The Reconstruction of Some 3D Convex Polyominoes from Orthogonal Projections*

Maciej Gębala

Institute of Mathematics, Wrocław University of Technology
Janiszewskiego 14, 50–370 Wrocław, Poland
mgc@im.pwr.wroc.pl

Abstract. The reconstruction of discrete two- or three-dimensional sets from their orthogonal projections is one of the central problems in the areas of medical diagnostics, computer-aided tomography, and pattern recognition. In this paper we will give a polynomial algorithm for reconstruction of some class of convex three-dimensional polyominoes that has time complexity $O(n^7 \log n)$.

1 Introduction

A unit cube is a cube of volume one, whose centre belongs to $\mathbb{Z}^3$ and whose vertices belong to the lattice $(\mathbb{Z} + 1/2)^3$. We do not distinguish between a unit cube and its centre, thus (x, y, z) denotes the unit cube of centre (x, y, z). A three-dimensional polyomino is a finite connected union of unit cubes. In this paper we consider polyominoes contained in a finite lattice $\{1, \ldots, n\}^3$. The lattice with a polyomino corresponds to the three-dimensional binary matrix R, when the 1's correspond to filling positions in the lattice, and 0's correspond to empty positions.

The slices are two-dimensional sections of the matrix R. We define the slices S_X^i, S_Y^j and S_Z^k by $S_X^i[j,k] = R[i,j,k]$, $S_Y^j[i,k] = R[i,j,k]$ and $S_Z^k[i,j] = R[i,j,k]$, respectively, for $i, j, k \in \{1, \ldots, n\}$.

The bars are one-dimensional sections of the matrix R. We define the bars $B_X^{j,k}$, $B_Y^{i,k}$ and $B_Z^{i,j}$ by $B_X^{j,k}[i] = R[i,j,k]$, $B_Y^{i,k}[j] = R[i,j,k]$ and $B_Z^{i,j}[k] = R[i,j,k]$, respectively, for $i, j, k \in \{1, \ldots, n\}$.

For a polyomino P in the matrix R we define three two-dimensional matrices of orthogonal projections (i.e. the number of 1's in each bar of matrix R): P_T, P_F and P_S by

$$P_T(P)[i,j] = \sum_{k=1}^{n} R[i,j,k], \qquad P_F(P)[i,k] = \sum_{j=1}^{n} R[i,j,k],$$

$$P_S(P)[j,k] = \sum_{i=1}^{n} R[i,j,k],$$

for $i, j, k \in \{1, \ldots, n\}$.

* Supported by the KBN grant No. 7 T11C 03220

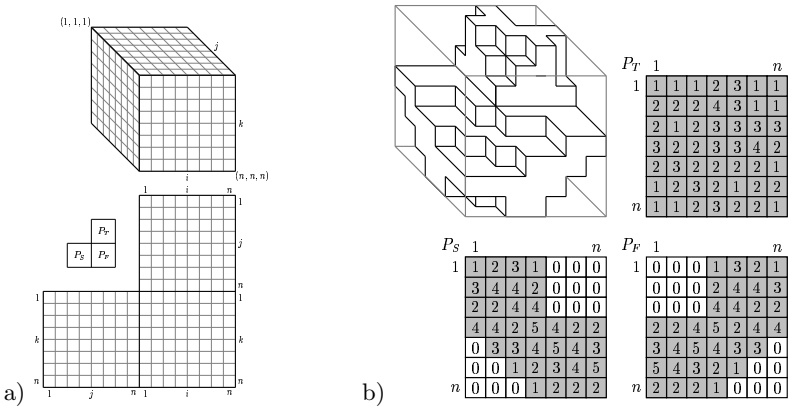

Fig. 1. a) three-dimensional matrix R and adequate matrices of orthogonal projections; b) example of full convex 3D polyomino with the matrices of orthogonal projections

Definition 1. *The three-dimensional polyomino P in the matrix R is a full convex polyomino if each slice of R contains a two-dimensional hv-convex polyomino (i.e. connected set of 1's with the property that in every row and every column of the slice the 1's forms a connected block) and at least one matrix of projections does not contain zeros (is full).*

We consider that always P_T is a full matrix. Thus each bar $B_Z^{i,j}$ contains at least one 1.

We can now define the problem of reconstructing a full convex 3D polyomino P from its orthogonal projections: Given three assigned matrices

$$P_F \in \{0,\ldots,n\}^{n\times n},\ P_S \in \{0,\ldots,n\}^{n\times n} \text{ and } P_T \in \{1,\ldots,n\}^{n\times n},$$

we examine whether there exists at least one matrix $R \in \{0,1\}^{n\times n\times n}$ with a full convex polyomino P such that $P_T(P) = P_T$, $P_F(P) = P_F$ and $P_S(P) = P_S$.

1.1 Previous Results

The reconstruction of polyominoes from orthogonal projections is one of classical problems of discrete tomography and has been an intensive study for some years. But all the results concern the two-dimensional case.

First Ryser [8], and subsequently Chang [2] and Wang [9] studied the existence of a pattern satisfying orthogonal projections (H, V) in the class of sets without any conditions. They showed that the decision problem can be solved in time $O(mn)$. These authors also developed some algorithms that reconstruct the pattern from (H, V).

Woeginger [10] proved that the reconstruction problem in the class of polyominoes is an NP-complete problem. Barcucci, Del Lungo, Nivat, Pinzani [1]

showed that the reconstruction problem is also NP-complete in the class of h-convex polyominoes and in the class v-convex polyominoes.

The first algorithm that establishes the existence of an hv-convex polyomino satisfying a pair of assigned vectors (H, V) in polynomial time was described by Barcucci et al. in [1]. Its time complexity is $O(m^4n^4)$ and it is rather slow. Gębala [5] showed the faster version of this algorithm with complexity $O(\min(m^2, n^2) \cdot mn \log mn)$. The latest algorithm described by Chrobak and Dürr in [3] reconstructs the hv-convex polyomino from orthogonal projection in time $O(\min(m^2, n^2) \cdot mn)$.

Moreover Gębala in [6] shows that the problem of the reconstruction two-dimensional polyominoes from approximately orthogonal projections is NP-complete in the classes of (1) polyominoes, (2) horizontal convex polyominoes and (3) vertical convex polyominoes. And proves that, for an arbitrary chosen function of error, the problem is in P for the class of hv-convex polyominoes and shows the algorithm with complexity $O(m^3n^3)$.

For three-dimensional case only one result was known. A general problem of the reconstruction of three-dimensional lattice sets from orthogonal projections is NP-complete [7,4].

1.2 Our Results

In this paper we show a polynomial algorithm for the reconstruction of full convex 3D polyominoes with complexity $O(n^7 \log n)$. In this reconstruction we use some properties of 2D convex polyominoes described in [5].

2 Some Properties of 2D hv-Convex Polyominoes

Let Q be a matrix which has $n \times n$ cells containing 0's and 1's. Let S be a set of cells containing 1's which represents an hv-convex 2D polyomino. Let h_i denotes the projection of i-th row and v_j denotes the projection of j-th column. Moreover let

$$H_k = \sum_{i=1}^{k} h_i \qquad \text{and} \qquad V_k = \sum_{j=1}^{k} v_j,$$

for $k \in \{1, \ldots, n\}$, and $H_0 = 0$, $V_0 = 0$ and $A = H_n = V_n$.

Definition 2. *For all $j \in \{1, \ldots, n\}$ we define*

$$\begin{aligned}
D_j^{\searrow} &= \min\{\ i \in \{1, \ldots, m-1\}\ :\ A - H_i < A - V_j \quad \vee \quad i = m\},\\
U_j^{\searrow} &= \max\{\ i \in \{2, \ldots, m\}\ :\ H_{i-1} < V_{j-1} \quad \vee \quad i = 1\}.\\
D_j^{\swarrow} &= \min\{\ i \in \{1, \ldots, m-1\}\ :\ A - H_i < V_{j-1} \quad \vee \quad i = m\}\\
U_j^{\swarrow} &= \max\{\ i \in \{2, \ldots, m\}\ :\ H_{i-1} < A - V_j \quad \vee \quad i = 1\}.
\end{aligned}$$

Lemma 1 ([5]). *For all $j \in \{1, \ldots, n\}$ we have $U_j^{\searrow} \leq D_j^{\searrow}$ and $U_j^{\swarrow} \leq D_j^{\swarrow}$. And for all $j \in \{1, \ldots, n-1\}$ we have $D_j^{\searrow} + 1 \geq U_{j+1}^{\searrow}$ and $D_{j+1}^{\swarrow} + 1 \geq U_j^{\swarrow}$.* □

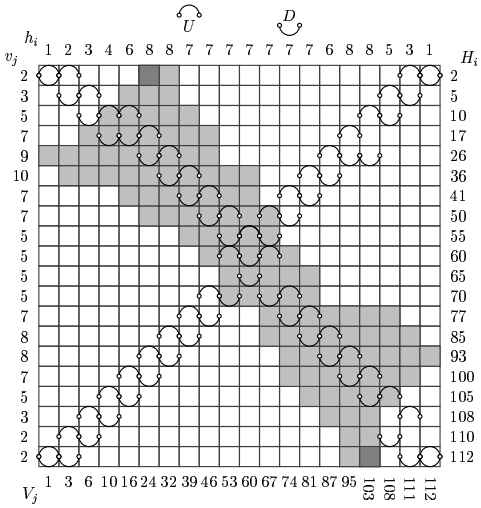

Fig. 2. Properties of 2D hv-convex polyomino

Lemma 2 ([5]). *Let cells* $[1, p_1]$ *and* $[n, p_2]$ *belong to the polyomino* S *in the matrix* Q*, for the fixed* p_1 *and* p_2*. Then*

1. *if* $p_1 \leq p_2$ *then cells* $[1, p_1], \ldots, [D^{\searrow}_{p_1}, p_1]$*, and* $[U^{\searrow}_j, j], \ldots, [D^{\searrow}_j, j]$ *for* $j \in \{p_1 + 1, \ldots, p_2 - 1\}$*, and* $[U^{\searrow}_{p_2}, p_2], \ldots, [n, p_2]$ *also belong to the polyomino* S*, or*
2. *if* $p_1 \geq p_2$ *then cells* $[U^{\swarrow}_{p_2}, p_2], \ldots, [n, p_2]$*, and* $[U^{\swarrow}_j, j], \ldots, [D^{\swarrow}_j, j]$ *for* $j \in \{p_2 + 1, \ldots, p_1 - 1\}$*, and* $[D^{\swarrow}_{p_1}, p_1], \ldots, [1, p_1]$ *also belong to the polyomino* S*.* □

Corollary 1 ([5]). *If we have at least one cell which belongs to* S *in the first and the last row of the matrix* Q *then we can compute at least one cell which belongs to* S *in each row of the matrix* Q*. These computations cost* $O(n)$*.* □

We can change symmetrically the word *row* by the word *column* in above lemmas and they still hold.

3 Algorithm of Reconstruction

In this section we describe a polynomial algorithm for the reconstruction of full convex 3D polyominoes. The main idea of the algorithm is to test all possible positions of 1's in four corner bars $B_Z^{1,1}$, $B_Z^{1,n}$, $B_Z^{n,1}$ and $B_Z^{n,n}$. If we fix any initial positions in corner bars we will use Lemma 2 for computing positions at least one 1 in each bar $B_Z^{i,j}$. If we have at least one 1 in each bar $B_Z^{i,j}$ we will start filling procedure that return the polyomino or the word *fail*, when a polyomino with these initial positions does not exist.

3.1 Initial Positions

We arbitrary fix the positions of 1's in corner bars $B_Z^{1,1}$, $B_Z^{1,n}$, $B_Z^{n,1}$ and $B_Z^{n,n}$. We call these positions by the initial positions. Now we consider two slices S_X^1 and S_X^n. Both will contain, by Definition 1, a two-dimensional hv-convex polyomino. Both contain in the first and the last column at least one 1. Thus from Corollary 1 we can compute positions of 1's in each column of these slices, i.e. in each bar $B_Z^{1,i}$ and $B_Z^{n,i}$, for $i \in \{1, \ldots, n\}$.

Now we consider all slices S_Y^j, for $j \in \{1, \ldots, n\}$. Each slice S_Y^j contain now at least one 1 in the first and the last column. We can use again Corollary 1 for each slice. Thus we have at least one 1 in each column of each slice S_Y^j. In other words we have at least one 1 in each bar $B_Z^{i,j}$, for $i, j \in \{1, \ldots, n\}$.

The set of positions computed above we denote by the start positions. It is easy to see that the following lemma is correct

Lemma 3. *If there exists the full convex 3D polyomino P that satisfies matrices of projections P_T, P_S and P_F, and if the initial positions belong to the polyomino P then all of the start positions also belong to P.* □

Lemma 4. *Computing the start positions costs $O(n^2)$.*

Proof. We perform $n+2$ times Corollary 1. Thus the cost is equal to $O(n^2)$. □

3.2 Filling Procedure

The procedure described below is three-dimensional modifications of the filling procedure described in [5].

We use balanced binary trees (like e.g. AVL) with following operations:

empty($tree$) - returning *true* if $tree$ is empty and *false* otherwise, with the complexity $O(1)$;
delete(k,$tree$) - deleting element k from $tree$, with the complexity $O(\log |tree|)$;
insert(k,$tree$) - inserting element k in $tree$ where $k \in tree$; or doing nothing otherwise, with the complexity $O(\log |tree|)$;
min($tree$) - returning a minimal element of $tree$, with complexity $O(\log |tree|)$;
max($tree$) - returning a maximal element of $tree$, with complexity $O(\log |tree|)$;

where $|tree|$ means the size of $tree$.

We have three global variables tree_X, tree_Y and tree_Z which are balanced binary trees. In these trees we will store the indices of bars, which we will review in the next step of the main loop of our procedure.

For each bar we define the following auxiliary variables: l, r, p, q, $\tilde{l}$, $\tilde{r}$, $\tilde{p}$, $\tilde{q}$ and free_0. The variable l is a minimal position containing 1, r is a maximal position containing 1, p is a minimal position without 0 and q is a maximal position without 0. The variables $\tilde{l}$, $\tilde{r}$, $\tilde{p}$ and $\tilde{q}$ are the temporary values of l, r, p and q, respectively. The variable free_0 is a balanced tree containing positions of 0's which are between $\tilde{p}$ and $\tilde{q}$. We initialise these variables as follow: $l = \tilde{l} = n + 1$,

$r = \tilde{r} = 0$, $p = \tilde{p} = 1$, $q = \tilde{q} = n$ and $\text{free}_0 = nil$, where nil means an empty tree.

Now we can introduce two auxiliary operations putting 1 and 0 in a bar:

put 0 bar $B_X^{j,k}$ at the position i ($[i,j,k]$):
if $R[i,j,k] = 1$ then exit(*fail*)
if $R[i,j,k] \neq 0$ then
  $R[i,j,k] \leftarrow 0$, insert$((i,k),\text{tree}_Y)$, insert$((i,j),\text{tree}_Z)$
  modify0 bar $B_Y^{i,k}$ at the position j with the matrix P_F
  modify0 bar $B_Z^{i,j}$ at the position k with the matrix P_T

put 1 bar $B_X^{j,k}$ at the position i ($[i,j,k]$):
if $R[i,j,k] = 0$ then exit(*fail*)
if $R[i,j,k] \neq 1$ then
  $R[i,j,k] \leftarrow 1$, insert$((i,k),\text{tree}_Y)$, insert$((i,j),\text{tree}_Z)$
  modify1 bar $B_Y^{i,k}$ at the position j with the matrix P_F
  modify1 bar $B_Z^{i,j}$ at the position k with the matrix P_T

Where suboperations **modify0** and **modify1** are defined consecutively by

modify0 bar $B^{j,k}$ at the position i with the matrix P:
with $B^{j,k}$ do
  if $r \geq l$ then
    if $i < l$ and $i \geq \tilde{p}$ then $\tilde{p} \leftarrow i + 1$
    if $i > r$ and $i \leq \tilde{q}$ then $\tilde{q} \leftarrow i - 1$
  else
    if $i < \tilde{p} + P[j,k]$ and $i \geq \tilde{p}$ then
      $\tilde{p} \leftarrow i + 1$
      while not empty(free_0) and $(m \leftarrow \min(\text{free}_0)) < \tilde{p} + P[j,k]$ do
        delete(m,free_0), $\tilde{p} \leftarrow m + 1$
    if $i > \tilde{q} - P[j,k]$ and $i \leq \tilde{q}$ then
      $\tilde{q} \leftarrow i - 1$
      while not empty(free_0) and $(m \leftarrow \max(\text{free}_0)) > \tilde{q} - P[j,k]$ do
        delete(m,free_0), $\tilde{q} \leftarrow m - 1$
    if $\tilde{p} + P[j,k] \leq i \leq \tilde{q} - P[j,k]$ then insert(i,free_0)

modify1 bar $B^{j,k}$ at the position i with the matrix P:
with $B^{j,k}$ do
  if $r < l$ then
    $l \leftarrow r \leftarrow \tilde{l} \leftarrow \tilde{r} \leftarrow i$
    if $\tilde{p} < i - P[j,k] + 1$ then $\tilde{p} \leftarrow i - P[j,k] + 1$
    if $\tilde{q} < i + P[j,k] - 1$ then $\tilde{q} \leftarrow i + P[j,k] - 1$
    while not empty(free_0) do
      $m \leftarrow \min(\text{free}_0)$, delete$(m,\text{free}_0)$
      if $m < i$ and $m + 1 > \tilde{p}$ then $\tilde{p} \leftarrow n + 1$
      if $m > i$ and $m - 1 < \tilde{q}$ then $\tilde{q} \leftarrow n - 1$
  else
    if $i < \tilde{l}$ then $\tilde{l} \leftarrow i$
    if $i > \tilde{r}$ then $\tilde{r} \leftarrow i$

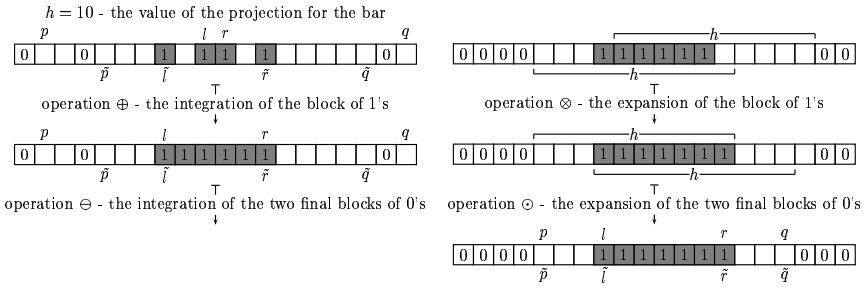

Fig. 3. Operations $\oplus$, $\ominus$, $\otimes$ and $\odot$ in the bar

The operations described above retain in memory the indices of bars, that are modified when we put a new symbol and modify adequate auxiliary variables.

Now we define operations putting new symbols in the bar (see Fig. 3):

operation $\oplus$ in bar $B_X^{j,k}$:
`with` $B_X^{j,k}$ `do`
  `if` $\tilde{l} < l$ `then`
    `for` $i \leftarrow \tilde{l}$ `to` $l-1$ `do` **put 1** in bar $B_X^{j,k}$ at the position i
    $l \leftarrow \tilde{l}$
  `if` $\tilde{r} > r$ `then`
    `for` $i \leftarrow r+1$ `to` $\tilde{r}$ `do` **put 1** in bar $B_X^{j,k}$ at the position i
    $r \leftarrow \tilde{r}$

The operation $\oplus$ fills empty cells between 1's and integrates the block of 1's in the bar.

operation $\ominus$ in bar $B_X^{j,k}$:
`with` $B_X^{j,k}$ `do`
  `if` $p < \tilde{p}$ `then`
    `for` $i \leftarrow p$ `to` $\tilde{p}-1$ `do` **put 0** in bar $B_X^{j,k}$ at the position i
    $p \leftarrow \tilde{p}$
  `if` $q > \tilde{q}$ `then`
    `for` $i \leftarrow \tilde{q}+1$ `to` q `do` **put 0** in bar $B_X^{j,k}$ at the position i
    $q \leftarrow \tilde{q}$

The operation $\ominus$ integrates the two final blocks of 0's in the bar.

operation $\otimes$ in bar $B_X^{j,k}$:
`with` $B_X^{j,k}$ `do`
  `if` $l > r$ `and` $p + P_S[j,k] - 1 \geq q - P_S[j,k] + 1$ `then`
    $l \leftarrow \tilde{l} \leftarrow q - P_S[j,k] + 1$, $r \leftarrow \tilde{r} \leftarrow p + P_S[j,k] - 1$
    `for` $i \leftarrow l$ `to` r `do` **put 1** in bar $B_X^{j,k}$ at the position i
  `if` $l \leq r$ `and` $q - P_S[j,k] + 1 < l$ `then`
    `for` $i \leftarrow q - P_S[j,k] + 1$ `to` $l-1$ `do` **put 1** in bar $B_X^{j,k}$ at the position i

$l \leftarrow \tilde{l} \leftarrow q - P_S[j,k] + 1$
`if` $l \leq r$ `and` $p + P_S[j,k] - 1 > r$ `then`
`for` $i \leftarrow r+1$ `to` $p + P_S[j,k] - 1$ `do` **put 1** in bar $B_X^{j,k}$ at the position i
$r \leftarrow \tilde{r} \leftarrow p + P_S[j,k] - 1$

The operation $\otimes$ expands the block of 1's if the area without 0's is adequately narrow.

operation $\odot$ in bar $B_X^{j,k}$:
`with` $B_X^{j,k}$ `do`
`if` $l \leq r$ `and` $p \leq r - P_S[j,k]$ `then`
`for` $i \leftarrow p$ `to` $r - P_S[j,k]$ `do` **put 0** in bar $B_X^{j,k}$ at the position i
$p \leftarrow \tilde{p} \leftarrow r - P_S[j,k] + 1$
`if` $l \leq r$ `and` $q \geq l + P_S[j,k]$ `then`
`for` $i \leftarrow l + P_S[j,k]$ `to` q `do` **put 0** in bar $B_X^{j,k}$ at the position i
$q \leftarrow \tilde{q} \leftarrow l + P_S[j,k] - 1$

The operation $\odot$ expands the two final blocks of 0's if the block of 1's is adequately wide in the bar.

It is obvious that we define these operations for bars B_Y and B_Z analogously.

Moreover if the bar contains 1's then it has five following blocks of cells: the block of 0's, the block of empty cells, the block of 1's, the block of empty cells and the block of 0's. The size of each block of empty cells is equal to the number of the deficient 1's. Each bar containing 1's has this property after operations $\oplus$, $\ominus$, $\otimes$ and $\odot$.

The main loop of filling procedure has the following form now:

`repeat`
`while not` empty(tree$_X$) `do`
$(j,k) \leftarrow$min(tree$_X$), delete$((j,k)$,tree$_X$)
perform operations $\oplus, \ominus, \otimes, \odot$ in bar $B_X^{j,k}$
`while not` empty(tree$_Y$) `do`
$(i,k) \leftarrow$min(tree$_Y$), delete$((i,k)$,tree$_Y$)
perform operations $\oplus, \ominus, \otimes, \odot$ in bar $B_Y^{i,k}$
`while not` empty(tree$_Z$) `do`
$(i,j) \leftarrow$min(tree$_Z$), delete$((i,j)$,tree$_Z$)
perform operations $\oplus, \ominus, \otimes, \odot$ in bar $C_Z^{i,j}$
`until` empty(tree$_X$) `and` empty(tree$_Y$) `and` empty(tree$_Z$)
`if` there exist empty cells in the matrix R `then`
build and solve the adequate 2SAT formula

When we compute the start positions we put neither 0 nor 1 in the matrix R. We only modify variables $\tilde{p}$ and $\tilde{q}$ in bars B_Z and put the indices of these bars in tree$_Z$. Operations $\ominus$ and $\otimes$ put these symbols in the first step of the loop.

3.3 Correctness and Complexity

Lemma 5. *If we have at least one 1 in each bar B_Z then the filling procedure works correctly.*

Proof. When we start the filling procedure we have in each bar B_Z at least one 1, thus the number of empty cells in the matrix R will be equal to the double number of the deficient 1's. And these properties occur in the whole time of the procedure work.

If the filling procedure returns *fail*, we know that a convex 3D polyomino which has projections P_T, P_S and P_F and the fixed initial positions does not exist.

Otherwise, if trees tree_X, tree_Y and tree_Z are empty, we have two different cases:
case 1: Each cell of the matrix R contains 0 or 1. We have the solution. The set P of 1's is a full convex 3D polyomino and satisfies matrices of projections.
case 2: The matrix R contains empty cells. Its number is equal to the double number of the deficient 1's. In each bar we have two blocks of empty cells such as the length of the block is equal to the number of the deficient 1's in this bar (it follows the properties of operations $\oplus$, $\ominus$, $\otimes$ and $\odot$). We group these empty cells into labelling cycles.

If we have an empty cell we label it by, for example, x. This cell has exactly three empty neighbour cells in the cycle, one in each bar B_X or B_Y or B_Z containing this cell. These cells are in other block of empty cells and have the same positions in blocks in the same bar. We label these cells by negation of x ($\bar{x}$). And we repeat this step for these cells until we cannot find a new cell for labelling in this way. We create a three-dimensional cycle in the matrix R which cells are labelled by x and $\bar{x}$ alternately and each cell has three neighbours in the cycle.

We create cycles as long as we have empty cells in matrix R. Each cycle we label by a different letter. Cells in the cycles are valued alternately by 0 and 1 (true and false). But some cycles are valued dependently. Thus we build a 2SAT formula. If in some bar we have two successive cells, labelled by, for example, x and y, we add to the formula the implication $x \to y$ if y is between x and the block of 1's, or implication $y \to x$ otherwise. If the bar does not contain the block of 1's we create cycle of implications for each block of empty cells in this way that variables in the block have the same value.

This 2SAT formula has the size at most $O(n^3)$ and can be solved in the linear time. Thus if the 2SAT formula is satisfiable then there exists an adequate full convex 3D polyomino that satisfies matrices of projections. □

Lemma 6. *The filling procedure costs at most* $O(n^3 \log n)$.

Proof. We estimate the complexity of the main loop of the filling procedure. In each position $[i, j, k]$ we perform operation **put** only three times (one operation in each bar containing this position). Moreover, when we do operations $\oplus$, $\ominus$, $\otimes$ and $\odot$ in the bar in our algorithm, we execute at least one **put** operation. Hence, we review only $O(n^3)$ bars and the review of one bar costs $O(\log n)$+[cost of the **put** operations]. Therefore, the global cost of the main loop of the procedure is $O(n^3 \log n)$+[cost of all **put** operations].

Now we estimate the global cost of all **put** operations. In the fixed bar when we perform **put** operations we execute at most n insert operations in an adequate

tree. It costs $O(n \log n)$. Since the insert operations in free_0 in the bar we have done no more than one time for each position. There are no more than n delete operations, either. We execute the functions min and max only while modifying $\tilde{p}$ or $\tilde{q}$. Hence, the number of these operations is at most n. All operations in the tree free_0 cost at most $\mathrm{O}(n \log n)$. For all $3n^2$ bars the cost is equal to $O(n^3 \log n)$.

The complexity of all residual operations is at most $O(n^3)$. Hence, the cost of the filling procedure is $O(n^3 \log n)$. □

Theorem 1. *The problem of the reconstruction of full convex 3D polyominoes from orthogonal projections has the complexity* $O(n^7 \log n)$.

Proof. The number of all different initial positions is equal to n^4 and the cost of the reconstruction for fixed initial position is equal to $O(n^3 \log n)$. Thus the cost of testing all possible initial positions is equal to $O(n^7 \log n)$. □

4 Conclusion

In this paper, we have studied the reconstruction of three-dimensional convex polyominoes from their orthogonal projection. We have designed the polynomial-time algorithm for the reconstruction of some wide class of the 3D polyominoes called full convex. Yet some complexity problems are still unsolved: are the general reconstruction problem for convex three-dimensional polyominoes polynomial or NP-complete?

References

1. Barcucci, E., Del Lungo, A., Nivat, M., Pinzani, R.: Reconstructing Convex Polyominoes from Horizontal and Vertical Projections. TCS 155 (1996) 321–347 2, 3
2. Chang, S.K.: The Reconstruction of Binary Patterns from their Projections. Communications of the ACM 14 (1971) 21–25 2
3. Chrobak, M., Dürr, Ch.: Reconstructing hv-convex Polyominoes from Orthogonal Projections. IPL 69(6) (1999) 283–289 3
4. Gardner R.J., Gritzmann P., Prangenberg D.: On the Computional Complexity of Reconstructing Lattice Sets from Their X-Rays. Discrete Mathematics 202 (1999) 45–71 3
5. Gębala, M.: The Reconstruction of Convex Polyominoes from Horizontal and Vertical Projections. SOFSEM'98, LNCS 1521 (1998) 350–359 3, 4, 5
6. Gębala, M.: The Reconstruction of Polyominoes from Approximately Orthogonal Projections. SOFSEM'2001, LNCS 2234 (2001) 253–260 3
7. Irving R.W., Jerrum M.R.: Three-Dimensional statistical data security problems. SIAM Journal on Computing 23 (1994) 170-184 3
8. Ryser, H.: Combinatorial Mathematics. The Carus Mathematical Monographs Vol. 14 (The Mathematical Association of America, Rahway, 1963) 2
9. Wang, X.G.: Characterisation of Binary Patterns and their Projections. IEEE Transactions on Computers C-24 (1975) 1032–1035 2
10. Woeginger, G.J.: The Reconstruction of Polyominoes from Their Orthogonal Projections. Information Processing Letters 77(5-6) (2001) 225-229 2