Optimization of placement in the tasks of rapid prototyping and manufacturing of volumetric parts based on additive technologies

M A Verkhoturov[1], G N Verkhoturova[1], R R Yagudin[1], K V Danilov[1], R R Karimov[1], N V Kondratyeva[2] and S S Valeev[1, 2]

[1] Ufa State Aviation Technical University, K. Marx Str. 12, Ufa, 450008, Russia
[2] Sochi State University, Politekhnicheskaya Str. 7, Sochi, 354008, Russia

verhotur@vmk.ugatu.ac.ru

Abstract. The problem of optimizing the life cycle of complex three-dimensional objects in small-scale production is considered. Additive technologies and optimization algorithms for the placement of three-dimensional objects are considered as technologies to solve this problem. Using the multilevel synthesis method can significantly reduce the time for prototyping new products. It should be noted that since several independent parts can be manufactured at the same time, this problem belongs to the class of optimization geometric modelling problems, namely, the problem of three-dimensional irregular placement. An algorithmic solution is proposed for the most complex task - placing 3D objects in a container. The results of the analysis of the effectiveness of the proposed algorithms are discussed.

1. Introduction

An analysis of the stages of the life cycle of complex products in various industries shows that many of them are associated with the solution of optimization problems of placement. Finding the optimal or close solution allows you to significantly reduce the consumption of various resources and lower the cost of production. Such tasks are important from the point of view of saving resources, but difficult to make decisions.

On the other hand, the advent of additive technologies and their application for rapid prototyping and manufacturing of volumetric parts made a real revolution in high-tech industries, which are characterized by single or small-scale production: nuclear, aerospace, aviation, instrumentation, etc.

The use of layered synthesis technologies has led to the emergence of new methods for producing synthesis models and synthesis forms, which made it possible to drastically reduce the time to create new products [1]. The implementation of such technologies leads to the need to solve the problem of irregular optimization placement of three-dimensional objects in those cases when many independent parts can be manufactured in one cycle of the corresponding installation, which is advisable from the point of view of saving time, energy and other resources. In modern manufacturing processes, accuracy is always limited by material processing devices that are digital (discrete), therefore, this article discusses the presentation of parts in a discrete-logical representation. One of the common formats for storing 3D objects of arbitrary shape is STL. This is a format for storing three-dimensional models of objects for use in rapid prototyping technologies, usually by stereolithography, in which information about the object is a list of triangular faces that describe its surface. The voxel-based algorithms approach proposed by the authors in [2] is considered in detail.

2. Statement of the problem

Irregular placement of 3D objects consists in the following mathematical problem.

There is a set of three-dimensional objects $T = \{T_1, T_2, \ldots, T_n\}$, $i = \overline{1,n}$ presented in STL format, where $T_i = \{T_1, T_2, \ldots, T_m\}$ is an array of triangles that represent the figure.

Each triangle is represented in the form $Tr_i(A, B, C)$, where A, B, C are the vertices of the triangle with coordinates (x, y, z).

The packing area R is a three-dimensional figure in the form of a rectangular parallelepiped $R = \{L, W, H\}$, where L (length) and W (width) are constants, and H (height) is a variable.

Packaging is a set of object placement parameters that satisfies the following conditions:

$$\begin{cases} T_i \in T,\ T_j \in T,\ T_i \cap T_j = \varnothing,\ \forall i = \overline{1,n},\ \forall j = \overline{1,n},\ i \neq j, \\ \\ \forall T_i \in T,\ T_i \cap R = T_i,\ i = \overline{1,n}, \end{cases} \tag{1}$$

i.e., no two objects intersect each other, and all objects are located inside the placement area R.

It is required for objects T_i, $i = \overline{1,n}$ to find a package with such a set of placement parameters U so that the density of packed objects in the placement area R is maximum

$$\rho(T(U)) = \frac{\sum_{i=1}^{n} V(T_i(U))}{V(R)} \to \max,$$

where $V(R) = L*W*H$ is the volume R, which depends on the height of the packing area H, and $V(T_i)$ is the volume of the object T_i.

3. The basic principles of constructing dense packaging based on a discrete-logical representation of information (voxel)

An analysis of the methods for solving this problem allows us to draw the following conclusion: in the process of development, these methods underwent the following chain of changes in the operations on which they are built: “floating-point operations” [3] – “integer arithmetic” – “logical operations” [2].

The peculiarity of using the last of the above is that only logical operations with “0” and “1” are needed to determine the non-intersection of geometric objects.

The main idea of this approach is to simulate the dense motion of objects, i.e. building No-Fit Polyhedron (NFP) [3] in computer memory.

Namely, the actions corresponding to the basic operations: shift, determination of the direction of movement, determination of intersections and others, are carried out on the basis of the discrete-logical structure of RAM.

Various implementation options are possible based on a discrete-logical representation of information (voxel) and chain coding [2, 7-12].

Each shape T_k consists of a starting point P_0 and a vector $Path_k = \{V_1, \ldots, V_m\}, i = \overline{1,m}$ describing the surface of the shape. The nodes in the vector are stored as

$$V_i(N, N_{prev}, direct),$$

where

- N is the number of the voxel, determining its place in the vector,
- N_{prev} is the reference to the number of the parent voxel from which the voxel is built,
- *direct* is the direction in which the voxel is located relative to the parent.

Further details of the approach presented earlier in [2] are considered.

4. Description of object placement algorithms using the NFP construction method based on a discrete-logical representation of information and chain coding (DLPI and CC)

Let a set of objects $T = \{T_1, T_2, \ldots, T_n\}$, $i = \overline{1,n}$ be given, each of which is represented as a surface described by a set of triangles.

Pretreatment:

- Conversion of objects from a given set into a discrete-logical space into a voxel matrix by the **RASTERIZATION()** procedure.
- Formation of a voxel vector by the procedure **SET_VECTOR()**.

External procedure:

- Sorting shapes $T = \{T_1, T_2, \ldots, T_n\}$.

Internal procedure:

- For each object T_i, $i = \overline{1,n}$ from the list T:

  1. Entering T_i into the allocation area by the function **PUT_BEGIN(T_i, P_0)**.
  2. The construction of the NFP T_i relative to the packing area and each of the already placed figures by the **NFP()** procedure.
  3. Selecting from all NFP points a point P_{minz} with a minimum coordinate along z; in the case of several similar points, select a point with a minimum vector length $\overrightarrow{(x, y)}$.
  4. Restoration of an object T_i at a point P_{minz}.

Consider the above procedures separately.

4.1. Converting objects to discrete logical space RASTERIZATION ()

Figure 1 shows the initial (polygonal) representation of the object. The final (voxel) representation of the object is presented in figure 2.

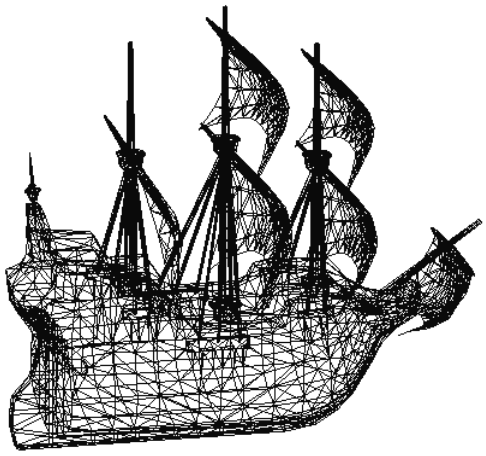

Figure 1. Representation of the object in polygonal form.

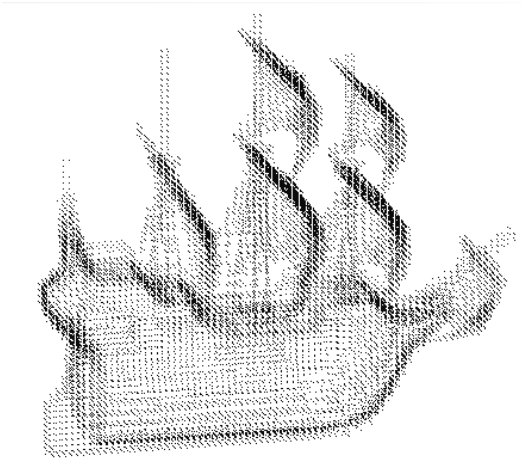

Figure 2. Representation of the object in the form of a matrix of voxels.

The algorithm for rasterizing one polygon $Tr_i(A, B, C)$ is as follows.

For intermediate storage, we select a three-dimensional *DataR* array. Iinitially the array is filled with zeros.

Building a matrix of voxels consists of the following steps.

1. Rasterize the sides of the polygon A, B, C according to the Bresenham's line algorithm into an array of points.

We obtain arrays of voxels that make up the corresponding parties:

$$AB = \{AB_1, \ldots, AB_n\},$$
$$BC = \{BC_1, \ldots, BC_m\},$$
$$AC = \{AC_1, \ldots, AC_p\}.$$

2. Divide the resulting arrays into groups $G = \{G_1, \ldots, G_r\}$. In each group G_i, all voxels have the same z coordinate,

$$G_i = \begin{cases} ab \in AB, \forall ab_j, z = i, j \in \overline{[1,n]}, \\ bc \in BC, \forall bc_j, z = i, j \in \overline{[1,m]}, i = \overline{1,r}, \\ ac \in AC, \forall ac_j, z = i, j \in \overline{[1,p]} \end{cases}$$

respectively, are in the same plane (figure 3).

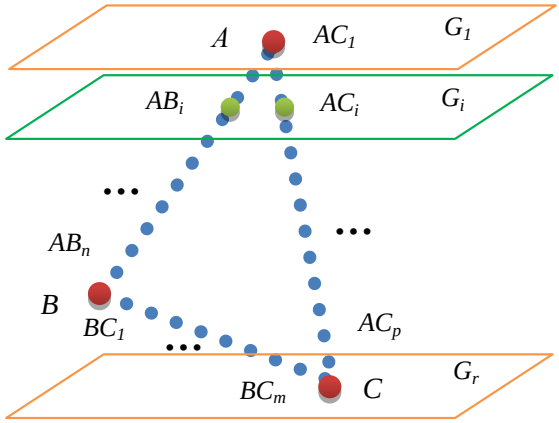

Figure 3. The distribution of voxels of the sides into groups $G = \{G_1, \ldots, G_r\}$.

3. For each of the obtained groups G, depending on the number of voxels:

- One voxel in figure 2 on the plane G1: the voxel is entered into the DataR array: the place in the array corresponding to the coordinates of the voxel is marked as 1.
- Two voxels in figure 2 on the plane Gi: based on 2 voxels, a segment is constructed that is also rasterized into n voxels; the corresponding n places in the DataR array are marked as 1.
- Three voxels and more: depending on how many voxels are included in group G from each of the sides AB, BC and CA, the following situations are possible:
  - All voxels are from one side. These voxels make up a segment; places corresponding to these voxels in the *DataR* array are marked as 1.
  - Voxels are from two sides. In the plane there are points forming two segments. We connect the corresponding ends of the segments. The resulting figure is a flat quadrangle, which is divided into two flat triangles. Triangles are rasterized by the rasterization algorithm of a flat triangle.
  - Voxels are from three sides. In the plane there are points from three segments and two of them are connected. We connect, as in figure 3, the ends of the corresponding segments. Then it is necessary to rasterize the resulting flat shape similarly to the

previous step. A special case of this stage is that all the point of all segments is in one group, which means that the entire triangle lies in one horizontal plane.

4.2. Building a voxel vector based on a three-dimensional matrix SET_VECTOR ()

Building a voxel vector consists of the following main steps.

1. Set the seed voxel $V_{current}(0, -1, -1)$.
2. Mark the place $P_{current}$ in the *DataR* space as passed.
3. Enter $V_{current}(N_{current}, N_{prev}, direct)$ into the vector *Path*.
4. Check the possibility of a shift $V_{current}$ in six directions of space $F(V_{current}) = \{\vec{v}_0,...,\vec{v}_S\}$ in the appropriate order (figure 4).
5. Select the first free direction $\vec{v}_i \in F(V_{current}), i = \overline{0,5}$, redefine $V_{current}=(N_{current}+1, N_{current}, i)$, go to step 2.
6. If there is no free direction for the shift $\vec{v}_i \in F(V_{current}), i = \overline{0,5}$, select the previous node in the *Path* array as the current node: $V_{current}=Path[N_{prev}]$, move the current position to the *direct* direction back: $P_{current} = P_{current} - P_{current}$. Go to step 4.

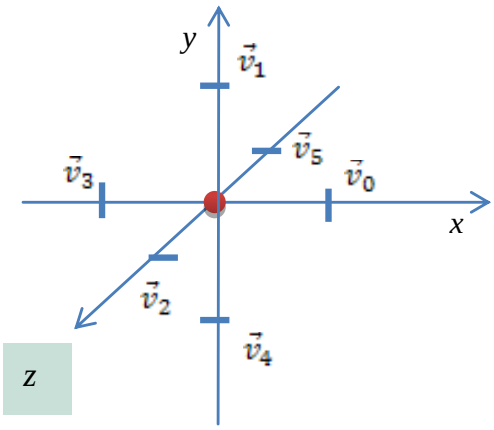

Figure 4. Shift directions for the current node $V_{current}$.

4.3. Placement of object T_0 in placement area PUT_BEGIN ()

Placement of object T_0 consists of the following steps.

1. For all already packed objects $T' = \{T_1, \dots, T_n\}$, their maximum height $h_{max} = \max(h(T_i))$ $\forall i = \overline{1,n}$ is selected; $h(T_i) = \max(P_j, z)$, $\forall j = \overline{1,m}$ where $\{P_1, \dots, P_m\}$ are the coordinates of the voxel points that make up the object T_i.
2. The entry point $P_{begin}(0,0,h_{max} + h_T(T_0))$ is selected.
3. If the object T_0 cannot be restored from P_0, then the object cannot be entered.

4.4. Building NFP placed object

The process of building NFP placed object consists of the following steps.

1. For each of the packed objects $T_i' \in \{T_0'...T_m'\}$, where *m* is the number of packed objects, the NFP of the placed object *T* is constructed relative to T'_i in the packing area *R* **LIST_CONSTRACTION (*T*, *T'i*).**
2. In each of these NFP $V_i \in \{V_{i0}...V_{im}\}$ some nodes $V_i' \in V_i$ are selected.

3. Resulting NFP is $V_{result} = \bigcup_{i=0}^{m} V_i'$.

4.5. Building NFP of two objects LIST_CONSTRACTION (T1, T2) using the "right hand" method.
The process of building NFP of two objects consists of the following steps.

1. Set the seed node $V_{current}$ at the starting point $P_{current} = P_0$.

2. Check the free neighboring nodes (figure 5). If the nodes in all directions of the shift are free, then we shift the node $V_{current}$ in the 0th direction until at least one of the directions is occupied.

3. Mark the place $P_{current}$ in the placement area as occupied.

4. Enter $V_{current}$ ($N_{current}$, N_{prev}, *direct*) into the list vector *Path*.

5. Check the possibility of a shift $V_{current}$ in six directions of space $F(V_{current}) = \{\vec{v}_1,...,\vec{v}_6\}$ in the appropriate order.

6. Select the first free direction $\vec{v}_i \in F(V_{current})$, transfer the object to the vector of this direction. At the same time, so that the object does not come off the surface when moving to the 0th ("right") direction, the priority shift directions are turned so that the 0th direction is in place of the first (by priority) inaccessible. Go to step 3.

7. If there is no free direction for the shift $V_i \in F(V_{current})$, select the previous node in the *Path* array: $V_{current}$=*Path*[N_{prev}] as the current node, move the current position to the *direct* direction backward: $P_{current} = P_{current} - \vec{v}_3$. Go to step 4.

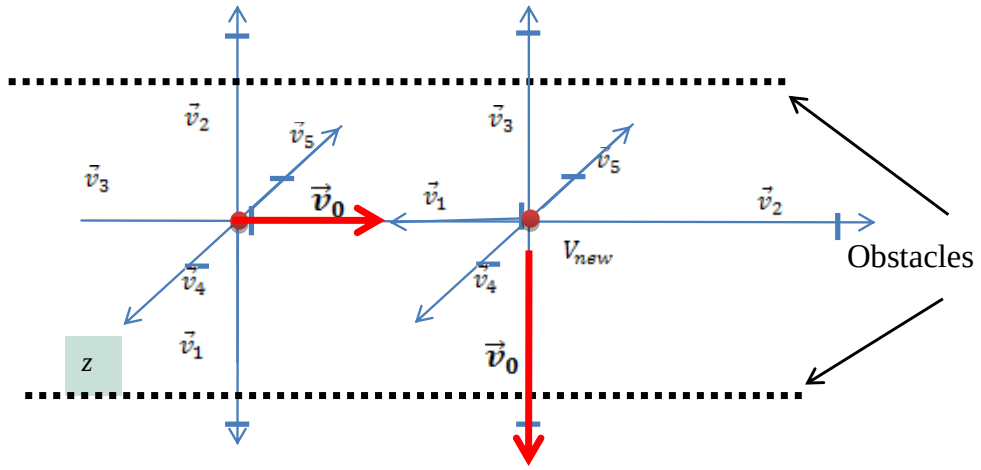

Figure 5. Checking the free neighbouring nodes in the shift directions for the current node $V_{current}$.

5. Analysis of the dense packing algorithm efficiency

To analyze the efficiency of the developed methods for placing three-dimensional objects in a container, a series of computational experiments was performed.

As input data, we used data placed in open databases, as well as data presented in articles by leading researchers in the field of analysis of algorithms for placing three-dimensional objects in a container [4, 5].

Sets of 20, 30, and 40 polyhedra of two, three, and four of each type were used. Objects were packaged in a container with a base size of 30 × 35 units. The packaging efficiency was estimated by the packing density parameter (%).

Table 1 presents the sets of three-dimensional objects and the results of solving the dense allocation problem. The algorithms used were:

- First local minimum (FLM) [4],

- Decremental neighborhood search (DNS))[4],
- Random search (RS))[4],
- First Fit (FF)[6],
- First Fit (FF) + Local Search (LS)[6],
- GRASP[6],
- GRASP + Local Search (LS) [6],
- Simple Heuristic (Voxel representation), considered in this paper [2].

Table 1. Datasets and packing algorithms.

Packing algorithms	Packing density (%)		
	20 objects	30 objects	40 objects
First local minimum (FLM)	17.71	19.7	19.03
Decremental neighborhood search (DNS)	24.2	23.71	24.5
Random search (RS)	21.75	23.71	23.37
First Fit (FF)	17.14	19.42	23.03
First Fit (FF) + Local Search (LS)	22.8	22.91	25.61
GRASP	19.32	18.54	17.89
GRASP + Local Search (LS)	24.47	21.78	20.53
Simple Heuristic (Voxel representation)	24.01	24.09	25.55

An example of dense packing for various sets of objects is presented in figure 6.

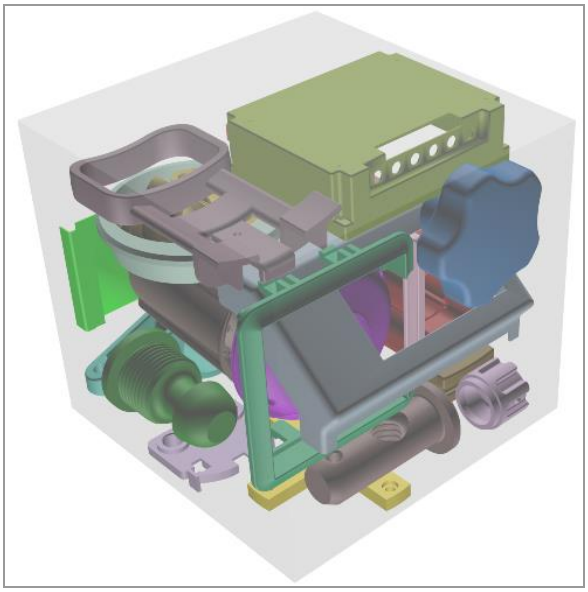

Figure 6. Dense packing for various sets of objects.

The results of analysis of the dense packing algorithms efficiency show that the best packing density is provided by:

- GRASP + Local Search (LS) algorithm (24.47%) for 20 objects;
- Simple Heuristic (Voxel representation) algorithm (24.09%) for 30 objects;
- First Fit (FF) + Local Search (LS) (25.61%) for 40 objects.

The packing density of the objects obtained by the voxel-based representation is slightly lower, since a simplified optimization procedure algorithm was used. Despite this, it should be noted the high efficiency of the method, because with certain accuracy parameters, it allows you to pack objects faster. The main advantages of the voxel approach are the correctness of the solution, that is, small changes in the source data do not lead to changes in the results, as well as the speed and reliability of the basic logical operations.

6. Conclusions

The paper considers an approach to solving the problem of packing complex three-dimensional objects in a parallelepiped container based on the use of a discrete-logical (voxel) representation of information, which allows one to obtain results that are different in time and accuracy of calculation.

Analysis of the dense packing algorithms efficiency shows the best packing density (average 24.72%) provided by GRASP + Local Search (LS) algorithm, Simple Heuristic (SH) algorithm and First Fit (FF) + Local Search (LS).

Also, the data obtained in the framework of the study show the actual independence of the packing time of objects from the accuracy of approximation of objects by polygons. This has a significant impact on the implementation of the algorithm for packing objects in object space.

References

[1] Meet the Fuse 1 [Electronic resource]. — Access mode: https://formlabs.com/3d-printers/fuse-1/ (20.12.2019)

[2] Verkhoturov M, Petunin A, Verkhoturova G, Danilov K and Kurennov D 2016 The 3D Object packing problem into a parallelepiped container based on discrete-logical representation *IFAC-PapersOnLine* **49** 12 pp 1–5

[3] Verkhoturov M, Verkhoturova G and Yagudin R 2012 Geometric optimizing modeling in the systems of 3-D objects placing *Workshop on computer science and information technologies CSIT'2012* (Ufa – Hamburg – Norwegian Fjords) pp 176–179

[4] Stoyan Yu, Gil N, Scheithauer G and Pankratov A 2004 Packing non-convex polytopes into a parallelepiped *TU Dresden* 32 p (Preprint MATH-NM-06-2004)

[5] Egeblad J, Benny K and Marcus B 2009 Translational packing of arbitrary polytopes *Elsevier. Computational geometry* **42** 4 pp 269–288

[6] Verkhoturov M, Verkhoturova G., Yagudin R. 2013 On one solution to the problem of dense packing of non convex polyhedra based on a dynamic scheme for determining the conditions of mutual non intersection *Workshop on information technologies for intelligent decision making Support ITIDS'2013* (Ufa, Russia, May 2013) pp 198-207

[7] Kaufman A and Shimony E 1993 3D scan-conversion algorithms for voxel-based graphics *Proc. ACM Workshop on Interactive 3D Graphics,* (Chapel Hill, NC, October 1986) pp 45–76

[8] Kaufman A 1987 Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes *Computer Graphics* **21** 4 pp 171-179

[9] Samuli Laine Tero Karra 2010 Efficient sparse voxel octrees – analysis, extensions, and implementation *NVIDIA Research*

[10] Requicha A A G 1980 Representations of rigid solids: theory, methods and systems *ACM Computing Surveys* **12** 4

[11] Wohlers T 1992 CAD meets rapid prototyping *Issue of Computer-Aided Engineering* **11** 4

[12] Baumgart B 1975 A polyhedron representation for computer vision *Proceedings of the National Computer Conference and Exposition* (ACM, Anaheim, California, 1975) pp 589-596