# Lightweight Mesh File Format Using Repetition Pattern Encoding for Additive Manufacturing

Benjamin Vaissier [a,b], Jean-Philippe Pernot [a,*], Laurent Chougrani [b], Philippe Véron [a]

[a] *Arts et Métiers Institute of Technology, LISPEN EA 7515, HeSam, Aix-en-Provence, France*
[b] *Poly-Shape, 235 rue des Canesteu, Salon-de-Provence, France*

*Keywords:*
Lightweight encoding
Weighted exact cover problem
Repetition patterns
Lattice and support structures
Additive manufacturing
File formats

## A B S T R A C T

To facilitate the transfer, storage and manipulation of intricate parts' geometry whose fabrication has been made possible thanks to the rise of Additive Manufacturing (AM) technologies, an encoding framework reducing the resulting file size has been developed. This approach leverages the fact that many AM parts are presenting repetition patterns, by encoding the repetition of similar geometry chunks. The decomposition of the part into chunks is a complex optimization problem, whose identification as a Weighted Exact Cover (WEC) problem allowed to develop a new heuristic algorithm dedicated to its fast resolution in linear time $O(n)$. The encoding strategy is implemented through a variation of the AMF file standard (for quick adoption of the format by existing software), and also through a new ad-hoc hybrid file format. To demonstrate the efficiency of the approach, the encryption of lattice and support structures through these two encoding strategies are compared to the results of several state-of-the-art encoding approaches. The way this data weight lightening strategy preserves the overall accuracy is discussed while considering different floating points encoding precisions with respect to the AM process requirements. This comparison exhibits file size reductions up to -84% in comparison with file sizes generated by state-of-the-art approaches. Not only the proposed repetition pattern encoding framework allows file size reductions, but it could also be exploited to optimize and speed-up some steps of the Product Development Process (PDP), including process planning phases.

## 1. Introduction

In the context of Industry 4.0, the recent scientific advances in terms of Additive Manufacturing (AM) are revolutionizing the fabrication of complex industrial parts so far inconceivable [1]. Thanks to the design freedom these technologies offer, intricate geometries with numerous small features and extended repetition patterns (e.g. lattice structures, topology optimized geometries or cooling inner channels) are being drawn and manufactured by both private experts and 3D printing enthusiasts. This fourth industrial revolution and the underlying digital transformation have naturally brought some needs for developing new innovative solutions to best exploit the potential of such breakthrough technologies. This is particularly true when considering the models, methods and tools to support the entire Product Development Process (PDP) of those value-added 3D printed products, including the needs to easily store, share and exchange digital data and notably the 3D ones often abundant.

Describing such intricate shapes requires a lot of information and generates large files when using the current AM file standards (i.e. STL or OBJ). As a consequence, AM machines are sometimes experiencing computing difficulties and slow-down when trying to process complex parts. For instance, geometries such as heat exchangers, composed largely of either lattice structures or thousands of intricate cooling channels [2], are sometimes impossible to slice with a reasonable amount of time and memory, because of the number of trajectories that need to be computed [3,4]. When the slice is realized on the AM machine, simultaneously with the actual production, computation time and memory allocation are all the more crucial. An overconsumption of the available resources can cause the build to fail at mid-production, leading to disastrous economical consequences, considering the amount of wasted material, but also the production time spent (several hundred of hours). Furthermore, large files are harder to exchange through communication channels (e.g. e-mails or online 3D geometry platforms), they require a lot of storage space for back-ups or archives, and are slower to manipulate when modifications must be made. These are some of the drawbacks that can slow down the adoption of AM technologies and the development of a full digital thread in the context of the Industry 4.0. Thus, there is a clear need to reduce the weight of the data manipulated all along the PDP, from the early design to the recycling steps, including process planning related phases.

[*] Corresponding author.
*E-mail address:* jean-philippe.pernot@ensam.eu (J.-P. Pernot).

Such data weight lightening should preserve an overall accuracy consistent with the adopted AM technologies.

To tackle this problem, the present work exploits the fact that many features present in AM part's geometries are repeated and can therefore be stored only once, together with the mechanisms to be used to retrieve the complete geometries. Though the application of this approach is not beneficial to geometries with no repeated features (like results from topology optimization for example), it is particularly interesting for parts suited for AM which are demonstrating a repeated feature in at least a portion of their volume (e.g. heat exchangers composed of lattice structures or parallel cooling channels [5], injection molds, static mixers, medical implants, water and gas turbines, crankcases, protection casings with multiple screw holes). This represents an important portion of the parts produced by AM. The underlying NP-complete problem behind the identification of the repetition pattern in such geometries is solved with a new heuristic algorithm that tends to minimize the overall size of the file used to store the compressed data. The repeated features are encoded within the file as the repetition (through 3D translations) of the mesh of a single feature. Through the description of similar features with only one mesh, this strategy avoids the encoding of similar facets. In addition, the way the geometric information are encoded is discussed with a particular focus on the accuracy required for AM, and two file formats are tested. Not only the proposed repetition pattern encoding framework allows file size reductions, but it can also be exploited at various steps of the Product Development Process (PDP): during the meshing and simulation phases so as to consider similar treatments for duplicated features (so as to reduce computation time or to ensure identical meshes), during pre-production modifications and for process planning to automatically repeat a required adjustment or treatment (such as closure of a hole or addition of a machining allowance), or even after fabrication to optimize repeated operations (e.g. automatic support removal). For all these steps, the repetition pattern can be exploited without having to decompress the whole part geometry, by applying the operation on the elementary feature and only repeating the computed outcome. More broadly, the preservation and exploitation of part repetition patterns throughout the production workflow thus permits to optimize and speed up the end-to-end AM process.

The proposed repetition patterns encoding framework works on AM file standards, i.e. on meshes stored in .STL or .OBJ files. Preserving the repetition patterns directly from the CAD model, and exploiting these information all along the PDP would be certainly more efficient. However, when dealing with AM parts, some digital operations are still performed at the level of the 3D mesh: geometry healing, lattice generation (in some cases) and support generation. For the moment, these operations cannot be carried out with classical CAD software, but only with dedicated ones and the geometry must be transferred from one software to another. Thus, preserving the repetition patterns throughout the whole numerical process would require modifications in many pieces of software, and this is not the option that has been considered in this work. Advantageously, the proposed re-identification of the repetition patterns is performed a posteriori, and the encryption into a dedicated file format is thus a functional and robust solution, that adapt smoothly to the current AM preparation process.

The contribution is threefold: (i) the NP-complete problem behind the compression of 3D geometries through repetition patterns is identified and formalized; (ii) a heuristic algorithm for its fast resolution is proposed; (iii) two implementations of this approach through a variation of the AMF (Additive Manufacturing File) standard and through a dedicated new file format are detailed.

This article is structured as follows. After a literature overview of the works focusing on file formats, pattern detection and mesh compression (Section 2), the proposed framework and its file size-reducing mechanisms are introduced in Section 3. The linear problem governing the optimization step of this framework is then identified and formalized in Section 4, followed by the description of the heuristic algorithm proposed for its fast resolution. The results presented in Section 5 analyze both the file sizes and compression times when running the proposed encoding strategy on several test cases. They clearly demonstrate the efficiency of this new encoding strategy within the proposed framework. Finally, Section 6 ends this article with conclusions and perspectives.

## 2. Related works

The work presented in this article covers several literature domains: *file formats* when considering the way geometric information can be stored and read in an efficient manner, *pattern detection* to identify patterns in geometric models, *mesh compression* to reduce the size of the data with or without loss of information.

*File formats.* The inadequacy and inefficiency of AM file formats have been pointed out by many authors. In recent articles, various file representations of AM data have been compared, including industrial standardized formats [6], academical representations [7], and CAD native formats [8]. Among other points, it has been identified that many superior standardized formats adapted to AM (such as AMF) are struggling to be adopted within the industrial sector, while the earliest simpler file formats (such as STL) are still widely used in spite of their poor definition level. By comparing various standardized file encoding formats, the poor performances of AMF have also been identified, especially regarding the time required to open and save lattice and organic structures [9]. The idea behind the work presented in our article is not only to propose a new lightweight mesh file format but also to show how the proposed encoding strategy can best exploit an existing file format, namely the AMF one. The diverted and enhanced use of the proposed AMF file format could broaden its adoption within the AM industry.

*Pattern detection.* The literature is rich in articles related to pattern detection. For instance, a few years ago, Pauly et al. proposed a framework identifying repetition patterns (including combinations of rotation, translation and scaling) with no prior knowledge of the 3D geometry structure [10]. More recently, Shi et al. completed this work by detecting symmetry and circular repetition patterns using the Lie-Algebra theory [11]. Nevertheless, two types of pattern detection in point clouds and meshes can be distinguished: *symmetry* detection and *similarity* detection. The first focuses on identifying planes or symmetry axes [11,12], whereas the other consists of localizing repetitions of the same feature in different parts of a mesh [10,13,14]. Many articles are applying these detections to urban structures reconstruction, such as building facades [15,16] or tunnels [17]. However, to the best of our knowledge, no article is exploiting these patterns to reduce geometry files size, which is the purpose of this work.

*Mesh compression.* In their recent survey, Maglo et al. gives a comprehensive overview of the existing mesh compression techniques that can be divided into 3 main categories [18]: *single-rate* approaches (compressing and decompressing the totality of an input mesh at once), *progressive* approaches (decompressing the mesh with consecutive level of details) and *random-accessible* approaches (enabling the decompression of only a part of the encoded mesh). Considering the AM context, and the needs to

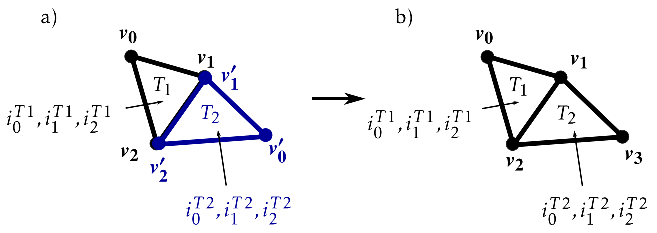

a)
b)
$v_0$
$v_1$
$v'_1$
$v_2$
$v'_2$
$v'_0$
$v_3$
$T_1$
$T_2$
$i_0^{T1}, i_1^{T1}, i_2^{T1}$
$i_0^{T2}, i_1^{T2}, i_2^{T2}$

**Fig. 1.** Vertex Indexing (VI) mechanism avoiding duplicated vertices.

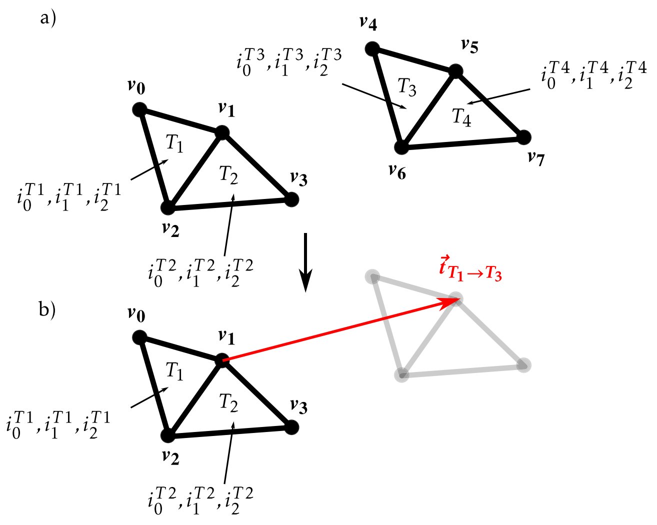

a)
b)
$v_0$
$v_1$
$v_2$
$v_3$
$v_4$
$v_5$
$v_6$
$v_7$
$T_1$
$T_2$
$T_3$
$T_4$
$i_0^{T1}, i_1^{T1}, i_2^{T1}$
$i_0^{T2}, i_1^{T2}, i_2^{T2}$
$i_0^{T3}, i_1^{T3}, i_2^{T3}$
$i_0^{T4}, i_1^{T4}, i_2^{T4}$
$\vec{t}_{T_1 \to T_3}$

**Fig. 2.** Repetition Pattern (RP) encoding mechanism exploiting the isometric properties of facets.

exchange with existing machines whose exchange protocols are standardized, only single-rate compression processes are of interest (even if they employ quantization as described in Section 5.1). In this domain, many articles have proposed algorithms demonstrating compression rates ranging from 11 bit per vertex (bpv) to 1.8 bpv (under certain circumstances). Some of them will be compared to the encoding framework proposed in this article. Regarding the reduction of AM geometry file sizes, implicit slicing has also been investigated [19].

As a conclusion, though many works have been focusing on reducing the size of 3D geometry files, fewer have tried to do it in the context of AM parts [20,21]. Because they sometimes present heavily repeated features, adapted strategies can exploit this particularity to further reduce file sizes as it will be demonstrated in this article. Again, the proposed repetition pattern encoding framework not only aims at reducing file sizes, but it can also help simplifying and optimizing repetitive treatments arising when processing patterned geometries all along the PDP, and notably to support process planning steps. Indeed, the identified patterns could also be used to speed up and further optimize the slicing for instance, but it is not directly the purpose of this paper that focuses on reducing files size. These perspectives are discussed in the conclusion.

## 3. Overall lightweight encoding framework

To reduce the encoding files size of 3D meshes, two encoding mechanisms have been identified and are leveraged within the lightweight encoding framework proposed in this article. Even though the following figures illustrate the mechanisms and algorithms on triangle meshes, the core of the proposed approach can also tackle other mesh types (such as quad-based or tetrahedron-based meshes) for which repetition patterns can be identified. The two mechanisms are as follows:

- *Vertex Indexing (VI) encoding mechanism* associates with each vertex of the mesh an index, and each facet is defined as a list of vertex indices (denoted $i_k^{T_j}$ for the facet $T_j$ in Figs. 1 and 2). It thus generates a non-redundant encryption of the vertices: when the same vertex is shared by two facets of the mesh, this vertex is encoded only once (Fig. 1). Though this mechanism is not adopted in the most widespread STL format, it is implemented within more efficient file standards such as OBJ, PLY or VRML [22].
- *Repetition Pattern (RP) encoding mechanism* reduces the file size by leveraging the presence of repetition patterns within the mesh: isometric facets are not encoded separately but as translations of a unique facet. Several facets can even be encoded through the translation of a multiple facets mesh, as illustrated in Fig. 2.

In order to exploit these two size-reducing mechanisms, an encoding and decoding framework is proposed and illustrated in Fig. 3. First, the optimization problem associated with the VI and RP mechanisms, formalized in Section 4.1, is resolved by a new heuristic algorithm run during the RP optimization step. The interest of using a heuristic is to provide a sub-optimal but good solution in a reasonable amount of time (see Section 4.3). This step is only required when the repetition pattern of the geometry is unknown, in case of a support structure for example (see Sections 5.4 and 5.5). However, in case of a lattice structure generated by the 3D repetition of a unit-cell mesh, the pattern is imposed by the lattice generation. This pattern is a priori known and can therefore directly be exploited during the encoding step in order to reduce the file size through the VI and RP encoding mechanisms. Similarly, if the mesh has been obtained from a CAD model on which patterns have been identified prior to its tessellation, then this information can directly be exploited within the proposed framework, and some steps might be bypassed.

Once the geometry encoded, the resulting file can be easily stored or transmitted to another party, as its size is greatly reduced. At reception, the file can be decoded through the adapted procedure (associated with the encoding strategy). Because of the RP mechanism, a final step of facets stitching is required in order to ensure the watertightness of the mesh (see details in Section 5). This operation is the same as the one required when importing an STL file, and will therefore not be extensively detailed in this article that mostly focuses on the encoding steps. However, as discussed in the conclusion, some of the PDP steps (e.g. slicing or geometry healing) could also benefit from this approach without having to decompress the whole part geometry. In this case, time-consuming operations could be performed just once and the resulting geometries could then be repeated according to the characteristics of the identified patterns.

## 4. Identification of repetition patterns in facet meshes

This section details the algorithm used to identify one of the most promising repetition patterns in a mesh $\mathcal{M}$, i.e. one that best reduces the encoding file weight. This identification results from the resolution of an optimization problem that tries to minimize the weight $w(\mathcal{M}, \kappa)$ of the file resulting from the encoding of $\mathcal{M}$ according to an encoding strategy $\kappa$. As there exists a huge amount of admissible repetition patterns decompositions when considering AM parts, a heuristic algorithm is proposed to solve this problem, and its efficiency is demonstrated in Section 5.

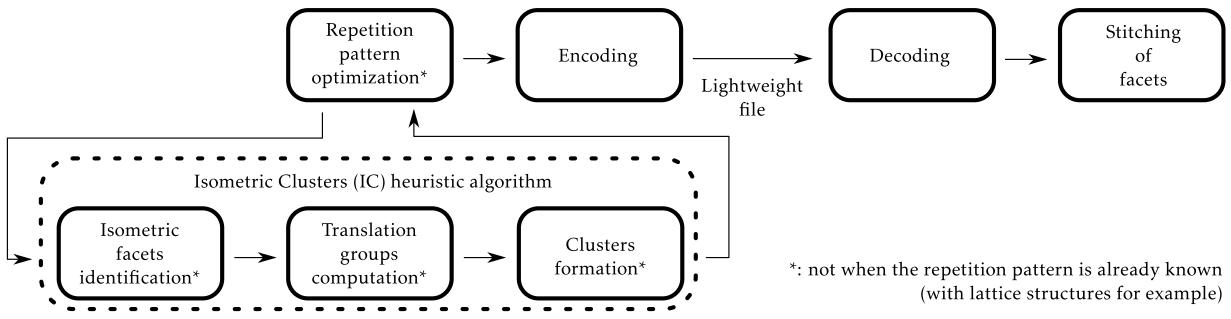

Repetition pattern optimization*
Encoding
Lightweight file
Decoding
Stitching of facets
Isometric Clusters (IC) heuristic algorithm
Isometric facets identification*
Translation groups computation*
Clusters formation*
*: not when the repetition pattern is already known (with lattice structures for example)

**Fig. 3.** Encoding and decoding framework used to generate easily transferable lightweight mesh files.

### *4.1. Mesh decomposition problem formalization*

Let us note $\mathcal{F}$ the set of all facets $f$ of the mesh $\mathcal{M}$, and $S$ a subset of $\mathcal{F}$, i.e. $S \subset \mathcal{F}$. Encoding $S$ according to a certain encoding strategy $\kappa$ generates a certain file weight $w_S^\kappa$ that contributes to the resulting overall file weight. From these definitions, the encoding optimization problem can be defined as a Weighted Exact Cover (WEC) problem, where the objective is to find a decomposition of $\mathcal{F}$ into subsets, minimizing the overall encoding cost function, i.e. the sum of the file weights of the encoded subsets. This can be written as the following Integer Linear Problem (ILP), where each subset is associated with a boolean activation variable $x_S$ defining whether or not $S$ is part of the solution: Minimize

$$w(\mathcal{M},\kappa) = \sum_{S\in\mathcal{A}_S} w_S^\kappa . x_S \tag{1a}$$

subject to

$$\forall f \in \mathcal{F}, \sum_{S\in\mathcal{A}_S : f\in S} x_S = 1 \tag{1b}$$

$$x_S \in \{0, 1\} \tag{1c}$$

where $\mathcal{A}_S$ corresponds to the partition of $\mathcal{F}$ (i.e. the set of all the possible subsets of $\mathcal{F}$). The cardinality of $\mathcal{A}_S$ (i.e. $|\mathcal{A}_S| = 2^{|\mathcal{F}|}$) can be very large when considering AM parts potentially defined by hundreds of thousands of facets. The exact nature of the problem is formalized by Eq. (1b), which constrains each facet of $\mathcal{F}$ to be contained by only one subset of the solution.

The WEC problem has been extensively studied, and has been proven to be NP-complete [23]. One of the most famous strategy to solve this problem is the *Dancing Link (DL)* algorithm [24]. It is a simple greedy recursive back-tracking algorithm allowing to efficiently identify all the solutions of the WEC problem, i.e. all the divisions of the universe $U$ (that is $\mathcal{F}$ in the present case) into disjoint subsets covering all the elements of $U$. The concept is straight forward: first, a subset $S$ is selected to be part of the first solution. Then all the subsets containing at least one facet of $S$ are removed from the list of selectable subsets, in order to ensure the exact nature of the solution (i.e. with disjoint subsets). By identifying the facets of the universe that are not already contained in one of the subsets of the solution, a sub-universe $U' \subset U$ is obtained on which the DL algorithm can recursively be executed. The main issue with the DL algorithm is that it requires to list all the possible subsets of $\mathcal{F}$, and this enumeration of $2^{|\mathcal{F}|}$ subsets can be very time-consuming. Therefore, in this article, a heuristic based on 3 considerations has been developed to solve the identified WEC problem, and its execution is detailed in Section 4.3.

### *4.2. Exploitation of the RP encoding mechanism*

In order to best exploit the previously introduced RP encoding mechanism (and thus find out a decomposition of $\mathcal{M}$ into disjoint subsets that reduces the overall encoding file weight), the isometries between the facets of a subset $S \subset \mathcal{F}$ can be exploited, reducing its encoding weight.

In this article, only the case of isometries by translation between the facets is considered. Two facets $(f, f') \in \mathcal{F}^2$ are isometric by translation if it exists a translation $t_{f\to f'}$ (associated with a 3D vector $\vec{t}_{f\to f'}$) by which $f'$ is the image of $f$ and so that $f' = t_{f\to f'}(f)$. This implies that the two facets have the same angles, but also the same edge lengths, the same normal and the same area. The benefit of only considering isometries by translation instead of all the similarities (such as rotation and scaling) is that no rounding error is introduced. Indeed, the translation of a point by any vector can be obtained only through addition and subtraction, whereas is it not the case for the other similarities for which inaccuracies can be introduced depending on the angle of rotation or the scaling factor.

Let us define an isometric subset as a subset $I \subset \mathcal{F}$ in which all the facets are isometric by translation. Let us define the facet translation group $\Gamma_f^I$ associated with a facet $f \in I$ as the set of all translations transforming $f$ into another facet of $I$:

$$\Gamma_f^I = \{t_{f\to f'} : f' \in I\} \tag{2}$$

Let us also define the translation group $\Gamma^I$ associated with the isometric subset $I$ as the union of the facet translation groups associated with each facet $f \in I$:

$$\Gamma^I = \bigcup_{f\in I} \Gamma_f^I \tag{3}$$

Because of the two size-reducing mechanisms identified in Section 3 (VI and RP mechanisms), one particular case for $S$ is of interest to compute its encoding weight: the case where $S$ can be written as the Cartesian product of a set of facets $F_S \subset \mathcal{F}$ and a set of 3D translation vectors $T_S$ with $\vec{0} \in T_S$. This is noted $S = F_S \times T_S$. This means that if the translation $t$ (associated with each vector $\vec{t} \in T_S$) is applied to each facet $f \in F_S$, each facet of $S$ is obtained only once.

Finally, let us define $V_S$ the set of non-redundant vertices of $S$. Therefore, the computation of the encoding cost of a subset $S$ of facets falls into two cases:

- if $S$ can be written as $F_S \times T_S$:

$$w_S^\kappa = w_V^\kappa.|V_S| + w_F^\kappa.|F_S| + w_T^\kappa.|T_S| + w_{cste}^\kappa \tag{4a}$$

- otherwise,

$$w_S^\kappa = w_V^\kappa.|V_S| + w_F^\kappa.|S| + w_{cste}^\kappa \tag{4b}$$

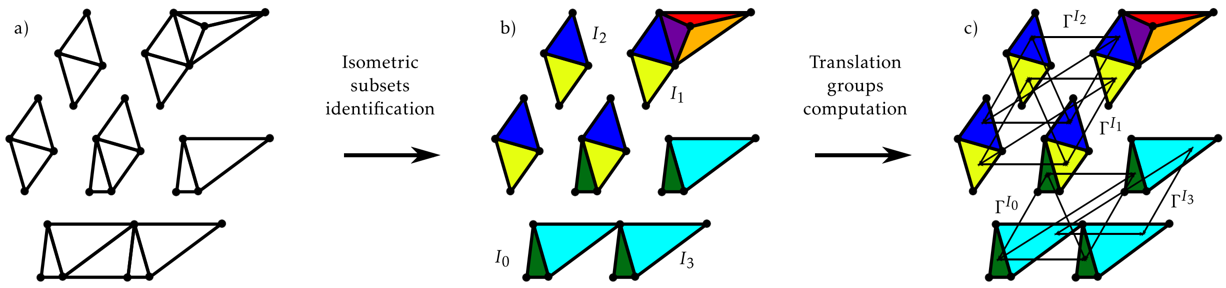


**Fig. 4.** Isometric subsets identification (b) and translation groups computation (c) on a facet mesh (a) : first two steps of the Isometric Clusters (IC) heuristic algorithm.

As it will be discussed in Section 5.3, the coefficients $w_V^\kappa$, $w_F^\kappa$, $w_T^\kappa$ and $w_{cste}^\kappa$ depend on the adopted encoding strategy $\kappa$ and file format in which the information is encoded. Because encoding a facet is usually more expensive than encoding a translation ($w_F^\kappa \geq w_T^\kappa$), encoding Cartesian product based subsets of facets is more beneficial than encoding regular subsets. This is particularly interesting to reduce the resulting file size. These two functions are finalizing the previously introduced ILP definition (Section 4.1).

### 4.3. *Heuristic resolution*

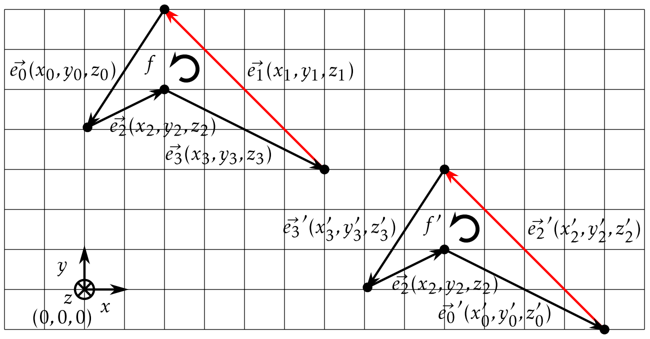
To solve the ILP associated with the RP optimization step, the *Isometric Clusters (IC)* heuristic algorithm has been developed. It is composed of three consecutive stages, namely the isometric subsets identification, the translation group computation and the clusters formation, which executions are detailed in the next paragraphs. In comparison to using a metaheuristic algorithm, the IC heuristic produces a good solution almost instantly. The computation time is especially important for such an encoding problem for which it should not exceed a few seconds.

#### 4.3.1. *Isometric subsets identification*

In the first stage of the IC heuristic algorithm, the isometric facets are grouped together through the creation of isometric subsets. This is illustrated in Fig. 4, where each isometric subset is identified by a specific color. In order to speed up the association of facets, a hash table data structure is used. Through the definition of a hash function invariant by translation, i.e. a function outputting the same value for two facets isometric by translation (this value being the common hash key of the considered facets), a hash table data structure enables to group isometric facets together with a linear $O(n)$ worst-case complexity (compared to the quadratic $O(n^2)$ complexity of a one-to-one comparison). For example, the same technique is employed to reduce the time of triangles stitching during a mesh reconstruction operation [25]. In our implementation, this hash key is defined as the concatenation of the edge vectors coordinates of the facet, starting from the longest edge and contouring the facet according to its orientation, as illustrated in Fig. 5. This hash function is adapted for any oriented polygon regardless of its number of edges. Thus, two facets are guaranteed to have the same hash key if and only if they are isometric by translation. In the rest of this paper, the hash key corresponding to a facet $f$ will be denoted as $H_1(f)$

#### 4.3.2. *Translation groups computation*

Before combining subsets together so as to form facet clusters, each isometric subset $I$ must also be associated with a specific hash key. This second hash function denoted $H_2$, must associate the same key to isometric subsets demonstrating the same repetition pattern, so as to clusterize them together. To do so, in our implementation, a reference facet $f_{ref}^I$ is first selected for each


$\vec{e_0}(x_0, y_0, z_0)$
$f$
$\vec{e_1}(x_1, y_1, z_1)$
$\vec{e_2}(x_2, y_2, z_2)$
$\vec{e_3}(x_3, y_3, z_3)$
$\vec{e_3}'(x_3', y_3', z_3')$
$f'$
$\vec{e_2}'(x_2', y_2', z_2')$
$\vec{e_2}(x_2, y_2, z_2)$
$\vec{e_0}'(x_0', y_0', z_0')$
$y$
$z$
$x$
$(0, 0, 0)$

Hash keys:

$$H_1(f) = H_1(f')$$

$$x_1 y_1 z_1 x_0 y_0 z_0 x_2 y_2 z_2 x_3 y_3 z_3 = x_2' y_2' z_2' x_3' y_3' z_3' x_1' y_1' z_1' x_0' y_0' z_0'$$

$$-4\ 4\ 0\ -2\ -3\ 0\ 2\ 1\ 0\ 4\ -2\ 0 = -4\ 4\ 0\ -2\ -3\ 0\ 2\ 1\ 0\ 4\ -2\ 0$$

**Fig. 5.** Hash function for a fast identification of isometric facets.

isometric subset $I$. The hash key associated with $I$ is then defined as the facet translation group corresponding to $f_{ref}^I$:

$$H_2(I) = \Gamma_{f_{ref}^I}^I \tag{5}$$

Let us consider $I_0$ and $I_1$ two isometric subsets demonstrating the same repetition pattern. To ensure that $H_2$ associates the same key to $I_0$ and $I_1$, their respective reference facet $f_{ref}^{I_0}$ and $f_{ref}^{I_1}$ must be identically located (with respect to $I_0$ and $I_1$) in order to have the same facet translation group $\Gamma_{f_{ref}^{I_0}}^{I_0} = \Gamma_{f_{ref}^{I_1}}^{I_1}$. Therefore, in our implementation, the reference facet $f_{ref}^I$ of each isometric subset $I$ is defined as its left-most facet (i.e. having the vertex with the smallest x-coordinate. In case of equality, the disambiguation is done on the vertices y-coordinates, and then on the z-coordinates).

Let us now consider any isometric subset $I$. For the identification of the translation group $\Gamma_{f_{ref}^I}^I$, each facet $f \in I$ must be associated with a reference point $r_f$, identically located with respect to $f$. Considering two facets $(f_0, f_1) \in I^2$, the associated reference points $r_{f_1}$ and $r_{f_0}$ must ensure that $f_1 = t_{f_0 \to f_1}(f_0)$ where the associated 3D translation vector of $t$ is $\vec{t}_{f_0 \to f_1} = r_{f_1} - r_{f_0}$. The use of the facet barycenter could lead to inconvenient approximations, therefore in our implementation, the left-most vertex (i.e. with the smallest x-coordinate) of each facet is considered as a reference point, as illustrated in Fig. 6. Here again, in case of equality, the disambiguation is performed on the y-coordinates, and then on the z-coordinates).

#### 4.3.3. *Clusters formation*

Once the first two stages of the IC algorithm are executed (as illustrated in Fig. 4), the isometric subsets are grouped according

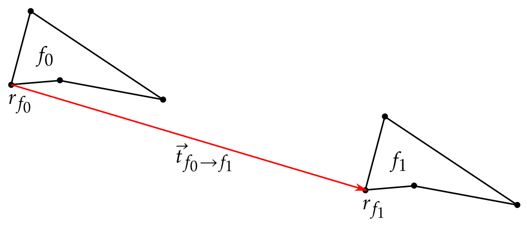


**Fig. 6.** Reference points $r_{f_0}$ and $r_{f_1}$ used for the computation of the translation vector $\vec{t}_{f_0 \to f_1}$.

to their translations group $\Gamma^I$: isometric subsets with the same translation group are joined together in a so-called *isometric cluster*. Then, the IC heuristic algorithm decomposes each cluster, according to the three following considerations (Fig. 7):

- *Identical isometric subsets:* if a cluster contains several isometric subsets (i.e. if two or more isometric subsets have identical translation groups), it is beneficial to encode them together, as the repetition of a generating mesh by a set of translations. In that case, the generating mesh contains one facet of each isometric subset, and all of these facets have the same facet translation group (defining the repetition pattern of the generating mesh). Fig. 7.d1 to g1 illustrate this process.
- *Isolated isometric subset:* if an isometric subset $I$ does not have any counterpart with the same translation group $\Gamma^I$ (i.e. is the only item of its cluster), but contains more than one facet, it is beneficial to encode this subset as the repetition of one of its facets $f_0$ by its own translation group $\Gamma^I_{f_0}$. This is illustrated in Fig. 7.d2 and e2.
- *Mono-facet subsets:* all the isometric subsets with only one facet can be encoded together within a single mesh in order to benefit from the Vertex Indexing (VI) mechanism (see Fig. 7.d3 and e3).

The pseudo-code of algorithm 1 details the execution of the IC heuristic algorithm. Using this heuristic, all the facets of the initial mesh are encoded only once. Indeed, each facet within the initial mesh belongs exclusively either to a multiple facets isometric subset, or to a mono-facet subset.

Let us define $n = |\mathcal{F}|$ so as to compute the time complexity of the IC algorithm. Thanks to the definition of the hash function $H_1$ and the use of a hash table data structure, the regroupment of the facets of $\mathcal{F}$ into isometric subsets is realized with a linear time complexity $O(n)$. For each isometric subset $I$, the complexity of the reference facet identification is linear (i.e. in $O(k_I)$, where $k_I = |I| \leq n$), and the same applies to the computation of $H_2(I)$. This ultimately leads to a linear complexity of the isometric subsets regroupment into isometric clusters. Finally, the last steps of the IC algorithm, which corresponds to the actual encoding of the repetition pattern, are also realized in linear time. Indeed, the translation group associated to each isometric subset reference facet $\Gamma^I_{f^I_{ref}}$ has been computed for the calculation of $H_2(I)$. Consequently, the overall time complexity of the IC algorithm is $O(n)$.

**Algorithm 1** Isometric Clusters heuristic algorithm

```
Require: $\mathcal{F}$: list of facets
  $A_I \leftarrow$ group $\mathcal{F}$ by $f \longmapsto H_1(f)$
  $G \leftarrow$ group $A_I$ by $I \longmapsto H_2(I) = \Gamma^I_{f^I_{ref}}$
  for all isometric cluster $c \in G$ do
    $\Gamma^c \leftarrow \Gamma^{I_0}_{f^{I_0}_{ref}}$ where $I_0$ is first element of $c$
    for all isometric subset $I \in c$ do
      $M^c \leftarrow$ add $f^I_{ref}$
    end for
    encode $(M^c, \Gamma^c)$
  end for
```

## 5. Implementation and results

In order to demonstrate the efficiency of the proposed heuristic algorithm to solve the file weight optimization problem identified in Section 4.1, three geometries presenting repetition

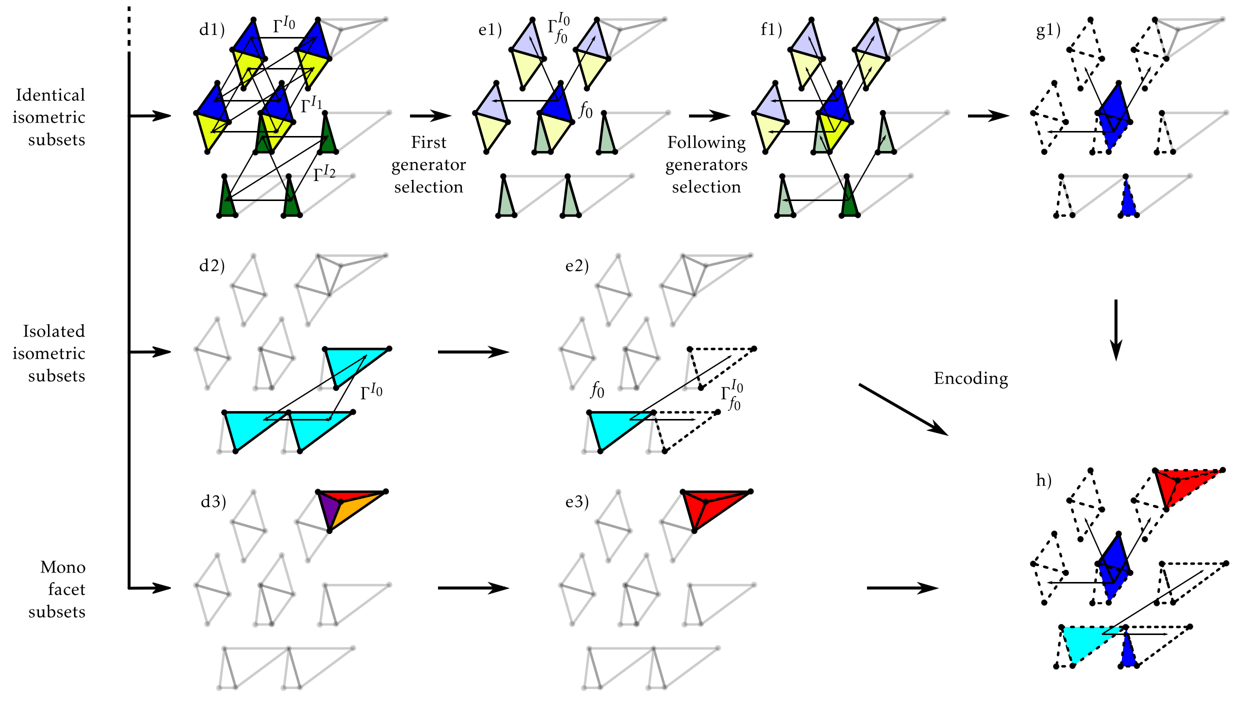


**Fig. 7.** 3-considerations based clusters formation: third step of the Isometric Clusters (IC) heuristic algorithm.

patterns are encoded according to different encoding strategies ($\kappa$). The resulting file sizes are compared to the ones of state-of-the-art approaches such as the OpenCTM format (a compression file format and library created by an open-source project) implemented in the MeshLab software and the compression algorithm proposed by Alliez et al. [26]. Because the encoding precision of each strategy is different, some are more efficient than others in generating lighter files. The various encoding precisions are discussed in this section, along with the assets and efficiency of each approach.

### 5.1. Quantization and additive manufacturing

In a 3D geometry file, many objects such as points or vectors must be encoded as floating-points. Two types of encodings can be distinguished: loss-less (ensuring that the decoding of any finite floating-point produces exactly the same floating-point with all the same digits) and lossy encodings (producing a close but not necessarily identical floating-point at decoding).

In case of an ASCII encoding, any finite floating-point can be encoded in a loss-less manner. In this case, all the digits from both the integer and the decimal part of any floating-point are represented by ASCII characters. Thus, the more precise the floating-point needs to be, the longest the ASCII chain is. In case of a binary encoding, only a finite number of floating-points can be encoded. The loss-less nature of the encoding therefore depends on the floating-points to encode and on its precision. Three main encoding precisions are usually considered, based on the IEEE 754-2008 standard: the half, single and double-precision formats, requiring respectively 16, 32 and 64 bits to encode a floating-point [27]. For each of these approaches, the resulting code is divided into 3 sections: the sign, the exponent and the fraction part (as illustrated by Fig. 8). In all cases, only a finite number of floating-points can be encoded (respectively $2^{16}$, $2^{32}$ and $2^{64}$): this is called floating-point *quantization*.

In this article, the encoded geometries used for performance evaluation are a lattice structure, a support structure and a static mixer, all designed for AM production. Assuming the geometry coordinates are defined in mm, it is admitted that a precision of $10^{-3}$ mm is sufficient. Though large AM machines are emerging, the common size of a printing platform is 250 mm $\times$ 250 mm. Let us thus consider that the coordinates of the mesh are lying between $-125$ and 125. In the IEEE 754-2008 floating point format standard, the precision of the encoded number depends on its value: the bigger a number, the less precise its encoding. For example, a floating-point between 64 and 128 will be encoded with a $2^{-4} = 0.0625$ precision with the half-precision standard, and with a $2^{-17} \approx 0.000008$ precision with the single-precision standard. Therefore, the half-precision quantization scheme does not meet the $10^{-3}$mm constraint imposed by the AM process, but the single-precision standard does and this is the one that should be privileged for AM applications. However, in this paper, encodings with the half-precision format will be realized in order to make some fair comparisons with the algorithm proposed by Alliez et al. [26].

In the framework of Fig. 3, the floating-points quantization must be performed before the isometric facets identification, in order to maximize the compression rate of the heuristic algorithm. Indeed, two facets can be isometric after quantization even if they were not before this operation.

### 5.2. Implementation of the RP encoding mechanism

The RP mechanism has been implemented and tested following three encoding strategies ($\kappa$) introduced in the next paragraphs: RP-AMF, RP-32 and RP-16. The RP-AMF one directly exploits the formalism of the AMF (Additive Manufacturing File) format. AMF is a standard specification developed specifically for AM applications. Its creation was motivated by the need to supplement the low-information and inefficient formats still in use within the AM domain (i.e. STL and OBJ). Among other features, an AMF file can associate with a mesh-based part various pieces of information, regarding its color, texture, 3D orientation and so on. Moreover, it natively includes a VI mechanism in order to reduce the resulting file size. Finally, it can encapsulate *constellations* containing several instances of the same part. Here, an instance refers to an object (containing a mesh) through an object id (see Appendix B) and includes a set of translations and rotations locating a copy of the object in the 3D space. The aim of the AMF instance objects was originally to locate several identical parts onto the build platform, without needing to duplicate vertices and facets of the mesh.

The same VI and RP mechanisms can also be found in the emerging 3MF file format, also dedicated to Additive Manufacturing applications. In this format, elements called *components* are equivalent to the *constellations* objects of the AMF standard, and are used to repeat several instances of the same triangle mesh. Because of these similar entities, the proposed approach can seamlessly be applied with the AMF or 3MF specifications, and would demonstrate the same performances. Therefore, it has been chosen in this work to exploit only the AMF standard specification to develop the RP-AMF file format.

*RP-AMF encoding strategy* directly exploits the RP mechanism and AMF standard to reduce the overall size of AMF files while avoiding multiple definitions of highly repeated geometries. Let us consider a part with a specific feature repeated several times. Using the RP-AMF encoding strategy, the mesh corresponding to this feature is encoded only once, and each repetition of this feature is encoded as an instance of the previous mesh. The rest of the part facets is encoded into another independent mesh. This encoding strategy is not restricted to one-feature Repetition Patterns only, and this example can easily be extended to parts presenting multiple repeated features. The RP-AMF encoding strategy preserves the AMF formalism: ASCII encoding, standard tags, vertices collection and triangles indices structure. The *.amf extension is also preserved, ensuring that any software able to open an AMF file format is also able to open an RP-AMF file. At opening, all the individual meshes are repeated according to its corresponding encoded instances. However, as intended by the AMF standard, each fragment repetition will be considered as an independent part on the building platform. Therefore, a stitching operation is required in order to ensure the watertightness of the opened mesh. This is similar to what is needed when importing an STL file. Indeed, the overall geometry being composed of several meshes, the points at the border of each mesh are generally duplicated within the definition of another mesh. The aim of the stitching step is thus to identify and merge duplicate vertices and edges, thus connecting neighboring triangles. By preserving the AMF formalism, the RP-AMF exchange is instantly exploitable, without any software modification required. However, the AMF formalism is not a concise 3D geometry description, and thus creates heavy files in comparison to other formalisms. Indeed, it is an ASCII description, employing redundant tags, separators as well as other artifacts.

*RP-32* and *RP-16* encoding strategies make use of a binary format to further reduce the size of the files. These new formats stand out from the RP-AMF because the encoding of each floating-point is realized through an IEEE 754-2008 floating point format standard. Thus two versions of this format must be distinguished: the RP-32, using the single-precision standard, and the RP-16, using the half-precision standard. The structure of this format is similar to the one of the AMF file format, in so that it implements a 3-blocks VI mechanism. For each mesh, a collection of points is

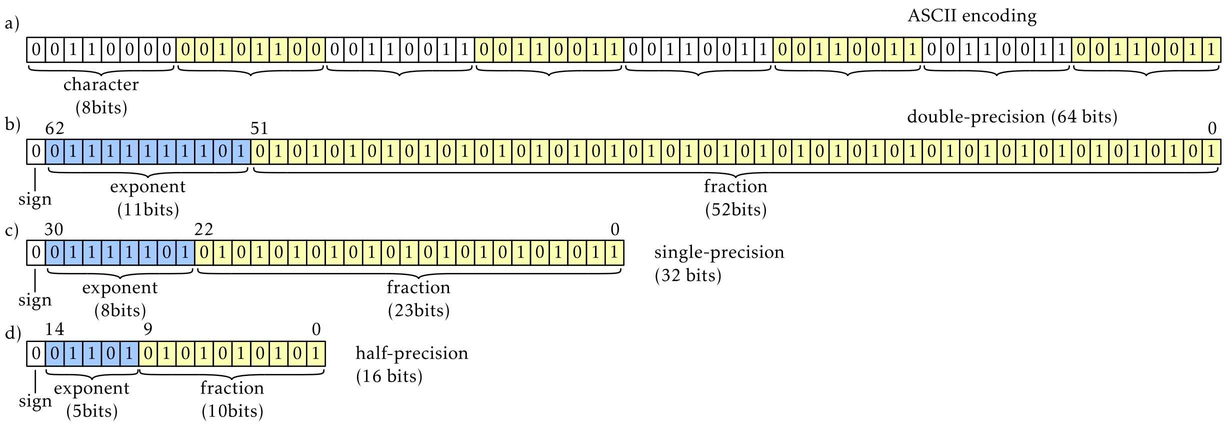


**Fig. 8.** IEEE 754-2008 standard: half (a), single (b) and double (c)-precision floating point formats.

defined, followed by a collection of point indices triplets (each triplet representing a triangle): these two blocks are encapsulating the VI mechanism. The last block consists of a collection of 3D vectors locating the multiple instances of the mesh in the 3D space. The overall part geometry is thus defined by a succession of block triplets. To further lighten the file, the unnecessary tags of the AMF encoding have been removed: no separation between floating points are required since the bit length of each encoded number is constant in binary. Only a separation between a block and the next is required, and is realized by a line jump.

## 5.3. *Cost coefficients*

In Section 4.1, the ILP governing the RP optimization has been identified and formalized. Eqs. (4a) and (4b) reveal the use of cost coefficients which values depend upon the adopted encoding strategy ($\kappa$). Table 1 summarizes the coefficients of the file formats considered in this article (for triangle meshes only). Because the RP-AMF is an ASCII based formalism, the number of bits used to encode each vertex, facet and translation (respectively $w_V^\kappa$, $w_F^\kappa$ and $w_T^\kappa$) is not constant. Indeed, the number of ASCII characters used depends upon the number of digits of the floating point (or the integer in the case of vertex indices for the facet definitions). However, because the AMF format uses encompassing tags for defining each of these objects, an average number of characters can be given, around which the variations are small (considering the process precision requirements of $10^{-3}$mm as discussed in Section 5.1). For the binary formats, the number of octets used to encode a floating-point is constant (because it is a binary encoding) and is only defined by the IEEE 754-2008 standard employed (4 octets by floating point for the single-precision in RP-32 and 2 octets by floating point for the half-precision in RP-16). However, looking at the encoding cost per facet $w_F^\kappa$, the number of octets used to encode an integer is the same for both versions of the binary format since the chosen binary precision must not impact the number of vertices that can be encoded. Indeed, 4 octets are used to encode each vertex index ensuring that each positive integer between 0 and $2^{32} \approx 4\,294\,967\,295$ can be encoded. These costs are also much lower because the unnecessary tags of the AMF have been removed with the use of binary formalism.

## 5.4. *Experimentation on lattice structures with known RP*

This section illustrates how the proposed RP encoding framework can be used to reduce the file size when storing geometries for which the repetition patterns are already known. This is the case when the software used to model the part is also the one

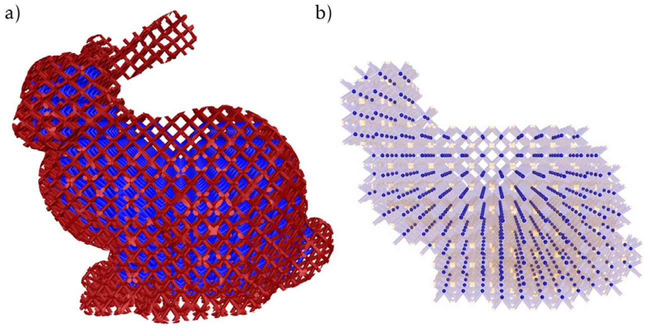


**Fig. 9.** Full lattice structure of the Stanford Bunny (a) with repeated features mesh (blue) and residual mesh (red), and repetition pattern (b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 1**
Cost coefficients associated to each file format $\kappa$ for the RP optimization.

| File format ($\kappa$) | $w_V^\kappa$ (in octets) | $w_F^\kappa$ (in octets) | $w_T^\kappa$ (in octets) |
|---|---|---|---|
| RP-AMF | ≈77(±9) | ≈64(±14) | ≈97(±9) |
| RP-32 | 12 | 12 | 12 |
| RP-16 | 6 | 12 | 6 |

encoding its geometry into an exchange file. In this case, the first framework steps of pattern identification (illustrated in Fig. 3 with an asterisk) can be bypassed.

To demonstrate the efficiency of the proposed framework, the compression of a lattice structure generated within the famous Stanford Bunny has been realized (Fig. 9). This lattice structure has been generated through the repetition of a simple BCC unit cell of 4 mm side. The border unit cells have been trimmed to the envelope of the bunny, resulting in a total of 1330 complete unit cells, each one composed of 154 triangles (using the lattice meshing approach of Chougrani et al. [28]), and a residual mesh composed of 312 834 triangles (i.e. 517 654 triangles total). In the case of a known RP, the geometry is initially divided into two meshes: a repeated features mesh (in blue on Fig. 9.a), associating the facets of each repeated feature with the corresponding repetition pattern ( Fig. 9.b), and a residual mesh, gathering all the individual facets (in red on Fig. 9.a). Only the repeated feature mesh benefits from the use of the proposed repetition encoding framework. Since the residual mesh does not present any repetition pattern, its encoding through the proposed framework is generally not optimal, but is necessary to complete the overall part geometry during the decoding step.

**Table 2**
Compressed file sizes for a known repetition pattern part (Stanford Bunny lattice structure).

| File format ($\kappa$) | Bits per floating point | Lattice structure 133 440 vertices Unzipped | Zipped | File size reduct. |
|---|---|---|---|---|
| AMF | 8–48 (ASCII) | 76 511 Ko | 4564 Ko | |
| RP-AMF | 8–48 (ASCII) | 32 774 Ko | 3062 Ko | −33% |
| OpenCTM | 32 (binary) | n/a | 3524 Ko | |
| RP-32 | 32 (binary) | 1773 Ko | 1316 Ko | −63% |
| Alliez et al. [26] | 16 (binary) | n/a | 933 Ko | |
| RP-16 | 16 (binary) | 872 Ko | 646 Ko | −31% |

Six encoding strategies have been experimented: the classical AMF format, the newly developed variation of this format exploiting RP and encoded in ASCII (RP-AMF), with a single-precision binary format (RP-16) and with a half-precision binary format (RP-32), the OpenCTM format (a compression file format and library created from an open-source project) implemented in the MeshLab software and a compression algorithm proposed by Alliez et al. [26]. Because the OpenCTM and the format proposed by Alliez et al. are both implementing binary-wise data compression schemes (hence the "n/a" abbreviations in the "unzipped" column of the following result tables), the zipped version of the other formats is also considered.

The file sizes resulting from these various encoding approaches are summarized in Table 2 along with the file size reductions obtained comparing same precision encodings. From these results, RP-based formats are demonstrating lower file sizes, for both their unzipped and zipped versions, with file size reductions reaching −63%. These results well illustrate the interest of the proposed framework with respect to state-of-the-art approaches.

## 5.5. *Experimentation on support structures with unknown RP*

Beyond lattice structures, the proposed framework can also be used to efficiently compress any 3D shape presenting repetition patterns. Such repeated geometries are more common in AM than in other manufacturing fields due to the design freedom permitted by the "layer-by-layer" approach, and support structures are great examples of such geometries. Generated after the design of the part and before the slicing step, they usually are regularly perforated with rhombus-shaped holes, in order to ensure the good evacuation of the powder once the part fabrication is completed (with the Laser Beam Melting technology in particular).

Therefore, the efficiency of the proposed framework has been demonstrated with the compression of a support structure geometry (Fig. 10) and a static mixer geometry also presenting several repeated features (Fig. 11). The considered encoding strategies and file formats are the same as the ones used for the compression of the lattice structure in Section 5.4. Initially composed of 13 196 and 21 610 triangles respectively, the support structure and the static mixer are decomposed respectively into 69 and 71 RP once the IC heuristic algorithm is executed. However, these patterns are particularly different: most of the support structure generating meshes are composed of less than 10 facets repeated up to 440 times, whereas the biggest generating mesh of the static mixer contains 317 triangles duplicated only twice. The support structure presents also more repeated facets since its residual mesh contains only 1629 triangles compared to 14 042 facets for the one of the static mixer.

Similarly to the lattice structure, the file sizes resulting from the proposed encoding strategies of these two test cases are lower than the ones of the files generated by the other strategies (Table 3). The file size reduction ratios are also indicating the

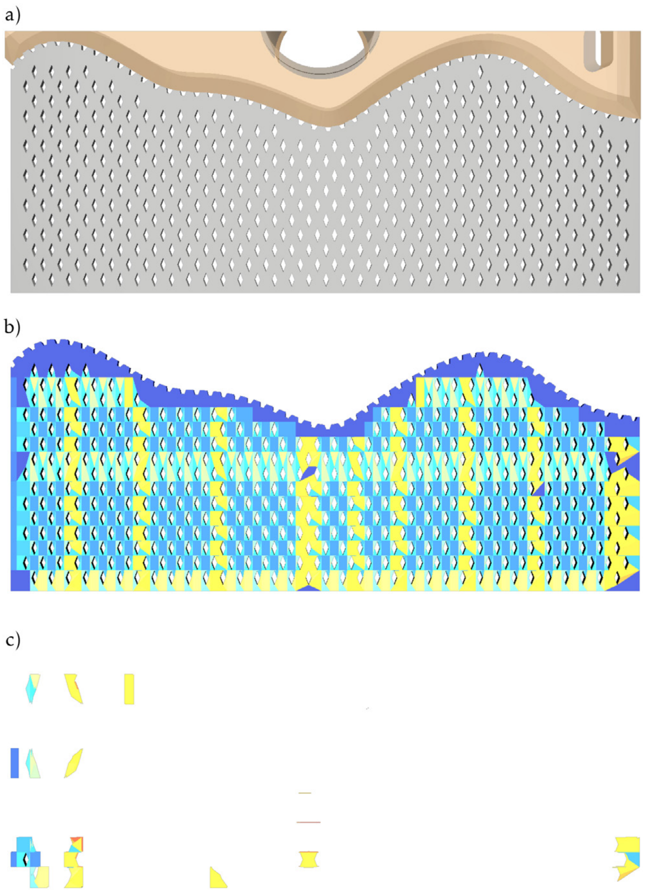


**Fig. 10.** Support structure test case: initial mesh and supported part (a), mesh decomposed after repetition pattern encoding (b) and its generating meshes (c).

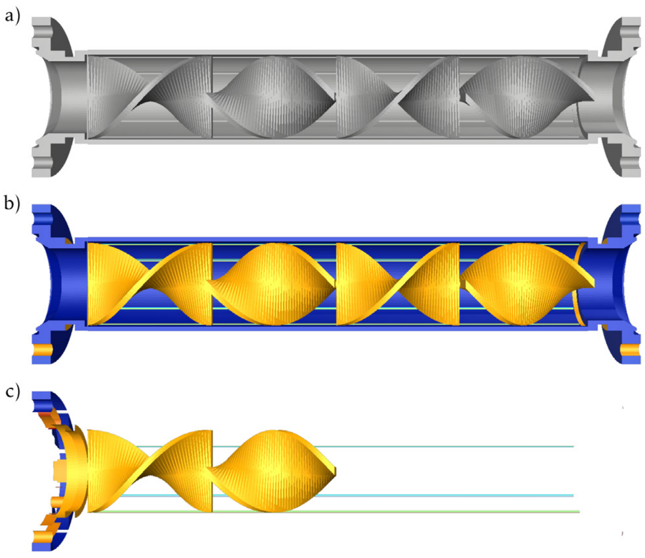


**Fig. 11.** Static mixer test case: initial mesh (a), mesh decomposed after repetition patterns encoding (b) and its generating meshes (c).

same trend. Comparing the various RP-based encoding strategies between the support structure and the static mixer, the file size reduction ratios are similar, even though the number of repetition patterns (and the generating mesh facet counts) are different between the two geometries.

**Table 3**
Compressed file sizes for the two test cases (support structure and static mixer) with unknown RP.

| File format ($\kappa$) | Bits per floating point | Support structure 5721 vertices | | File size | Static mixer 10 791 vertices | | File size |
|---|---|---|---|---|---|---|---|
| | | Unzipped | Zipped | reductions | Unzipped | Zipped | reductions |
| AMF | 8–48 (ASCII) | 1827 Ko | 89 Ko | | 3180 Ko | 167 Ko | |
| RP-AMF | 8–48 (ASCII) | 346 Ko | 15 Ko | −83% | 389 Ko | 26 Ko | −84% |
| OpenCTM | 32 (binary) | n/a | 42 Ko | | n/a | 75 Ko | |
| RP-32 | 32 (binary) | 53 Ko | 9 Ko | −79% | 31 Ko | 14 Ko | −81% |
| Alliez et al. [26] | 16 (binary) | n/a | 20.2 Ko | | n/a | 28.1 Ko | |
| RP-16 | 16 (binary) | 27 Ko | 6 Ko | −70% | 16 Ko | 9 Ko | −68% |

**Table 4**
IC heuristic algorithm computation times for the two considered test cases.

| Framework steps | Support structure | Static mixer |
|---|---|---|
| Isometric subsets identification | 169 ms | 372 ms |
| Translation groups computation | 390 ms | 576 ms |
| Clusters formation | 18 409 ms | 556 ms |

**Table 5**
Overall encoding times for the different encoding strategies considered.

| File format ($\kappa$) | Bits per floating point | Lattice structure | Support structure | Static mixer |
|---|---|---|---|---|
| AMF | 8–48 (ASCII) | 9 s | 1 s | 1 s |
| RP-AMF | 8–48 (ASCII) | 13 s | 25 s | 5 s |
| OpenCTM | 32 (binary) | 11 s | 4 s | 3 s |
| RP-32 | 32 (binary) | 9 s | 27 s | 4 s |
| Alliez et al. [26] | 16 (binary) | 82 s | 2 s | 3 s |
| RP-16 | 16 (binary) | 6 s | 30 s | 3 s |

### 5.6. Encoding times comparison

One of the obstacles to the adoption of a compressed format is the encoding time taken to obtain the resulting files. In order to analyze computation time issues, the duration of each IC heuristic algorithm stage is first measured and the results are presented in Table 4. In the case of a lattice structure, the IC heuristic algorithm is not run as the RP is known a priori and can thus be directly exploited during the encoding step.

These results are illustrating the different behaviors that can arise from the execution of the IC algorithm on geometries with different unknown repetition patterns. For example, in the case of a mesh with a simple feature repeated a lot of times such as the one of the support structure, numerous isometric subsets must be clustered together, and the clusters formation is therefore the more time-consuming step. Inversely, when the part presents several complex features replicated only a few times like for the static mixer, the duration of the clusters formation step is considerably reduced.

To compare the speed of the proposed framework to the ones of the literature, the overall encoding times of the previous test cases have also been measured and listed in Table 5. On an indicative basis, the times to encode the three considered structures in AMF have also been added, though it does not include any optimization mechanism.

The computation time of the lattice structure encoding demonstrates the speed of the proposed framework regarding geometries for which the repetition pattern is known in advance. Indeed, for this particular test case, the compression algorithms of the literature (namely the Open CTM format, and the algorithm proposed by Alliez et al.) are presenting longer optimization times. However, in the case of geometries where the repetition pattern is not a priori known, such as the support structure and the static mixer, the proposed framework optimization times are not competing with the literature algorithms.

Focusing on the RP-encodings results, though the lattice structure is composed of more facets, its encoding is faster than the one of the support structure, since the RP optimization step does not need to be executed (Fig. 3).

However, comparing the results obtained with the RP-based encodings between the static mixer and the support structure, one can identify the longest time requirements for the latter. This can be explained by the difference between the two pattern structures (as detailed in Section 5.5). Since the longest operation is the clusters formation, and since the pattern structure of the static mixer is composed of larger generating meshes with fewer repetitions than the ones of the support structure, its encoding is performed more rapidly. This corroborates with the results displayed in Table 4.

In conclusion, though the proposed framework performances are surpassing the ones of the literature algorithms in terms of compression rate, this comes at a small temporal cost. This cost can vary according to the prior knowledge of the repetition patterns, and according to its balance between the number of facets and the number of repetitions of each generating mesh. However, the order of magnitude of the RP-based encodings optimization durations are remaining reasonable, ergonomically speaking, for a compression encoding operation.

## 6. Conclusions and perspectives

This article has introduced a new framework for the file compression of faceted 3D geometries presenting repetition patterns. This approach diminishes the size of the 3D geometry file by encoding a repeated feature as one mesh and several repetitions of it, rather than encoding each similar mesh once. This framework is particularly suited for parts produced by AM machines and often presenting repetition patterns. It can straightforwardly be applied on geometries for which the repetition pattern is already known (such as lattice structures). But in the broader case, the geometry's 3D mesh must first be decomposed into repeated submeshes, in order to ensure a high compression rate. The identification of repeated submeshes is performed by a new heuristic algorithm especially developed to solve the underlying NP-complete Weighted Exact Cover (WEC) problem.

The benefits of this framework have been demonstrated on three test cases and for several file encodings: a variation of the standard AMF format (the RP-AMF format), and a dedicated binary file format with different floating point precisions (the RP-32 and the RP-16 formats). By exploiting the numerical objects included within the AMF standard definition, this work encourages its adoption within the AM community. The results of the three test cases have demonstrated both good compression rates and reasonably short computation times when compared to state-of-the-art approaches.

This work only takes advantage of the repetitions consisting of 3D translations. Other similarities could be exploited (such as rotation, scaling or symmetries) to further enhance the compression rate. The identification algorithms for such patterns proposed by Shi et al. could therefore be employed [11]. However, encoding circular repetition patterns, scaling or symmetries can lead to approximation errors, resulting in a non-watertight mesh. Furthermore, many AM parts (especially the ones with lattice structures features) mostly present translated pattern repetitions, which already makes the proposed framework useful.

To further increase the efficiency of the proposed framework, the computation time required for the RP optimization could be reduced. Indeed, the more time-consuming step of the IC heuristic algorithm has been identified, namely the clusters formation stage, and could be accelerated by using GPU parallelization for example. Moreover, to further decrease the encoding file sizes generated by the framework, the quality of the mesh decomposition obtained after the RP optimization step could be improved by implementing a more complex algorithm. Such an algorithm could for example benefit from a splitting operation applied on an isometric subset, in order to create a more efficient clustering. In the same intent, a multi-scale approach could also be developed to encode repetitions of repetition patterns (nested repetitions) within the resulting file.

Moreover, to further increase the compression ratio, the proposed framework could be coupled with another mesh description encoding (such as the one of Alliez et al. [26]). Indeed, the RP mechanism is decomposing the part geometry into independent meshes to be encoded with or without repetition, and the encoding of each mesh can be more complex than the one employed in the RP formats (i.e. listing all the 3D points and then listing all the facets as triplets of point indexing). The RP mechanism is therefore a compression layer that can be coupled with any existing geometry compression approach.

Finally, not only the proposed repetition pattern encoding framework allows file size reductions, but it can also be exploited at various steps of the Product Development Process (PDP), including process planning phases. For instance, the proposed framework could benefit to the slicing step, wherein the repetition patterns could be exploited without having to decompress the whole part geometry. Indeed, knowing the elementary feature (that is repeated) and the locations of its various occurrences, the slicing operation can be applied once, and the so computed trajectories can be repeated to match the other locations of the considered feature. More broadly, the preservation and exploitation of part repetition patterns throughout the production workflow thus permits to optimize and speed up the end-to-end AM process.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A. AMF mesh formalism

```
<?xml version="1.0">
<amf unit="millimeter">
  <object id="1">
    <mesh>
      <vertices>
        <vertex>
          <coordinates>
            <x>1.2</x><y>5.6</y><z>3.468</z>
          </coordinates>
        </vertex>
        ...
      </vertices>
      <volume>
        <triangle>
          <v1>1</v1><v2>2</v2><v3>3</v3>
        </triangle>
        ...
      </volume>
    </mesh>
  </object>
</amf>
```

## Appendix B. AMF constellation formalism

```
<?xml version="1.0">
<amf unit="millimeter">
  <object id="1">
  ...
  </object>
  <constellation id="2">
    <instance objectid="1">
      <deltax>5</deltax>
      <deltay>10</deltay>
      <deltaz>0.5</deltaz>
      <rx>-10</rx>
      <ry>10</ry>
      <rz>180</rz>
      ...
    </instance>
  </constellation>
</amf>
```

## References

[1] Dilberoglu UM, Gharehpapagh B, Yaman U, Dolen M. The role of additive manufacturing in the era of industry 4.0. Procedia Manuf 2017;11:545–54.
[2] Jafari D, Wits WW. The utilization of selective laser melting technology on heat transfer devices for thermal energy conversion applications: A review. Renew Sustain Energy Rev 2018;91:420–42.
[3] Gao W, Zhang Y, Ramanujan D, Ramani K, Chen Y, Williams CB, Wang CC, Shin YC, Zhang S, Zavattieri PD. The status, challenges, and future of additive manufacturing in engineering. Comput Aided Des 2015;69:65–89.
[4] Shi K, Cai C, Wu Z, Yong J. Slicing and support structure generation for 3D printing directly on B-rep models. Vis Comput Ind Biomed Art 2019;2(1):3.
[5] Dixit T, Ghosh I. Review of micro- and mini-channel heat sinks and heat exchangers for single phase fluids. Renew Sustain Energy Rev 2015;41:1298–311.
[6] Qin Y, Qi Q, Scott PJ, Jiang X. Status, comparison, and future of the representations of additive manufacturing data. Comput Aided Des 2019.
[7] McMillan M, Jurg M, Leary M, Brandt M. Programmatic lattice generation for additive manufacture. Proc Technol 2015;20:178–84.
[8] Azman AH, Vignat F, Villeneuve F. Evaluating current CAD tools performances in the context of design for additive manufacturing, no 44. 2014, p. 1–7.
[9] Savio G, Meneghello R, Rosso S, Concheri G. 3D model representation and data exchange for additive manufacturing. In: von Schweinitz D, Ure B, editors. Kinderchirurgie. Berlin, Heidelberg: Springer Berlin Heidelberg; 2019, p. 412–21.
[10] Pauly M, Mitra NJ, Wallner J, Pottmann H, Guibas LJ. Discovering structural regularity in 3D geometry. 2008, p. 11.
[11] Shi Z, Alliez P, Desbrun M, Bao H, Huang J. Symmetry and orbit detection via Lie-Algebra voting. Comput Graph Forum 2016;35(5):217–27.

[12] Sipiran I, Gregor R, Schreck T. Approximate symmetry detection in partial 3D meshes. Comput Graph Forum 2014;33(7):131–40.
[13] Mellado N, Guennebaud G, Barla P, Reuter P, Schlick C. Growing least squares for the analysis of manifolds in scale-space. Comput Graph Forum 2012;31(5):1691–701.
[14] Biasotti S, Thompson EM, Barthe L, Berretti S, Giachetti A, Lejemble T, Mellado N, Moustakas K, Manolas I, Dimou D, Tortorici C, Velasco-Forero S, Werghi N, Polig M, Sorrentino G, Hermon S. SHREC'18 track: Recognition of geometric patterns over 3D models. 2018, p. 8.
[15] Nan L, Sharf A, Zhang H, Cohen-Or D, Chen B. SmartBoxes for interactive urban reconstruction. ACM Trans Graph 2010;29(4):1.
[16] Shen C-H, Huang S-S, Fu H, Hu S-M. Adaptive partitioning of urban facades. In: Proceedings of the 2011 SIGGRAPH Asia conference on - SA '11. Hong Kong, China: ACM Press; 2011, p. 1.
[17] Yi C, Lu D, Xie Q, Liu S, Li H, Wei M, Wang J. Hierarchical tunnel modeling from 3D raw LiDAR point cloud. Comput Aided Des 2019;114:143–54.
[18] Maglo A, Lavoué G, Dupont F, Hudelot C. 3D mesh compression: Survey, comparisons, and emerging trends. ACM Comput Surv 2015;47(3):1–41.
[19] Steuben JC, Iliopoulos AP, Michopoulos JG. Implicit slicing for functionally tailored additive manufacturing. Comput Aided Des 2016;77:107–19.
[20] Lee KH, Woo H. Direct integration of reverse engineering and rapid prototyping. Comput Ind Eng 2000;38(1):21–38.
[21] Vaissier B, Pernot J-P, Chougrani L, Véron P. Genetic-algorithm based framework for lattice support structure optimization in additive manufacturing. Comput Aided Des 2019;110:11–23.
[22] Taubin G, Horn W, Lazarus F, Rossignac J. Geometry coding and VRML. Proc IEEE 1998;86(6):1228–43.
[23] Garey MR, Johnson DS. Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman and Company; 1979.
[24] Knuth DE. Dancing links. 2000, arXiv:cs/0011047.
[25] Skala V, Hrádek J, Kucha M. Hash function for triangular mesh reconstruction. 2009, p. 6.
[26] Alliez P, Desbrun M. Valence-driven connectivity encoding for 3D meshes. Comput Graph Forum 2001;20(3):480–9.
[27] IEEE standard for floating-point arithmetic 754-2019. IEEE; 2019, p. 16.
[28] Chougrani L, Pernot J-P, Véron P, Abed S. Lattice structure lightweight triangulation for additive manufacturing. Comput Aided Des 2017;90:95–104.