specified directly through the weight map.

The associated graph G can also be used to solve the weighted version of the reconstruction problem. Define the mapping $C : E \to \mathbb{R}$ as follows:

$$C(e) = \begin{cases} -W(\Phi(e)), & \text{for } e \in E_p \ , \\ 0, & \text{otherwise} \ . \end{cases} \tag{9.11}$$

The *cost* $C(Y)$ of a flow Y in G is defined as $\sum_{e \in E} C(e)Y(e)$. The *min cost flow problem* in G deals with finding an integral flow Y of a prescribed size $\bar{T}$ in G such that the cost $C(Y)$ is minimal. If we choose $\bar{T} = S^{(1)} = S^{(2)}$, any integral flow Y of size $\bar{T}$ is a maximum flow in G and corresponds to a solution of Problem 1. The total weight of the solution that corresponds to a flow Y equals $-C(Y) = W(F_Y)$. Therefore, solving the integral min cost flow problem in G yields a solution of the reconstruction problem of maximum weight, solving Problem 2.

Just as for the max flow problem, efficient algorithms are available for solving the (integral) min cost flow problem. However, most such algorithms assume that the edge costs are integer values. If the edge costs are all in $\mathbb{Q}$, we can simply multiply all edge costs by the smallest common multiple of the denominators to obtain integer costs. If the edge costs are not in $\mathbb{Q}$, the solution of Problem 2 can be approximated by multiplying all edge costs with a large integer and rounding the resulting costs.

In [22] Slump and Gerbrands described an application of Problem 2 to the reconstruction of the left ventricle of the heart from two orthogonal angiographic projections. They used a min cost flow approach to solve a specific instance of Problem 2.

Having the ability to solve Problem 2 can be very helpful in solving a variety of reconstruction problems. We will describe two such problems. These problems deal with the reconstruction of *binary images*, i.e., images for which all pixels are either black or white. Each pixel in the image corresponds to a lattice point. A binary image corresponds to the lattice set $F \subseteq A$, where F contains the lattice points of all white pixels in the image.

Example 1. As an application of Problem 2, consider an industrial production line, where a large amount of similar objects has to be produced. Suppose that a *blueprint* is available, which specifies what the objects should look like. Occasionally, flaws occur in the production process, resulting in objects that don't match the blueprint. To check for errors, the factory uses a tomographic scanner that scans the objects in two directions: horizontal and vertical. To obtain a meaningful reconstruction from only two projections, the blueprint is used as a *model image*. For each object on the factory line, the reconstruction is computed that matches the blueprint in as many points as possible.

This problem can be formulated in the context of Problem 2. Suppose we want to reconstruct an $n \times n$ image. Put $A = \{1, \ldots, n\} \times \{1, \ldots, n\}$, $v^{(1)} = (1, 0)$, and $v^{(2)} = (0, 1)$. Let F_M be the lattice set that corresponds to the blueprint. We want to compute the solution F of Problem 1 such that

$$|(F \cap F_M) \cup (A \backslash F \cap A \backslash F_M)| = |A| - |F \triangle F_M| \tag{9.12}$$

is maximal, where $\triangle$ denotes the symmetric set difference. The term $F \cap F_M$ represents the white pixels shared by F and F_M; the term $A \backslash F \cap A \backslash F_M$

represents the shared black pixels. To formulate this problem as an instance of Problem 2, put

$$W(x) = \begin{cases} 1 & \text{if } x \in F_M \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{9.13}$$

The solution of Problem 2 for this weight map maximizes $|F \cap F_M|$, the number of common elements of F and F_M, subject to the constraints $P_F^{(1)} = p^{(1)}$ and $P_F^{(2)} = p^{(2)}$.

For the symmetric difference $F \triangle F_M$, the following equality holds:

$$|F \triangle F_M| = (|F| - |F \cap F_M|) + (|F_M| - |F \cap F_M|) \ . \tag{9.14}$$

Noting that $|F| = S^{(1)}$ is constant for all solutions of Problem 1 yields

$$|(F \cap F_M) \cup (A \backslash F \cap A \backslash F_M)| \tag{9.15}$$

$$= |A| - (S^{(1)} - |F \cap F_M|) - (|F_M| - |F \cap F_M|) \tag{9.16}$$

$$= 2|F \cap F_M| + (|A| - |F_M| - S^{(1)}) \ . \tag{9.17}$$

The term $(|A| - |F_M| - S^{(1)})$ is constant, which shows that maximizing $|(F \cap F_M) \cup (A \backslash F \cap A \backslash F_M)|$ is equivalent to maximizing $|F \cap F_M|$. We conclude that the given weight map indeed computes the reconstruction that corresponds to the blueprint in as many pixels as possible.

Figure 9.3(a) shows a blueprint image that represents a semiconductor part. The white pixels correspond to the wiring; the black pixels correspond to the background. Suppose that the object shown in Fig. 9.3(b) passes the scanner. The object clearly contains a gap that is not present in the blueprint and should be detected. Figure 9.3(c) shows a reconstruction computed from the horizontal and vertical projection data of the faulty object, using the blueprint image of Fig. 9.3(a). It has the same projections as the image in Fig. 9.3(b) and corresponds to the blueprint in as many pixels as possible. Although the reconstruction is not perfect, the gap is clearly visible and the object can be easily identified as faulty. For comparison, consider the image in Fig. 9.3(d), which also has the same projections as the images in Fig. 9.3(b) and (c). This time, the reconstruction corresponds to the blueprint in as *few* pixels as possible. Comparing this reconstruction to the original image of the faulty part shows how severely underdetermined the reconstruction problem is when only two projections are available. Of course, using a blueprint image does not guarantee that the reconstruction resembles the scanned object, but it is likely that the reconstruction will be much better than if no prior knowledge is used at all.

Example 2. Another practical problem that can be formulated in the framework of Problem 2 is how to obtain a 0–1 reconstruction from an already computed real-valued reconstruction. Computing a 0–1 reconstruction from

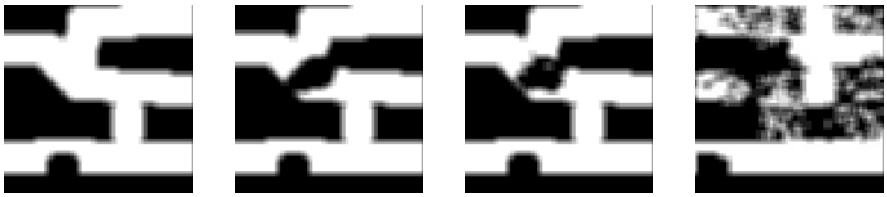


Fig. 9.3. (**a**) Blueprint image of a semiconductor part. (**b**) Test image, containing a gap in one of the wires. (**c**) Reconstruction of the test image from the horizontal and vertical projections, using the image from (a) as a model image. (**d**) Reconstruction using an inverted version of the blueprint image as a model image.

more than two projections is a computationally hard problem, but for computing a real-valued reconstruction several algorithms are available, such as the algebraic reconstruction technique (ART, see Chapter 7 of [17]). These algorithms typically require many projections to compute an accurate reconstruction. Figure 9.4(a) shows an ART reconstruction of the image in Fig. 9.3(b) from six projections. If we want the reconstruction to be binary, this reconstruction can be "rounded," such that all pixel values less than 1/2 become 0 and all pixel values of 1/2 or more become 1. The result is shown in Fig. 9.4(b). A different way to obtain a binary reconstruction is to solve Problem 2 using the pixel values of the original image as the weight map: the higher the gray value of a pixel in the continuous reconstruction, the higher the preference for this pixel to be assigned a value of 1 in the binary reconstruction. In this way the reconstruction will perfectly satisfy two of the projections, while "resembling" the continuous reconstruction. Figure 9.4(c) and (d) show two such reconstructions. The reconstruction in Fig. 9.4(c) was obtained using $v^{(1)} = (1, 0)$, $v^{(2)} = (0, 1)$. For the second reconstruction, the lattice directions $v^{(1)} = (0, 1)$, $v^{(2)} = (1, 1)$ were used. Both reconstructions are better than the one in Fig. 9.4(b) at some features, but it is not clear how to detect automatically which one is better, or how the two solutions can be combined into one superior solution. In Section 9.6, we describe how the reconstructions for different pairs of lattice directions can be combined to compute a single, more accurate reconstruction (see Fig. 9.9).

9.4 Reconstruction from Noisy Projections

The network model from Sections 9.2 and 9.3 is only suitable for computing reconstructions from *perfect* projection data. In simulation experiments, it is easy to compute perfect projections of a given image, but data that are obtained by physical measurements are usually polluted by noise. As an example of what happens in the network of Section 9.2, when the projection data contain errors, consider the possibility that $S^{(1)} \neq S^{(2)}$. In this case, it

Fig. 9.4. (**a**) ART reconstruction of the image in Fig. 9.3(b) from six projections. (**b**) Rounded ART reconstruction. (**c**) Solution of Problem 2, using the ART reconstruction as the weight map, for lattice directions $(1, 0)$ and $(0, 1)$. (**d**) Solution of Problem 2 using lattice directions $(0, 1)$ and $(1, 1)$.

is clear that no perfect solution of the reconstruction problem exists. One can still compute a maximum flow in the associated graph G. Due to the line arc capacity constraints, such a flow will always have size at most $\min(S^{(1)}, S^{(2)})$. If the measured projection for a line ℓ is lower than the number of points on that line in the original object, that line will always contain too few points in the reconstruction, regardless of the measured line projections in the other direction, because of the capacity constraint on the corresponding line edge of ℓ.

In this section we consider a modification of the associated graph which can be used to compute a reconstruction F for which the norm of the residue, i.e., the difference between the projections of F and the two prescribed projections is minimal. This network does not have the drawbacks that we described above of the network from Section 9.2.

Let $F \subseteq A$. For $k = 1, 2$, the projections $P_F^{(k)}$ of F have finite domains, so we can regard $P_F^{(k)}$ as a vector of $|L^{(k)}|$ elements. We denote the sum-norm of this vector by $|P_F^{(k)}|_1$. For a given prescribed projection $p^{(k)}$, the norm

$$|P_F^{(k)} - p^{(k)}|_1 = \sum_{\ell \in L^{(k)}} |P_F^{(k)}(\ell) - p^{(k)}(\ell)| \tag{9.18}$$

equals the total summed projection difference over all lines in $L^{(k)}$. Another norm that is often used in tomography is the Euclidean norm $|\cdot|_2$. The sum-norm is better suited for incorporation in the network flow approach. We now define a generalization of Problem 1 that allows for errors in the prescribed projections.

Problem 3. Let A, $v^{(1)}$, $v^{(2)}$, $p^{(1)}$, $p^{(2)}$ be given as in Problem 1. Let $\bar{T} \in \mathbb{N}_0$. Construct a set $F \subseteq A$ with $|F| = \bar{T}$ such that $|P_F^{(1)} - p^{(1)}|_1 + |P_F^{(2)} - p^{(2)}|_1$ is minimal.

Problem 3 asks for a set F that has a prescribed number of $\bar{T}$ elements such that F corresponds as well as possible to the two prescribed projections, according to the sum-norm. If Problem 1 has a solution, we can find all solutions by putting $\bar{T} = S^{(1)}$ and solving Problem 3. We will show that Problem 3

can be solved within the network flow model. For any n-dimensional vector $p \in \mathbb{R}^n$, define

$$|p|^+ = \sum_{i=1}^{n} \max(p_i, 0) \; . \tag{9.19}$$

To solve Problem 3, we need to make some modifications to the associated graph. Before introducing the modified graph, we prove the following lemma.

Lemma 1. *Let $F \subseteq A$, $|F| = \bar{T}$. Then, for $k = 1, 2$,*

$$|P_F^{(k)} - p^{(k)}|_1 = 2|P_F^{(k)} - p^{(k)}|^+ + S^{(k)} - \bar{T} \; . \tag{9.20}$$

Proof. Let $k \in \{1, 2\}$. By definition, we have

$$|P_F^{(k)} - p^{(k)}|_1 = |P_F^{(k)} - p^{(k)}|^+ + |p^{(k)} - P_F^{(k)}|^+ \; . \tag{9.21}$$

For each line $\ell \in L^{(k)}$, we have

$$P_F^{(k)}(\ell) = p^{(k)} + \max(P_F^{(k)}(\ell) - p^{(k)}(\ell), 0) - \max(p^{(k)}(\ell) - P_F^{(k)}(\ell), 0) \; . \tag{9.22}$$

Summing this equation over all lines $\ell \in L^{(k)}$, it follows that

$$\bar{T} = S^{(k)} + |P_F^{(k)} - p^{(k)}|^+ - |p^{(k)} - P_F^{(k)}|^+ \; ; \tag{9.23}$$

hence,

$$|P_F^{(k)} - p^{(k)}|_1 = 2|P_F^{(k)} - p^{(k)}|^+ + S^{(k)} - \bar{T} \; . \tag{9.24}$$

□

Lemma 1 shows that solving Problem 3 is equivalent to finding a set F with $|F| = \bar{T}$ for which

$$|P_F^{(1)} - p^{(1)}|^+ + |P_F^{(2)} - p^{(2)}|^+ \tag{9.25}$$

is minimal, since $S^{(1)}$, $S^{(2)}$, and $\bar{T}$ are constant.

We will now describe how the associated graph can be modified for solving Problem 3. The network from Section 9.2 forms the basis for the new network. From this point on we refer to the line edges of the network from Section 9.2 as *primary line edges*. As before, we denote the sets of all primary line edges for directions $v^{(1)}$ and $v^{(2)}$ by E_1 and E_2, respectively. Let $\ell \in L^{(k)}$ be any lattice line for direction $v^{(k)}$, and let $e \in E_k$ its corresponding primary line edge. The capacity of e imposes a hard upper bound on the number of points on ℓ in the network flow reconstruction. To relax this hard constraint, we add a second edge for each lattice line, the *excess edge*. The excess edges are parallel to their corresponding primary line edges and have the same orientation. We denote the set of excess edges for directions $v^{(1)}$ and $v^{(2)}$ by E_1' and E_2', respectively.

The resulting graph G' is shown in Fig. 9.5. The capacities of the primary line edges remain unchanged. The excess edges have unbounded capacities. Effectively, this means that the total flow through a primary line edge and its corresponding excess edge — both belonging to a line $\ell \in L^{(k)}$ — is bounded by $|A \cap \ell|$, as all outgoing flow from the line edges must pass through $|A \cap \ell|$ point edges where each has capacity 1. Therefore, it is still possible to assign finite capacities to the excess edges.

The primary line edges of the new graph are still assigned a cost of 0, as in the original network. The excess edges are assigned a cost of K, where K is a positive constant. In this way it is possible to allow more points on a line ℓ than $p^{(k)}(\ell)$, but only at the expense of a cost increase.

Now consider the problem of finding a min cost flow in G' of size $\bar{T}$. Without computing such a flow, we can already be sure that any excess edge will only carry flow if its corresponding primary line edge is saturated up to its capacity. Otherwise, the cost could be decreased by transferring flow from the excess edge to the primary edge.

Suppose that $Y : E \to \mathbb{Z}$ is a min cost flow in G' of size $\bar{T}$. The total cost of Y, given by

$$C(Y) = K(\sum_{e \in E'_1} Y(e) + \sum_{e \in E'_2} Y(e)) \,. \tag{9.26}$$

Let F_Y be the set of points for which the corresponding point edges in Y carry a positive flow, as in the Proof of Theorem 1. For any line $\ell \in L^{(k)}$, the total flow through the primary and excess edges of ℓ must equal $P^{(k)}_{F_Y}(\ell)$, because of the flow conservation constraints. Therefore, we have

$$\sum_{e \in E'_k} Y(e) = |P^{(k)}_{F_Y} - p^{(k)}|^+ \,; \tag{9.27}$$

hence,

$$C(Y) = K(|P^{(1)}_{F_Y} - p^{(1)}|^+ + |P^{(2)}_{F_Y} - p^{(2)}|^+) \,. \tag{9.28}$$

Applying Lemma 1, we conclude that a min cost flow in G' of size $\bar{T}$ yields a solution of Problem 3.

The new network can also be used to solve an extended version of Problem 2.

Problem 4. Let A, $v^{(1)}$, $v^{(2)}$, $p^{(1)}$, $p^{(2)}$ be as given in Problem 2. Let $\bar{T} \in \mathbb{N}_0$, $\alpha \in \mathbb{R}_{>0}$. Construct a set $F \subseteq A$ with $|F| = \bar{T}$ such that

$$\alpha(|P^{(1)}_F - p^{(1)}|_1 + |P^{(2)}_F - p^{(2)}|_1) - \sum_{x \in F} W(x) \tag{9.29}$$

is minimal.

Similar to the procedure for solving Problem 2, we set $C(e) = -W(\Phi(e))$ for all $e \in E_p$. Assuming that an excess edge only carries flow if its corresponding primary line edge is completely full, the total cost of an integral flow $Y \in \mathcal{Y}$ now becomes

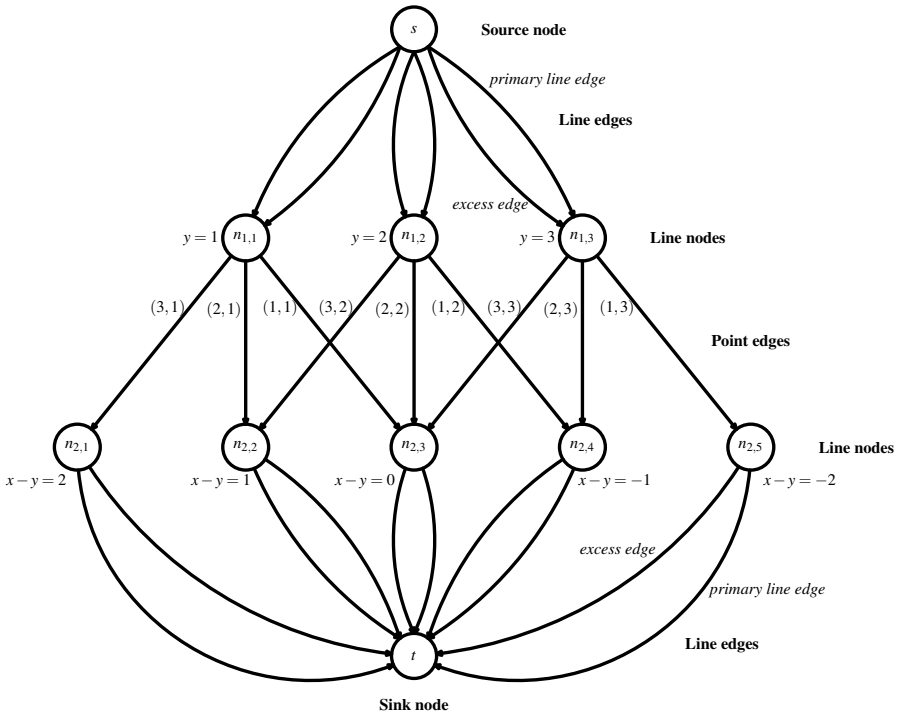


Fig. 9.5. Modified associated graph G' for the triple $(A, v^{(1)}, v^{(2)})$ from Fig. 9.1.

$$C(Y) = K(|P_{F_Y}^{(1)} - p^{(1)}|^+ + |P_{F_Y}^{(2)} - p^{(2)}|^+) - \sum_{x \in F_Y} W(x) \ . \tag{9.30}$$

Setting $K = 2\alpha$ and using Eq. (9.24) yield

$$C(Y) = \alpha(|P_{F_Y}^{(1)} - p^{(1)}|_1 + |P_{F_Y}^{(2)} - p^{(2)}|_1) - \sum_{x \in F_Y} W(x) - C_0 \ , \tag{9.31}$$

where C_0 is a constant. We conclude that if Y is an integral min cost flow of size $\bar{T}$ in G', then F_Y is a solution to Problem 4.

9.5 Algorithms and Implementation

As described in the previous sections, Problem 1, 2, 3, and 4 can all be solved as instances of network flow problems. Both the max flow problem and the min cost flow problem have been studied extensively. The book [1] provides an overview of available algorithms. A survey of the time complexities of various

network flow algorithms can be found in [21] (max flow: Chapter 10; min cost flow: Chapter 12).

We now assume that the reconstruction lattice A is a square of size $N \times N$, and we fix a pair $(v^{(1)},v^{(2)})$ of lattice directions. It is clear that the number of points in A on each lattice line parallel to $v^{(1)}$ or $v^{(2)}$ is $O(N)$. It is also clear that the number of lattice lines parallel to $v^{(1)}$ or $v^{(2)}$ that have a nonempty intersection with A is $O(N)$.

Problem 1 can be solved as an instance of the max flow problem in the associated graph. In [10], Goldberg and Rao describe an algorithm to compute a maximum flow in a graph with n nodes, m edges, and maximum edge capacity c in $O(n^{2/3}m \log(n^2/m) \log c)$ time. The associated graph of the triple $(A, v^{(1)}, v^{(2)})$ has $n = O(N)$ nodes, $m = O(N^2)$ edges, and a maximum edge capacity of $c = O(N)$. Therefore, Problem 1 can be solved in $O(N^{8/3} \log N)$ time.

Problems 2 and 3 can both be solved as instances of the min cost flow problem, i.e., the problem of finding a flow of fixed size that has minimal cost. The min cost flow problem can be reformulated as a minimum-cost circulation problem by adding an edge from the sink node t to the source node s; see Section 12.1 of [21]. In [11], Goldberg and Tarjan describe an algorithm to compute a minimum-cost circulation in a graph with n nodes, m edges, and maximum (integral) edge cost K in $O(nm \log(n^2/m) \log(nK))$ time. For the associated graph from Section 9.3, as well as for the modified associated graph from Section 9.4, this yields a time complexity of $O(N^3 \log(NK))$ for solving the min cost flow problem.

The problem of finding a maximum flow in the associated graph is known in the literature as *simple b-matching*. A flow that saturates all line edges is called a *perfect simple b-matching*, and the weighted variant of finding a perfect b-matching is known as *perfect weighted b-matching*; see Chapter 21 of [21]. For these particular network flow problems, special algorithms have been developed that are sometimes faster than general network flow algorithms.

Implementing fast network flow algorithms is a difficult and time-consuming task. The fastest way to use such algorithms is to use an existing implementation. Several network flow program libraries are available, some commercially and some for free. The ILOG CPLEX solver [15] performs very well for a wide range of network flow problems. The CS2 library from Goldberg [9] performs well and is free for noncommercial use. The same holds for the RelaxIV library from Bertsekas [4].

9.6 Extension to More Than Two Projections

As shown in the previous sections, the reconstruction problem from two projections is well understood and can be solved efficiently. We now move to the case where more than two projections are available.

Problem 5. Let $n > 2$ and let $v^{(1)}, \ldots, v^{(n)}$ be given distinct lattice directions. Let $A \subseteq \mathbb{Z}^2$ be a given lattice set. For $k = 1, \ldots, n$, let $p^{(k)} : L^{(k)} \rightarrow \mathbb{N}_0$ be given functions. Construct a set $F \subseteq A$ such that $P_F^{(k)} = p^{(k)}$ for $k = 1, \ldots, n$.

When more projections are available, the reconstruction problem is less underdetermined and we would like to be able to use the additional projections to increase the reconstruction quality. However, the reconstruction problem for more than two projections is NP-hard. Therefore, we have to resort to approximation algorithms. In this section we will describe an iterative algorithm that uses only two projections in each iteration. Within an iteration, a new pair of projections is first selected. Subsequently, an instance of Problem 2 is solved to obtain a reconstruction that satisfies the current two projections. The reconstruction from the previous iteration, which was computed using a different pair of projections, is used to construct the weight map of Problem 2 in such a way that the new reconstruction will resemble the previous one. In this way the other projections are incorporated in the reconstruction procedure in an implicit way.

Compute the start solution F^0;
$i := 0$;
while (stop criterion is not met) **do**
begin
  $i := i + 1$;
  Select a new pair of directions v_a and v_b $(1 \leq a < b \leq n)$;
  Compute a new weight map W^i from the previous solution F^{i-1};
  Compute a new solution F^i by solving Problem 2 for directions v_a and v_b, using the weight map W^i;
end

Fig. 9.6. Basic steps of the algorithm.

Figure 9.6 describes the basic structure of the algorithm. In the next subsections each of the steps will be described. The algorithm relies heavily on the methods for solving two-projection subproblems, which we described in the previous sections.

9.6.1 Computing the Start Solution

At the start of the algorithm, there is no "previous reconstruction"; a start solution has to be computed for the iterative algorithm. Ideally, the start solution should satisfy two criteria:

(a) **Accuracy**. The start solution should correspond well to the prescribed projection data.
(b) **Speed**. The start solution should be computed fast (compared with the running time of the rest of the algorithm).

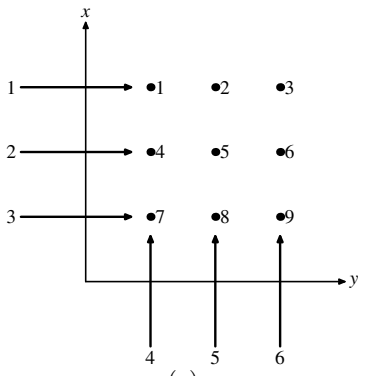
These are conflicting goals. Computing a highly accurate binary reconstruction will certainly take too much time, as the reconstruction problem is NP-hard.

There are several options for computing the start solutions, each having a different tradeoff between speed and accuracy. The first option is to choose the empty set $F^0 = \emptyset$ as a start solution, i.e., an image that is completely black.

A better alternative is to use a very fast approximate reconstruction algorithm, such as one of the greedy algorithms described in [12]. The running time of these algorithms is comparable to the time it takes to solve a single network flow problem in the body of the main loop of our algorithm.

A third possibility is to start with a continuous reconstruction. A binary start solution can then be computed from the continuous reconstruction, as described in Example 2 of Section 9.3. One class of reconstruction algorithms that can be used consists of the *algebraic reconstruction algorithms* (see Chapter 7 of [17]). The basic idea of these algorithms is to describe Problem 5 as a system of linear equations:

$$Mx = b\,. \tag{9.32}$$



(a)

$$\begin{pmatrix} 1&1&1&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0\\ 0&0&0&0&0&0&1&1&1\\ 1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&1&0&0&1&0&0&1 \end{pmatrix} \begin{pmatrix} x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\\x_8\\x_9 \end{pmatrix} = \begin{pmatrix} b_1\\b_2\\b_3\\b_4\\b_5\\b_6 \end{pmatrix}$$

(b)

Fig. 9.7. (**a**) Numbering scheme for the lattice points and the lattice lines in a rectangular reconstruction lattice. (**b**) System of equations corresponding to the numbering in (a).

Figure 9.7 shows an example 3×3 grid with the corresponding system of equations for two directions, $v^{(1)} = (1, 0)$ and $v^{(2)} = (0, 1)$. Each entry of the vector x represents an element of A. The entries of the vector b correspond

to the line projections for lattice directions $v^{(1)}, \ldots, v^{(n)}$. Each row of the binary matrix M represents a lattice line. The entry M_{ij} is 1 if, and only if, its corresponding lattice line i passes through point j.

The system (9.32) is usually underdetermined. The shortest solution of the system with respect to the Euclidean norm $|\cdot|_2$, which we denote as x^*, is a good choice for a start solution in discrete tomography. It can be shown that if Problem 5 has several solutions, then the Euclidean distance of x^* to any of these solutions is the same, so x^* is "centered" between the solutions. In addition, if the system (9.32) has binary solutions, any of these solutions has minimal norm among all integral solutions. Therefore, a short solution is likely to be a good start solution. We refer to [13] for the details of these arguments. The shortest solution of (9.32) can be computed efficiently by iterative methods, as described in [23]. After this solution has been computed, a pair $(v^{(a)}, v^{(b)})$ of lattice directions has to be selected for computing the binary start solution. The start solution is computed by solving Problem 2, using the pixel values in x^* as the weight map.

9.6.2 Computing the Weight Map

In each iteration of the main loop an instance of Problem 2 is solved. The weight map for this reconstruction problem is computed from the reconstruction of the previous iteration; it does not depend on the selected pair of lattice directions.

The weight map should be chosen in such a way that the new reconstruction resembles the reconstruction from the previous iteration. In the new instance of Problem 2, only two of the projections are used. If the new reconstruction is similar to the previous reconstruction, which was computed using a different pair of projections, the new image will also approximately adhere to the prescribed two projections from the previous iteration. Repeating this intuitive argument, we would hope that the new image also satisfies the projections from the iteration before the previous one, from the iteration before that one, etc.

The most straightforward way to make the new reconstruction resemble the previous one is to follow the approach from Example 1 in Section 9.3. If we put

$$W^i((x,y)) = \begin{cases} 1 & \text{if } (x,y) \in F^{i-1}\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{9.33}$$

the new reconstructed image F^i will have the same pixel value as F^{i-1} in as many pixels as possible. Unfortunately, this choice usually does not lead to good results. Typically, the main loop of the algorithm does not converge, making it difficult to decide when the algorithm should be terminated. This behavior is by no means surprising. The reconstruction problem from a small number of projections is severely underdetermined. If no additional

prior knowledge is used, a small number of projections (e.g., four or five) may not even be nearly enough data to uniquely determine a reconstruction.

To deal with this problem, we focus on the reconstruction of images that satisfy additional properties. *Smoothness* is a property that can often be observed in practical images: Images consist of large areas that are completely black or completely white, instead of exhibiting completely random pixel patterns. A nice property of the smoothness concept is that it can be measured *locally*. We say that an image F is *perfectly smooth* at pixel $x \in A$ if all neighboring points of x have the same value as x. Of course, this notion requires the definition of a *neighborhood* of x, which we will describe below.

From this point on, we assume that the reconstruction lattice A is rectangular. If this assumption is not satisfied, we can use any square reconstruction lattice A' for which $A \subseteq A'$, as increasing the size of the reconstruction lattice does not affect the projections.

Let F^{i-1} be the reconstructed image from the previous iteration. As a neighborhood of the point $p = (x_p, y_p) \in A$, we choose a square centered in (x_p, y_p). The reason for preferring a square neighborhood over alternatives is that the required computations can be performed very efficiently in this case. Let $p = (x_p, y_p) \in A$. Let r be a positive integer, the *neighborhood radius*. Put

$$N_p = \{\, (x, y) \in A \mid x_p - r \leq x \leq x_p + r,\, y_p - r \leq y \leq y_p + r \,\} \,. \tag{9.34}$$

N_p contains all pixels in the neighborhood of p, including p. In case p is near the boundary of A, the set N_p may contain fewer than $(2r+1)^2$ pixels. Let s_p be the number of pixels $q \in N_p$ for which $F(p) = F(q)$. Define

$$f_p \;=\; \frac{s_p}{|N_p|} \,. \tag{9.35}$$

We call f_p the *similarity fraction* of p. A high similarity fraction corresponds to a smooth neighborhood of p.

Let $g : [0,1] \to \mathbb{R}_{>0}$ be a nondecreasing function, the *local weight function*. This function determines the preference for locally smooth regions. We compute the pixel weight $W(p)$ of p as follows:

$$W(p) = 2(F(p) - \frac{1}{2}) g(f_p) \,. \tag{9.36}$$

Note that $2(F(p) - \frac{1}{2})$ is either -1 or $+1$.

When we take $g(f) = 1$ for all $f \in [0,1]$, there is no preference for local smoothness. To express the preference, we make the local weight function g an increasing function of f_p. Now a pixel having a value of 1 that is surrounded by other pixels having the same value will obtain a higher weight than such a pixel that is surrounded by 0-valued pixels. A higher weight expresses a preference to retain the value of 1 in the next reconstruction. The same reasoning holds for pixels having a value of 0, except that in this case the pixel weights are negative. Three possible choices for the local weight function are

(a) $g(f_p) = f_p$,
(b) $g(f_p) = \sqrt{f_p}$,
(c) $g(f_p) = f_p^2$.

The last choice results in a strong preference for pixels that are (almost) perfectly smooth. Of course, many other local weight functions are possible. In [3], extensive results are reported for the local weight function

$$g(f_p) = \begin{cases} 1 & (f_p \leq 0.65) , \\ 4f & (0.65 < f_p < 1) , \\ 9 & (f_p = 1) . \end{cases} \tag{9.37}$$

The choice for this particular function is somewhat arbitrary. In each case, a preference is expressed for retaining the pixel value of p in the next reconstruction, instead of changing it. In the case that the whole neighborhood of p has the same value as p, this preference is very strong. If the neighborhood contains a few pixels having a different value, the preference is less. If there are many pixels in the neighborhood that have a different value, the preference is even smaller.

So far we have not discussed how the neighborhood radius should be chosen. If the start solution is already a good approximation of the final reconstruction, using a fixed value of $r = 1$ works well. For this neighborhood radius, the differences between consecutive reconstructions F^i are typically small. It is usually better to start with a larger neighborhood radius, e.g., $r = 5$ or $r = 8$. This typically results in large changes between consecutive reconstructions. Only very large regions of pixels that have the same value obtain a strong preference to keep this value. Regions that are less smooth can easily change. A choice that works well for the range of images studied in [3] is to start the algorithm with $r = 8$ and to set $r = 1$ after 50 iterations.

9.6.3 Choosing the Pair of Directions

In each iteration of the main loop of the algorithm, a new pair of lattice directions is selected. There is no selection scheme that is "obviously best" in all cases. Yet there are several ways for choosing the direction pairs that perform well in practice.

A good choice for the new direction pair is to choose the lattice directions $v^{(a)}$, $v^{(b)}$, for which the total projection error

$$|P_F^{(a)} - p^{(a)}|_1 + |P_F^{(b)} - p^{(b)}|_1 \tag{9.38}$$

is largest. After solving the new instance of Problem 2, the total projection error for these two lattice directions will be zero, assuming perfect projection data. This also guarantees that if at least two projections have a positive projection error after the previous iteration, both new lattice directions will be different from the ones used in the previous iteration.

If the number of projections is very small (e.g., four or five) the projection error is not a good criterion for selecting the new projection pair. For the case of four projections, this scheme leads to cycling behavior between two pair of projections. The other projection pairs are not used at all. To avoid this behavior, it is better to use a fixed order of direction pairs, in which all pairs occur equally often. Such schemes, for four and five projections, are shown in Table 9.1.

Table 9.1. (left) Lattice Direction Scheme for Four Projections (Each projection pair is used equally often. No projection pair is used in two consecutive iterations.) (right) Lattice Direction Scheme for Five Projections

Iteration	1	2	3	4	5	6
1st dir.	1	3	1	2	1	2
2nd dir.	2	4	3	4	4	3

Iteration	1	2	3	4	5	6	7	8	9	10
1st dir.	1	3	5	2	4	1	2	3	4	5
2nd dir.	2	4	1	3	5	3	4	5	1	2

9.6.4 Stop Criterion

In general, it is not easy to determine when the iterative algorithm from Fig. 9.6 should terminate, because there is no guaranteed convergence. Yet, the experiments from [3] show that if enough projections are available, the algorithm often converges to the exact solution of Problem 5. Detecting that an exact solution has been found is easy, and the algorithm always terminates in that case.

To measure the quality of the current reconstruction F, the *total projection difference*

$$D(F) = \sum_{k=1}^{n} |P_F^{(k)} - p^{(k)}|_1 \tag{9.39}$$

can be used. This distance is 0 for any perfect solution of Problem 5 and greater than 0 otherwise. The total projection difference can be used for defining termination conditions. If no new minimal value is found for the total projection distance during the last T iterations, where T is a positive integer, the algorithm terminates. We used $T = 100$ for all experiments in the next subsection.

9.6.5 Some Results

We will now show some experimental results obtained by the iterative algorithm from Fig. 9.6. The performance of the algorithm, as for any other

general discrete tomography algorithm, depends heavily on the type of image that is being reconstructed and the number of available projections. In order to give extensive statistical results about the reconstruction quality, a *class of images* has to be defined first. All images in this class should have similar characteristics. The performance of the algorithm can then be measured for this particular image class. In [3], reconstruction results were reported for several different image classes. The results in this section show a varied set of test images with their reconstructions, rather than providing extensive quantitative results. Figure 9.8 shows six test images, each having different characteristics, and their reconstructions. The number n of projections that was used is shown in the figure captions. The reconstructions of the first five images (a–e) are all *perfect reconstructions*, obtained using the weight function in Eq.(9.37) from Section 9.6.2. For the first four images (a–d), the linear local weight function also works very well, even faster than the function in Eq.(9.37). The image in Fig. 9.8(e) contains many fine lines of only a single pixel's thickness. In this case, the local weight function $g(f_p) = \sqrt{f_p}$ works well. Which local weight function is best for a certain class of image depends strongly on characteristics of the particular class. This is also true for the *number of projections* required to reconstruct an image.

For reconstructing the image in Fig. 9.8(a), four projections suffice. The structure of the object boundary is fairly complex, but the object contains no "holes." The iterative algorithm reconstructed the image perfectly from four projections (horizontal, vertical, diagonal, and antidiagonal).

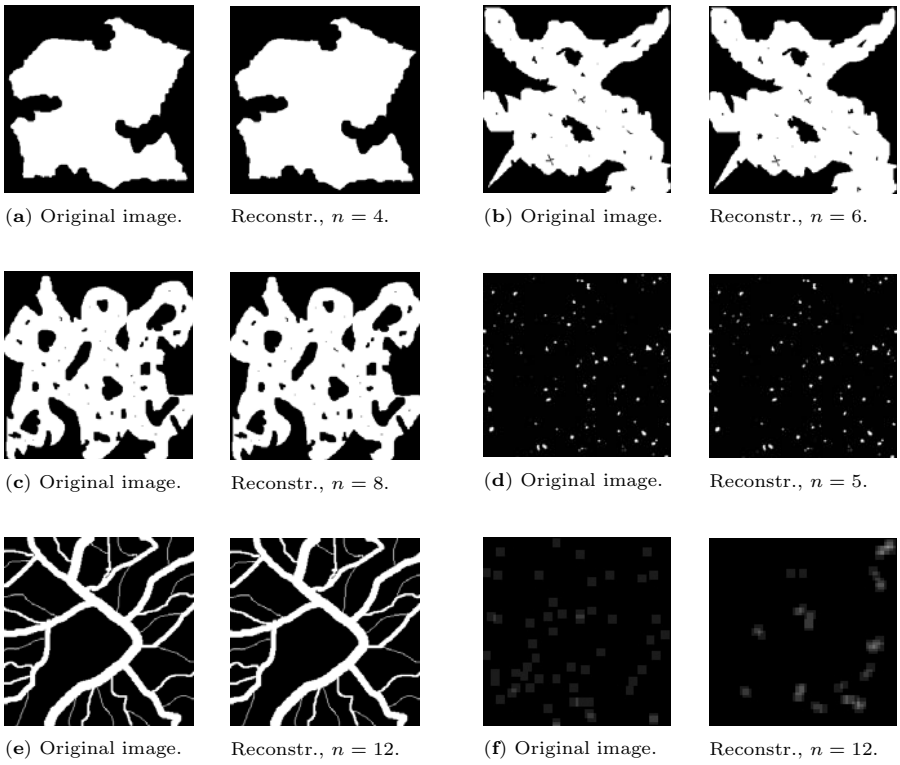
Figure 9.8(b) shows a much more complicated example. The object contains many cavities of various sizes and has a very complex boundary. Some of the black holes inside the white region are only a single pixel in size. In this case, our algorithm requires six projections to compute an accurate reconstruction. Some of the fine details in this image are not smooth at all. Still, the fine details are reconstructed with great accuracy. The image from Fig. 9.8(c) is even more complex. It requires eight projections to be reconstructed perfectly.

The image in Fig. 9.8(d) has a lot of local smoothness in the black areas, but it contains no large white areas. Still, the image is smooth enough to be reconstructed perfectly from only five projections. This example also illustrates that very fine, nonsmooth details can be reconstructed by the algorithm, as long as the entire image is sufficiently smooth.

Figure 9.8(e) shows a vascular system containing several very thin vessels. The iterative algorithm can reconstruct the original image perfectly from 12 projections. This is quite surprising, since the very thin vessels have a width of only one pixel, so they are not smooth. Still, the smoothness of the thicker vessels and the background area provides the algorithm with enough guidance to reconstruct the original image.

When the image contains no structure at all, the algorithm performs very badly. Figure 9.8(f) shows an image of random noise. The reconstruction from 12 projections shows that our algorithm has a preference for connected areas

of white and black pixels. In this case, however, the smoothness assumption is obviously not satisfied by the original image. The distance between the projections of the image found by our algorithm and the prescribed projections is very small, however.



(**a**) Original image. Reconstr., $n = 4$. (**b**) Original image. Reconstr., $n = 6$.

(**c**) Original image. Reconstr., $n = 8$. (**d**) Original image. Reconstr., $n = 5$.

(**e**) Original image. Reconstr., $n = 12$. (**f**) Original image. Reconstr., $n = 12$.

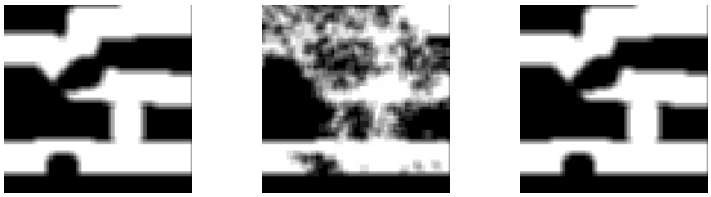
Fig. 9.8. Six original images and their reconstructions. The number n of projections is shown in the figure caption. [The images in Fig. 9.8(f) show a zoomed portion of the center of the actual images, to make the details clearly visible.]

For reconstructing the images in Fig. 9.8, a sufficiently large set of projections was used. Figures 9.9 and 9.10 demonstrate the result of using the algorithm if too few projections are available.

Figure 9.9 shows the results of reconstructing the semiconductor image of Fig. 9.3(b) from three and four projections, respectively. When we use only three projections, a reconstruction is found that has exactly the prescribed projections, but the reconstruction is very different from the original image.

If we use too few projections, the algorithm may also get stuck in a local minimum of the projection distance, which is shown in Fig. 9.10. The original image can be reconstructed perfectly from five projections, but when only

four projections are used, the algorithm fails to find a good reconstruction. The projections of the reconstructed image are significantly different from the four prescribed projections, yet the algorithm is unable to find a better reconstruction.



Original image. Reconstruction, $n = 3$. Reconstruction, $n = 4$.

Fig. 9.9. (**a**) Original image. (**b**) Reconstruction from three projections (horizontal, vertical, diagonal) that has exactly the prescribed projections. (**c**) Perfect reconstruction of the original image from four projections (horizontal, vertical, diagonal, antidiagonal).

Original image. Reconstruction, $n = 4$. Reconstruction, $n = 5$.

Fig. 9.10. (**a**) Original image. (**b**) Reconstruction from four projections, which does not have the prescribed projections. The horizontal and vertical projections are identical to those of the first image. The diagonal and antidiagonal projections have a total projection difference (sum of absolute values) of 184 and 126, respectively. (**c**) Perfect reconstruction of the original image from five projections.

9.7 Reconstructing 3D Volumes

So far our approach has been concerned with the reconstruction of two-dimensional images. In many practical applications of tomography, it is important to obtain 3D reconstructions. Computing 3D reconstructions is usually a computationally very demanding task, as large amounts of data are involved. There is a slight difference in terminology between 2D and 3D reconstructions. *Pixels* in 2D images are usually called *voxels* in the context of 3D images, where they represent a unit cube in the 3D volume.

If there exists a plane H in $\mathbb{Z}^3$ such that all projection directions lie in H, all algorithms for 2D reconstruction can be used directly for 3D reconstruction as well, reconstructing the volume as a series of slices. All slices can be reconstructed in parallel, which allows for a large speedup if several processors are used. A disadvantage of reconstructing all slices independently is that certain types of prior knowledge cannot be exploited. For example, if we generalize the preference for local smoothness from Section 9.6 to 3D, voxels from adjacent slices are required to compute the neighborhood density of a certain voxel. Therefore, the reconstruction computations of the slices are no longer independent.

If the projection directions are not coplanar, reconstructing the volume as a series of slices is not possible. This situation occurs, for example, in the application of atomic resolution electron microscopy. The crystal sample is tilted in two directions to obtain as many useful projections as possible.

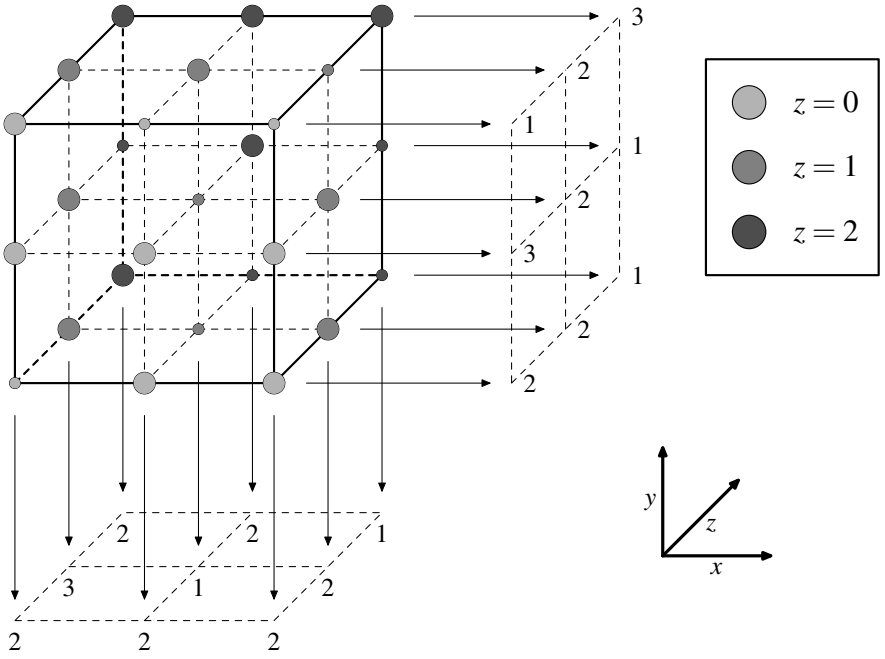


Fig. 9.11. $3 \times 3 \times 3$ binary volume with its projections in directions $v^{(1)} = (1, 0, 0)$ and $v^{(2)} = (0, 1, 0)$. A large circle indicates a value of 1; a small circle indicates a value of 0.

We will now show how the algorithm for 2D reconstruction from the previous section can be generalized to the 3D case. Figure 9.11 shows an example of a $3 \times 3 \times 3$ volume A with its projections parallel to the lattice directions

$v^{(1)} = (1, 0, 0)$ and $v^{(2)} = (0, 1, 0)$. Lattice points that have a value of 1 (i.e., lattice points included in the set F) are indicated by large dots. Similar to the associated network from Section 9.2, each two-projection problem in 3D also has an associated graph. The associated graph for the volume in Fig. 9.11 is shown in Fig. 9.12. Just as in the 2D case, the associated graph contains a line edge for every projected lattice line. The middle layer of edges contains one edge for every voxel, connecting the two line nodes for which the corresponding lines intersect with that voxel.

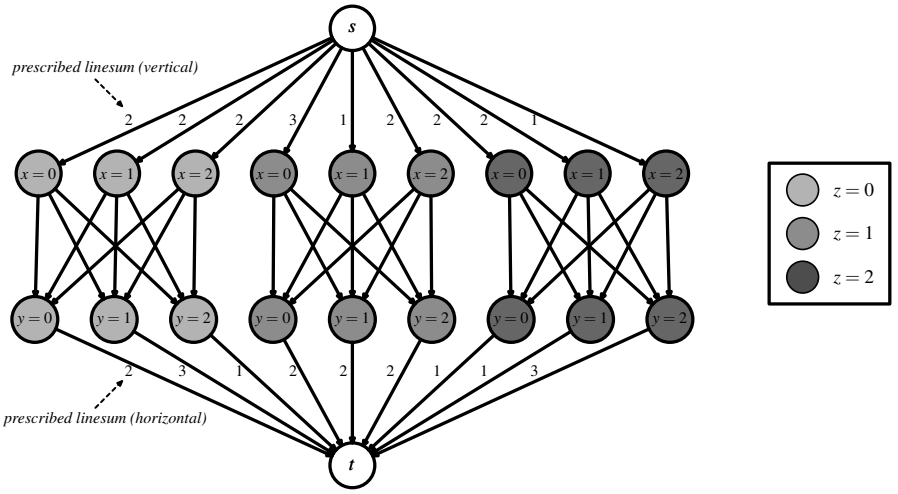
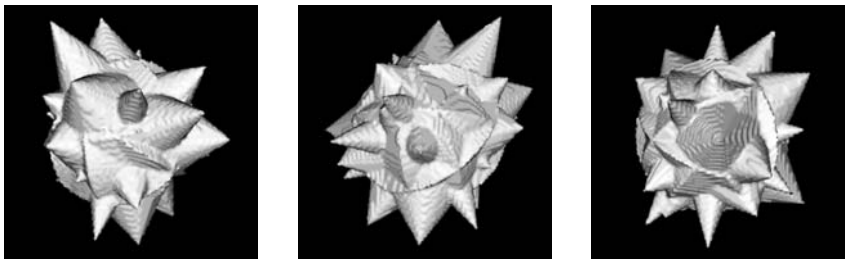
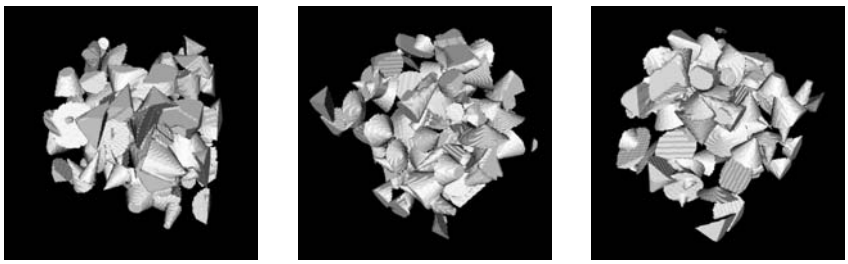


Fig. 9.12. Network corresponding to the two-projection reconstruction problem in Fig. 9.11.

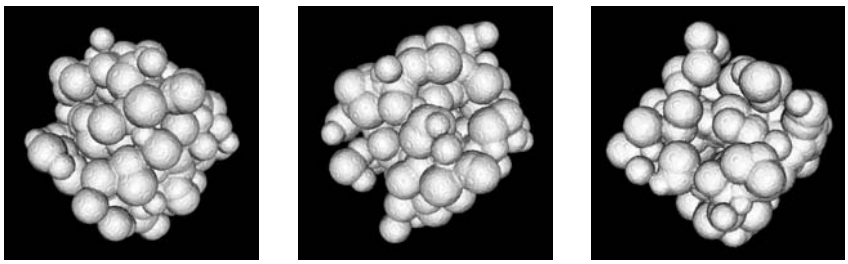
Figure 9.12 shows a nice property of the two-projection reconstruction problem. For any point edge $(n_{1,i}, n_{2,j}) \in E_p$ in the associated graph, the lines $\ell_{1,i}$ and $\ell_{2,j}$ have a nonempty intersection in A, so there is a plane in $\mathbb{Z}^3$ that contains both $\ell_{1,i}$ and $\ell_{2,j}$. Since $\ell_{1,i}$ and $\ell_{2,j}$ are translates of $v^{(1)}$ and $v^{(2)}$, respectively, this plane will always be a translate of the plane spanned by $v^{(1)}$ and $v^{(2)}$. If two lines $\ell_{1,i}$ and $\ell_{2,i}$ lie in different translates of this plane, there will be no voxel edge connecting the corresponding line nodes. Therefore, the max flow problem can be solved for each translate of the plane independently. In the example network of Fig. 9.12, the subproblems for each of the planes $z = 0$, $z = 1$, and $z = 2$ can be solved independently. This property holds for any pair $(v^{(a)}, v^{(b)})$ of lattice directions, although the sizes of the subproblems depend on the direction pair. The number of point edges in each subproblem is bounded by the maximal number of voxels in A that lie in a single plane.



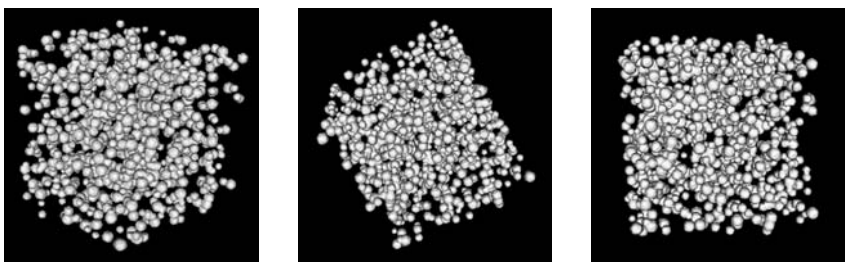
(**a**) Cones pointing out, 128×128×128: perfect reconstruction from 4 projections.



(**b**) Random cones, 128×128×128: perfect reconstruction from 6 projections.



(**c**) 100 spheres, 169×169×169: perfect reconstruction from 6 projections.



(**d**) 1000 small spheres, 139×139×139: perfect reconstruction from 6 projections.

Fig. 9.13. Reconstruction results for four 3D volumes.

Figure 9.13 shows four different test volumes, each displayed from three different viewing directions. The directions were selected to provide a clear view of the volume; they are not parallel to any of the projection directions. The iterative network flow algorithm can reconstruct each of these images from projections along the six lattice directions $(1,0,0), (0,1,0), (0,0,1), (1,1,0)$, $(1,0,1)$, and $(0,1,1)$. The image dimensions are shown in the figure. For the test image in Fig. 9.13(a), a perfect reconstruction is already found if only the first four projections are used. The test volumes are surrounded by a black background, which is not counted in the image dimensions. For all four test volumes, the algorithm computed the 3D reconstruction within 7 minutes on a standard 2.4GHz Pentium IV PC.

9.8 Extension to Plane Sets

So far we have considered the reconstruction of lattice sets in 2D and 3D. This model is well suited for the application of nanocrystal reconstruction at atomic resolution in electron microscopy [16]: Atoms in a crystalline solid are arranged regularly in a lattice structure. In many other applications of tomography there is no "intrinsic lattice." In this section, we consider the reconstruction of subsets of $\mathbb{R}^2$ from its projections in a small number of directions. We will also refer to such subsets as *planar black-and-white images*. In this new context, the projection along a line is no longer a sum over a discrete set; rather it is a *line integral* or *strip integral* of a function $\mathbb{R}^2 \to \{0,1\}$, which yields a real value.

Binary tomography problems without an intrinsic lattice structure occur often in practice, for example, in medical imaging [14]. Besides using a pixel representation for the reconstructed image, other representations have also been proposed. If the object of interests can be approximated well by a polygon, for example, one can use a polygonal, representation in the reconstruction algorithm (see [18]).

The iterative network flow algorithm can be adapted in such a way that it can be used for the reconstruction of planar black-and-white images. We will only give a high-level overview of the algorithm. The details will be described in a future publication.

Figure 9.14 shows an example of a planar black-and-white image along with two of its projections. If strip projections are used, the total amount of "white" (or black) in a set of consecutive strips parallel to the projection direction is measured. As it is impossible to represent all planar images in a computer program, they are approximated by representing the images on a pixel grid. Each of the pixels can have a value of either 1 (white) or 0 (black).

The weighted reconstruction problem for two projections (i.e., Problem 2 in the context of lattice sets) can also be solved efficiently in the case of planar subsets. However, a specifically chosen pixel grid must be used, which depends on the two projection directions. Figure 9.15 shows how the grid is

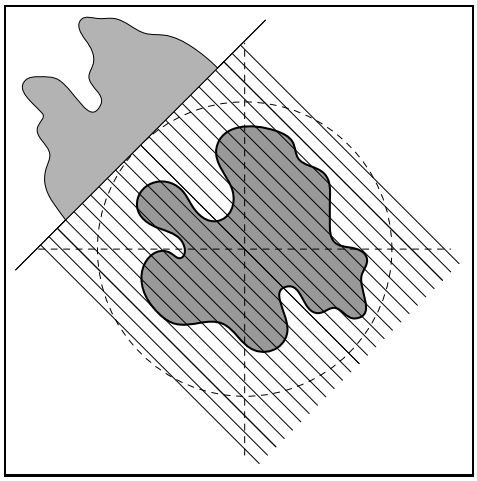


Fig. 9.14. A planar black-and-white image with one of its projections. If strip projections are used, the total amount of "black" in a set of consecutive strips parallel to the projection direction is measured.

formed from two projections. Every pixel on the grid is the intersection of a strip in the first direction with a strip in the second direction. The network flow approach can be used for this pixel grid, to compute a 0–1 reconstruction from the given two projections.

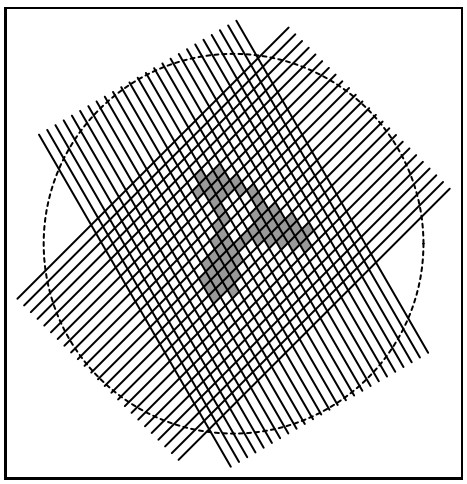


Fig. 9.15. Two parallel beams span a pixel grid. On this pixel grid, network flow methods can be used to solve the two-projection 0–1 reconstruction problem.

In every iteration of the iterative network flow algorithm, a new pair of projections is selected. Therefore, the pixel grid on which the new solution is computed is different in each iteration. To compute the weight map for the new pixel grid, the image from the previous iteration (defined on a different pixel grid) is first converted to a grayscale image on the pixel grid by interpolation. Recall that the computation of the *similarity fraction* for a pixel p does not require that neighboring pixels of p are *binary*. Therefore, the new weight map can be computed in a straightforward way. An overview of the adapted version of the iterative algorithm is shown in Fig. 9.16.

Compute the start solution F^0 on the standard pixel grid;
$i := 0$;
while (stop criterion is not met) **do**
begin
  $i := i + 1$;
  Select a new pair of directions $v^{(a)}$ and $v^{(b)}$ $(1 \leq a < b \leq n)$;
  Convert the previous solution F^{i-1} to an image $\hat{F}^{i-1}$ that is defined on the pixel grid spanned by directions $v^{(a)}$ and $v^{(b)}$;
  Compute a new weight map W^i from the image $\hat{F}^{i-1}$;
  Compute a new solution F^i on the grid spanned by $v^{(a)}$ and $v^{(b)}$ by solving a variant of Problem 4, using the weight map W^i.
end

Fig. 9.16. Basic steps of the algorithm for plane sets.

Just as for lattice images, the iterative network flow algorithm for plane sets is capable of computing very accurate reconstructions if a sufficient number of projections is available. However, if the original image does not have an intrinsic lattice structure, we cannot expect a reconstruction that perfectly matches the projection data, as it is unlikely that the original image can be represented as a binary image on the pixel grid used by the algorithm.

Acknowledgments

The author would like to thank Robert Tijdeman, Herman te Riele, and Willem, Jan Palenstijn for their valuable comments and suggestions.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs, NJ (1993).

2. Anstee, R.P.: The network flows approach for matrices with given row and column sums. *Discr. Math.*, **44**, 125–138 (1983).
3. Batenburg, K.J.: Reconstructing binary images from discrete X-rays. *Tech. Rep. PNA-E0418*, CWI, Amsterdam, The Netherlands (2004).
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, Cambridge, MA (1994).
5. Dürr, C.: Ryser's algorithm can be implemented in linear time. http://www.lix.polytechnique.fr/~durr/Xray/Ryser/.
6. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. *Can. J. Math.*, **8**, 399–404 (1956).
7. Gale, D.: A theorem on flows in networks. *Pac. J. Math.*, **7**, 1073–1082 (1957).
8. Gardner, R.J., Gritzmann, P., Prangenberg, D.: On the computational complexity of reconstructing lattice sets from their X-rays. *Discr. Math.*, **202**, 45–71 (1999).
9. Goldberg, A.V.: An efficient implementation of a scaling minimum-cost flow algorithm. *J. Algebra*, **22**, 1–29 (1997).
10. Goldberg, A.V., Rao, S.: Beyond the flow decomposition barrier. *J. ACM*, **45**, 783–797 (1998).
11. Goldberg, A.V., Tarjan, R.E.: Finding minimum-cost circulations by successive approximation. *Math. Oper. Res.*, **15**, 430–466 (1990).
12. Gritzmann, P., de Vries, S., Wiegelmann, M.: Approximating binary images from discrete X-rays. *SIAM J. Opt.*, **11**, 522–546, (2000).
13. Hajdu, L., Tijdeman, R.: Algebraic aspects of discrete tomography. *J. Reine Angew. Math.*, **534**, 119–128 (2001).
14. Herman, G.T.,Kuba, A.: Discrete tomography in medical imaging. *Proc. IEEE*, **91**, 380–385 (2003).
15. *ILOG CPLEX*, http://www.ilog.com/products/cplex/.
16. 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. *Mat. Res. Soc. Symp. Proc.*, **839**, 4.5.1–4.5.6 (2004).
17. Kak, A.C., Slaney, M.: *Principles of Computerized Tomographic Imaging*. SIAM, Philadelphia, PA (2001).
18. Mohammad-Djafari, A.: Binary polygonal shape image reconstruction from a small number of projections. *ELEKTRIK*, **5**, 127–139 (1997).
19. Ryser, H.J.: Combinatorial properties of matrices of zeros and ones. *Can. J. Math.*, **9**, 371–377 (1957).
20. Ryser, H.: *Combinatorial Mathematics*, Mathematical Association of America, Washington, DC (1963).
21. Schrijver, A.: *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Heidelberg, Germany (2003).
22. Slump, C.H., Gerbrands, J.J.: A network flow approach to reconstruction of the left ventricle from two projections. *Comp. Graph. Image Proc.*, **18**, 18–36 (1982).
23. Tanabe, K.: Projection method for solving a singular system. *Num. Math.*, **17**, 203–214 (1971).