

Reconstruction of Binary Images with Few Disjoint Components from Two Projections

Péter Balázs*

Department of Image Processing and Computer Graphics
University of Szeged
Árpád tér 2., H-6720, Szeged, Hungary
pbalazs@inf.u-szeged.hu

Abstract. We present a general framework for reconstructing binary images with few 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 information about the components of the image to be reconstructed can be incorporated into the reconstruction process. In addition, we can keep control over the number of components which can increase the speed and accuracy of the reconstruction. Experimental results are also presented.

1 Introduction

Reconstruction tomography is an imaging procedure for obtaining the two dimensional cross-sections of three dimensional objects by using projections of the object being studied. Reconstruction algorithms have a wide area of applications in non-destructive testing, angiography, radiology, crystallography, image processing, and so on. In most of those applications the reconstructed image should contain only a small number of grey-intensities that are known beforehand. In the same time, there is often a practical limitation that only a few (usually at most about 10) projections of the object can be made. *Discrete tomography* (DT) [10,11] investigates how the knowledge that the reconstruction should only contain a few values can be exploited to eliminate the problems arising from using a small number of available projections. *Binary tomography* (BT) arises as a special case of DT, where the task is to reconstruct a binary image (also called discrete set). BT also has a variety of applications, for example, in angiography, electron microscopy, and non-destructive testing.

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. In this paper we present a general framework for reconstructing discrete sets from two projections. The prior assumption we use this time is that the discrete set to be reconstructed should consist of few disjoint components.

* This work was supported by OTKA grant T048476.

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. Finally, in Sect. 5 we summarize our experiences.

2 Preliminaries

The finite subsets of the two-dimensional integer lattice are called *discrete sets*. A discrete set is defined up to a translation and it can be represented by a binary image or a binary matrix as well (see Fig. 1). The *horizontal* and *vertical* projections of a discrete set F are the vectors $\mathcal{H}(F) = (h_1, \dots, h_m)$, and $\mathcal{V}(F) = (v_1, \dots, v_n)$, respectively, where

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

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

For example, the discrete set F in Fig. 1 has the horizontal and vertical projections $\mathcal{H}(F) = (1, 3, 1, 3, 4, 3)$, and $\mathcal{V}(F) = (2, 3, 4, 2, 2, 2)$, respectively. In the rest of the paper (without losing generality) we always assume that discrete sets do not have empty rows and columns.

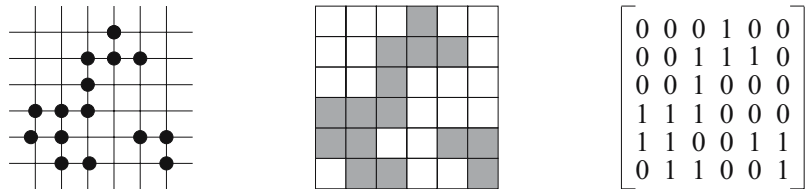


Fig. 1. A discrete set represented by its elements (left), a binary image (center), and a binary matrix (left)

In the reconstruction two vectors, H and V are given and the task is to construct a discrete set such that $\mathcal{H}(F) = H$ and $\mathcal{V}(F) = V$. To reduce the number of possible solutions of this task we are usually interested in reconstructing discrete sets belonging to a certain class defined by some geometrical properties like connectedness or convexity [3,4,5,6,7,8,12,13].

Two positions $P = (p_1, p_2)$ and $Q = (q_1, q_2)$ in a discrete set are said to be *4-adjacent* if $|p_1 - q_1| + |p_2 - q_2| = 1$. A discrete set F is *4-connected* (with an other term *polyomino*) if for any two points P and Q in F there is a sequence of distinct positions $P_0 = P, \dots, P_k = Q$ in the discrete set F such that P_l is 4-adjacent to P_{l-1} , for each $l = 1, \dots, k$. The discrete set F is *hv-convex* if all the rows and columns of F are 4-connected. An *hv-convex polyomino* P is

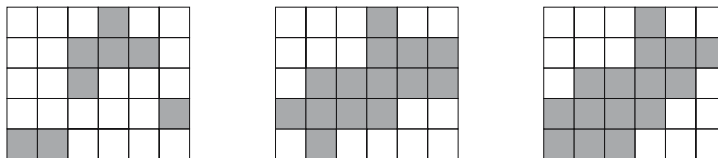


Fig. 2. An *hv*-convex discrete set (left), an *hv*-convex polyomino (center), and an *hv*-convex directed polyomino (right)

called *directed* if the point $(m, 1)$ belongs to P . Figure 2 shows some examples of *hv*-convex discrete sets.

A discrete set can always be partitioned into maximal 4-connected subsets, in a uniquely determined way. Those partitions are called the *components* of the discrete set. Clearly, if the discrete set is a polyomino then it has just one component. Let F be a discrete set 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 $J_l \cap J_{l'} = \emptyset$. Note that this statement is stronger than just saying that the minimal bounding rectangles of the components are pairwise disjoint.

3 The Reconstruction Framework

We develop our reconstruction framework in such a way that it will be easy to incorporate a priori knowledge of the components into the reconstruction process. Thus, we will always assume that the components belong to a certain class \mathcal{S} of polyominoes. When no prior knowledge is available then we simply assume that \mathcal{S} is the whole class of polyominoes.

We introduce a notation. For each $1 \leq i_1 \leq i_2 \leq m$ and $1 \leq j_1 \leq j_2 \leq n$ let

$$H_{[i_1, i_2]} = \sum_{i=i_1}^{i_2} h_i, \quad \text{and} \quad V_{[j_1, j_2]} = \sum_{j=j_1}^{j_2} v_j. \tag{3}$$

If $i_1 = 1$ or $j_1 = 1$ then - as special cases - we obtain the cumulated vectors of F which are used in several reconstruction algorithms [3,4,5,7]. We will refer to the partial sums defined in (3) as the sums of the intervals $[i_1, i_2]$ of the vector H and $[j_1, j_2]$ of the vector V .

Definition 1. 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} , if there exists a polyomino $P \in \mathcal{S}$ with $\mathcal{H}(P) = (h_{i_1}, \dots, h_{i_2})$ and $\mathcal{V}(P) = (v_{j_1}, \dots, v_{j_2})$.

For the sake of simplicity we will denote $[i_1, i_2] \sim_{\mathcal{S}} [j_1, j_2]$ if $[i_1, i_2]$ of H and $[j_1, j_2]$ of V are compatible, w.r.t. the class \mathcal{S} . Obviously, if $[i_1, i_2] \sim_{\mathcal{S}} [j_1, j_2]$ for an arbitrary class \mathcal{S} then the followings hold simultaneously

- (i) $i \leq j_2 - j_1 + 1$ for each $i_1 \leq i \leq i_2$,
- (ii) $j \leq i_2 - i_1 + 1$ for each $j_1 \leq j \leq j_2$,
- (iii) $H^{[i_1, i_2]} = V^{[j_1, j_2]}$.

We will exploit this necessary condition to make our algorithm faster. The following proposition offers the basis of our algorithm.

Proposition 1. *Let F be a discrete set 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 $\varphi : \{I_1, \dots, I_k\} \rightarrow \{J_1, \dots, J_k\}$ such that $I_i \sim_{\mathcal{S}} \varphi(I_i)$ ($i = 1, \dots, k$).*

Proof. Let F be a discrete set 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 smallest containing discrete 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 $\varphi : \{I_1, \dots, I_k\} \rightarrow \{J_1, \dots, J_k\}$ where $I_i \mapsto J_i$ ($i = 1, \dots, k$), $I_i \sim_{\mathcal{S}} \varphi(I_i)$ also holds. □

Remark 1. The partitionings defined in Proposition 1 determine the smallest containing discrete rectangles $I_i \times \varphi(I_i)$ ($i = 1, \dots, k$) of the disjoint components belonging to the class \mathcal{S} of a set F' with $\mathcal{H}(F') = \mathcal{H}(F)$ and $\mathcal{V}(F') = \mathcal{V}(F)$. However, $F' = F$ not necessarily holds. For example, if $F = \{(1, 1), (2, 1), (3, 2)\}$, $I_1 = [1, 1]$, $I_2 = [2, 3]$, $\varphi(I_1) = J_1 = [2, 2]$, and $\varphi(I_2) = J_2 = [1, 1]$, then $F' = \{(1, 2), (2, 1), (3, 1)\}$ (here \mathcal{S} can be the class of all polyominoes).

Remark 2. The counterpart of Proposition 1 does not always hold. Let $F = \{(1, 2), (2, 1), (3, 2)\}$. Clearly, the components of F are not disjoint. However, there exists a partitioning $I_1 = [1, 1], I_2 = [2, 3]$ of $\mathcal{H}(F)$ and a partitioning $J_1 = [2, 2], J_2 = [1, 1]$ of $\mathcal{V}(F)$ such that $I_1 \sim_{\mathcal{S}} J_1$ and $I_2 \sim_{\mathcal{S}} J_2$ (with \mathcal{S} being the class of all polyominoes).

On the basis of Proposition 1, 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 labelled 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 point 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

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

For the backtracking we need the following tree operators.

Father. It is the simplest operator. The *Father* of $\langle l_1, l_2, \dots, l_r \rangle$ is the empty list $\langle \rangle$ if $r = 1$, otherwise it is $\langle l_1, l_2, \dots, l_{r-1} \rangle$.

Brother. 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*.

Son. The *Son* of $\langle l_1, l_2, \dots, l_{r-1}, [i_{1r}, i_{2r}, j_{1r}, j_{2r}] \rangle$ points to *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 tree node p which son points to *nil* is called leaf.

Our framework is called DisjointReconstruction. We now give some technical details on how it works. It takes two vectors, $H \in \mathbb{N}^m$ and $V \in \mathbb{N}^n$ as input, and the class \mathcal{S} from which the components arise must be also specified. First the algorithm 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} which takes $O(m^2n^2)$ iterations. To prove compatibility of the intervals it is necessary to construct a polyomino $P \in \mathcal{S}$ with the corresponding projections determined by the intervals. This modul of the framework must be set according to the class \mathcal{S} . Its execution can be time-consuming, depending on the reconstruction complexity in the class \mathcal{S} . Therefore we execute it only if the actual pair of intervals satisfies the necessary conditions (i), (ii), and (iii) of compatibility. 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.

After the preprocessing step 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 unreached 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). Without proof we mention here that for certain classes this step can be performed in polynomial, while for others in superpolynomial time. The following theorem proves the correctness of the algorithm.

Theorem 1. *Every leaf p of the search tree built by Algorithm DisjointReconstruction 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

$$\langle [i_{11}, i_{21}, j_{11}, j_{21}], [i_{12}, i_{22}, j_{12}, j_{22}], \dots, [i_{1l}, i_{2l}, j_{1l}, j_{2l}] \rangle \quad (5)$$

be its label. The definition of operator *Son* guarantees 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

$$H_{[1, m]} = \sum_{r=1}^l H_{[i_{1r}, i_{2r}]} = \sum_{r=1}^l V_{[j_{1r}, j_{2r}]} = V_{[1, n]}. \quad (6)$$

Furthermore, there exists a discrete set with disjoint components from the class \mathcal{S} such that the smallest containing discrete rectangles of the components are $[i_{1r}, i_{2r}] \times [j_{1r}, j_{2r}]$ ($r = 1, \dots, l$). Those components are reconstructed in the preprocessing step.

From the above points it follows that the smallest containing discrete rectangles of the components of the discrete set 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

The algorithm in this form 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.

Example 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 $[1, 1] \sim_{\mathcal{S}} [1, 1]$, $[1, 1] \sim_{\mathcal{S}} [4, 4]$, $[1, 2] \sim_{\mathcal{S}} [1, 2]$, $[1, 3] \sim_{\mathcal{S}} [3, 4]$, $[2, 3] \sim_{\mathcal{S}} [1, 2]$, $[2, 4] \sim_{\mathcal{S}} [3, 4]$, $[3, 3] \sim_{\mathcal{S}} [1, 1]$, $[3, 3] \sim_{\mathcal{S}} [4, 4]$, $[3, 4] \sim_{\mathcal{S}} [2, 2]$, $[3, 5] \sim_{\mathcal{S}} [3, 4]$, $[4, 4] \sim_{\mathcal{S}} [1, 1]$, $[4, 4] \sim_{\mathcal{S}} [4, 4]$, $[4, 5] \sim_{\mathcal{S}} [1, 2]$. The other intervals violate one or more of the necessary properties (i), (ii), and (iii) of compatibility, or there is no polyomino with the projections determined by those intervals. The array used by the algorithm stores the following elements in lexicographical order: $[1, 1, 1, 1]$, $[1, 1, 4, 4]$, $[1, 2, 1, 2]$, $[1, 3, 3, 4]$, $[2, 3, 1, 2]$, $[2, 4, 3, 4]$, $[3, 3, 1, 1]$, $[3, 3, 4, 4]$, $[3, 4, 2, 2]$, $[3, 5, 3, 4]$, $[4, 4, 1, 1]$, $[4, 4, 4, 4]$, $[4, 5, 1, 2]$. The tree built by the algorithm and the solutions of the reconstruction task are presented in Fig. 3. 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 $\langle [1, 3, 3, 4], [4, 5, 1, 2] \rangle$.

4 Experimental Results

In order to test the effectiveness of our algorithm we conducted some experiments. In each cases we assumed that the components of the set to be reconstructed are *hv*-convex directed polyominoes. We decided to choose this class

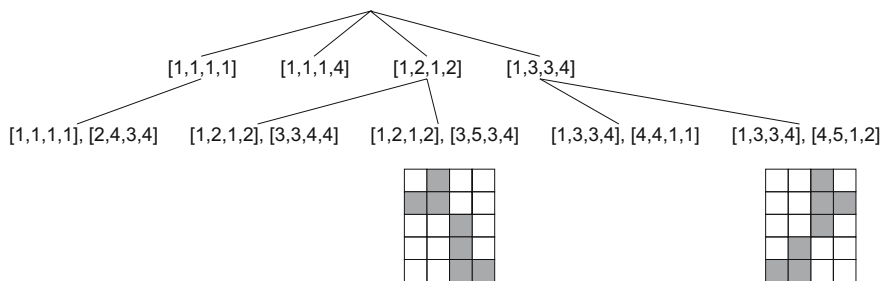


Fig. 3. A search tree built by Algorithm DisjointReconstruction

since its elements are uniquely determined by the horizontal and vertical projections [9], and they can be reconstructed from those projections in $O(mn)$ time [13]. Thus, it was possible to find not just one, but all the solutions of a given reconstruction problem. Each test data set consisted of 100 binary images having the same size and a fixed number of disjoint components. Those images were generated with uniform random distributions by applying the method described in [1]. 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 column of Table 1 represents the structure of the test data sets used in our first experiment. It describes the size of the squared binary test images (only one dimension is given) followed by the number of components of the binary images. The remaining columns show the average number of solutions (second column), and the running times of finding the first (third column) and all (fourth column) solutions 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 often uniquely determined. In addition, there is usually no significant difference regarding the execution time of finding just one or all the solutions. The explanation of this can be that the most amount of the execution time is due to the preprocessing step of the algorithm. Unfortunately, the running time grows rapidly as the number of components and/or the size of the set increases. Furthermore, the more components the image has, the more likely will the reconstruction be ambiguous.

In our second experiment we tested the performance of our algorithm on more complex sets having 8, 9, and 10 components. In this test we set a time limit of 30 seconds for the reconstruction. The second column of Table 2 reports the number of reconstructions that ran in less than 30 seconds out of the 100 datas. 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 has not 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

Table 1. Reconstruction results for sets with few components

size/#comp.	#solutions	first (s)	all (s)
20/2	1.13	0.0006	0.0006
20/3	1.42	0.0007	0.0008
20/4	2.41	0.0020	0.0019
20/5	8.61	0.0014	0.0025
40/2	1.03	0.0081	0.0081
40/3	1.22	0.0094	0.0094
40/4	1.45	0.0104	0.0104
40/5	2.98	0.0145	0.0147
60/2	1.03	0.0378	0.0377
60/3	1.14	0.0448	0.0445
60/4	1.24	0.0485	0.0485
60/5	2.02	0.0580	0.0582
80/2	1.02	0.1421	0.1421
80/3	1.12	0.1512	0.1513
80/4	1.20	0.1707	0.1710
80/5	1.80	0.1936	0.1948
100/2	1.01	0.3847	0.3814
100/3	1.01	0.4209	0.4206
100/4	1.12	0.4561	0.4562
100/5	1.24	0.5038	0.5016

Table 2. Reconstruction results for more complex sets with (bottom half of the table) and without (top half of the table) prior information

size/#comp.	success	#solutions	first (s)	all (s)
80/8	99	10.18	0.33	0.60
80/9	96	11.90	0.82	1.53
80/10	93	228.68	0.45	1.38
100/8	100	7.97	0.90	0.96
100/9	96	15.58	1.09	1.25
100/10	93	40.41	1.56	2.01
80/8	99	9.75	0.33	0.56
80/9	97	10.80	0.73	1.30
80/10	95	38.2	0.63	1.34
100/8	100	7.78	0.83	0.89
100/9	97	14.56	0.98	1.11
100/10	95	26.11	1.45	1.84

more ambiguous solutions. Fortunately, it is easy to incorporate some further knowledge into the framework. The bottom half of Table 2 presents the results obtained on the same data sets as before but pruning the search tree with the prior information that the image consists of more than 7 and less than 11 components. Obviously, this a priori information yielded an improvement both on

the accuracy and the overall speed of the reconstruction as well. For an extreme example, in the test set 80/10 we found a task with 16204 (!) solutions, 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. The additional information usually also had a positive effect on the running time for finding the first solution. However, it can occur that finding a solution that satisfies more constraints can take more time than just finding one for which the conditions not necessarily hold (see third and ninth rows of the table).

Even if we do not have explicit knowledge about the number of the components of the set to be reconstructed, in some cases we can gain such statistical information directly from the size of the discrete set – as it was presented in [2].

5 Conclusions

We have developed a general backtracking algorithm to reconstruct discrete sets with few disjoint components from the horizontal and vertical projections. It is easy to incorporate a priori knowledge about the components into our reconstruction framework. We conducted experiments to test the speed and accuracy of our algorithm under several circumstances, assuming that the components of the set to be reconstructed are directed. Experimental results prove that the algorithm is fast and accurate if the number of the components are relatively small. If the binary image to be reconstructed is more complex (i.e. it has more components) then 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.

There have been many reconstruction algorithms developed for the class of *hv*-convex sets and its subclasses (see, e.g., [3,4,5,7,8,12,13]). Since every *hv*-convex set has disjoint components, it is also possible to apply the presented method to reconstruct *hv*-convex discrete sets from two projections, which problem is known to be NP-hard in general [14].

The algorithm can also be adapted to exploit further information about the set to be reconstructed, e.g., the size or the approximate shape of the components, the relative positions of them, or if some positions of the set are fixed in advance. An other extension would be to reduce the search space by using more than just two projections for the reconstruction. In general, it needs further investigation how the search tree can be pruned in an appropriate way for given reconstruction tasks.

Our framework can be extended to 3 dimensional reconstruction problems in a straightforward way.

References

1. Balázs, P.: A framework for generating some discrete sets with disjoint components by using uniform distributions. *Theor. Comput. Sci.* (2008), doi:10.1016/j.tcs.2008.06.010

2. Balázs, P.: On the number of hv -convex discrete sets. In: Brimkov, V.E., Barneva, R.P., Hauptman, H.A. (eds.) IWCIA 2008. LNCS, vol. 4958, pp. 112–123. Springer, Heidelberg (2008)
3. Balázs, P., Balogh, E., Kuba, A.: Reconstruction of 8-connected but not 4-connected hv -convex discrete sets. *Disc. Appl. Math.* 147, 149–168 (2005)
4. Balogh, E., Kuba, A., Dévényi, C., Del Lungo, A.: Comparison of algorithms for reconstructing hv -convex discrete sets. *Lin. Alg. Appl.* 339, 23–35 (2001)
5. Barucci, E., Del Lungo, A., Nivat, M., Pinzani, R.: Medians of polyominoes: A property for the reconstruction. *Int. J. Imaging Systems and Techn.* 9, 69–77 (1998)
6. Brunetti, S., Daurat, A.: An algorithm reconstructing convex lattice sets. *Theor. Comput. Sci.* 304, 35–57 (2003)
7. Brunetti, S., Del Lungo, A., Del Ristoro, F., Kuba, A., Nivat, M.: Reconstruction of 4- and 8-connected convex discrete sets from row and column projections. *Lin. Alg. Appl.* 339, 37–57 (2001)
8. Chrobak, M., Dürr, C.: Reconstructing hv -convex polyominoes from orthogonal projections. *Inform. Process. Lett.* 69(6), 283–289 (1999)
9. Del Lungo, A.: Polyominoes defined by two vectors. *Theor. Comput. Sci.* 127, 187–198 (1994)
10. Herman, G.T., Kuba, A. (eds.): *Discrete Tomography: Foundations, Algorithms and Applications*. Birkhäuser, Boston (1999)
11. Herman, G.T., Kuba, A. (eds.): *Advances in Discrete Tomography and its Applications*. Birkhäuser, Boston (2007)
12. Kuba, A.: The reconstruction of two-directionally connected binary patterns from their two orthogonal projections. *Comp. Vision, Graphics, and Image Proc.* 27, 249–265 (1984)
13. Kuba, A., Balogh, E.: Reconstruction of convex 2D discrete sets in polynomial time. *Theor. Comput. Sci.* 283, 223–242 (2002)
14. Woeginger, G.W.: The reconstruction of polyominoes from their orthogonal projections. *Inform. Process. Lett.* 77, 225–229 (2001)