

International Journal of Shape Modeling
© World Scientific Publishing Company

DISCRETE TOMOGRAPHIC RECONSTRUCTION OF BINARY IMAGES WITH DISJOINT COMPONENTS USING SHAPE INFORMATION

PÉTER BALÁZS

*Department of Image Processing and Computer Graphics, University of Szeged
Árpád tér 2., Szeged, H-6720, Hungary
pbalazs@inf.u-szeged.hu
<http://www.inf.u-szeged.hu/~pbalazs>*

Received (Day Month Year)

Revised (Day Month Year)

Accepted (Day Month Year)

We present a general framework for reconstructing binary images with disjoint components from the horizontal and vertical projections. We develop a backtracking algorithm that works for binary images having components from an arbitrary class. Thus, a priori knowledge about the components of the image to be reconstructed can be incorporated into the reconstruction process. In addition, we show how to extend the algorithm to obtain a branch-and-bound scheme useful to reconstruct images satisfying some further properties (for example similarity to a model image) as much as possible. Experimental results are also presented.

Keywords: Binary tomography; reconstruction; disjoint components; branch-and-bound; model image.

1991 Mathematics Subject Classification: 22E46, 53C35, 57S20

1. Introduction

Computerized tomography (CT) is an imaging procedure to obtain density information about the interior of an object from their projections without damaging or destroying the object itself. The reconstruction is often done slice-by-slice, i.e., 2D reconstructed slices are integrated together to form a 3D model of the object. Usually several hundreds of projections are needed to gain a satisfactory reconstruction of a single 2D slice.¹ However, in some applications there is often a practical limitation that only a few (usually at most about 10) projections of the object can be made. Classical reconstruction methods of CT are hardly applicable in such cases. For example, in electron microscopy, even 3-4 electron-beams transmitted through the specimen may modify the investigated macromolecule or crystalline, thus preventing to acquire further projections from the original structure.^{2,3} In angiography, the contrast agent injected (or inhaled) into the human organs can be present just for a very short time therefore there is a chance to obtain just a few

projections.^{4,5} Finally, in neutron-tomography applied for non-destructive testing, the acquisition of projections can be very expensive which again causes that the number of available projections is very small.⁶

Fortunately, in those applications it is often known beforehand that the reconstructed image should contain only a small number of known grey-intensity values. *Discrete tomography* (DT) investigates how this prior knowledge can be exploited to eliminate the problems arising from using a small number of available projections.^{7,8} *Binary tomography* (BT) arises as a special case of DT, where the task is to reconstruct a binary image, and it has its own mathematical background.

Due to the small number of available projections, the reconstruction in BT is usually very underdetermined. Thus, additional knowledge is needed to avoid intractability and reduce the number of possible solutions. The aim of this paper is to present a general framework for reconstructing binary images from just two projections. The prior assumption we use this time is that the image to be reconstructed should consist of disjoint components.

This paper is structured as follows. The necessary definitions are given in Sect. 2. Section 3 describes the reconstruction framework. In Sect. 4 we give experimental results. In Sect. 5 we show how the general framework can be improved to obtain a branch-and-bound technique in order to get faster and more accurate reconstructions. Especially, we will focus on the reconstruction when the approximate shape of the image is known beforehand. Finally, in Sect. 6 we summarize our experiences.

2. Preliminaries

A binary image will be considered as a set of unitary cells centered on the 2D integer lattice. It also can be represented in a uniquely determined way by a binary matrix F of a minimal size $m \times n$ or by a discrete set $\hat{F} = \{(i, j) \in \mathbb{Z}^2 \mid f_{ij} = 1\}$ of the non-zero positions of F as well. Figure 1 shows a binary image represented by the discrete set $\hat{F} = \{(1, 4), (2, 2), (2, 3), (2, 4), (2, 5), (3, 2), (4, 1), (4, 2), (4, 3), (5, 1), (5, 2), (5, 3), (5, 4)\}$. In the followings we will simply use the notation F for the binary image, and the corresponding binary matrix and discrete set, too. To avoid confusion we mention here that on the size of the binary image we mean the size of the binary matrix (i.e. not the elements of the corresponding discrete set) and the positions of the minimal bounding rectangle of the binary image will be marked by the corresponding matrix positions. That is, the top-left position is marked by (1,1). The *horizontal* and *vertical* projections of a binary image F are the vectors $\mathcal{H}(F) = (h_1, \dots, h_m)$, and $\mathcal{V}(F) = (v_1, \dots, v_n)$, respectively, where

$$h_i = \sum_{j=1}^n f_{ij} \quad (i = 1, \dots, m), \quad (1)$$

$$v_j = \sum_{i=1}^m f_{ij} \quad (j = 1, \dots, n). \quad (2)$$

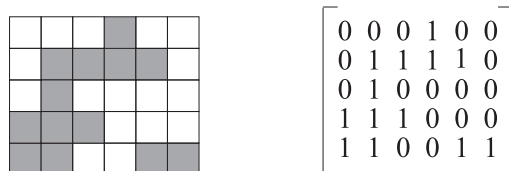


Fig. 1. A binary image of size 5×6 (left) and its matrix representation (right).

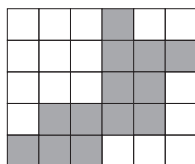


Fig. 2. An hv -convex directed polyomino.

For example, the binary image F in Fig. 1 has the horizontal and vertical projections $\mathcal{H}(F) = (1, 4, 1, 3, 4)$, and $\mathcal{V}(F) = (2, 4, 2, 2, 2, 1)$, respectively. In the rest of the paper (without losing generality) we always assume that none of the projections' coordinates are equal to 0.

In the reconstruction two vectors, H and V are given and the task is to construct a binary image such that $\mathcal{H}(F) = H$ and $\mathcal{V}(F) = V$. The first method to solve this problem was published in Ref. 9 where it was also shown that the number of solutions of the same reconstruction task can be extremely large. To reduce the number of possible solutions of this task we are usually interested in reconstructing images belonging to a certain class of shapes defined by some geometrical properties like connectedness or convexity.^{10,11,12,13,14,15,16,17} Two positions $P = (p_1, p_2)$ and $Q = (q_1, q_2)$ in a binary image are said to be 4 -adjacent if $|p_1 - q_1| + |p_2 - q_2| = 1$. A binary image F is 4 -connected (with an other term *polyomino*) if for any two points $P, Q \in F$ there is a sequence of distinct positions $P_0 = P, \dots, P_k = Q$ of F such that P_l is 4-adjacent to P_{l-1} , for each $l = 1, \dots, k$. We say that the binary image F is hv -convex if all the rows and columns of F are 4-connected. An hv -convex polyomino P is called *directed* if the point $(m, 1)$ belongs to P . Figure 2 shows an hv -convex directed polyomino.

A binary image can always be partitioned into maximal 4-connected images, in a uniquely determined way. Those partitions are called the *components* of the discrete set. Clearly, if the binary image is a polyomino then it has just one component. Let F be a binary image with k ($k \geq 1$) components such that $I_l \times J_l = [i_l, i'_l] \times [j_l, j'_l]$ is the minimal bounding rectangle of the l -th component of F . We say that the components of F are *disjoint* if for any $1 \leq l, l' \leq k$ the inequality $l \neq l'$ implies that $I_l \cap I_{l'} = \emptyset$ and $J_l \cap J_{l'} = \emptyset$ (see Fig. 3). Note that this statement is stronger than

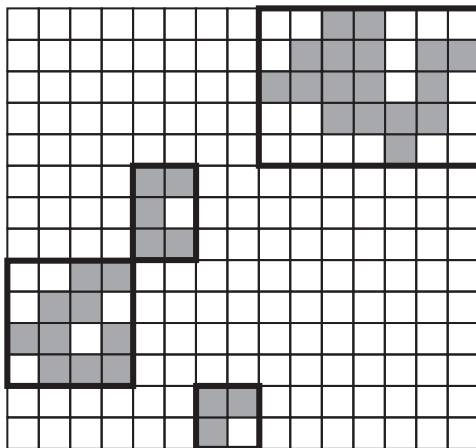


Fig. 3. A binary image with four disjoint components. The minimal bounding rectangles of the components are drawn bold.

just saying that the minimal bounding rectangles of the components are pairwise disjoint. While in this latter case the reconstruction for certain classes of discrete sets can be extremely difficult (see, e.g. Ref. 18), as a consequence of the above definition of disjointness the support sets of the projections can be partitioned which can facilitate the reconstruction.

3. The Reconstruction Framework

Some reconstruction algorithms have been already supplied for certain subclasses of binary images having disjoint components using four projections.^{19,20} Our aim is to develop a general reconstruction framework which can effectively exploit shape information of the image to be reconstructed and uses just two projections. We always will assume that the components of the image belong to a given class \mathcal{S} of polyominoes satisfying some shape restrictions. When no prior knowledge of the components is available then we simply assume that \mathcal{S} is the whole class of polyominoes. Basically, the method we present here is not bound by the horizontal and vertical projections, but – for the sake of technical simplicity – we will concentrate on those projections. We begin with two definitions.

Definition 3.1. Let $H = (h_1, \dots, h_m)$ and $1 \leq i_1 \leq i_2 \leq m$. The *partial sum of H on the interval $[i_1, i_2]$* is defined by

$$H_{[i_1, i_2]} = \sum_{i=i_1}^{i_2} h_i . \quad (3)$$

If $i_1 = 1$ then – as special cases – we obtain the cumulated horizontal vectors of F which are used in several reconstruction algorithms.^{10,11,12,14} Similar definition

can be given in case of the vertical projection V as well.

Definition 3.2. Let \mathcal{S} be a class of polyominoes, $H \in \mathbb{N}^m$ and $V \in \mathbb{N}^n$. We say that the intervals $[i_1, i_2]$ of H ($1 \leq i_1 \leq i_2 \leq m$) and $[j_1, j_2]$ of V ($1 \leq j_1 \leq j_2 \leq n$) are *compatible* with respect to the class \mathcal{S} (and write $[i_1, i_2] \sim_{\mathcal{S}} [j_1, j_2]$), if there exists a polyomino $P \in \mathcal{S}$ with $\mathcal{H}(P) = (h_{i_1}, \dots, h_{i_2})$ and $\mathcal{V}(P) = (v_{j_1}, \dots, v_{j_2})$.

Now, we give necessary conditions for the compatibility of an interval-pair.

Proposition 3.1. *Let $H \in \mathbb{N}^m$ and $V \in \mathbb{N}^n$. If $[i_1, i_2] \sim_{\mathcal{S}} [j_1, j_2]$ for an arbitrary class \mathcal{S} then the followings hold simultaneously*

- (a) $H_{[i_1, i_2]} = V_{[j_1, j_2]}$,
- (b) $h_i \leq j_2 - j_1 + 1$ for each $i_1 \leq i \leq i_2$,
- (c) $v_j \leq i_2 - i_1 + 1$ for each $j_1 \leq j \leq j_2$.

Proof. If $[i_1, i_2] \sim_{\mathcal{S}} [j_1, j_2]$ then there exists a polyomino $P \in \mathcal{S}$ with $\mathcal{H}(P) = (h_{i_1}, \dots, h_{i_2})$ and $\mathcal{V}(P) = (v_{j_1}, \dots, v_{j_2})$. The minimal bounding rectangle of P is $[i_1, i_2] \times [j_1, j_2]$. Then, property (a) necessarily holds. Furthermore, no coordinates of the horizontal/vertical projection of P can be greater than the width/height of the minimal bounding rectangle of P , respectively, thus properties (b) and (c) do also hold. \square

The following proposition offers the basis of our algorithm.

Proposition 3.2. *Let F be a binary image with disjoint components from the class \mathcal{S} . Then there exists a partitioning of $\mathcal{H}(F)$ into intervals I_1, \dots, I_k , a partitioning of $\mathcal{V}(F)$ into intervals J_1, \dots, J_k ($k \geq 1$), and a bijective mapping $\varphi : \{I_1, \dots, I_k\} \longrightarrow \{J_1, \dots, J_k\}$ such that $I_i \sim_{\mathcal{S}} \varphi(I_i)$ ($i = 1, \dots, k$).*

Proof. Let F be a binary image with k ($k \geq 1$) disjoint components from the class \mathcal{S} , and let $I_l \times J_l = [i_l, i'_l] \times [j_l, j'_l]$ be the minimal bounding rectangle of the l -th component ($l = 1, \dots, k$). Then I_1, \dots, I_k gives a partitioning of $\mathcal{H}(F)$, J_1, \dots, J_k gives a partitioning of $\mathcal{V}(F)$, and with the bijection $\varphi : \{I_1, \dots, I_k\} \longrightarrow \{J_1, \dots, J_k\}$ where $I_i \mapsto J_i$ ($i = 1, \dots, k$), $I_i \sim_{\mathcal{S}} \varphi(I_i)$ also holds. \square

Remark 3.1. The partitionings defined in Proposition 3.2 determine the minimal bounding rectangles $I_i \times \varphi(I_i)$ ($i = 1, \dots, k$) of the disjoint components belonging to the class \mathcal{S} of a binary image F' with $\mathcal{H}(F') = \mathcal{H}(F)$ and $\mathcal{V}(F') = \mathcal{V}(F)$. However, $F' = F$ not necessarily holds, i.e. uniqueness of the solution is not always guaranteed. In particular, it is also possible that the original binary image F does not have disjoint components. Figure 4 shows three different binary images (one with non-disjoint components) which have the same horizontal and vertical projections. For all three images the partitionings $I_1 = [1, 1]$, $I_2 = [2, 3]$, and $J_1 = [2, 2]$, $J_2 = [1, 1]$,

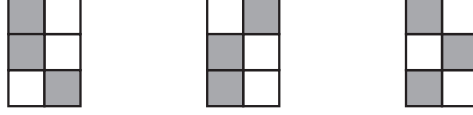


Fig. 4. Two different binary images with disjoint components having the same projections (left and center) and a third one with the same projections which has non-disjoint components (right).

and the mapping $\varphi(I_1) = J_1$, $\varphi(I_2) = J_2$ satisfy the conditions of Proposition 3.2 since those conditions depend on the projections, exclusively.

On the basis of Proposition 3.2, to reconstruct a set with disjoint components from the class \mathcal{S} and with the horizontal and vertical projections H and V , respectively, it is sufficient to find the proper partitionings of H and V , and the appropriate mapping between them, whenever the reconstruction in the class \mathcal{S} can be performed from two projections. This can be achieved by a backtracking algorithm which builds a tree where each node is labeled with a list containing 4-tuples of the form $[i_1, i_2, j_1, j_2]$ such that $[i_1, i_2] \sim_{\mathcal{S}} [j_1, j_2]$. We define the classical lexicographical order \prec over these 4-tuples. Moreover, for an arbitrary node p with the label $\langle [i_{11}, i_{21}, j_{11}, j_{21}], [i_{12}, i_{22}, j_{12}, j_{22}], \dots, [i_{1r}, i_{2r}, j_{1r}, j_{2r}] \rangle$ we denote

$$S_H(p) = \bigcup_{l=1}^r [i_{1l}, i_{2l}] , \quad \text{and} \quad S_V(p) = \bigcup_{l=1}^r [j_{1l}, j_{2l}] . \quad (4)$$

For the backtracking we need to define the classical tree operators *Father*, *Brother*, and *Son*. The *Father* of $\langle l_1, l_2, \dots, l_r \rangle$ is *nil* if $r = 1$, otherwise it is $\langle l_1, l_2, \dots, l_{r-1} \rangle$. The *Brother* of $\langle l_1, l_2, \dots, l_{r-1}, [i_{1r}, i_{2r}, j_{1r}, j_{2r}] \rangle$ is a list of the form $\langle l_1, l_2, \dots, l_{r-1}, [i_{1r}, i'_{2r}, j'_{1r}, j'_{2r}] \rangle$ where $[i_{1r}, i'_{2r}, j'_{1r}, j'_{2r}]$ is the smallest element according to the ordering \prec such that

- $[i_{1r}, i_{2r}, j_{1r}, j_{2r}] \prec [i_{1r}, i'_{2r}, j'_{1r}, j'_{2r}]$, and
- if $r > 1$ then $S_V(\langle l_1, l_2, \dots, l_{r-1} \rangle) \cap [j'_{1r}, j'_{2r}] = \emptyset$

if such an element exists (otherwise the *Brother* points to *nil*). Finally, the *Son* of $\langle l_1, l_2, \dots, l_{r-1}, [i_{1r}, i_{2r}, j_{1r}, j_{2r}] \rangle$ is *nil* if $i_{2r} = m$. Otherwise, it is $\langle l_1, l_2, \dots, l_r, [i'_{1r}, i'_{2r}, j'_{1r}, j'_{2r}] \rangle$ where $[i'_{1r}, i'_{2r}, j'_{1r}, j'_{2r}]$ is the smallest element according to the ordering \prec such that

- $i'_{1r} = i_{2r} + 1$, and
- $S_V(\langle l_1, l_2, \dots, l_r \rangle) \cap [j'_{1r}, j'_{2r}] = \emptyset$

if such an element exists (otherwise the *Son* points to *nil*). A node p with $Son(p) = nil$ is called leaf. Our suggested reconstruction framework takes two vectors, $H \in \mathbb{N}^m$ and $V \in \mathbb{N}^n$ as input, and the class \mathcal{S} of polyominoes from which the components must arise should be also specified. We first give an outline of the algorithm.

Algorithm DisjointRec

Step 1 Find and store all compatible interval-pairs $[i_1, i_2]$ of H and $[j_1, j_2]$ of V ;

Step 2 Starting out from the first row connect interval-pairs such that

- (a) the rows of the intervals of H are consecutive and disjoint, and
- (b) the columns of the intervals of V are disjoint;

We now go into technical details on how the algorithm works. In Step 1 it identifies all the compatible intervals $[i_1, i_2]$ of H and $[j_1, j_2]$ of V w.r.t. the class \mathcal{S} . The number of intervals of H is $\binom{m}{2}$ and similarly, the number of intervals of V is $\binom{n}{2}$. Therefore there are $\frac{m(m-1)n(n-1)}{4}$ interval-pairs of H and V . For each interval-pair – to prove compatibility – we first check whether the necessary conditions of Proposition 3.1 are satisfied. If so, then – as a final condition of compatibility – we try to construct a polyomino $P \in \mathcal{S}$ with the corresponding projections determined by the intervals. The time complexity of this step depends on the class \mathcal{S} from which the components arise. For example, if \mathcal{S} is the class of hv -convex polyominoes then an element of \mathcal{S} of size $k \times l$ can be reconstructed in $O(kl \cdot \min\{k^2, l^2\})$ time¹¹ while in the class of hv -convex directed polyominoes the reconstruction complexity is of $O(kl)$.¹⁷ The compatible intervals can be stored in a lexicographical order without further computation in a 4-dimensional array in the form $[i_1, i_2, j_1, j_2]$. In addition, we also store the polyominoes corresponding to the intervals as they possibly will serve as the components of the solution. Furthermore, for each $i = 1, \dots, m$ we define a pointer to the smallest element of the array which has i as the first component. This yields fast execution of the tree operators.

In Step 2 we explore the tree in a preorder way using the tree operators starting out from the node which has the smallest label according to the ordering \prec . We stop the search if there are no unvisited nodes left (in this case there is no solution of the given reconstruction problem) or if we find a leaf p with $S_H(p) = \{1, \dots, m\}$ (which corresponds to a solution). In the described form the algorithm finds just one solution of the given reconstruction problem but it can be modified in a straightforward way to find all the solutions, assuming that it is possible to reconstruct all the elements of \mathcal{S} having the same two projections. In both cases it can happen that the tree built by our algorithm will have an exponential number of nodes causing the running time of this step to be exponential.

Example 3.1. Consider that $H = (1, 2, 1, 1, 2)$, $V = (1, 2, 3, 1)$, and \mathcal{S} is the class of directed polyominoes. In this case the array used by the algorithm stores the following elements in lexicographical order

$$\begin{array}{cccc}
 [1, 1, 1, 1] & [2, 3, 1, 2] & [3, 3, 1, 1] & [4, 4, 1, 1] \\
 [1, 1, 4, 4] & [2, 4, 3, 4] & [3, 3, 4, 4] & [4, 4, 4, 4] \\
 [1, 2, 1, 2] & & [3, 4, 2, 2] & [4, 5, 1, 2] \\
 [1, 3, 3, 4] & & [3, 5, 3, 4] &
 \end{array} \quad (5)$$

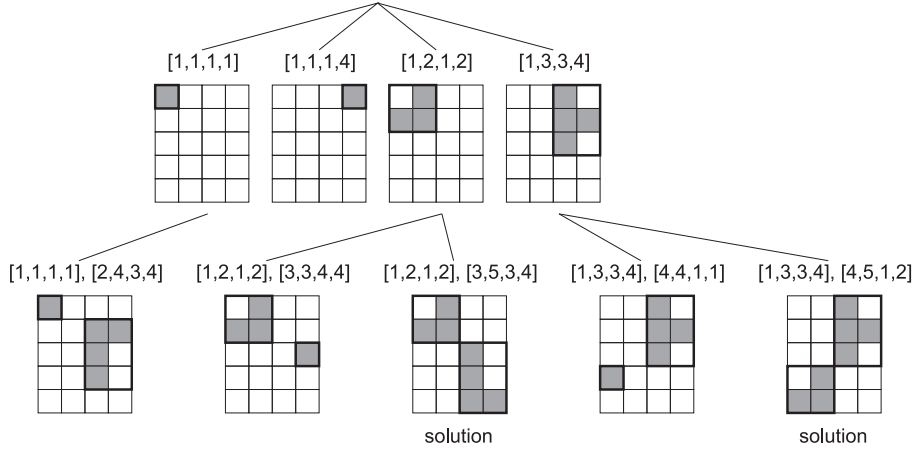


Fig. 5. A search tree built by the reconstruction algorithm.

The tree built by the algorithm and the solutions of the reconstruction task are presented in Fig. 5. The algorithm finds a solution at node $\langle [1, 2, 1, 2], [3, 5, 3, 4] \rangle$, and an other one – after the whole search tree had been scanned – at node $\langle [1, 3, 3, 4], [4, 5, 1, 2] \rangle$.

The following theorem proves the correctness of the algorithm.

Theorem 3.1. *Every leaf p of the search tree built by Algorithm DisjointRec for which $S_H(p) = \{1, \dots, m\}$ corresponds to at least one solution of the corresponding reconstruction problem. If the reconstruction in the class \mathcal{S} is uniquely determined from the horizontal and vertical projections then the correspondence is one-to-one.*

Proof. Let p be a leaf for which $S_H(p) = \{1, \dots, m\}$, and

$$\langle [i_{11}, i_{21}, j_{11}, j_{21}], [i_{12}, i_{22}, j_{12}, j_{22}], \dots, [i_{1l}, i_{2l}, j_{1l}, j_{2l}] \rangle \quad (6)$$

be its label. The definition of operator *Son* guaranties that the intervals $[i_{1r}, i_{2r}]$ ($r = 1, \dots, l$) are pairwise disjoint. From the definition of the operators *Son* and *Brother* it also follows that the intervals $[j_{1r}, j_{2r}]$ ($r = 1, \dots, l$) are pairwise disjoint as well. Moreover, $[i_{1r}, i_{2r}] \sim_{\mathcal{S}} [j_{1r}, j_{2r}]$, therefore $H_{[i_{1r}, i_{2r}]} = V_{[j_{1r}, j_{2r}]}$ ($r = 1, \dots, l$). Then $S_V(p) = \{1, \dots, n\}$ since

$$H_{[1, m]} = \sum_{r=1}^l H_{[i_{1r}, i_{2r}]} = \sum_{r=1}^l V_{[j_{1r}, j_{2r}]} = V_{[1, n]}. \quad (7)$$

Furthermore, there exists a binary image with disjoint components from the class \mathcal{S} such that the minimal bounding rectangles of the components are $[i_{1r}, i_{2r}] \times [j_{1r}, j_{2r}]$ ($r = 1, \dots, l$). Those components are reconstructed in Step 1.

From the above points it follows that the minimal bounding rectangles of the components of the binary image corresponding to p are uniquely determined, i.e., the size of all the components and their relative positions are fixed. There might be differences at component-level, but it is possible only if two different polyominoes of \mathcal{S} have the same projections. \square

Discrete tomographic reconstruction plays an important role in certain data compression and data security tasks, too (see, e.g. Ref. 21 and the references given there). As a direct consequence of Theorem 3.1 we obtain a result for the compact representation of some binary images having disjoint components.

Corollary 3.1. *Let F be a binary image of size $n \times n$ having disjoint components which are uniquely reconstructible from their projections in the class \mathcal{S} in $O(f(n))$ time. Then F can be represented in a compressed form on $O(\frac{\log n}{n})$ bits and the decompression time is $O(f(n))$.*

Proof. The size of the original binary image F is of $n \times n$ therefore n^2 bits are needed to describe all its elements directly. If the components are uniquely reconstructible from their horizontal and vertical projections then, on the basis of Theorem 3.1 the image itself is uniquely determined by the minimal bounding rectangles of its components and the horizontal and vertical projections. The components are disjoint therefore there are at most n of them. The collection of the minimal bounding rectangles of the components can be represented, say, by the bottom left corners of each rectangle. For this we need $O(n \log n)$ bits. Finally, the horizontal and vertical projections can be stored on $2n \log n$ bits. Thus the first statement follows. If the minimal bounding rectangles are known then the task of the decomposition is simply to reconstruct the components in the corresponding parts of the image, which takes $O(f(n))$ time. \square

Recall that for hv -convex polyominoes $f(n) = n^4$ while for hv -convex directed polyominoes $f(n) = n^2$.

4. Experimental Results

In order to test the effectiveness of our algorithm we conducted two experiments. In the first one, we designed 20 collections of binary images with disjoint components. The components always arose from the class of hv -convex directed polyominoes. We decided to choose this class since its elements are uniquely determined by the horizontal and vertical projections²², and they can be reconstructed from those projections in $O(mn)$ time.¹⁷ Thus, it was possible to find not just one, but all the solutions of a given reconstruction problem. Each collection consisted of 100 binary images having the same size (20×20 , 40×40 , 60×60 , 80×80 , and 100×100) and a fixed number (2, 3, 4, and 5) of disjoint components (see Fig. 6 for an example). So in total we had $5 \cdot 4 = 20$ collections and $20 \cdot 100 = 2000$ images. The images

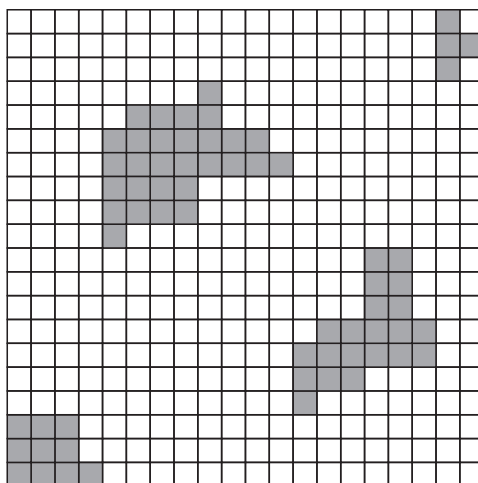


Fig. 6. A test image of size 20×20 having four components.

of each collection were chosen from uniform random distributions by applying the method described in Ref. 23. The programs were written in C++ and the tests were performed on an Intel Pentium 4 CPU with 3.2 GHz and 1GB RAM under Debian/GNULinux (kernel 2.6.17.13).

The first two columns of Table 1 represent the size and the number of components of the binary test images. The remaining columns show the average number of solutions (third column), and the running times of finding the first (fourth column) and all solutions (fifth column) in seconds rounded to four digits. From this table we can observe that if the set is of moderate size and it consists of just a few components then our algorithm finds the solution quite fast. Moreover, the average number of solutions is close to 1 in almost all studied cases, i.e., the reconstruction is usually uniquely determined. In addition, finding all solutions of the given reconstruction task usually does not take much more time than just finding one solution. The explanation of this can be that the most amount of the execution time is due to the preprocessing step of the algorithm. It becomes even more evident if we take a closer look at the average depth and the number of nodes (both rounded to two digits) of the search tree built for the several collections of the images (sixth and seventh column of Table 1, respectively). Unfortunately, the running time of our algorithm (mostly due to the preprocessing step) grows rapidly as the number of components and/or the size of the image increases (see also Fig. 7).

In our second experiment we tested the performance of our algorithm on more complex images having 8, 9, and 10 components. We used here 6 collections of 100-100 images. For each collection, the size of the images and the number of their components are presented in the first and second column of Table 2, respectively. In this test we set a time limit of 30 seconds for the reconstruction. The third

Table 1. Reconstruction results for binary images with few components.

Size	# Comp.	# Sol.	First sol. (sec)	All sol. (sec)	Depth	Nodes
20 × 20	2	1.13	0.0006	0.0006	2.32	18.95
20 × 20	3	1.42	0.0007	0.0008	3.25	44.78
20 × 20	4	2.41	0.0020	0.0019	4.42	9570.15
20 × 20	5	8.61	0.0014	0.0025	5.54	3258.98
40 × 40	2	1.03	0.0081	0.0081	2.23	29.87
40 × 40	3	1.22	0.0094	0.0094	3.29	56.76
40 × 40	4	1.45	0.0104	0.0104	4.39	43711.70
40 × 40	5	2.98	0.0145	0.0147	5.44	6653.58
60 × 60	2	1.03	0.0378	0.0377	2.15	10.77
60 × 60	3	1.14	0.0448	0.0445	3.43	757.49
60 × 60	4	1.24	0.0485	0.0485	4.23	95.95
60 × 60	5	2.02	0.0580	0.0582	5.54	9444.93
80 × 80	2	1.02	0.1421	0.1421	2.15	12.03
80 × 80	3	1.12	0.1512	0.1513	3.36	104.59
80 × 80	4	1.20	0.1707	0.1710	4.38	304.80
80 × 80	5	1.80	0.1936	0.1948	5.72	13925.90
100 × 100	2	1.01	0.3847	0.3814	2.19	25.32
100 × 100	3	1.01	0.4209	0.4206	3.27	196.48
100 × 100	4	1.12	0.4561	0.4562	4.29	376.95
100 × 100	5	1.24	0.5038	0.5016	5.44	1118.99

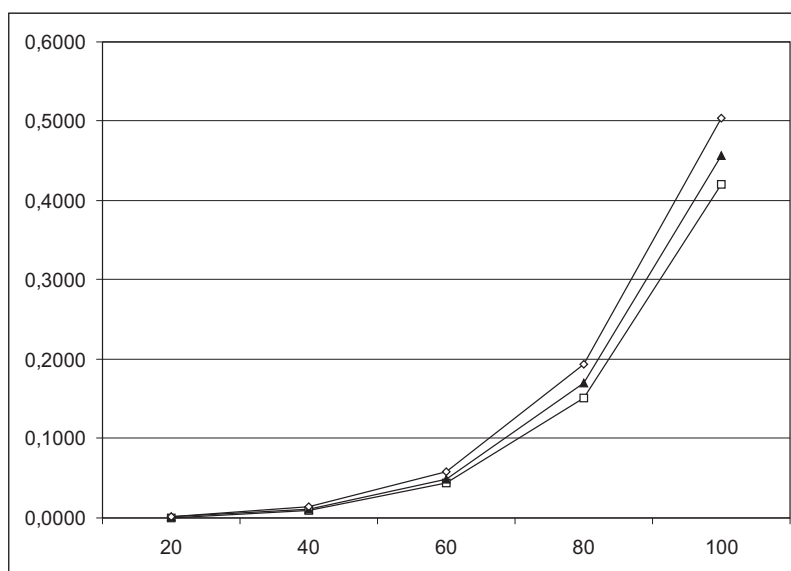


Fig. 7. The average running time of the algorithm (vertical axis) as it depends on the size of the image (horizontal axis) if it has three (□), four (▲), and five (◇) components.

Table 2. Reconstruction results for more complex binary images.

Size	Components	Success	Avg. solutions	First solution (sec)	All solutions (sec)
80×80	8	99	10.18	0.33	0.60
80×80	9	96	11.90	0.82	1.53
80×80	10	93	228.68	0.45	1.38
100×100	8	100	7.97	0.90	0.96
100×100	9	96	15.58	1.09	1.25
100×100	10	93	40.41	1.56	2.01

column of Table 2 reports the number of reconstructions which ran in less than 30 seconds out of the 100 images of the given collection. The remaining columns show the average number of solutions, and the reconstruction times for finding one, and all solutions, from left to right, respectively, omitting the cases where the reconstruction have not been finished in 30 seconds. From Table 2 we can again deduce that the reconstruction of more complex sets (i.e. sets with more components) takes more time and gives usually more ambiguous solutions.

5. Pruning with Shape Information

There are two main drawbacks of the algorithm. First, for more complex images the running time increases. On the other hand, the reconstruction task is often very underdetermined due to small number of projections, and thus many (very different) images with the same two projections may exist. Fortunately, it is easy to incorporate some further knowledge into the framework. Obviously, such information yields an improvement on the accuracy. In the same time, with an effective pruning of the search tree the reconstruction can be speeded up, too. Table 3 presents the results obtained on the same data sets as in our second experiment but pruning the search tree with the prior information that the set consists of more than 7, and less than 11 components. For an extreme example, in the second experiment we found an image of size 80×80 having 10 components for whose projections 16204 (!) different images could be reconstructed, all having the same projections. By using the information that the set has 8-10 components the number of solutions could be reduced to 748. While the additional information usually also had a positive effect on the running time for finding the first solution, it can occur that finding a solution that satisfies more constraints can take more time (see again Table 2 for a comparison, as well).

The approximate number of components is just a simple knowledge that can be useful in the reconstruction. In the followings we describe in more general how information about the shape of the expected image may facilitate the reconstruction. We will outline a branch-and-bound algorithm to find the solution (or solutions) that optimally fits the prior information. We here only will consider maximization problems (minimization problems can be treated in the same way by multiplying the objective function by -1). Assume that we want to find the binary image with

Table 3. Reconstruction results for more complex binary images by pruning the search tree with the knowledge about the number of components.

Size	Components	Success	Avg. solutions	First solution (sec)	All solutions (sec)
80 × 80	8	99	9.75	0.33	0.56
80 × 80	9	97	10.80	0.73	1.30
80 × 80	10	95	38.20	0.63	1.34
100 × 100	8	100	7.78	0.83	0.89
100 × 100	9	97	14.56	0.98	1.11
100 × 100	10	95	26.11	1.45	1.84

disjoint components that has the given horizontal and vertical projections and satisfies some additional shape priors as much as possible. For the sake of convenience we introduce the following notations. Let \mathcal{D} denote the class of binary images with disjoint components and $f : \mathcal{D} \rightarrow \mathbb{R}_0^+$ be the image function that assigns a non-negative real number to each binary image representing how much a given image satisfies the shape priors. For the pruning we introduce the function Φ which assigns a positive real value to each node of the search tree built by Algorithm DisjointRec. Denoting the set of tree nodes by \mathcal{T} the function $\Phi : \mathcal{T} \rightarrow \mathbb{R}_0^+$ has to satisfy the following criteria:

- (α) for every inner node $t \in \mathcal{T}$ and an arbitrary node s in the subtree rooted by t $f(s) \leq \Phi(t)$, and
- (β) if t is a leaf corresponding to a solution then $\Phi(t) = f(t)$.

In this way Φ defines an upper bound on f . In addition, we globally store the best solution (i.e. a solution with the maximal value f_{max} of f) found so far (initially f_{max} is set to be zero). If $\Phi(t) < f_{max}$ for an actual node t then the subtree rooted by t can not contain solutions for which the function f can take greater value than f_{max} , therefore this subtree can be pruned.

A wide range of shape descriptors can be used in this framework, possibly in combination as well. We concentrate here on one important application of this approach. In several practical reconstruction tasks there is a natural information that the consecutive slides do not differ from each others (recall that 3D reconstruction is often done slice-by-slice). Then, a previously reconstructed slice can be regarded as a model for the reconstruction of the forthcoming one (this is the case for example in biplane-angiography). The same situation occurs when a blueprint image is available in applications of non-destructive testing. Although the reconstruction of a binary image from two projections that has the most common black pixels with a given image is a well-known polynomial-time problem²⁴, the task becomes difficult if additional constraints are also present. Therefore we will study this problem in more detail. Our aim is to find a binary image with disjoint components which has the maximal number of common black pixels with a given model image (which itself not necessarily has disjoint components and/or not even the same projections as

the binary image we are searching for). The similarity of the reconstructed image R and the model image M will therefore be defined by

$$f_M(R) = \text{card}(\{(i, j) \mid r_{ij} = m_{ij} = 1\}) . \quad (8)$$

Now, let $t \in \mathcal{T}$ be a node of the search tree built by Algorithm DisjointRec having the label $\langle [i_{11}, i_{21}, j_{11}, j_{21}], [i_{12}, i_{22}, j_{12}, j_{22}], \dots, [i_{1r}, i_{2r}, j_{1r}, j_{2r}] \rangle$ and let R be the binary image corresponding to t . Define $\Phi_M(R) = 0$ if t is the root of the tree (i.e., the corresponding binary image is empty) and otherwise by

$$\begin{aligned} \Phi_M(R) = & \text{card}(\{(i, j) \mid (i, j) \in U \wedge r_{ij} = m_{ij} = 1\}) + \\ & + \text{card}(\{(i, j) \mid (i, j) \notin U \wedge m_{ij} = 1\}) , \end{aligned} \quad (9)$$

where

$$U = [1, i_{2r}] \times (\cup_{l=1}^r [j_{1l}, j_{2l}]) . \quad (10)$$

The following lemma shows that the functions defined by (8) and (9) can be used to correctly prune the tree.

Lemma 5.1. *The function-pair $f_M(R)$ and $\Phi_M(R)$ defined by (8) and (9), respectively, satisfy properties (α) and (β) .*

Proof. Let t be an arbitrary node of the tree and R be the corresponding binary image. Then the set U defined by (10) covers exactly the rows and the columns of the components of R . We know that $\{(i, j) \mid (i, j) \notin U \wedge r_{ij} = m_{ij} = 1\} \subseteq \{(i, j) \mid (i, j) \notin U \wedge m_{ij} = 1\}$. Then $f_M(R) = \text{card}(\{(i, j) \mid r_{ij} = m_{ij} = 1\}) = \text{card}(\{(i, j) \mid (i, j) \in U \wedge r_{ij} = m_{ij} = 1\}) + \text{card}(\{(i, j) \mid (i, j) \notin U \wedge r_{ij} = m_{ij} = 1\}) \leq \text{card}(\{(i, j) \mid (i, j) \in U \wedge r_{ij} = m_{ij} = 1\}) + \text{card}(\{(i, j) \mid (i, j) \notin U \wedge m_{ij} = 1\}) = \Phi_M(R)$. Thus $f_M(R) \leq \Phi_M(R)$. In the special case when t is a leaf corresponding to a solution the set U covers all rows and columns. Then $\text{card}(\{(i, j) \mid (i, j) \notin U \wedge m_{ij} = 1\}) = 0$ thus $f_M(R) = \Phi_M(R)$ and property (β) immediately follows.

Our basic assumption is that the components of the reconstructed image have to be disjoint. Then it follows that an arbitrary image produced from R by Algorithm DisjointRec – by adding one or more components to R – can have no components in U (except those which are already present in R as well). This additionally means that for each such image the number of matching black pixels related to those of R can only increase by at most $\text{card}(\{(i, j) \mid (i, j) \notin U \wedge m_{ij} = 1\})$ yielding that property (α) also holds. \square

Let us see on an example how the reconstruction based on a model image can reduce the search space and increase accuracy.

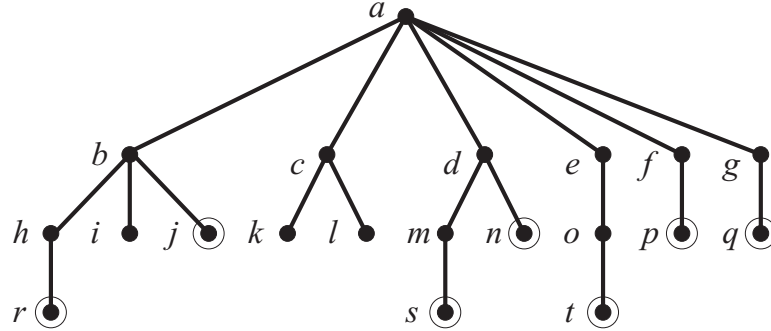


Fig. 8. The search tree built by the execution of the task given in Example 5.1. Circles mark nodes corresponding to solutions of the reconstruction task.

Example 5.1. Let $H = (1, 2, 2, 2, 1)$, $V = (1, 2, 2, 2, 1)$, and \mathcal{S} be the class of directed polyominoes. In this case the interval-pairs stored by the algorithm are

$$\begin{array}{l}
 [1, 1, 1, 1] \quad [2, 3, 2, 3] \quad [3, 4, 2, 3] \quad [4, 5, 4, 5] \quad [5, 5, 1, 1] \\
 [1, 1, 5, 5] \quad [2, 3, 3, 4] \quad [3, 4, 3, 4] \quad [5, 5, 5, 5] \\
 [1, 2, 1, 2] \quad [2, 5, 2, 5] \quad [3, 5, 3, 5] \\
 [1, 2, 4, 5] \\
 [1, 3, 1, 3] \\
 [1, 4, 1, 4]
 \end{array} \quad . \quad (11)$$

The tree built by the algorithm is presented in Fig. 8. The labels of the nodes are $a : \langle \rangle$, $b : \langle [1, 1, 1, 1] \rangle$, $c : \langle [1, 1, 5, 5] \rangle$, $d : \langle [1, 2, 1, 2] \rangle$, $e : \langle [1, 2, 4, 5] \rangle$, $f : \langle [1, 3, 1, 3] \rangle$, $g : \langle [1, 4, 1, 4] \rangle$, $h : \langle [1, 1, 1, 1], [2, 3, 2, 3] \rangle$, $i : \langle [1, 1, 1, 1], [2, 3, 3, 4] \rangle$, $j : \langle [1, 1, 1, 1], [2, 5, 2, 5] \rangle$, $k : \langle [1, 1, 5, 5], [2, 3, 2, 3] \rangle$, $l : \langle [1, 1, 5, 5], [2, 5, 2, 5] \rangle$, $m : \langle [1, 2, 1, 2], [3, 4, 3, 4] \rangle$, $n : \langle [1, 2, 1, 2], [3, 5, 3, 5] \rangle$, $o : \langle [1, 2, 4, 5], [3, 4, 2, 3] \rangle$, $p : \langle [1, 3, 1, 3], [4, 5, 4, 5] \rangle$, $q : \langle [1, 4, 1, 4], [5, 5, 5, 5] \rangle$, $r : \langle [1, 1, 1, 1], [2, 3, 2, 3], [4, 5, 4, 5] \rangle$, $s : \langle [1, 2, 1, 2], [3, 4, 3, 4], [5, 5, 5, 5] \rangle$, and $t : \langle [1, 2, 4, 5], [3, 4, 2, 3], [5, 5, 1, 1] \rangle$. The seven different solutions of the reconstruction task are shown in Fig. 9(a)-(g).

Now, consider that we are interested in finding the solution that is the most similar to the model image given in Fig. 9(h). With the aid of the functions defined in (8) and (9) it is possible to prune some nodes of the search tree. Figure 10 shows the effect of the pruning. Four out of the 20 nodes do not need to be visited (the pruned branches are drawn by dashed arcs) since the upper bounds defined by the function Φ are smaller at the root of those subtrees as the value of f_{max} assigned to the best solution found so far. The solution that fits best to the model shape is the image at node q having six common elements with the model ($f_{max} = 6$).

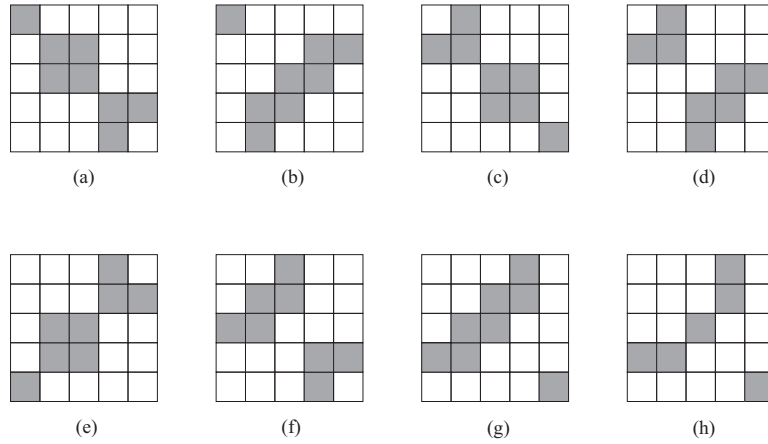


Fig. 9. The seven different solutions of the reconstruction task given in Example 5.1, corresponding to the nodes r , j , s , n , t , p , and q (from (a) to (g), respectively) of the tree in Fig. 8, and a model image (h) which is used for pruning the tree.

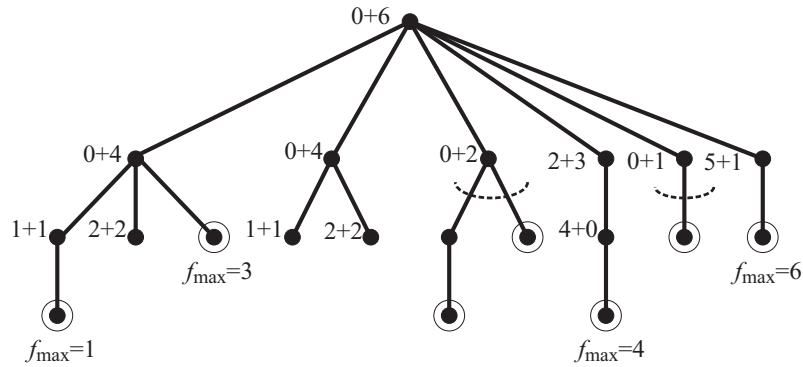


Fig. 10. The tree obtained in Example 5.1 by pruning the tree of Fig. 8 using the model image of Fig. 9(h). The corresponding values of the function Φ are also presented at each node.

6. Conclusions

We have developed a general backtracking algorithm to reconstruct binary images with disjoint components from the horizontal and vertical projections. It is easy to incorporate a priori knowledge about the components into our reconstruction framework. Since every hv -convex set has disjoint components, it is also possible to apply the presented method to reconstruct hv -convex binary images from two projections which problem is proved to be NP-hard in general (see Ref. 25) but in certain special cases it can be solved in polynomial time (see, e.g., Refs. 10, 11, 12,

14, 15, 17).

In our experiments we investigated a subclass of *hv*-convex images and assumed that the components are directed. The algorithm can find all solutions of a given reconstruction task, and experimental results show that the method is fast and accurate if the number of components is small related to the size of the image. For more complex images (i.e. ones which has relatively many components) we found that the running time of the algorithm increases considerably, and the size of the search tree built by the algorithm becomes in some cases intractably large. We also presented a simple pruning technique that exploits further information on the number of components in order to reduce the search space and gain more accurate reconstructions.

The pruning technique can be generalized, and the presented backtracking algorithm can be extended to a branch-and-bound method to reduce the search space. In this way not only the accuracy of the reconstruction can be increased but also the execution time can be decreased. A large variety of shape informations (e.g., the approximate size and/or shape of the components, or the relative positions of them) known before the reconstruction can be exploited in this way, and one can search for the image that satisfies those shape descriptions optimally. As an example, we studied the problem of reconstructing a binary image most similar to a given model image.

The speed of the branch-and-bound method (and the number of pruned branches) strongly depends on the ordering how the feasible solutions are reached during the search. If a relatively good solution can be found at an early stage of the process then possibly more subtrees can be pruned, yielding faster reconstruction. It needs further investigation how this observation can be exploited to speed up our branch-and-bound technique.

We also must note that the applications reported in the introduction usually do not satisfy the requirements of our algorithm, therefore new reconstruction methodologies should be developed in the near future for real data. In our further work we intend to study the reconstruction of binary images having disjoint components when more than two projections are available. Possible extensions of the concept of disjointness for non-orthogonal directions and for higher dimensions form also parts of our further research.

Acknowledgments

This contribution was supported by grant OTKA T048476. A preliminary version of this work was presented at the International Symposium on Visual Computing, Las Vegas, NV, 2008, Special Track on Discrete and Computational Geometry and their Applications in Visual Computing.²⁶ The author of this paper would like to thank the reviewers for their valuable suggestions.

References

1. A.C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging* (IEEE Press, New York, 1988).
2. K.J. Batenburg and W.J. Palenstijn, On the reconstruction of crystals through discrete tomography, *Lecture Notes in Comput. Sci.* **3322** (2004) 23–37.
3. J.M. Carazo, C.O. Sorzano, E. Rietzel, R. Schröder and R. Marabini, Discrete tomography in electron microscopy, In [7], pp. 405–416 (1999).
4. B.M. Carvalho, G.T. Herman, S. Matej, C. Salzberg and E. Vardi, Binary tomography for triplane cardiography, *Lecture Notes in Comput. Sci.* **1613** (1999) 29–41.
5. D.G.W. Onnasch and G.M.P. Prause, Heart chamber reconstruction from biplane angiography, In [7], pp. 385–403 (1999).
6. J. Baumann, Z. Kiss, S. Krimmel, A. Kuba, A. Nagy, L. Rodek, B. Schillinger and J. Stephan, Discrete tomography methods for non-destructive testing, In [8], pp. 303–332 (2007).
7. G.T. Herman and A. Kuba (eds.), *Discrete Tomography: Foundations, Algorithms and Applications* (Birkhäuser, Boston, 1999).
8. G.T. Herman and A. Kuba (eds.), *Advances in Discrete Tomography and its Applications* (Birkhäuser, Boston, 2007).
9. H.J. Ryser, Combinatorial properties of matrices of zeros and ones, *Canad. J. Math.* **9** (1957) 371–377.
10. P. Balázs, E. Balogh and A. Kuba, Reconstruction of 8-connected but not 4-connected $h\nu$ -convex discrete sets, *Disc. Appl. Math.* **147** (2005) 149–168.
11. E. Balogh, A. Kuba, Cs. Dévényi and A. Del Lungo, Comparison of algorithms for reconstructing $h\nu$ -convex discrete sets, *Lin. Alg. Appl.* **339** (2001) 23–35.
12. E. Barucci, A. Del Lungo, M. Nivat and R. Pinzani, Medians of polyominoes: A property for the reconstruction, *Int. J. Imaging Systems and Techn.* **9** (1998) 69–77.
13. S. Brunetti and A. Daurat, An algorithm reconstructing convex lattice sets, *Theor. Comput. Sci.* **304** (2003) 35–57.
14. S. Brunetti, A. Del Lungo, F. Del Ristoro, A. Kuba and M. Nivat, Reconstruction of 4- and 8-connected convex discrete sets from row and column projections, *Lin. Alg. Appl.* **339** (2001) 37–57.
15. M. Chrobak and Ch. Dürr, Reconstructing $h\nu$ -convex polyominoes from orthogonal projections, *Inform. Process. Lett.* **69(6)** (1999) 283–289.
16. A. Kuba, The reconstruction of two-directionally connected binary patterns from their two orthogonal projections, *Comp. Vision, Graphics, and Image Proc.* **27** (1984) 249–265.
17. A. Kuba and E. Balogh, Reconstruction of convex 2D discrete sets in polynomial time, *Theor. Comput. Sci.* **283** (2002) 223–242.
18. E. Barucci, A. Del Lungo, M. Nivat and R. Pinzani, Reconstructing convex polyominoes from horizontal and vertical projections, *Theor. Comput. Sci.* **155** (1996) 321–347.
19. P. Balázs, On the ambiguity of reconstructing $h\nu$ -convex binary matrices with decomposable configurations, *Acta Cybernetica* **18(3)** (2008) 367–377.
20. P. Balázs, A decomposition technique for reconstructing discrete sets from four projections, *Image and Vision Computing* **25(10)** (2007) 1609–1619.
21. R.J. Gardner, P. Gritzmann and D. Prangenberg, On the computational complexity of reconstructing lattice sets from their X-rays, *Discrete Math.* **202** (1999) 45–71.
22. A. Del Lungo, Polyominoes defined by two vectors, *Theor. Comput. Sci.* **127** (1994) 187–198.
23. P. Balázs, A framework for generating some discrete sets with disjoint components

- by using uniform distributions, *Theor. Comput. Sci.* **406** (2008) 15–23.
24. K.J. Batenburg, An evolutionary algorithm for discrete tomography, *Disc. Appl. Math.* **151** (2005) 36–54.
 25. G.W. Woeginger, The reconstruction of polyominoes from their orthogonal projections, *Inform. Process. Lett.* **77** (2001) 225–229.
 26. P. Balázs, Reconstruction of binary images with few disjoint components from two projections, in *Advances in Visual Computing, Proc. 4th Int. Symp. Visual Computing*, eds. G. Bebis et al., LNCS Vol. 5359 (Springer, Berlin, 2008) pp. 1147–1156.