Computers & Graphics (2018)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Fabrication Oriented Shape Decomposition Using Polycube Mapping

Filippo A. Fanni[a], Gianmarco Cherchi[a], Alessandro Muntoni[a,b], Alessandro Tola[a], Riccardo Scateni[a]

[a] *Department of Mathematics and Computer Science, University of Cagliari, Cagliari, Italy*
[b] *Visual Computing Laboratory, ISTI-CNR, Pisa, Italy*

ARTICLE INFO

Article history:
Received October 3, 2018

3D Meshes, Geometry Processing, Polycubes, Fabrication

ABSTRACT

In recent years, fabrication technologies have developed at a breakneck pace. However, some limitations on shape and dimension still apply both to additive and subtractive manufacturing, and one way to bypass them could be the partition of the object to build. We present here a novel algorithm, based on the polycube representation of the original shape, able to decompose any model into smaller parts simpler to fabricate. We first map the shape in a polycube and, then, split it to take advantage of the polycube partitioning. In this way, we obtain quite easily a partition of the model. In this work we also study and analyze pros and cons of this partitioning scheme for fabrication, when using both the additive and subtractive pipelines. Our proposed partitioning scheme is computationally light, and it produces high-grade results, especially when applied to models that we can map onto polycubes with a high compactness value.

1. Introduction

The introduction of cheap and small 3D printers has boosted the research in the field of digital model representation for fabrication. Novel algorithms and techniques flourished to let almost everybody reliably 3D print accurate and cheap reproductions of digital objects. At the base of this explosion, there are many manufacturers which are selling low-priced entry-level 3D printers, leading to a sound diffusion between hobbyists. As smaller and cheaper a 3D printers is, as fewer functionalities and features, compared to the high-level ones, it has. A noticeable difference is the **size** of the printing chamber (the maximum printable volume). The only possibility to print big objects is, thus, to decompose them into multiple portions, print them separately and, later on, reassemble the object.

Besides 3D printing, that we can call *Additive Manufacturing*, a different approach to the fabrication of digital shapes, usually called Machining or *Subtractive Manufacturing*, has a broad diffusion in the field of mechanical engineering. This latter approach makes use of CNC milling machines and has been routinely used since decades in the industry, to fabricate parts out of metals blocks or, sometimes, other materials like wood or foam. As with 3D printing, the dimension of the object can be a constraining factor, which we can address with the same techniques.

In both 3D printing and machining, other essential constraints apply to the **shape** of the object, and a way to bypass them is to subdivide the object into pieces that satisfy the required features:

- a 3D printer cannot produce, without introducing extra-structures - the supports -, parts with an overhanging larger than a fixed amount, usually set at 45 degrees;
- a 3-axis milling machine can produce only parts which are height-fields with a flat base;
- a 4-axis milling machine can produce parts which are radially height-fields but machining a piece at a time.

A solution to this class of problems is a shape decomposition guided by the above constraints and size constraints. One can obtain a straightforward decomposition using cutting planes to fit each part in size, but it would probably be meaningless in shape. Moreover the cutting planes are keen to cut other portions of the shape in an uncontrolled way. On the other hand, it could be difficult to control the size of parts obtained using fancy

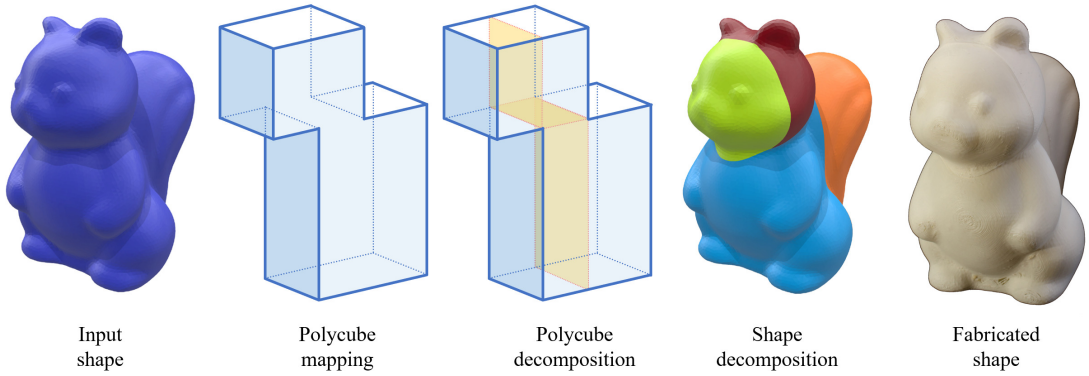


Fig. 1. Our pipeline, from left to right: the mesh representing the input shape; the polycube computed from the input model; the polycube decomposed in orthogonal parallelepipeds with our sweeping algorithm; the decomposition mapped back onto the input shape; the final fabricated real object.

and efficient decompositions which take into full account the semantics of the shape.

We propose here a simple and low-cost - computational wise - approach to the decomposition of a three-dimensional shape which passes through its parametrization in polycube space. Polycubes are simple polyhedra composed only of orthogonal faces. This compact representation of a mesh has been proposed first by Tarini et al. [1], to obtain a seamless texturing. Polycubes have to respect three main constraints: axis-aligned faces, only 90-degree dihedral angle and integer coordinates. These restrictions cause the polycubes to be a simplified representation of the original mesh which can catch the low-frequency semantics of the shape. Using the state-of-the-art polycube generation software Polycut [2], we can control the decomposition of the input shape fine-tuning the parameters which determine the compactness and fidelity of the result. This choice allows us to produce decompositions in a time varying from seconds to few minutes, letting us converge timely to the desired solution.

Our main contributions are as follows:

- A pipeline to obtain an object decomposition that is guaranteed to be **printable** with a 3D printer of a given chamber volume without or with reduced use of supports.
- A checker to verify if the obtained decomposition could be **milled** using a 3-axis or a 4-axis milling machine.

It is worth to mention again that our pipeline strictly relies on the preliminary decomposition induced by the polycube mapping. Without an efficient polycube generator, the whole process does not hold.

We presented preliminary results in “Polycube-based Decomposition for Fabrication” [3]. In this new article, we propose a substantial improvement of the previous partitioning pipeline and a new result set.

2. State of the art

Our work and analysis are related to different research topics that we cover separately in the following sections.

2.1. Polycube maps

Polycube maps offer a compact and simplified representation of a digital model, using a set of face-connected orthogonal parallelepipeds. The polycube map is a bijective mapping function between the original model and the polycube space so that every vertex, triangle (and tetrahedron, in the volumetric case) of the model is mappable onto the polycube. As a consequence, it is possible to modify the polycube’s elements, and, then, map back the result to the original model. A significant advantage of this approach is that every single part of the decomposition is a parallelepiped, all the dihedral angles are right, and the modification is simple and efficient. Its original usage, proposed in [1] exploits this property using the surface of the polycube as a domain for texturing the object. When the polycubes offer a low distortion mapping, this technique is efficient and has almost no drawbacks.

Polycube mapping can apply to trivariate spline fitting, volumetric texturing and hexahedral meshing. By its very nature, any polycube can be trivially gridded, obtaining an utterly regular hex mesh [4]. If we extend the polycube mapping to the whole, volumetric, domain, it is possible to map this hex mesh to the original shape. The resulting hex mesh is highly regular (with the topology computed on the polycube parametric space), and its quality depends on the polycube mapping, with each corner in the polycube mapped back to the hex mesh as a singularity.

2.1.1. Polycube-map construction

There are several algorithms in literature for creating polycubes, each one addressing specific characteristics. In 2008 Lin et al. [5] proposed the first automatic algorithm for polycube generation. In 2009 He et al. [6] proposed an alternative method, based on the principle of *Divide et Impera*. Gregson et al. in 2011 [7] analyzed the problem of computing hex meshes using the polycube parametrization. To do so, they developed a *Deformation based* approach to create polycubes. In 2013 Livesu et al. [2] proposed an approach (Polycut) based on the use of a graph-cut classification of the triangles of the mesh, followed by an hill-climbing optimization of the boundaries. In the algorithm one can set a single parameter to tune the fidelity vs. compactness trade-off, thus allowing for several possible - coarse to fine - polycube mappings. Another approach is the one

by Huang et al. in [8]: they obtain polycubes aiming to keep low the distortion and the corner count. In our work, we use PolyCut for generating the polycube maps, but any other method can produce our input polycubes.

More recently another work focused on polycube post-processing. Cherchi et al. [9] in 2016, proposed a shape optimization, beneficial for the analysis presented in this paper. This work introduced an algorithm to align polycubes' singularities. A polycube of "good-quality", with a low number of corners and proper alignment of singularities is very helpful in our work.

2.2. *Fabrication*

Fabrication is a fast emerging field of research in geometry processing. It collects the study and implementation of all the processes and techniques that can be used to produce real objects from digital models. The most relevant technologies are but are not limited to, 3D printers and CNC milling machines. These represent two different approaches to fabrication, usually referenced as additive and subtractive manufacturing.

Research in the fabrication field is mostly related to additive manufacturing, while a few works address subtractive manufacturing. We will briefly review the most relevant literature regarding both topics.

2.2.1. *Additive manufacturing*

Additive manufacturing (or additive fabrication) makes use of machines that build the final object layer by layer. These machines are the 3D printers, and they can use multiple materials. The most common printers use thermoplastic polymers and deposit the fused filament to build the layer. However, other technologies are available to use other materials, such as liquid resins, metals, and various powders.

This kind of manufacturing does not impose any constraint on the model's shape. However, some models require external support structures, as 3D printers can not directly print steep overhangs or islands (see Figure 2). One has to remove these structures manually after printing, and in an industrial context, they represent a significant waste of material and time. To avoid this waste Hu et al. proposed in [10] an algorithm to subdivide the model in *approximate pyramidal shape*, printable without supports. Herholz et al. in [11] suggest a similar approach, by exploiting the surface deformation to reduce the number of pieces.

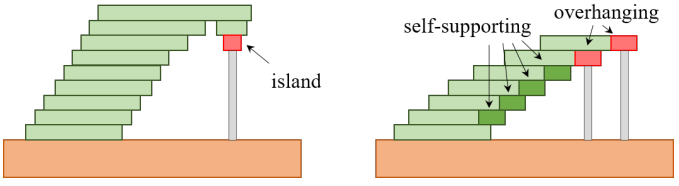


Fig. 2. Islands and overhangs need external support structures.

The hardest constraint imposed by 3D printers regards the size of the object, as it is undoubtedly impossible to print anything greater than the printing chamber. The solution to this problem is, again, to partition the model into smaller portions, print them, and reassemble them back. Many works that face this problem appeared in the last years, and the most remarkable ones are in [12], [13] and [14]. The algorithm proposed by Song et al. in [13] creates self-interlocking structures, to avoid the use of glue or connectors, and it obtains a stable structure that can be disassembled and reassembled multiple times. The algorithm proposed by Hao et al. in [14] tries to minimize the aesthetic impact of seams. Lastly, the algorithm proposed by Luo et al. in [12] generates a partitioning of the input model that optimizes a set of objective functions, including printability of every block in the working volume, assemblability, avoiding small blocks and optimal position of the seams (both for aesthetics or structure). They subdivide the model using cutting planes, and a BSPTree gives the order of cut. This approach does not allow to keep apart semantically separate portions of the object. Furthermore, they don't consider the supports that are necessary to print every block of their partition, which can complicate the assemblability due to the presence of connectors in the planar portions of the blocks. An extensive discussion on pros and cons of additive fabrication, partitioning and related issues can be found in the survey of Livesu et al. [15].

2.2.2. *Subtractive manufacturing*

Subtractive manufacturing, also known as machining or subtractive fabrication, consists in removing material from a starting block until only the desired shape is left. CNC milling machines have a crucial role in mechanical manufacturing since decades, but only recently experiments on automatic free-form shape production started.

Milling, unlike 3D printing, enables manufacturing objects with a large variety of materials, like wood, metal or stone. Despite this significant plus, the usage of subtractive techniques for free-form production still struggles in the digital fabrication field, due to the hard constraints that they impose on the geometry of the objects. There are three main categories of milling machines, which differ for the degrees of freedom of the milling tool.

3-axis. The most diffused, inexpensive and easy to use milling machines can move their tool on the three axes of the Cartesian system and, thus, they have three degrees of freedom. These characteristics limit the class of objects they can produce: millable shapes can be only height-fields with flat bases. In other words, each line parallel to the z axis can cross the shape only once, as shown in Figure 3. Even if there is a vast bibliography on the production of mechanical parts with CNC manufacturing, there is still limited literature on the subject of decomposing generic free-form shapes into a set of millable parts. Alemanno et al. [16] define a user-assisted method for decomposing 3D shapes into height-field in the domain of cultural heritage. Their method is manually driven and overlaps between pairs of blocks are resolved using an interlocking zipper pattern. Herholz et al. [11] use 3-axis milling machines to create millable molds. These molds can be used to obtain the final model by solidifying a liquid material which decants inside the glued molds. Muntoni et al. [17] perform a decomposition in height-field blocks that can be manufactured in a single pass with a 3-axis milling machine, using a fully automatic algorithm in two steps. They first identify all the bounding boxes containing height-fields and then

select a subset determining a partition of the input shape. As they explain, a geometry manufacturable with a 3-axis machine must respect several constraints (such as height-field geometry w.r.t. a given direction, flat polygonal base, etc.). Our work does not focus on complying with these constraints. We first obtain a decomposition of a 3D digital model induced by its polycube, and then we analyze every piece of the decomposition. This analysis also includes a check for the manufacturability of the part with a 3-axis milling machine.

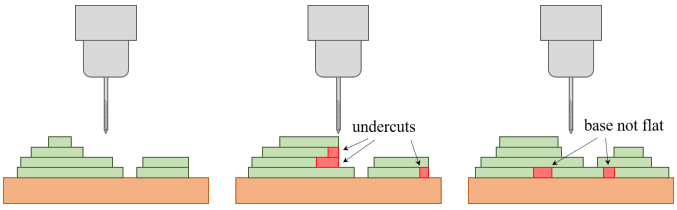


Fig. 3. The model on the left is millable (it is a height-field); the model in the middle is not millable (it is not a height-field and, thus, it has undercuts), the model on the right is not millable even if it is a height-field because its base is not flat.

4-axis or more. More complex machines have higher degrees of freedom, typically moving the tools over four, five, and six axes. These devices impose looser constraints over the machined shape, but, at the same time, they are more expensive than the 3-axis ones, and they require more sophisticated software and analysis which enables the automatic generation of tool-paths. Typically, in fact, the user generates the tool-paths manually based on his or her own experience. It is also possible to add accessories to a 3-axis machine having a 4^{th} degree of freedom given by the rotation axis. This add-on is quite useful since a 4-axis machine can produce all the models that, given a rotation axis, exposes every point of the surface in at least one rotation. This constraint is weaker than the one imposed by the 3-axis machines. Recently Hou et al. [18] improved the results obtained previously by Frank et al. [19], and using the global visibility map (GVM) of the shape can determine the best rotational axes for machining it. The authors show results obtained on mechanical parts and the computational effort reported is in the order of tens of minutes for shapes just more complex than a cube with pockets. Their approach is more exhaustive than our proposed checker but far more expensive in time.

3. Problem overview

The goal of our work is to decompose complex models into simpler parts that better suit limitations in current fabrication processes, both additive and subtractive. We do this computing the polycube map of the shape, splitting the polycube in orthogonal parallelepipeds, and mapping back this partition in the original model. This process allows us to have a set of smaller parts to manage, each one of them with desired features.

As well stated in [15], when planning the production of an object in additive manufacturing, it is possible to decide to partition the object into multiple pieces. This partition can be due to multiple reasons, but one of the most common situations is when the object is greater than the printing chamber.

We propose an alternative way of partitioning the shape. Our method is conceptually simple if a polycube map is available for the input shape. We then induce a partition on the shape using this map, smartly and efficiently. In our experiments, we show the proposed method worked satisfactorily for a variety of examples (see Figure 7).

Our work requires only one parameter: the compactness vs. fidelity term of Polycut for building the polycube. Furthermore, we try to partition the shape keeping in mind the requirements of both additive and subtractive manufacturing.

Our proposed production pipeline can be summarized as follows:

1. We start from a 3D input shape (a triangle mesh representing the surface of the model and a tetrahedral mesh of the interior)
2. We compute its representation in polycube space
3. We partition the polycube in orthogonal parallelepipeds
4. We use the partition computed at step 3. to subdivide the original model in shape space using Boolean operations

All the steps listed above are fully automatic. If the results do not respect the constraints (45 degrees maximum overhang for printing and height-field for milling), we can readily repeat the last step after manually splitting one or more parallelepipeds in polycube space with a plane orthogonal to the Cartesian axes. The splitting is trivial working in polycube space.

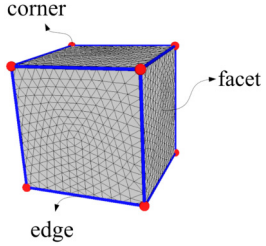
The following sections explain the third and fourth steps of this pipeline which are the primary focus of this work. In Figure 1 there is a sketched representation of the pipeline.

The third step of the pipeline explained in details in Section 4, takes in input the topology of the polycube and, using a queue-based sweeping algorithm, outputs its partition in orthogonal parallelepipeds.

The fourth step (Section 5) maps back each parallelepiped found in the previous step to parts of the original model. We cannot use a simple mapping since it would not produce the flat surfaces we need. We, thus, use the boxes as parameters for intersections with the original shape. The final result is manually evaluated to verify if one or more parallelepipeds needs further partitioning step.

4. Polycube partitioning

As shown in the inset, in the polycube, we call *corner* a vertex with at least three adjacent triangles (or faces) having three different normals and *edge* the shortest rectilinear path of triangle edges that connect two corners. We call *facet* a closed chain of edges and corners containing a set of triangles with the same normal. For the following steps, we can ignore the triangle mesh structure and focus only on these elements.



The primary step of our pipeline is the decomposition of the polycube in orthogonal parallelepipeds. We follow the idea that every concave edge

in the polycube defines a partial decomposition of the model. Since every edge is axis-aligned, it lies at the intersection two planes parallel to xy, yz or zx. By construction, the two planes are orthogonal. The intersections between the two planes and the polycube induce the partial decomposition we mentioned before. Iterating the decompositions obtained visiting all the concave edges we obtain the decomposition in orthogonal parallelepipeds of the input polycube.

We, first, make sure that each corner coordinates is rounded to integer values. In this way - fitting the polycube into an integer lattice - it is simple to create a uniform discrete grid inside the polycube.

We apply a sweep line algorithm along all the three axes, and we split the lattice at every concave edge. In Figure 4, for the sake of compactness, we represent both steps as they were only one pass; notice that we evidence the concave edges marking the explementary convex angles.

This method works fine for all polycubes except for self-intersecting ones. It is, anyway, practically impossible to have self-intersections in the class of polycubes used for the scope of this paper.

We are now ready to compute the cutting planes. We apply the back mapping - from polycube back to the original shape - only to each part's corner. Since we have at our disposal the tetrahedrization of the original shape and the polycube, we rely on them for this mapping. For each corner $P(x, y, z)$ on the surface of the part we determine in which tetrahedron of the polycube it lies, indexing them using an octree, and expressing its position in barycentric coordinates: $\omega_0, \omega_1, \omega_2, \omega_3$. We then compute the new corner position P' applying these barycentric coordinates to the tetrahedron in the original model. If A, B, C, D are the vertices of the chosen tetrahedron, we have that $P' = \omega_0 \cdot A + \omega_1 \cdot B + \omega_2 \cdot C + \omega_3 \cdot D$. The computational complexity is $O(n_c \cdot log(n_t))$, where n_c is the number of corners in the polycube portions and n_t is the number of tetrahedra in the original model.

In this way, we identify a set of eight vertices for each internal parallelepiped of the polycube. We can now compute, for each quadruple of vertices on a face of the parallelepiped, the plane that better approximate them and, repeating it for all the internal parts, obtain the set of cutting planes. We further explain this step in the next section.

5. Cutting planes

To be able to fabricate each part of the decomposed model, as it will be evident in the next section, it is beneficial for 3D printing, and mandatory for 3-axis milling to have at least one *side* planar. With the word *side* here we mean the set of triangles mapped from one facet of a parallelepiped. It's worth to remind that in polycube space since each component is an orthogonal parallelepiped, each face is a planar rectangle. To map back the parallelepiped in $\mathbb{R}^3$ we use the inverse function of the projection in polycube space. This inverse function only seldom maps a rectangle onto a planar portion of the surface: almost always the four vertices of the rectangle do not lie on a plane. We, thus, modify the position of these four vertices so that they will lie on a plane and call this step **flattening**. We are not allowed to change the position of the mesh vertices on the original surface of the input shape, of course, since we do not want to deform the input shape.

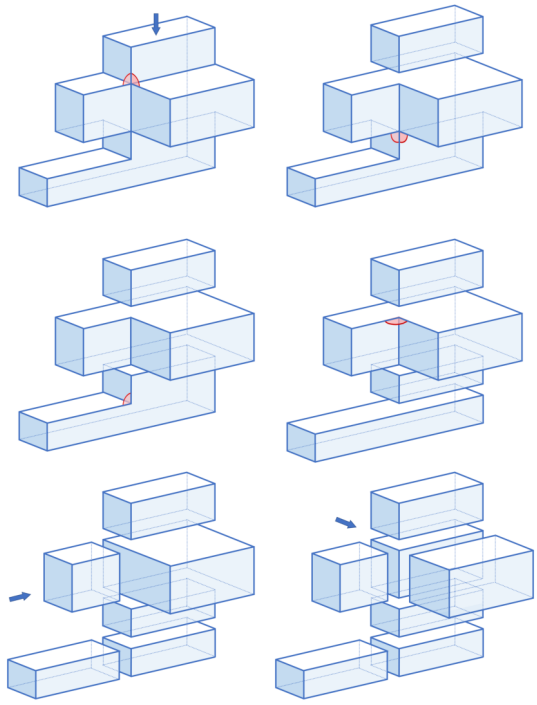


Fig. 4. The space sweeping partitioning of the polycube. We depict here a step in the direction marked by the arrow (top-down): any time we encounter one or more edges delimiting an internal concave angle (for the sake of understanding we mark, in red, the explementary convex angles) we split the polycube complex.

To reach our goal we use an iterative method that works in $\mathbb{R}^3$ using the topology of the polycube and, thus, in the following, we use the term side instead of facet for making this clear. It works as follows, using only a queue Q as the data structure to support the process:

1. Pick an external side of the model
2. Check if one or more of the other five sides of the same parallelepiped are internal sides
3. Put all the internal sides found in Q
4. Take the first side in Q, say f, flatten it and remove it from Q
5. Move to the parallelepiped incident on f not already visited and go to step 2

The process ends when we have visited all the cubes and Q is empty.

5.1. Side flattening

We use different approaches to flatten a side, depending on how many vertices are free to move. If all the **four** vertices are free to move (e.g., for the very first side to flatten) we apply a least square method to find the best fitting plane, and we project the four vertices on it. There are two more possible cases: (i) **two** vertices can move only on a given plane; (ii) **one** vertex can move only on a given line. In both cases, the solution is

straightforward. Finally, consider that when we flatten sides neighbor to already flat ones since we cannot move the edge in common, we use it as a pivot and allow the other two vertices only lay on a plane passing through the pivot.

In figure 5 we illustrate the whole process on a simple example that, for simplicity, is in polycube space.

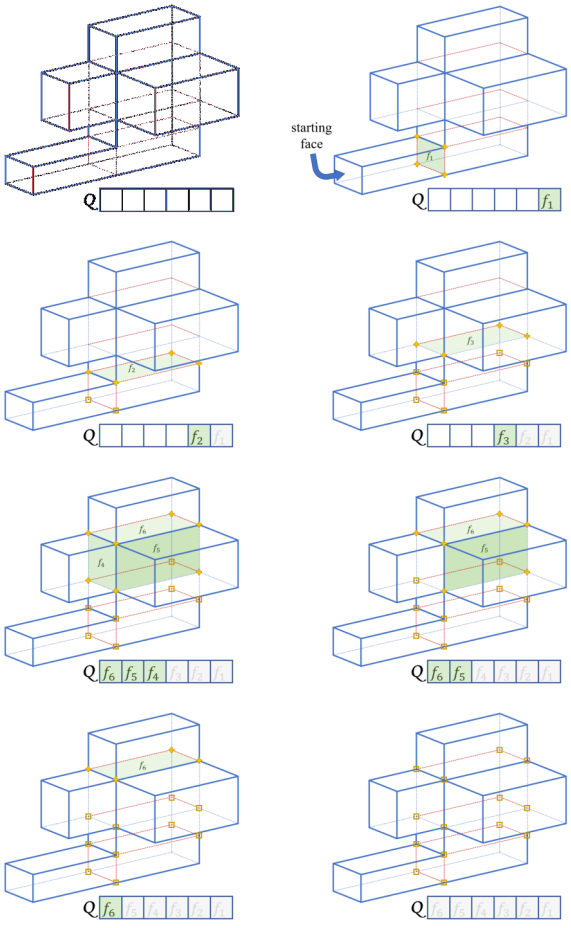


Fig. 5. From left to right and top to bottom we can follow the whole process of flattening. The vertices to be moved are the ones marked in yellow, once fixed they are marked with a yellow square.

6. Final decomposition

Once we have detected all the cutting planes onto which the internal faces lie, the last step is the definition of the geometry of each part of the partition in order to proceed to the fabrication feasibility analysis. To perform this refinement we use exact Boolean operations as described by Zhou et al. in [20], which permit to obtain the surface mesh resulting from a Boolean operation (intersection, union, difference) of two surface meshes. We obtain the final geometry of the part performing an intersection between the surface extracted from the input triangle mesh and a box enclosing the part. The boundaries of the enclosing box are given by the cutting planes of the part and the bounding box of the whole mesh.

If the cut results in more than one connected component, we select only the component containing the four vertices generating the cutting plane, and we ignore the other cuts. Each portion stems from a single orthogonal parallelepiped of the polycube, and therefore it can be trivially split - if necessary - using an appropriate axis-aligned plane that will map back onto the original shape. The result of this final step is a partition made by a set of triangle meshes.

7. Feasibility checking for fabrication

7.1. 3D printing support control

We can fabricate each piece of the resulting partition with a 3D printer since we used the chamber size as a control in the last step. We focused our feasibility check on the usage of supports.

Our algorithm guarantees to generate a set of pieces having from one to six flat polygonal facets. For each piece, we automatically check all the possible printing directions - one to six - given by its bounding cutting planes, and we select the printing orientation that gives the best results in terms of needed supports. The best orientation is the one with less surface triangles exceeding the overhang angle of the 3D printer. Since the number of cutting planes bounding a piece is at most six, the overall complexity of the check is linear with respect to the triangle number of the piece. We analyze the results in the next section.

Note that, as we stated in Section 6, with our algorithm we can split a piece with a new cutting plane if the volume of the part exceeds the chamber size. This operation permits also to avoid any outer supports iterating this split step until having only parts not needing supports. This splitting we add a new flat base to each of the two new pieces - one of them has already a flat face - allowing to choose a new printing direction for both pieces.

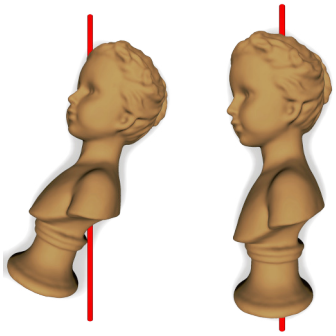
7.2. 3-axis milling checking

While for 3D printing we can guarantee results, for milling we are only able to perform a check on the obtained decomposition to verify the fabrication feasibility. We devised two checkers which allows to state if we can or cannot manufacture a block with both 3- and 4-axis machinery. The 3-axis milling checker is very simple: it just checks if a piece of our decomposition is a height-field. The milling direction is orthogonal to one of the cutting planes, and, again, at most, we perform this check six times per piece. We orient the piece in all its possible milling directions and, for every orientation, we check if it has triangles having normal with an angle greater than 90° for the milling direction. We exclude from the check all the triangles which belong to the polygonal base generated by the selected cutting plane.

7.3. 4-axis milling checking

We introduce a method for checking if an individual part of an object can be feasibly machined with a 4-axis milling machine. One of the main challenges for the fabrication of an object with a 4-axis milling machine is the identification of the rotation axis. This problem has been achieved by [18],

but they focus the study mostly on mechanical and very regular objects with a low number of triangles, emphasizing that the computational effort dramatically increases if the visibility resolution raises. We focus on free-form geometries and, thus, the more reasonable solution is to leave the user to choose the rotation axis, assisted by an automatic initial orientation of the model. We derive the suggested orientation as explained in [17], maximizing the alignment between the global axes and the face normals of the shape. As shown by [17], since this method is a heuristic it can fail. As we show in the inset, the rotation axes automatically detected for *BU* (left) is skewed while the manual choice of taking the vertical line passing through the center of the base (right) is the correct one. Therefore, we provide a tool that allows the user to adjust the orientation if he deems it necessary. This is the only user-controlled step of this checker.



A surface can be (theoretically, see the end of section) manufactured using a 4-axis milling machine if the milling tool can reach every surface point considering all the possible rotations of the model along the selected rotation axis. Since all the possible rotations along the 4th axis are infinite, once we choose the rotation axis, we sample with a small set of angles that generates a family of planes which intersect each other along the selected axis. We, then trace, for each triangle and at every rotation of the model, a ray orthogonal to the plane associated to the i-th rotation and passing through the barycenter of the triangle. If the ray intersects more than one triangle, we mark the farthest away from the plane as visible from the milling tool and, therefore, millable in the present orientation.

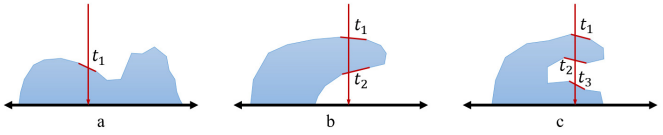


Fig. 6. Three main possible cases of ray-triangle intersection.

As you can see in Figure 6 we can have three primary cases:

- In case *.a* the ray traverses only one triangle.
- In case *.b* the ray traverses a whole portion of the shape and two triangles (one front-facing and one back-facing the milling tool).
- Case *.c* is an example of summation of both previous cases, that can sum up even more; in all these cases the ray traverses three or more triangles.

In the two cases sketched in Figure 6.*b* and in Figure 6.*c* we have triangle belonging to the surface and not directly reachable by the milling tool. If this happens, we need to verify if for some rotation angle the triangle could be visible. We cannot mill all the triangles never visible. Either we change axis, or we remove them from the surface filling the holes with a mesh repairing tool.

Checking the visibility of the barycenter is an approximation. If the barycenter is visible, it does not guarantee that the entire triangle is visible from the milling tool. However, this approximation is good enough for our purposes because in the worst case we lose precision for portions of triangles. In our experiments, these problems never appear. We tested our checker with oversampled meshes (keeping the same geometry but doubling or tripling the number of triangles), resulting with the same percentages of samples visible/not visible, with minimal differences (less than 0.1%).

As mentioned before, the selection of the rotation axis is user-assisted. The sampling number is an input parameter. In our experiments (the results are in Table 2) we used forty rotation planes uniformly distributed in the interval $[0, 2\pi)$.

The whole checking process does not take into account some practical details like the thickness and the height of the milling tool. They depend on the machine used for the manufacturing process. Even if these are essential aspects for the real feasibility of the fabrication of the pieces, they can be integrated into the checker using a configuration left to future extensions.

8. Results and analysis

We applied our pipeline to an extensive set of models, having different characteristics in term of details and complexity. We show a gallery of our results in Figure 7. Other peculiar or fabricated results are reported and commented separately (e.g. see Figure 9).

The polycube of each model is an input for our pipeline and, thus, to make their production process clear, in Table 1 we report, for each model, the compactness factor used to compute these polycubes with Polycut, with the number of resulting orthogonal parallelepipeds. The number of parallelepipeds in the polycube is the number of parts of the decomposition since we do not use any merging post-processing step. We do not report the timing for the cutting planes calculation because they are negligible (always less than a second). We also do not report the timing for polycube computation and for the generation of the pieces using exact Boolean operations since we use external tools for these steps. To give an idea of the order of magnitude, the computation of the polycube map never exceeds three minutes, and Boolean operations always stay under one minute, for all models except for *Fandisk*. The longest step of the whole pipeline is the polycube computation, that is the pre-processing step. The entire timing is, by no means, a problem when compared to the fabrication time.

We do not assure that our partition is optimal in term of the number of parts, but it is unquestionably easy and fast to compute. We do not propose, in fact, to improve in absolute the results in [12], but we suggest how it can be possible to partition a mesh for fabrication purposes with a simple method controlled by the user and using just one parameter in the whole pipeline (the polycube compactness). Our work gives excellent results in efficiency.

Model	Polycube compactness	Parts
Angel	5	4
Bird	4	5
Bu	9	9
Dea	7	4
Duck	5	5
Fandisk	4	50
Hole3	3	18
Max Plank	7	3
Moai	3	9
Ramses	9	9
Sphynx	10	6
Squirrel	3	4

Table 1. Results of the polycube computation on the models used in our experiments. We optimize the polycubes with the algorithm presented in [9]. The second column lists the compactness values of the polycubes.

The compactness of the polycube influences our final decomposition. The use of optimized polycubes as described in [9] leads to a better result, with a sound reduction of small and not semantically relevant pieces.

8.1. Partitioning in additive manufacturing

Since the partitioning allows the complete model to be larger than the printing chamber, with our pipeline, it is possible to print almost any free-form shape, using supports to handle overhanging features.

For every part of our partition, we analyzed the surface percentage needing support during the fabrication. The base for the printing is the flat face giving the lowest percentage. We analyze our results comparing the percentage of the surface of the whole mesh needing supports with the percentage of surface needing supports in our decomposition (Table 2, columns %3DPST and %3DPPST). Our decomposition allows fabricating the final model guaranteeing to have less percentage of surface needing support. Additionally, since the seams between pieces are planar, we can guarantee that the matching areas are as regular as the production process allows, and, therefore, the parts accurately match during the assembly.

8.2. Partitioning in subtractive manufacturing

Subtractive technologies impose stricter constraints on the model shape. For 3-axis fabrication, our polycube-based decomposition does not always produce a suitable partition. Even if we have flat faces, we cannot guarantee that the pieces are height-fields. Testing our 3-axis milling checker on our results, we verified that the height-field constraint is too hard to respect without taking into account specific precautions during the decomposition. As we can see in Table 2 (column %3AM), only five pieces of our decompositions (two belonging to the same model) have tiny percentages of surface that cannot be reached by the milling tool because occluded (less than 3%). These pieces can be manufactured using 3-axis milling machines, at the cost of losing the details of the occluded parts and, in some cases, introducing discontinuities between adjacent blocks if the occlusion involves one of the flat faces of the block. However, no pieces of our decompositions are strictly height-fields, and none of our results is composed only by pieces with negligible percentages of non-millable surface. This is a limitation to our method, and in Section 10 we propose some possible solutions to it. We compare two of our results with [17] in Figure 9. Our method guarantees a regular decomposition inherited from the polycube partitioning, where every block can be produced with 4 axis milling machines or 3D-printed without supporting structures. On the other hand, the method proposed in [17] guarantees blocks that can be milled using 3 axis milling machines..

The use of 4-axis machines allows for more flexible constraints but requires to identify first the rotation axis. Using the checking procedure described in Section 7 we demonstrate the feasibility of almost all the parts obtained from the experimented models. The results show that the machining tool cannot reach only a limited percentage of the surface (Table 2, column %4AM). In Figure 8 we show the chosen axes that would guarantee the fabrication of each piece of the *Duck* (note that this model cannot be fabricated with the 3-axis technology). All our results use the hypothesis of an ideal machining tool of indefinite length and infinite narrow size. Should one manufacture the parts, it would require to revise the checking procedure to take in account size, length, and shape of the tool. The change would not substantially modify the results.

8.3. 3D printing examples

We fabricated some of the computed decompositions using additive manufacturing.

We fabricated five models: *Duck*, *Sphynx*, *Angel*, *Max Plank*, and *Squirrel*. These models have different polycube mappings, all very simple and they decompose, respectively into five, six, four, four, and three pieces (see img:mosaicFigures 7, 11, and 1 for the decompositions). Photos of the models of the *Duck* and *Angel* are in Figure 10. In Figure 12 we show photos of *Max Planck* with the total height (24.5 cm) shown. The chamber of our 3D printer, a Flashforge Creator Pro, is $227 \times 148 \times 150$ mm, and thus it could not be possible to print the model in this size without decomposing it. Furthermore, a large number of external supports would have been necessary to fabricate this model without partition it.

9. Limitations

The polycube-based partitioning is not well suited when used to decompose models with large almost flat surfaces not orthogonal. The *Fandisk* model, for instance, is quite regular and straightforward but, due to its geometric features, decomposes in 50 portions (see Figure 13). This characteristic is a drawback when compared to pure semantical approaches. As an example, a manual segmentation can easily partition the fandisk in much fewer portions. But the fully automatic pipeline still makes our approach advantageous on models without these characteristics. Since we use optimized polycubes, we decompose in as few parts as possible, but we can still produce some small pieces as one can notice in the Ramses model of Figure 7. Another

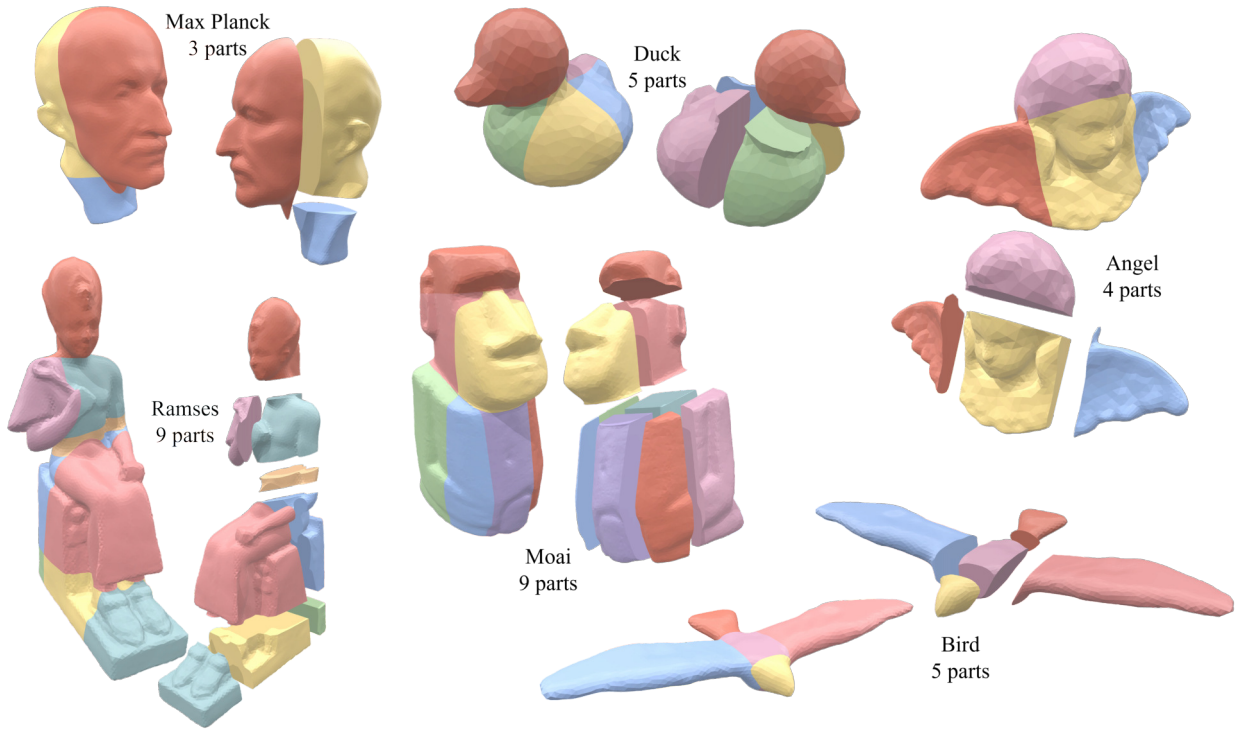


Fig. 7. For each shape, on the left the whole partitioned model, and, on the right, its exploded view.

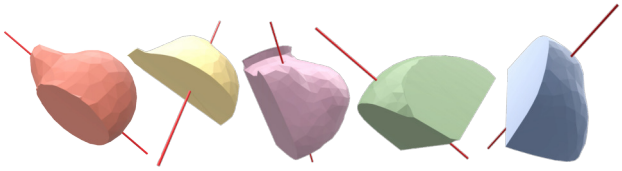


Fig. 8. The five parts of the *Duck* model decomposition, all with at least one flat face. The red cylinder is the rotation axis used for the 4-axis milling.

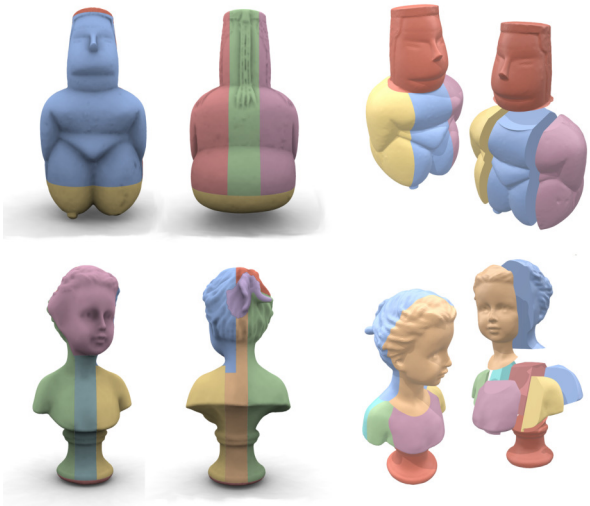


Fig. 9. Decompositions of Dea and Bu models obtained with the the method of [17] (left) and with our method (right).

Fig. 10. The models of *Duck* and *Angel*. On the top left the five printed parts of the *Duck*, and on the top right two views of the assembled model. On the bottom left the four parts of the *Angel*, and on the bottom right the assembled model.

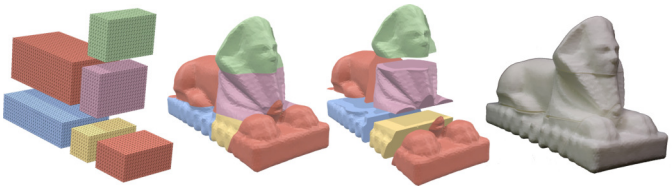


Fig. 11. The *Sphynx* model, from left to right: the polycube mapping; the partitioned model; the exploded set of parts; the ABS model.

Model	%3DPST	BID	%3DPPS	%3DPPST	%3AM	%4AM
Angel	4.6	0	0.0	0.0	13.7	0.0
		1	0.0		41.2	0.1
		2	0.0		17.7	0.0
		3	0.0		22.0	0.0
Bird	32.0	0	0.0	0.0	45.3	0.0
		1	0.0		2.9	0.0
		2	0.0		44.7	0.0
		3	0.0		16.6	0.0
		4	0.0		44.8	0.0
Bu	11.8	0	17.8	10.1	47.5	0.0
		1	5.0		16.9	0.0
		2	5.3		32.7	0.1
		3	5.9		28.8	0.0
		4	1.8		36.1	0.0
		5	1.5		48.0	0.1
		6	2.9		12.7	0.1
		7	4.6		9.4	0.0
		8	3.9		11.6	0.0
Dea	11.7	0	0.0	0.06	39.6	0.1
		1	0.0		0.9	0.0
		2	0.1		36.7	0.3
		3	0.0		0.3	0.0
Duck	14.5	0	3.0	0.7	37.9	0.0
		1	0.0		32.5	0.0
		2	0.0		0.4	0.0
		3	0.0		14.1	0.0
		4	0.0		23.4	0.0
Max Plank	21.7	0	0.0	0.0	6.2	0.0
		1	0.0		32.6	0.0
		2	0.0		31.2	0.0
Moai	6.0	0	0.0	0.1	11.6	0.0
		1	0.0		21.3	0.0
		2	0.0		1.8	0.0
		3	0.0		28.8	0.0
		4	0.5		23.9	0.0
		5	0.0		26.2	0.0
		6	0.0		19.8	0.0
		7	0.0		7.1	0.5
		8	0.0		30.0	2.4
Ramses	3.0	0	1.7	1.8	48.2	0.0
		1	0.0		16.6	0.0
		2	0.6		16.4	0.0
		3	4.3		23.4	0.0
		4	0.0		23.4	0.0
		5	1.4		34.0	0.0
		6	0.6		36.9	0.0
		7	3.0		28.8	0.0
		8	0.0		20.7	0.0
Sphynx	9.8	0	0.0	0.2	29.4	0.9
		1	0.0		29.9	0.0
		2	0.5		30.9	0.1
		3	0.0		37.7	0.4
		4	0.0		12.6	0.0
		5	0.0		24.3	0.0
Squirrel	9.8	0	0.0	0.0	30.1	0.6
		1	0.0		29.6	1.2
		2	0.0		6.9	0.0
		3	0.0		23.7	0.0

Table 2. Summary of the fabricability of most of the models listed in Table 1. The column labels have the following meanings: percentage of surface covered by supports if the entire model is printed (%3DPST); block identifier (BID); percentage of surface covered by supports for each piece (%3DPPS); percentage of surface covered by supports on the entire subdivided model (%3DPPST); percentage of surface not visible from the machine tool during the 3-axis milling machining (%3AM); percentage of samples not visible from the machine tool during the 4-axis milling machining (%4AM).

Fig. 12. The *Max Planck* model fabricated. The three separated parts on the left, and the assembled model on the right.

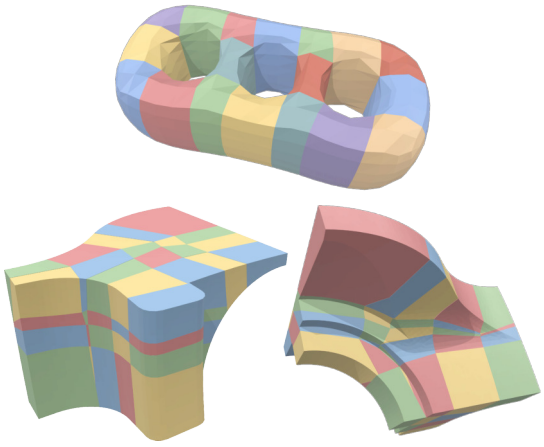


Fig. 13. The *Hole3* model decomposed in eighteen parts (top) and the *Fandisk* model decomposed in fifty parts (bottom).

limitation of our pipeline is its application to models like the *Hole3* one. This model has genus three and is an elementary CSG object that a human could easily split into two parts with flat bases which would print with no supports. Our algorithm decomposes it in eighteen pieces (see Figure 13) because each hole induces additional partitions in polycube space. Theoretically, the compactness of the polycube can be reduced to one single cube but only for objects of genus zero. When the genus is higher than zero, the theoretical limitations do not allow a compact subdivision.

10. Conclusion and future work

We presented a simple and effective, polycube based, decomposition scheme able to manipulate digital shapes in view of their fabrication. Our method allows fabricating any shape using a 3D printer of user's choice. It also includes two checking procedures that permit to verify if the decomposition is usable in 3- and 4-axis milling.

We plan to improve our proposed partitioning pipeline in several ways.

One improvement is related to the fabrication with 3-axis machine. We always produce parts with a flat base but only this property does not guarantee that the parts are height-fields. A solution for this problem could pass through splitting the not height-field portions into sub-portions, using cutting planes.

The choice of appropriate planes would lead to split a part into height-fields. The iterative application of the splitting step would produce an entirely fabricable set of portions. The optimal choice of cutting planes and the demonstration of the termination of the iterative method, apart from trivial solutions are open issues.

Another interesting topic to further investigate is a post-processing step to reduce the number of portions. A strategy to face this problem pass through the merging of adjacent pieces in clusters. This step would not be trivial, as we would have to apply the right constraints to maintain the partition suitable for fabrication. The constraints are: size, to avoid to generate clusters greater than the printing chamber; and shape, to prevent the increase of supports and to make sure that milling constraints are still satisfied.

Acknowledgements

We are indebted to Marco Livesu for his Cinolib library[21] that we used to store and process the 3D models. We also used the CGAL library [22] to implement some functions of the tool described in 7, LibIGL [23] for performing the boolean operations, and CG3Lib [24] for basic algorithms and data structures. This work was partly financed by the DSURF PRIN 2015 (2015B8TRFM) project.

References

[1] Tarini, M, Hormann, K, Cignoni, P, Montani, C. Polycube-maps. ACM Trans Graph 2004;23(3):853–860. doi:10.1145/1015706.1015810.

[2] Livesu, M, Vining, N, Sheffer, A, Gregson, J, Scateni, R. Polycut: Monotone graph-cuts for polycube base-complex construction. ACM Trans Graph 2013;32(6):171:1–171:12. doi:10.1145/2508363.2508388.

[3] Fanni, FA, Cherchi, G, Scateni, R. Polycube-based decomposition for fabrication. In: Giachetti, A, Pingi, P, Stanco, F, editors. Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference. The Eurographics Association. ISBN 978-3-03868-048-2; 2017, p. 1–7. doi:10.2312/stag.20171220.

[4] Livesu, M, Muntoni, A, Puppo, E, Scateni, R. Skeleton-driven adaptive hexahedral meshing of tubular shapes. Computer Graphics Forum 2016;35(7):237–246. doi:10.1111/cgf.13021.

[5] Lin, J, Jin, X, Fan, Z, Wang, CCL. Automatic polycube-maps. In: Chen, F, Jüttler, B, editors. Advances in Geometric Modeling and Processing: 5th International Conference, GMP 2008, Hangzhou, China, April 23-25, 2008. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-79246-8; 2008, p. 3–16. doi:10.1007/978-3-540-79246-8_1.

[6] He, Y, Wang, H, Fu, CW, Qin, H. A divide-and-conquer approach for automatic polycube map construction. Computers & Graphics 2009;33(3):369 – 380. doi:10.1016/j.cag.2009.03.024; iEEE International Conference on Shape Modelling and Applications 2009.

[7] Gregson, J, Sheffer, A, Zhang, E. All-hex mesh generation via volumetric polycube deformation. Computer Graphics Forum 2011;30(5):1407–1416. doi:10.1111/j.1467-8659.2011.02015.x.

[8] Huang, J, Jiang, T, Shi, Z, Tong, Y, Bao, H, Desbrun, M. ℓ1basedd construction of polycube maps from complex shapes. ACM Trans Graph 2014;33(3):25:1–25:11. doi:10.1145/2602141.

[9] Cherchi, G, Livesu, M, Scateni, R. Polycube simplification for coarse layouts of surfaces and volumes. Computer Graphics Forum 2016;35(5):11–20. doi:10.1111/cgf.12959.

[10] Hu, R, Li, H, Zhang, H, Cohen-Or, D. Approximate pyramidal shape decomposition. ACM Trans Graph 2014;33(6):213:1–213:12. doi:10.1145/2661229.2661244.

[11] Herholz, P, Matusik, W, Alexa, M. Approximating free-form geometry with height fields for manufacturing. Computer Graphics Forum 2015;34(2):239–251. doi:10.1111/cgf.12556.

[12] Luo, L, Baran, I, Rusinkiewicz, S, Matusik, W. Chopper: Partitioning models into 3d-printable parts. ACM Trans Graph 2012;31(6):129:1–129:9. doi:10.1145/2366145.2366148.

[13] Song, P, Fu, Z, Liu, L, Fu, CW. Printing 3d objects with interlocking parts. Computer Aided Geometric Design 2015;35:137 – 148. doi:10.1016/j.cagd.2015.03.020; geometric Modeling and Processing 2015.

[14] Hao, J, Fang, L, Williams, RE. An efficient curvature?based partitioning of large?scale stl models. Rapid Prototyping Journal 2011;17(2):116–127. doi:10.1108/13552541111113862.

[15] Livesu, M, Ellero, S, Martínez, J, Lefebvre, S, Attene, M. From 3d models to 3d prints: an overview of the processing pipeline. Computer Graphics Forum 2017;36(2):537–564. doi:10.1111/cgf.13147.

[16] Alemanno, G, Cignoni, P, Pietroni, N, Ponchio, F, Scopigno, R. Interlocking pieces for printing tangible cultural heritage replicas. In: Klein, R, Santos, P, editors. Eurographics Workshop on Graphics and Cultural Heritage. Eurographics Association; 2014, p. 145–154. doi:10.2312/gch.20141312.

[17] Muntoni, A, Livesu, M, Scateni, R, Sheffer, A, Panozzo, D. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. ACM Transactions on Graphics (TOG) 2018;Accepted.

[18] Hou, G, Frank, MC. Computing the global visibility map using slice geometry for setup planning. Journal of Manufacturing Science and Engineering 2017;139(8):081006. doi:10.1115/1.4036423.

[19] Frank, MC, Wysk, RA, Joshi, SB. Determining setup orientations from the visibility of slice geometry for rapid computer numerically controlled machining. Journal of manufacturing science and engineering 2006;128(1):228–238. doi:10.1115/1.2039100.

[20] Zhou, Q, Grinspun, E, Zorin, D, Jacobson, A. Mesh arrangements for solid geometry. ACM Transactions on Graphics (TOG) 2016;35(4):39. doi:10.1145/2897824.2925901.

[21] Livesu, M. cinolib: a generic programming header only C++ library for processing polygonal and polyhedral meshes. 2017. Https://github.com/maxicino/cinolib/.

[22] The CGAL Project, . CGAL User and Reference Manual. 4.11 ed.; CGAL Editorial Board; 2017. URL: http://doc.cgal.org/4.11/Manual/packages.html.

[23] Jacobson, A, Panozzo, D, et al. libigl: A simple C++ geometry processing library. 2016. Http://libigl.github.io/libigl/.

[24] Muntoni, A, Nuvoli, S, et al. CG3Lib: A structured C++ geometry processing library. 2018. Https://github.com/cg3hci/cg3lib.