Acta Cybernetica **21** (2013) 149–163.

An Empirical Study of Reconstructing hv-Convex Binary Matrices from Horizontal and Vertical Projections*

Zoltán Ozsvár† and Péter Balázs‡

Abstract

The reconstruction of hv-convex binary matrices (or equivalently, binary images) from their horizontal and vertical projections is proved to be NP-hard. In this paper we take a closer look at the difficulty of the problem. We investigate different heuristic reconstruction algorithms of the class, and compare them from the viewpoint of running-time and reconstruction quality. Using a large set of test images of different sizes and with varying number of components, we show that the reconstruction quality can depend not only on the size of the image, but on the number and location of its components, too. We also reveal that the reconstruction time can also be affected by the number of the so-called switching components present in the image.

Keywords: discrete tomography, reconstruction algorithm, hv-convex binary matrix, kernel-shell method, simulated annealing

1 Introduction

Tomography is a method of producing a 3D image of the internal structure of an object from its projections, without damaging it. This is usually achieved by reconstructing 2D slices from the projections and then assembling them. Applications of computerized tomography arise from various fields of science: image processing, medical imaging, nondestructive testing, electron microscopy, etc. The Filtered

*This work was supported by the European Union and co-funded by the European Social Fund under the grant agreement TÁMOP-4.2.2/B-10/1-2010-0012, "Broadening the knowledge base and supporting the long term professional sustainability of the Research University Centre of Excellence at the University of Szeged by ensuring the rising generation of excellent scientists". The work of Péter Balázs was also supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and the OTKA PD 100950 grant of the Hungarian Scientific Research Fund.

†Faculty of Science and Informatics, University of Szeged, Aradi vértanúk tere 1., H-6720 Szeged, Hungary, E-mail: Ozsvar.Zoltan@stud.u-szeged.hu

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

DOI: 10.14232/actacyb.21.1.2013.11

Backprojection and variants of the Algebraic Reconstruction Methods are the most commonly used algorithms to reconstruct images from their projections [8, 12]. However, they require several hundreds of projections to ensure an acceptable image quality. In *Binary Tomography* (BT) we assume that the examined object is homogeneous, thus the image to be reconstructed contains only black (object) and white (background) pixels. With this additional information, algorithms of BT can often produce images of good quality, even from just a small amount (say, at most 10-20) of projections [9, 10]. The reconstruction of a binary image is also feasible from just its horizontal and vertical projections [15]. Nevertheless, in that case the task is usually extremely underdetermined, i.e., there may be numerous different binary images with the same two projections. To overcome this problem, we can assume that the image satisfies certain geometrical conditions, too. In this paper we study the reconstruction of hv-convex binary images from the horizontal and vertical projections, from an experimental point of view. We perform several tests to compare different algorithms for reconstructing hv-convex binary images. The structure of the paper is the following. Chapter 2 is for the preliminaries. In Chapter 3 we describe three heuristical algorithms for solving the abovementioned task. In Section 4 we give our experimental results. Finally, Section 5 is for the conclusion.

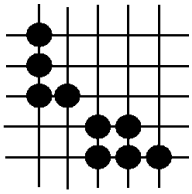
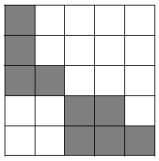
2 Preliminaries

A *binary image* is a digital image containing just black (also called as object or foreground) and white (background) pixels. A binary image of size $m \times n$ (where $m, n \in \mathbb{N}$) can also be represented by a binary matrix $F = (f_{ij})_{m \times n}$ where value 1 (respectively, value 0) indicates that the color of the corresponding pixel is black (respectively, white). *Discrete sets* (finite subsets of the 2D integer lattice $\mathbb{Z}^2$) can also be used to represent a binary image, with the agreement that the rows are numbered from top to bottom (and the columns are numbered from left to right). A position of the lattice belongs to the discrete set if and only if the corresponding matrix position has value 1. In this paper we will use the three concepts equivalently. Since rows/columns with 0 projection value can be reconstructed easily in a preprocessing step and then be eliminated, we also will assume that each row and column of the matrix contains at least one 1. Figure 1 shows the three different representations of the same binary image.

The *horizontal* and *vertical projection* of the image F is the vector $\mathcal{H}(F) = H = (h_1, \ldots, h_m)$ and $\mathcal{V}(F) = V = (v_1, \ldots, v_n)$, respectively, where

$$h_i = \sum_{j=1}^{n} f_{ij} \ \ (i = 1, \ldots, m) \quad \text{and} \quad v_j = \sum_{i=1}^{m} f_{ij} \ \ (j = 1, \ldots, n). \tag{1}$$

For example, the binary image F in Fig. 1 has the horizontal and vertical projection $\mathcal{H}(F) = (1, 1, 2, 2, 3)$, and $\mathcal{V}(F) = (3, 1, 2, 2, 1)$, respectively. The class of all binary matrices having the horizontal projection H, and vertical projection V will be denoted by $\mathcal{BM}(H, V)$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Figure 1: Different representations of the same object as a discrete set, as a binary image, and as a binary matrix (from left to right).

A *switching component* of a matrix is a 2×2 submatrix of the form

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2}$$

To avoid confusion, we emphasize that the above submatrix is formed by selecting two (not necessarily consecutive) rows and two (not necessarily consecutive) columns of the matrix. It is easy to see that the horizontal and vertical projections of a matrix do not change if we invert the values in the four positions of a switching component. A much stronger statement is also true.

Theorem 1. *[15] Let $A, B \in \mathcal{BM}(H, V)$ $(B \neq A)$. Then A is transformable into B (or vice versa) by a finite sequence of switching components.*

The *reconstruction task* consists in finding a binary image with the given horizontal and vertical projections. However, due to the presence of switching components, those projections usually do not uniquely determine the image itself. Thus, further prior information is needed in order to reduce the number of possible solutions. Two positions $P = (p_1, p_2)$ and $Q = (q_1, q_2)$ in a discrete set F are said to be *4-adjacent* if $|p_1 - q_1| + |p_2 - q_2| = 1$. The positions P and Q are *4-connected* if there is a sequence of distinct positions $P_0, \ldots, P_k$ in F such that $P_0 = P$, $P_k = Q$, and P_l is 4-adjacent to P_{l-1} (for each $l = 1, \ldots, k$). A discrete set F is *4-connected* if any two points in F are 4-connected. Every discrete set F can be partitioned (in a uniquely determined way) into maximal 4-connected subsets, which are called the *components* of F. The discrete set is called *h-convex* (respectively, *v-convex*) if its elements follow consecutively in each row (respectively, in each column). The discrete set is called *hv-convex* if it is both h- and v-convex. Figure 2 demonstrates the above concepts.

3 Algorithms for Reconstructing hv-Convex Binary Images

It is known, that the reconstruction of general hv-convex discrete sets from their horizontal and vertical projections is an NP-hard problem [16]. On the other hand,

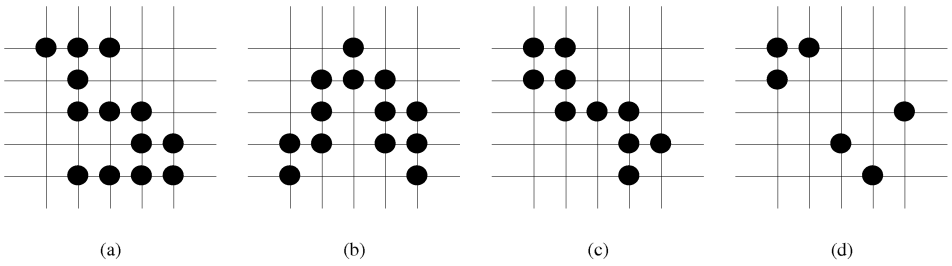


Figure 2: An h-convex 4-connected discrete set (a), a v-convex 4-connected discrete set (b), an hv-convex 4-connected discrete set (c), and a general hv-convex discrete set with 4 components (d).

if the hv-convex discrete set is also 4-connected, then the reconstruction can be solved in polynomial time [4, 5, 6]. Our goal is to reveal the reason of being the more general problem computationally hard. In this section, we shortly describe three heuristic algorithms from previous works to solve the problem.

3.1 Kernel-Shell Algorithm

The kernel-shell (or core-envelope) algorithm (see Algorithm 1) is a greedy type heuristic algorithm which approximates the discrete set F to be reconstructed by two sequences of discrete sets [13]. The first sequence is nondecreasing, and it consists of the so-called *core sets* which satisfy

$$C_0 \subseteq C_1 \subseteq \cdots \subseteq F, \tag{3}$$

while the second (nonincreasing) sequence of so-called *envelope sets* satisfies

$$S_0 \supseteq S_1 \supseteq \cdots \supseteq F. \tag{4}$$

Denoting the minimal bounding rectangle of F by T, as initial core and envelope sets we can use

$$C_0 = \emptyset \quad \text{and} \quad S_0 = T. \tag{5}$$

In every iteration, the new core set is constructed as the maximal hv-convex discrete set (operator J) from the current core set C_k, extending it to horizontal and vertical directions by taking into account the horizontal and vertical projection values (operator c) and the current envelope set S_k (Step 2). Similarly, the new shell is constructed as the intersection of the maximal possible extensions of the current core C_k, by taking into account the horizontal and vertical projection values (operator s) and the current envelope set S_k (Step 3).

If the kernel and the shell coincide, then we found a solution (Steps 4-6). Otherwise, if $C = C_k$ (i.e., the core set does not change in an iteration), then the core cannot be further increased. Thus, we use a stack memory P, and guess all the

positions of the shell not belonging to the kernel whether the core can be extended with that given position (Steps 7-9). This heuristic step might be repeated (if needed) several times (Steps 10-11). If we get that $C_k \not\subseteq S_k$, then the guess led to a contradiction, and we do a backtrack step (by deleting the tested position from the shell) if possible, i.e., if the stack is not empty (Steps 15-18). Otherwise, there is no solution (Steps 13-14).

Algorithm 1 Core-Shell Algorithm

Input: the vectors $H \in \mathbb{N}^m$ and $V \in \mathbb{N}^n$

Output: the binary matrix F with $\mathcal{H}(F) = H$, $\mathcal{V}(F) = V$ or the message "no solution"

1: $C_0 = \emptyset, S_0 = T, k = 0$, P is an empty stack
2: $C := J(c(S_k, H) \cup c(S_k, V) \cup C_k)$
3: $S := s(C_k, H) \cap s(C_k, V) \cap S_k$
4: **if** $C_k = S_k$ **then**
5: **return** F
6: **end if**
7: **if** $C = C_k$ **then**
8: $(C, S, ((i,j) \in S \backslash C) \to P); C := C \cup \{(i,j)\}; C_{k+1} := C; S_{k+1} := S; k := k+1$
9: **end if**
10: **if** $C_k \subset S_k$ **then**
11: **goto** Step 2
12: **else**
13: **if** $C_k \not\subseteq S_k$ and P is empty **then**
14: FAIL (no solution)
15: **else**
16: $P \to (C, S, (i,j)); C_{k+1} := C; S_{k+1} = S_k \setminus \{(i,j)\}; k := k+1$; **goto** Step 2
17: **end if**
18: **end if**

3.2 Algorithm Based on Simulated Annealing

The next algorithm is based on Simulated Annealing (SA) [14] where the objective is to maximize the number of adjacent ones in an unknown binary matrix $X = (x_{ij})_{m \times n}$, i.e., the following function

$$f(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n} x_{ij} x_{i+1,j} + \sum_{i=1}^{m} \sum_{j=1}^{n-1} x_{ij} x_{i,j+1} \tag{6}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = h_j \ (i = 1, \ldots, m) \tag{7}$$

$$\sum_{i=1}^{n} x_{ij} = v_j \ (j = 1, \ldots, n) \tag{8}$$

with $x_{i,j} \in \{0, 1\}$ ($i = 1, \ldots, m$ and $j = 1, \ldots, n$). Constraints (7) and (8) ensure that the matrix X satisfies the given projections.

Algorithm 2 outlines this method that was published in [11] on the basis of [7]. For generating an initial solution that satisfies the projections, Ryser's algorithm is used [15]. The neighborhood of a solution matrix is defined as the set of all matrices obtained by a single switching. As Theorem 1 says, by applying such switchings, all the binary matrices satisfying the given projections can be constructed. In Step 3, $p \in (0, 1)$ is a random variable, generated in each iteration from a uniform random distribution. The initial temperature was set to 5, and the algorithm was terminated when the best solution did not improve for the last 1000 iterations or when the temperature reached 0.0005. Those parameters as well as the cooling factor were set empirically by a long process of trial and error.

Algorithm 2 Algorithm Based on Simulated Annealing

Input: X_{act} = computed initial solution, $Temp = 5$, $Nbr = 0$
Output: the binary matrix X with $\mathcal{H}(X) = H$ and $\mathcal{V}(X) = V$

1: **while** ($Temp > 0.0005$ and $Nbr < 1000$) **do**
2: $X_{next} \longleftarrow$ invert a randomly chosen switching component in X_{act}
3: **if** ($f(X_{act}) < f(X_{next})$) or ($p < exp(-|f(X_{act}) - f(X_{next})|/Temp)$ **then**
4: $X_{act} \leftarrow X_{next}$; $Temp := Temp \cdot 0.9995$; $Nbr := 0$;
5: **else**
6: Nbr++;
7: **end if**
8: **end while**

3.3 Algorithm Based on the Location of the Components

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 ($l = 1, \ldots, k$). We say that the components of F are *disjoint* if for any $l \neq l'$ (where $1 \leq l, l' \leq k$) $I_l \cap I_{l'} = \emptyset$ and $J_l \cap J_{l'} = \emptyset$ (see Fig. 3).

In [1] the author presented an algorithm to reconstruct discrete sets having disjoint components from their horizontal and vertical projections, by locating the possible positions of the components. It is clear, that the hv-convex images naturally consist of disjoint components. Moreover, those components are hv-convex 4-connected images, which can be reconstructed in polynomial time ([4, 5, 6]). Thus, this algorithm is also capable of the fast reconstruction of hv-convex images. The outline of the method is given as Algorithm 3, which uses the following definition.

Definition 1. *Let $\mathcal{S}$ be a class of 4-connected binary images, $H \in \mathbb{N}^m$ 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 ($1 \leq j_1 \leq$*

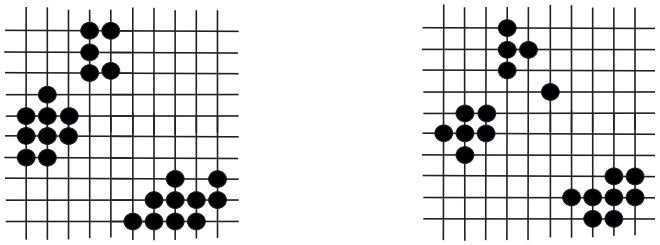


Figure 3: A discrete set with three disjoint components (left), and an hv-convex discrete set with four disjoint components (right).

$j_2 \leq n$*) are* compatible with respect to the class $\mathcal{S}$ *if a 4-connected binary image* $P \in \mathcal{S}$ *exits with* $\mathcal{H}(P) = (h_{i_1}, \ldots, h_{i_2})$ *and* $\mathcal{V}(P) = (v_{j_1}, \ldots, v_{j_2})$.

Algorithm 3 Algorithm Based on the Location of the Components

1: find and store all compatible interval-pairs $[i_1, i_2]$ of H and $[j_1, j_2]$ of V w.r.t. the class of hv-convex 4-connected binary images;
2: starting out from the first row connect interval-pairs found in Step 1, until all the components of the image are found, such that
a) the rows of the intervals of H are consecutive and pairwisely disjoint, and
b) the columns of the intervals of V are pairwisely disjoint;

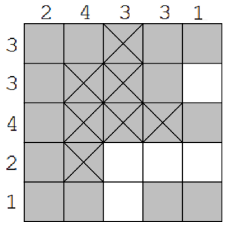
4 Experimental Results

4.1 Implementation Details

We implemented the algorithms of Section 3 in order to study their performance from the viewpoint of running-time and reconstruction quality.

The kernel-shell algorithm was implemented with two different data structures. We designed an *array data type* where we represented the set of core and shell positions by two two-dimensional arrays of size $m \times n$. The core positions were marked by 1s, and the shell points by 2s. The second data structure (called *first-last type*) stored the first and the last elements of the core and the shell in each row and each column (or alternatively -1, if there was no element of the set in the given row or column). This structure was suggested in [13] and it uses four one-dimensional arrays of size m and another four one-dimensional arrays of size n. The two data structures are presented in Fig. 4.

The two variants of the kernel-shell heuristic, the algorithm based on Simulated Annealing and the method based on the location of the components were implemented in JAVA, and the test run on an AMD Athlon X2, 2.1 GHz with 2GB RAM under Ubuntu 11.01. In the experiments we used a large set of hv-convex images,



$$\text{core}=\begin{bmatrix}0&0&1&0&0\\0&1&1&0&0\\0&1&1&1&0\\0&1&0&0&0\\0&0&0&0&0\end{bmatrix}\quad \text{shell}=\begin{bmatrix}2&2&2&2&2\\2&2&2&2&0\\2&2&2&2&2\\2&2&0&0&0\\2&2&0&2&2\end{bmatrix}$$

Core:	Shell:
RowFirst: 3,2,2,2,-1	RowFirst: 1,1,1,1,1
RowLast: 3,3,4,2,-1	RowLast: 5,4,5,2,5
ColumnFirst: -1,2,1,3,-1	ColumnFirst: 1,1,1,1,1
ColumnLast: -1,4,3,3,-1	ColumnLast: 5,5,3,5,5

Figure 4: Example image, where × represents the core and the gray pixels form the shell. The state is represented both by the array and the first-last data types.

with different sizes (from 10×10 to 50×50) and varying number of components (from 1 to 4), by picking them from uniform random distributions, using the methods of [2, 3].* For each size and number of components the data set contained 100 images. Since the algorithm based on Simulated Annealing uses random values, for each test data we repeated the reconstruction 5 times, and took the average speed and correctness of the five runs. In the following we present results just for a part of the test images, but we made the same observations by investigating the entire data set.

4.2 The Quality of the Reconstructions

First, we studied the quality of the reconstructions. Even if the image to be reconstructed is hv-convex, there can be a significant difference between two images having the same horizontal and vertical projections, due to the presence of the switching components (see, e.g., Fig. 5). To measure the error of reconstruction, we computed the conventionally used RME (relative mean error) [17] with the formula

$$RME = \frac{\sum_{i,j} |m^o_{i,j} - m^r_{i,j}|}{\sum_{i,j} m^o_{i,j}} \ , \tag{9}$$

where $M^o = (m^o_{i,j})$ is the original binary matrix (the expected image), and $M^r = (m^r_{i,j})$ is the binary matrix of the reconstruction.

The mean value and the variance of the RME for some of the data sets are given in Table 1. Note that the reconstruction quality of the kernel-shell method does not depend on the applied data structure. We can observe that – due to the fact that

* We used the benchmark collection available at `http://www.inf.u-szeged.hu/~pbalazs/benchmark/benchmark.html`

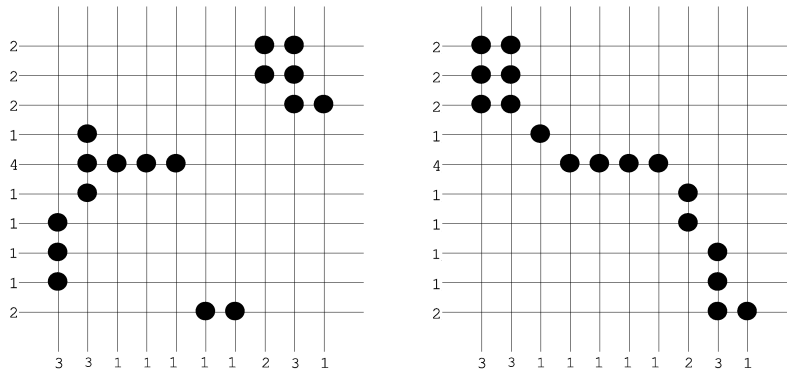


Figure 5: Two different hv-convex discrete sets with the same projections.

the SA algorithm does not guarantee hv-convexity – the quality of its results are usually worse than those of the other two methods. The core-shell method usually outperforms the algorithm based on the location of the components, although not significantly. However, all algorithms show an increasing trend in the RME value (yielding worse and worse quality) with the increasing number of components of the image. We found that images consisting of fewer components have usually significantly fewer switching components than those with more components, which could be an explanation for this trend.

Table 1: Quality of the reconstructions in RME as a function of the number of components (first column), and the size of the image (horizontally)

Core-Shell - Array/First-last data type										
	10×10		20×20		30×30		40×40		50×50	
	mean	variance	mean	variance	mean	variance	mean	variance	mean	variance
1	0.0181	0.0918	0.0027	0.0128	0.0002	0.0017	0.0002	0.0017	0.0001	0.0008
2	0.1005	0.3577	0.0138	0.0406	0.0062	0.0186	0.0320	0.1712	0.0114	0.0999
3	0.2669	0.5426	0.0483	0.1130	0.0173	0.0433	0.0668	0.2929	0.1234	0.3405
4	0.7000	0.7730	0.1171	0.2279	0.0570	0.1608	0.1044	0.4083	0.1890	0.5122
Simulated Annealing										
	10×10		20×20		30×30		40×40		50×50	
	mean	variance	mean	variance	mean	variance	mean	variance	mean	variance
1	0.3771	0.3239	0.5606	0.3573	0.6905	0.3711	0.7091	0.4536	0.6965	0.3957
2	0.7919	0.5172	1.0104	0.4949	1.0682	0.5091	1.1263	0.5371	1.1255	0.5326
3	0.9133	0.6889	1.1019	0.6027	1.2006	0.5825	1.2678	0.6125	1.2609	0.5963
4	1.0685	0.8217	1.3180	0.6548	1.3653	0.6394	1.3086	0.6827	1.2806	0.6590
Algorithm Based on the Location of the Components										
	10×10		20×20		30×30		40×40		50×50	
	mean	variance	mean	variance	mean	variance	mean	variance	mean	variance
1	0.0123	0.0401	0.0022	0.0094	0.0033	0.0019	0.0023	0.0014	0.0012	0.0009
2	0.0382	0.1105	0.0055	0.0257	0.0126	0.1002	0.0303	0.2217	0.0058	0.2406
3	0.1852	0.3217	0.0290	0.0676	0.0346	0.1772	0.1380	0.4023	0.1466	0.5303
4	0.5757	0.6457	0.0839	0.3402	0.1123	0.4233	0.1560	0.6073	0.2964	0.7169

It is easy to see that if the $X \in \mathcal{BM}(H,V)$ image is hv-convex, then

$$f(X) = \max\{f(M)|M \in \mathcal{BM}(H,V)\} = \sum_{m}^{i=1}(h_i - 1) + \sum_{n}^{j=1}(v_j - 1)\ . \qquad (10)$$

However, SA can also produce non-hv-convex images, which occasionally can have better RME values than others which are not hv-convex (see Fig. 6 for an example). For this reason, we also calculated how the criteria of convexity defined in (6) is satisfied. Table 2 depicts for each reconstructed set the value of (6) divided by its possible maximum determined by (10). From this table it becomes evident that – regarding the SA based algorithm – not only the RME value but also the convexity of the reconstructed image gets worse and worse, as the size of the image and the number of its components is increasing.

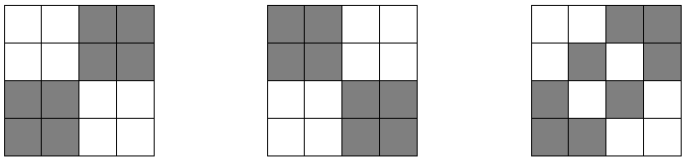


Figure 6: Original image (a). An hv-convex solution with RME=1 (b). A non-hv-convex image (c) with the same projections but better RME value (RME=0.5) than image (b).

Table 2: Quality of the reconstruction measured as the fraction of the number of adjacent ones and its possible maximum

Number of adjacent ones - Simulated Annealing										
	10×10		20×20		30×30		40×40		50×50	
	mean	variance	mean	variance	mean	variance	mean	variance	mean	variance
1	0.94	0.04	0.89	0.06	0.86	0.07	0.86	0.09	0.87	0.07
2	0.87	0.05	0.79	0.06	0.77	0.06	0.76	0.05	0.76	0.06
3	0.87	0.05	0.77	0.06	0.73	0.06	0.71	0.06	0.70	0.05
4	0.87	0.05	0.72	0.05	0.69	0.05	0.68	0.06	0.69	0.06

We also analyzed the effectiveness of the switching operators in the SA based algorithm. We calculated the difference of $f(S) - f(R)$ and again, divided by the maximum value given by (10), where S and R denoted the final reconstruction and the initial solution matrix provided by the Ryser algorithm, respectively. The results are shown in Table 3. We can deduce that the more components the discrete set has, the greater the difference is in quality between the initial and the final solutions. This again, can be justified by the observation, that – owing to the more switching components in the image – Ryser's algorithm gives a worse initial solution regarding the hv-convex property, in case of bigger number of components.

Table 3: Quality of the correction as a function of the number of components (first column) and the size of the image (horizontally)

Quality of the correction - Simulated Annealing										
	10×10		20×20		30×30		40×40		50×50	
	mean	variance	mean	variance	mean	variance	mean	variance	mean	variance
1	0.08	0.05	0.06	0.04	0.05	0.03	0.05	0.04	0.04	0.03
2	0.18	0.07	0.10	0.04	0.09	0.03	0.08	0.03	0.08	0.03
3	0.22	0.07	0.13	0.04	0.10	0.03	0.09	0.02	0.10	0.02
4	0.27	0.08	0.16	0.04	0.12	0.03	0.10	0.02	0.10	0.02

4.3 Running Time of the Reconstructions

Table 4 shows the running time (clear CPU usage, without garbage collecting) of the algorithms. First of all, the running time of the SA based algorithm is more or less independent of the number of components. It is rather influenced by the cooling schedule and the number of the switching components which increases with the size of the image. Furthermore, the two versions of the core-shell algorithm show similar performance. Images with 1 or 2 components can be reconstructed much faster than images with 3 or 4 components. In both implementations, the running time increases rapidly both by increasing the size of the set, and the number of its components. On the contrary, the speed of the algorithm based on the location of the components is mostly influenced by the size of the image, and not by the number of components. Finally, the implementation with the array data type is much faster than the one using the first-last data structure. The reason is, that this latter one needs more calculation for processing the new kernel and shell, when a heuristic step is taken.

One more observation is that in case of the kernel-shell method the size of the components has an important effect on the speed of the reconstruction. If the image contains a relatively big component, then it is more likely that in the first step a non-empty kernel can be produced, yielding less or no need for using the time-costly stack operators. In case of smaller components, the stack must be used more often for guessing, which means many steps of backtracking and a lot of execution time (see Fig. 7).

Finally, it is worth to note that the average time of the reconstruction with the kernel-shell method and the algorithm based on the location of the components can significantly differ with the same components but aligning them in different ways. Figure 8 shows just an example in case of images of size 30×30 with 4 components, but we found similar trends in any other test cases.

Table 4: The average and the variance of the running times in milliseconds

Core-Shell - Array data type										
	10		20		30		40		50	
	average	variance	average	variance	average	variance	average	variance	average	variance
1	39.73	4.92	52.82	11.33	74.77	37.12	147.81	126.75	246.45	325.99
2	42.53	4.20	71.63	45.65	146.87	97.71	334.67	351.82	735.52	1172.44
3	45.08	4.87	102.21	83.98	341.86	478.53	872.21	1341.80	1526.40	2783.91
4	47.33	13.61	292.23	402.12	1032.04	2333.96	1697.22	2264.28	3094.40	3277.27
Core-Shell - First-Last data type										
	10		20		30		40		50	
	average	variance	average	variance	average	variance	average	variance	average	variance
1	37.67	1.87	50.16	9.54	73.98	27.93	124.15	74.56	215.61	143.69
2	42.22	3.99	80.52	34.43	166.89	103.44	530.95	534.85	1349.71	1467.54
3	45.82	4.99	169.31	138.09	570.69	495.88	1719.58	1574.60	2223.90	2043.90
4	45.41	5.88	755.18	824.58	2006.66	2155.59	2934.82	2408.63	5889.01	4992.17
Simulated Annealing										
	10		20		30		40		50	
	average	variance	average	variance	average	variance	average	variance	average	variance
1	145.36	63.91	319.39	154.14	492.51	248.76	673.49	351.06	852.51	456.82
2	182.08	72.76	421.86	168.49	638.97	266.82	839.62	367.29	1053.75	461.08
3	184.24	69.54	386.23	143.50	612.47	245.05	832.27	334.91	1037.32	419.95
4	182.63	60.67	366.21	117.53	571.41	201.29	752.02	274.17	940.79	352.44
Algorithm based on the location of the components										
	10		20		30		40		50	
	average	variance	average	variance	average	variance	average	variance	average	variance
1	1.07	0.43	2.03	1.32	1.66	12.45	2.86	3.74	9.05	71.72
2	1.64	1.79	7.01	14.22	44.86	128.45	77.67	325.58	446.63	945.98
3	3.02	2.48	91.56	423.13	217.65	505.17	267.18	510.26	687.54	1591.19
4	4.47	10.07	356.46	790.07	436.50	1214.05	575.66	507.71	962.66	2212.41

5 Conclusion

In this paper we studied different algorithms for reconstructing hv-convex binary images. Knowing from theory that the task is NP-hard, we found that the difficulty of the problem depends on several factors. In case of the core-shell algorithm, the reconstruction speed and quality is in connection with the size of the image, and the number, the position, and the size of the components. The efficiency of the SA based algorithm depends on the number of the switching components in the image. Finally, the reconstruction efficiency of the algorithm based on the location of the components is mostly determined by the size of the image, and the number and position of the components. Thus, a fast algorithm for reconstructing hv-convex binary images must somehow combine the core-shell operators, the localization of the components, and the switching operators, too. In a future work we also intend to investigate how prior information on the number and size of the components can facilitate the reconstruction. We believe that the deeper insight we gained through our experiments can help us to design more efficient reconstruction algorithms for this class, in the near future. We also hope that this knowledge could also reveal the difficultness of the reconstruction in other classes of binary images, too.

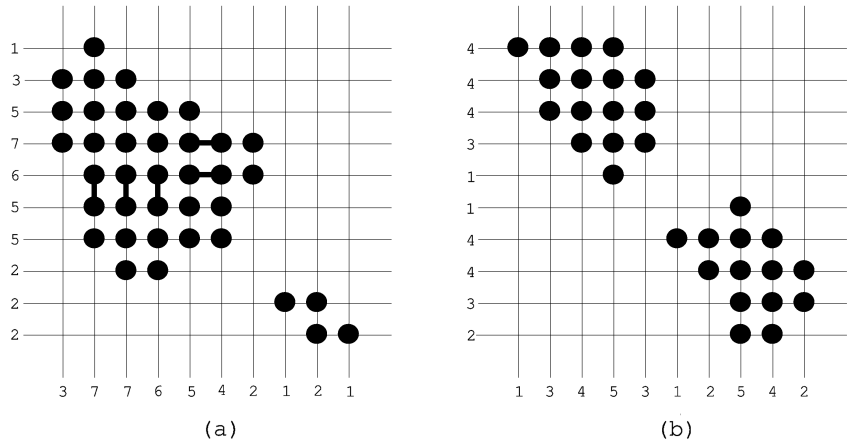


Figure 7: Two different images with the same number of components. (a) When the core-shell algorithm can give an initial non-empty kernel (indicated by thick lines), and (b) when the algorithm cannot give an initial non-empty kernel.

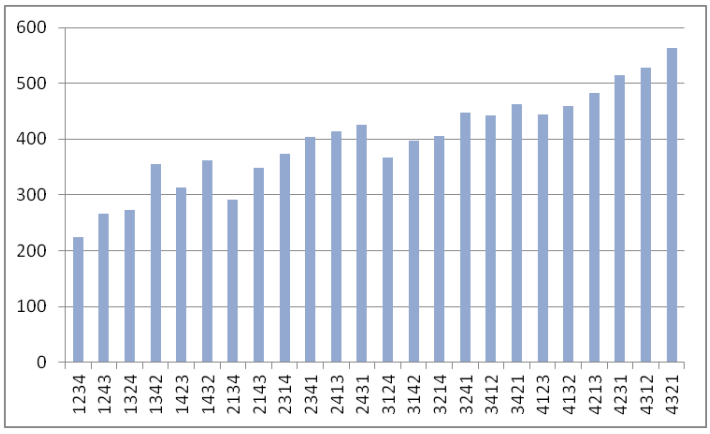


Figure 8: Average reconstruction time of the core-shell algorithm (with array data type) in milliseconds for images of size 30×30 with 4 components allocated them in different orders. Permutations on the horizontal axis indicate how the components are positioned relatively to each other. Components are numbered from top to bottom, and the permutation shows their orders from left to right.

References

[1] P. Balázs, Discrete tomographic reconstruction of binary images with disjoint components using shape information, *International Journal of Shape Modeling* **14:2** (2008) 189–207.

[2] P. Balázs, A benchmark set for the reconstruction of hv-convex discrete sets, *Discrete Appl. Math.* **157** (2009) 3447–3456.

[3] P. Balázs, A framework for generating some discrete sets with disjoint components by using uniform distributions, *Theoret. Comput. Sci.* **406** (2008) 15–23.

[4] E. Barcucci, A. Del Lungo, M. Nivat, R. Pinzani, Reconstructing convex polyominoes from horizontal and vertical projections, *Theor. Comput. Sci.* **155** (1996) 321–347.

[5] S. Brunetti, A. Del Lungo, F. Del Ristoro, A. Kuba, M. Nivat, Reconstruction of 4- and 8-connected convex discrete sets from row and column projections, *Lin. Algebra Appl.* **339** (2001) 37–57.

[6] M. Chrobak, C. Dürr, Reconstructing hv-convex polyominoes from orthogonal projections, *Inform. Process. Lett.* **69** (1999) 283–289.

[7] G. Dahl, T. Flatberg, Optimization and reconstruction of hv-convex (0,1)-matrices, *Discrete Appl. Math.* **151** (2005) 93–105.

[8] G.T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, 2nd edition, Springer, 2009.

[9] G.T. Herman, A. Kuba (Eds.), *Discrete Tomography: Foundations, Algorithms and Applications*, Birkhäuser, Boston, 1999.

[10] G.T. Herman, A. Kuba (Eds.), *Advances in Discrete Tomography and Its Applications*, Birkhäuser, Boston, 2007.

[11] F. Jarray, G. Tlig, A simulated annealing for reconstructing hv-convex binary matrices, *Electronic Notes in Discrete Mathematics* **36** (2010) 447–454.

[12] A.C. Kak, M. Slaney, *Principles of Computerized Tomographic Imaging*, IEEE Service Center, Piscataway, NJ., 1988.

[13] A. Kuba, Reconstruction of two-directionally connected binary patterns from their two orthogonal projections, *Computer Vision, Graphics, and Image Proc.* **27** (1984) 249–265.

[14] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculation by fast computing machines, *J. Chem. Phys.* **21** (1953) 1087–1092.

[15] H.J. Ryser, Combinatorial properties of matrices of zeros and ones, *Canad. J. Math.* **9** (1957) 371–377.

[16] G.J. Woeginger, The reconstruction of polyominoes from their orthogonal projections, *Inform. Process. Lett.* **77** (2001) 225–229.

[17] A. Kuba, G.T. Herman, S. Matej, A. Todd-Pokropek: Medical applications of discrete tomography, Discrete Mathematical Problems with Medical Applications, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS,* 55:195–208 (2000).