Fixing normal constraints for generation of polycubes

Dmitry Sokolov, Nicolas Ray

Fixing normal constraints for generation of polycubes

Sokolov Dmitry[1] and Ray Nicolas[2]

[1] ALICE team, Université Lorraine dmitry.sokolov@loria.fr
[2] ALICE team, INRIA Lorraine nicolas.ray@inria.fr

Abstract

Many efforts have been devoted to mesh a volume with hexahedral elements. One of the proposed methods consist in : deforming the original volume such that its boundary normal vector is always aligned with one of the majors axis i.e. constructing a polycube, voxelizing the deformed volume and deforming it back to the initial position. This pipeline recently proposed by Gregson *et al.* generates nice results, but is not sufficiently robust. More specificaly, it is required to determine a priori what will be the normal of the surface of the deformed object. However, nothing ensures that there exists a deformation compatible with these constraints. We propose a method able to determine if there exists a deformation of the object boundary that respects the normal constraints, and an automatic solution to edit the constraints to make the deformation possible.

1 Introduction

A polycube is an abstract representation of a volume that is very efficient for tasks such as texturing, deformation or remeshing. To convert a triangulated surface into a polycube, it is required to deform the surface such that its normal becomes always aligned with an axis of the scene. A natural choice of the axis to align the normal with is the closest axis to the original surface normal. However, it may not exist a deformation that satisfies these equality constraints on the surface normal. We present an algorithm able to detect such situations and fix the normal constraints. Fig. 1 presents some cases where the normal equality constraints needs to be edited to allow for a valid deformation.

This objective is motivated by the hexahedral remeshing application proposed by Gregson *et al.*'s [11], where invalid axis assignments were not always

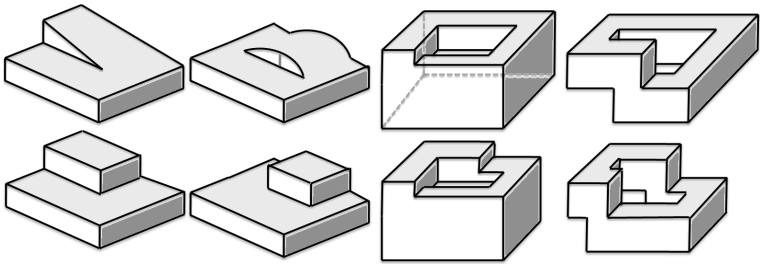

Fig. 1. Requiered editing of the normal constraint No deformation can align all charts with major axis without shrinking the surface (upper row). Editing the normal constraints by adding "'steps"' makes it possible (lower row).

fixed. Their pipeline (Fig. 2-up) starts with a tetrahedral mesh and can be summarized as follows :

1. it applies a soft rotation-based deformation to roughly align the surface normals with the major axis,
2. it determines which major axis have to be aligned with the normal on each point of the surface,
3. it deforms the mesh to respect these constraints.
4. the geometry is then filled by a Cartesian grid that is mapped to the original mesh position to provide the hex-remeshing. Standard post-processing removes too distorted hexahedrals close to the volume boundary.

A drawback of this method is that during the construction of the polycube, it is not always possible to deform the object subject to the surface orientation constraints (Step 3).

Our contribution is an algorithm that edits the normal constraints defined in Step 2, to ensure that there exists a deformation that respects these constraints. As illustrated in Fig. 2-Middle, the majors axis assignment of the object boundary is represented by a new mesh (referred to as meta-mesh and similar to the one introduced in [8]) that is easier to manipulate. This meta-mesh is embedded in the original surface, so local editing operations of the meta-mesh are equivalent to directly editing the axis assignment on the original surface.

Our method defines a deformation of the object boundary by affecting a geometry to the meta-mesh (Fig. 2-bottom). For each dimension e_i, each face of the meta-mesh is decomposed into quads, and the i^{th} coordinate of each meta-edge is determined by a constrained optimization algorithm. It ensures that each point of the surface has the prescribed normal when possible. If it is not possible, extra variables and constraints are added to determine where steps have to be created. A step creation is the basic operation that splits a meta-face's quad on the meta-mesh (Fig. 2-bottom, dimension e_3), and introduces a new chart of axis alignment on the original surface.

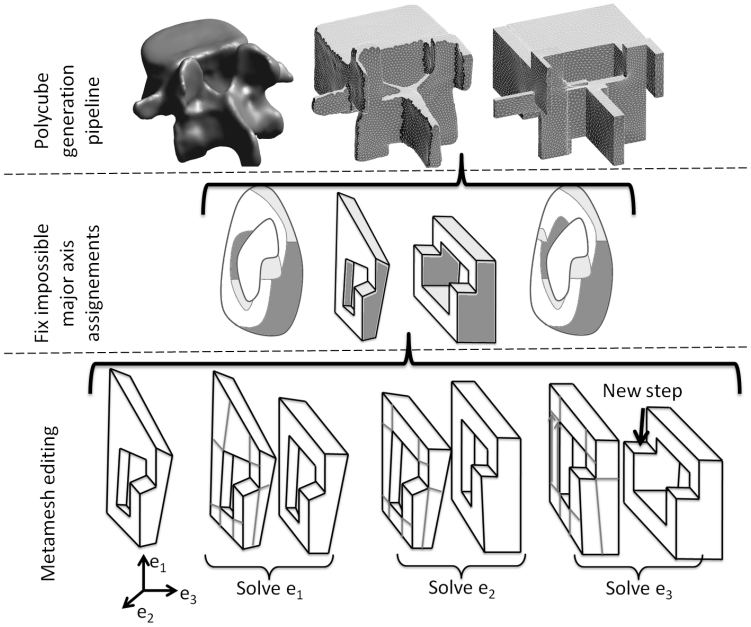

Fig. 2. Overview *Hex meshing pipeline* The volume is deformed to better align its boundary with majors axis, this alignment is enforced as a constraint, the deformed mesh is voxelized, and mapped back to the object original shape. *Fix impossible major axis assignments* The axis assignment is edited via a meta-mesh, that is easier to manipulate than the axis assignment. *meta-mesh editing* For each dimension e_i, each face of the meta mesh is decomposed into quads, and the geometry is resolved for the current dimension. Here, a solution was directly found for e_1 and e_2, but e_3 required to edit the meta-mesh topology.

Our contributions are an analysis of the possibility to deform an object with constrained normal, an algorithm that detects failure cases, and a solution to edit the normal constraints such that the deformation is made possible.

After a review of previous works, we present how the problem of finding a valid volumic deformation can be restated as a problem on its boundary (Section 2), how it can be formalized with a meta-mesh (Section 3), and an overview of our algorithm (Section 4). Sections 6, 7 and 8 provide details on each part of the algorithm. We conclude by presenting some results and discussing the method in sections 9 and 10.

Previous works

Polycubes were introduced in computer graphics by Tarini *et al.* [1] for seamless texturing surfaces. They were later used for hexahedral remeshing of shell

meshes [5], then extended to volumic remeshing [11]. In these applications, the construction of polycubes is not discussed beside in the framework of Gregson *et al.* [11] that we are improving in this paper.

Juncong *et al.* [3] proposed an algorithm to automatically construct polycubes. It is based on locally match simple polycube primitives, detect features such as protrusion, and merge them all together to produce a polycube. It is able to produce very convincing result for some classic computer graphics meshes, but it seems difficult to handle meshes from CAD/CAM where the features are not well characterized by their proxy.

Ying *et al.* [2] have presented an alternative solution based on cutting the volume into slices and iteratively add slices to the polycube. This solution produces fair results for meshes reasonably aligned with major axis.

Another contribution [4] in this domain is an optimization of the mapping between a polycube and a surface. It requires a valid polycube to start with, but it can improve any others results by adjusting the mapping.

2 Problem statement

Let's consider a volumic object O, and a major axis $N(P)$ associated to each point P of O's boundary. Our objective is to determine how to modify $N(P)$ to make it possible to define a deformation D such that for each point P of O's boundary, the normal $n(D(P))$ of the deformed volume at point $D(P)$ is equal to $N(P)$.

A natural deformation D have to be one-to-one. However, in our context global overlaps do not impact the final hexahedral mesh (Fig. 3) because the pre-image of each voxel in the voxelization of the deformed mesh (Step 4 of Gregson *et al.*'s pipeline) can be a set of hexes in the remeshed model.

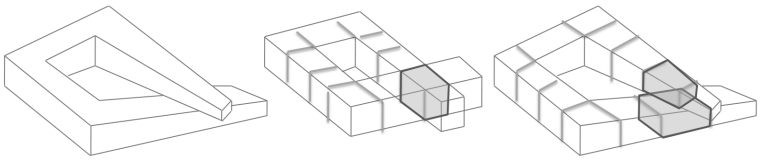

Fig. 3. Global overlaps Given a volume (Right), aligning its boundary with major axis (Middle) can create global overlaps (green voxel), but re projecting it to the original volume provides a valid hexahedral mesh.

Therefore, the constraints we need are rather local than global i.e. ensuring that D has no foldovers is a sufficient property for our application. As a consequence, we want to produce a locally one-to-one map, so D have to preserve the orientation of the volume i.e. the determinant of its Jacobian matrix $det(J(D))$ must be strictly positive.

There are two situations (Fig. 4) that can enforce a negative determinant of the Jacobian matrix of D: if the deformed boundary surface self-intersects (it is no more a boundary), or if there are some problems on the surface i.e. $det(J(D)) \leq 0$ on the surface. This paper deals only with surface issues, and a possible solution to prevent surface self-intersections is discussed in Section 10.4.

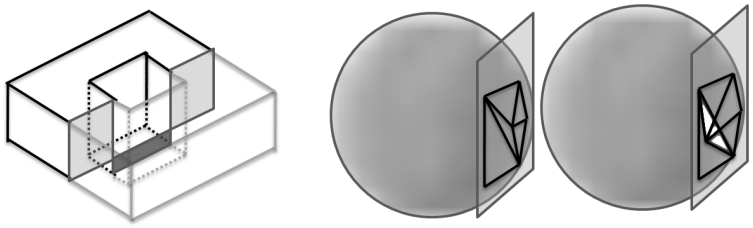

Fig. 4. Volume foldovers The volume is not defined if the deformed boundary surface self intersect (Left), or when the surface has foldovers (Right— white triangle).

3 Formalization

The object boundary is represented by a triangulated surface with an associated majors axis $N(f) \in \{e_i, -e_i\}_{i=1}^{i\leq 3}$ associated to each triangle f. We wish to edit N to ensure that the volume can be deformed without foldovers to make its boundary normal $n(f)$ equal to $N(f)$ (Fig. 5 for an exemple).

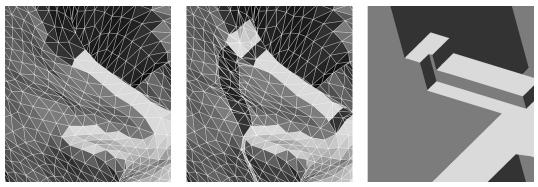

Fig. 5. Major axis assignment correction Grey level represents the axis that the deformed surface have to be orthogonal to. The volume cannot be directly deformed to align its boundary normals with their associated major axis (left). The desired major axis can be edited (middle) to make it possible (Right).

Axis assignment assumption

Our objective is to preserve as much as possible the original axis assignment. Therefore, we limit the possible edition of N to the local operation that creates

a step. As a consequence, it is impossible for our algorithm to deal with completely random axis assignments. For example, adding steps does not allow to make valid an object where all points are assigned to the same axis. More generally, the genus of the object strongly constraints the set of potentially valid axis assignments and creating steps do not allow to change it.

To be sure that the initial axis assignment corresponds to a similar object (same genus), we assume that it is given by the closest axis to the surface normal. Obviously, in Gregson *et al.*'s pipeline, it is the closest axis to the deformed surface normal.

Moreover, hard edges may generate adjacent triangles associated to opposite axis. It corresponds to a degenerated volume that can be directly handled by our algorithm if a new axis is assigned in a tiny band placed on the edge adjacent to conflicting triangles, as illustrated in Fig. 6. This operation can be interpreted geometrically as smoothing the hard edges.

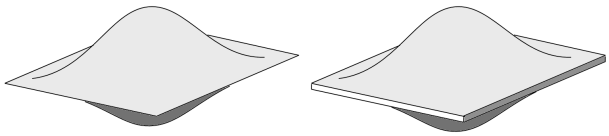

Fig. 6. Opposite axis can not be assigned to neigborg points (left). A tiny band is introduced to avoid this situation (right).

Meta-mesh

The major axis assignment is difficult to directly manipulate on the object boundary. It is easier to manipulate a $2D$ mesh embedded in the original volume boundary referred to as meta-mesh. By embedded, we mean that the meta-edge geometry are given as paths defined on the original surface as described by Li *e*t al. [10]. The meta-mesh is obtained from the original surface by merging all adjacent triangles f having the same $N(f)$ into a meta-face F. As a consequence, each meta-face F is associated to a major axis $N(F)$.

To manipulate the meta-mesh geometry, it is also required to define the major axis $N(E)$ associated to each oriented meta-edge E. The major axis $N(E)$ must be orthogonal to its adjacent meta-faces, but this is not sufficient to set its orientation (e_i or $-e_i$). We rely on an heuristic to resolve the ambiguity: if the projection of the edges on e_i is positive, e_i is affected, else $-e_i$ is affected. As a consequence, a meta-edge may be split into several parts during this process if the major axis associated to its original edges is not always the same like in Fig. 8.

As the meta-mesh is embedded in the original surface, there exists a one-to-one mapping between both. Therefore, if we determine a $3D$ geometry of the meta-mesh that is a boundary (no self intersections) and has no local shrinks

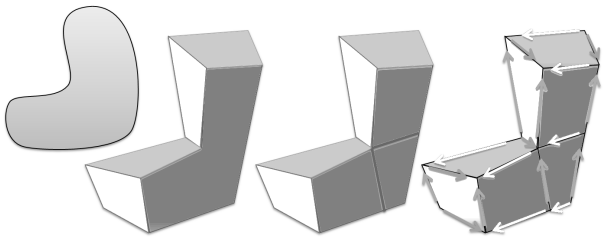

Fig. 7. meta mesh The surface can be decomposed into chart such that each chart is associated to a single preferred major axis. The meta mesh defines a facet of each such chart (middle left), and is decomposed into quads (middle right). Each edge is then affected to its preferred major axis.

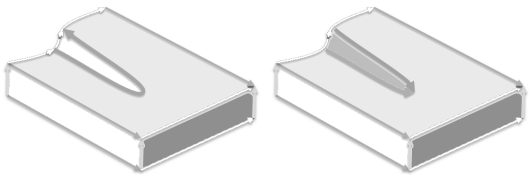

Fig. 8. Meta-edge axis assignment The major axis assignment on meta-edges can not be deduced from the axis of meta-faces. Here, the gray oriented edge (Left) goes direction e_1 at the beginning and $-e_1$ at the end. A coherent axis assignment requires to split this meta-edge (Right).

$(det(J(D)) = 0)$ nor foldovers$(det(J(D)) < 0)$, then the original surface can be mapped to it, and the deformation inside the volume can be interpolated by mean value coordinates [7].

As stated in the previous section, we focus here on the local shrinks and foldover problems (self intersections are discussed in section 10.4). To do so, the geometry of the meta-mesh must ensure that the normal $n(F)$ of each meta-face F is constant (flat meta-face) and defined everywhere (no shrink).

4 Algorithm overview

Having all meta-faces normals equal to their assigned axis can be ensured if all oriented meta-edge E are in the direction to their corresponding major axis $N(E)$, and if the meta-face boundary do not self intersect. Both conditions are enforced by decomposing each meta-face into quads, and solving the quad edges geometry such that their normal equals their assigned axis. The quad decomposition makes it impossible for meta-face boundaries to self-intersect because it would require some quad edges to be oriented in the wrong direction.

When determining the geometry of oriented meta-edges, the e_0, e_1, e_2 coordinates do not interact with each other, making it possible to deal with one

dimension at a time. Therefore, algorithm 1 performs on each dimension a decomposition of the meta faces into quads, determines the geometry of the quads edges with respect to their associated axis, possibly determine where to edit the meta-mesh by creation of steps, then create the steps and resolves again the geometry (Algorithm 1).

Algorithm 1: Algorithm overview

```
Data: Mesh m of the object boundary
Data: Major axis assignment N
for dim ← 1 to 3 do
    create meta-mesh M;
    decompose each meta-face of M into quads;
    resolve the e_dim coordinate of M's geometry (Section 6);
    if no valid solution is found then
        add extra degrees of freedom corresponding to step creation;
        resolve the e_dim coordinate of M's geometry (Section 7);
        create the corresponding steps (Section 8);
        introduce the steps vertice variables to the system;
        resolve the e_dim coordinate of M's geometry;
    end
end
```

To improve the clarity of the explanations, we will assume from now that the current dimension to be solved is the vertical dimension e_1. Therefore, meta-faces orthogonal to e_1, and meta-edges in direction e_2 and e_3 will be charaterized as **horizontal**, and other will be charaterized as **vertical**.

The rest of the paper presents how the meta-faces are decomposed into quads, the algorithm that determines if it is possible to directly construct an axis aligned meta-mesh (Section 6), an algorithm that determines where new steps should be created (Section 7), and the creation of a step in the meta-mesh corresponding to edit the axis assignment on the original surface (Section 8). Results are then presented and the benefits and drawbacks of the method are discussed.

5 Meta-face decomposition

Solving the meta-edges geometry enforces meta-faces to be flat. However, it is not sufficient to ensure that the surface normal is defined everywhere, because the boundary may self-intersect. It is prevented by decomposing the meta-faces into quads, and solving the geometry of quad's edges instead of meta-edges.

The decomposition is obtained by tracing curves on the original surface from the meta-vertices (see Fig. 9). The curves directions are determined by

two smooth tangent vector fields, each aligned with the subset of meta-edges that share a same associated axis. These smooth vector fields are computed by Ray *et al.*'s algorithm [6]. As explained in Section 10.2, the decomposition of a meta-face into quads have to depend on the current axis to be solved.

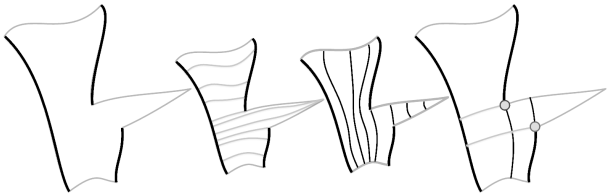

Fig. 9. Meta-face decomposition The axis associated to the meta-edges (Left) allows to interpolate two tangent vector fields in the meta-face (Middle). The quad decomposition is performed by following streamlines of these vector fields (Right).

Vertical meta-faces (See Figure 10-Middle)

For vertical meta-faces, the quad decompositions are obtained by tracing the set of curves from their vertices and aligned with e_1.

When two adjacent meta-edges are associated to opposite majors axis, a degenerated quad is created. In this case, an extra edge (Fig 10) is generated from a point close to the degenerated edge, and lying in a quad edge adjacent to the degenerated edge. This will force the algorithm to generate a special step.

Horizontal meta-faces (See Figure 10-Right)

For horizontal meta-faces, the quads decomposition is obtained by tracing curves in both directions from both the original meta-face vertices and the T-vertices introduced during the quad decomposition of other meta-faces.

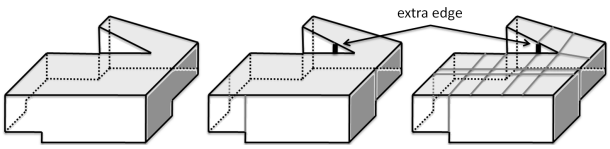

Fig. 10. When solving in the vertical direction, the vertical meta-faces are decomposed first by cutting them in the vertical direction (Middle), then the horizontal meta-faces are decomposed by cutting them in the two horizontal directions from all their vertices (Right).

6 Check assignment validity

A valid assignment of major axis requires each oriented edge h_i to go in the direction of its associated major axis $N(h_i)$. As meta faces are decomposed into quads, this will also ensure that each quads (and therefore all meta-faces) will have their prescribed orientation.

Let x_i represent the first (vertical) coordinate of vertex i and Δx_{ij} denotes the first coordinate of edge joining vertices i and j.

A valid geometry requires : that $\Delta x_{ij} = x_j - x_i$, that $\Delta x_{ij} > 0$ if the edge is associated to e_1, that $\Delta x_{ij} < 0$ if the edge is associated to $-e_1$, and that $\Delta x_{ij} = 0$ if the edge is horizontal. To find an unique solution, we minimize $\Sigma \|\Delta x_{ij}\|$ subject to the constrains that $\Delta x_{ij} = 0$ if the edge is not associated to e_1 or $-e_1$, and $\Delta x_{ij} \geq l_{ij}$, where l_{ij} is the geodesic distance of the meta-edge h_{ij}, if the edge is associated to e_1. For oriented edges associated to $-e_1$, the constraint on the opposite edge ensures that $\Delta x_{ji} > 0$ i.e. $\Delta x_{ij} < 0$.

This problem is solved by the revised simplex method. We use the implementation in the lp_solve library. If the problem has no solution, it means that there exists no deformation without foldover that aligns the surface normal with the current flagging N. Therefore more degree of freedom are required, corresponding to the creation of steps, as described in the next section.

7 Define where to create step

From the point of view of the meta-mesh geometry, creating a step on a quad strip allows to lift all vertices of one side of the quad strip. This can be represented (before creating the steps) in the meta-mesh geometry by allowing for horizontal edges to have non null e_1 component. However, it is subject to the constraint that on each quad (of the meta-face quad decomposition), opposite edges have the same e_1 geometry.

Simply introducing new geometry variables x_i to horizontal edges and setting the constraint of equality on opposite quad edges would probably create many useless steps. Indeed, even on a simple cube, it is possible to have a double step on meta-faces associated with e_1 and $-e_1$.

To avoid producing useless steps, the geometry variables Δx_{ij} of horizontal edges have their absolute value minimized $10000\Sigma \|\Delta x_{ij}\|$. The weight of 10000 is required to always prefer changing the geometry of vertical edges over horizontal edges that would lead to creating a new step. To do so, these Δx_{ij} are replaced by γ_{ij}^+ and γ_{ij}^- such that $\gamma_{ij}^+ + \gamma_{ij}^- = \Delta x_{ij}$, $\gamma_{ij}^+ \geq 0$, $\gamma_{ij}^- \leq 0$, leading to minimize $\Sigma\gamma_{ij}^+ - \Sigma\gamma_{ij}^-$, subject to the constraint that opposite quad edges have the same $\gamma_{ij}^+ + \gamma_{ij}^-$.

All horizontal edges are weighted by $10000 + \epsilon_{ij}$ in the objective function. The value of $\epsilon_{ij} \in [0..1000]$ is set to minimize the step length by setting it to be proportional to the corresponding quad strip length. Before introducing slack variables, the system minimizes:

$$\Sigma w_{ij} \|x_j - x_i\| \tag{1}$$

where $w_{ij} = 1$ if edge ij is vertical, and $w_{ij} = 10000 + \epsilon_{ij}$ if edge ij is horizontal,

subject to the constraints:

$x_j > x_i$ (resp. $x_j < x_i$) if $N(h_{ij} = e_1)$ (resp. $N(h_{ij} = -e_1)$)
$x_j - x_i = x'_j - x'_i$ if $h_{i'j'}$ and h_{ij} are opposite quad edges
$x_j - x_i \geq l_{ij}$ (resp. $x_j - x_i \leq -l_{ij}$) where l_{ij} is the length (geodesic distance) of edge ij, if $N(h_{ij} = e_1)$ (resp. $N(h_{ij} = -e_1)$)

In practice after introduction of slack variables Δx_{ij}, γ^+_{ij} and γ^-_{ij} (see Fig. 11), the final system minimizes :

$$\Sigma \|\Delta x_{ij}\| + \Sigma (10000 + \epsilon_{ij})(\gamma^+_{ij} - \gamma^-_{ij}) \tag{2}$$

and subject to the constraints:

$\Delta x_{ij} = \Delta x_{i'j'}$ if $h_{i'j'}$ and h_{ij} are opposite quad edges
$\Delta x_{ij} \geq l_{ij}$ (resp. $\Delta x_{ij} \leq -l_{ij}$) where l_{ij} is the length (geodesic distance) of edge ij, if edge ij is vertical
$\Delta x_{ij} = \gamma^+_{ij} + \gamma^-_{ij}$ if if edge ij is horizontal
$\gamma^+_{ij} \geq 0$
$\gamma^-_{ij} \leq 0$

Solving this system provides the height of vertices (x_i). Horizontal edges with different values of their extremities forms a set of quads strips where new steps needs to created.

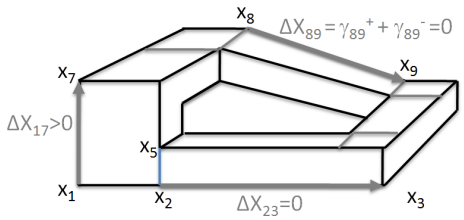

Fig. 11. Variables of the system: x_i are the vertices height, Δx_{ij} are the height component of edges, γ^+_{ij} and γ^-_{ij} are slack variables associated to horizontal edges.

8 make step operation

All modifications of the major axis assignment N rely on a single editing operation on the meta-mesh: the creation of a step (Fig. 12). This operation is local to a quad strip generated during the decomposition the meta-faces

into quads. In the original mesh, it corresponds to define a band crossing all quads of the quad strip, and affecting to it the major axis associated to the quad edges not crossed by the band. It determines the axis e_i of the new band: setting e_i creates an ascending step in the e_i direction, whereas taking the opposite $-e_i$ creates a descending step in the e_i direction.

The creation of a step invalidates the decomposition of meta-faces that are adjacent to the created band. Since two adjacent meta-faces can not have the same associated axis, creating a step will not invalidate the quad decomposition of other meta-faces associated to the same axis. As a consequence, all steps to be created on meta-faces associated to the same axis e_i can be created at the same time. On the other side, steps to be created on meta-faces associated with other axis requires a new decomposition of meta-faces into quads.

Crossing step It is possible that two steps need to be created on the same quad: one in each direction. In this case, at the bands intersection, the flagging comes for the value of N for the the highest step, as illustrated in Fig. 14.

Degenerated quad Degenerated quads require a special treatment because the band (where the major axis must be redefined) must touch an edge of the quad. Indeed, if the band is defined as usual in the middle of the quad, a part of the adjacent meta-face will be shrunk. However, it is not sufficient to place it at the border as proposed by Gregson *et al.*, because another meta-face may have to shrink (Fig. 15). To overcomes those issues, we move the degenerated quad edge prior to placing the step (Fig. 13).

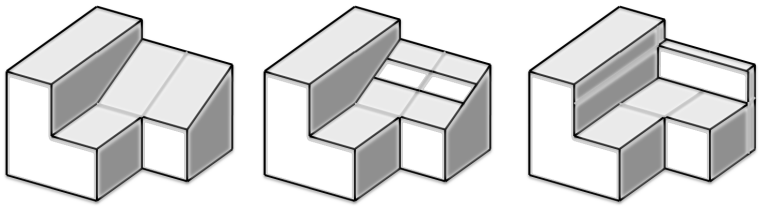

Fig. 12. CreateStep Assign a band of a metaface to another major axis (middle) creates a step that relax the planarity constraint between each sides of the band (Right).

9 Results

Our algorithm running time is very low (always less than one minute) because we are working on the meta-mesh. We have tested our algorithm on different type of objects to obtain the following results.

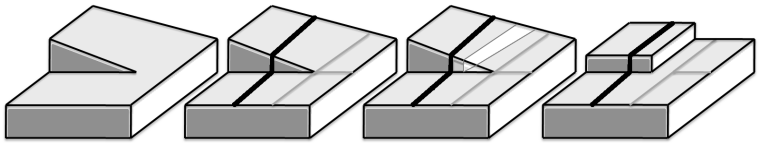

Fig. 13. When two adjacent meta-edges are assigned to opposite axis, the quad decomposition isolates the degenerated quad (bolt cut). The axis assignment close to the degenerated edge is changed to the axis associated to the adjacent meta-face, and a step is created.

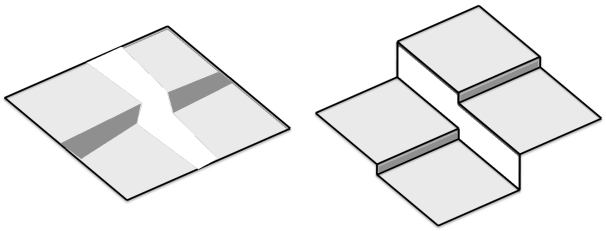

Fig. 14. Double step Two orthogonal steps can be created on the same meta face. At the intersection of the two bands, the axis assignment is set to the higher step size (Left) to ensure possible .

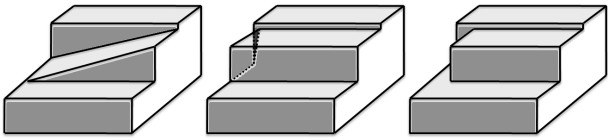

Fig. 15. Degenerated quads require to place a step adjacent to the degenerated edge (Left). It may be impossible to do so without shrinking a part of an adjacent meta-face (Middle). Our solution first removes the extremity of the degenerated edge, leading to a valid polycube (Right).

CAD/CAM objects (Fig. 16)

The axis alignement can work without editing the normal constraints (upper-row). If the object is mostly aligned with majors axis (middle row) it is not always sufficient to avoid editing the normal constraints (see close-up between the bars on the torus). As illustrated in the bottom row, our system is able to find a solution even with noisy initial normal constraints.

Other objects (Fig. 17)

On scanned meshes like the Stanford bunny, it is often requiered to add many steps, but most of them are due to degenerated quads. On the synthetic example of the screw, determining where steps must be placed can not be done locally. Our formulation with a global constrained linear system allows to deal

with these complex conflicts. The bottom row shows a step (in circle) created due to a contraint on a long loop (white).

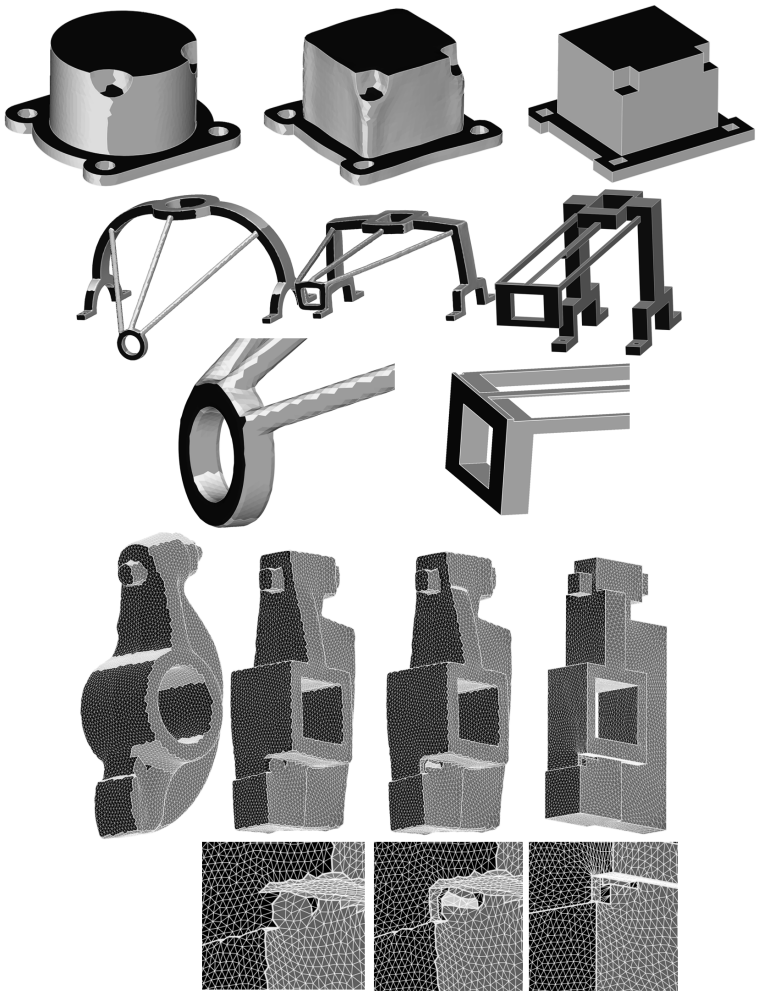

Fig. 16. Some CAD/CAM objects

10 Discussion

For clarity, we did not give all details and justifications during the presentation of our algorithm. We first better characterize for which is the target type of object of this method. We explain why the quad decomposition have to

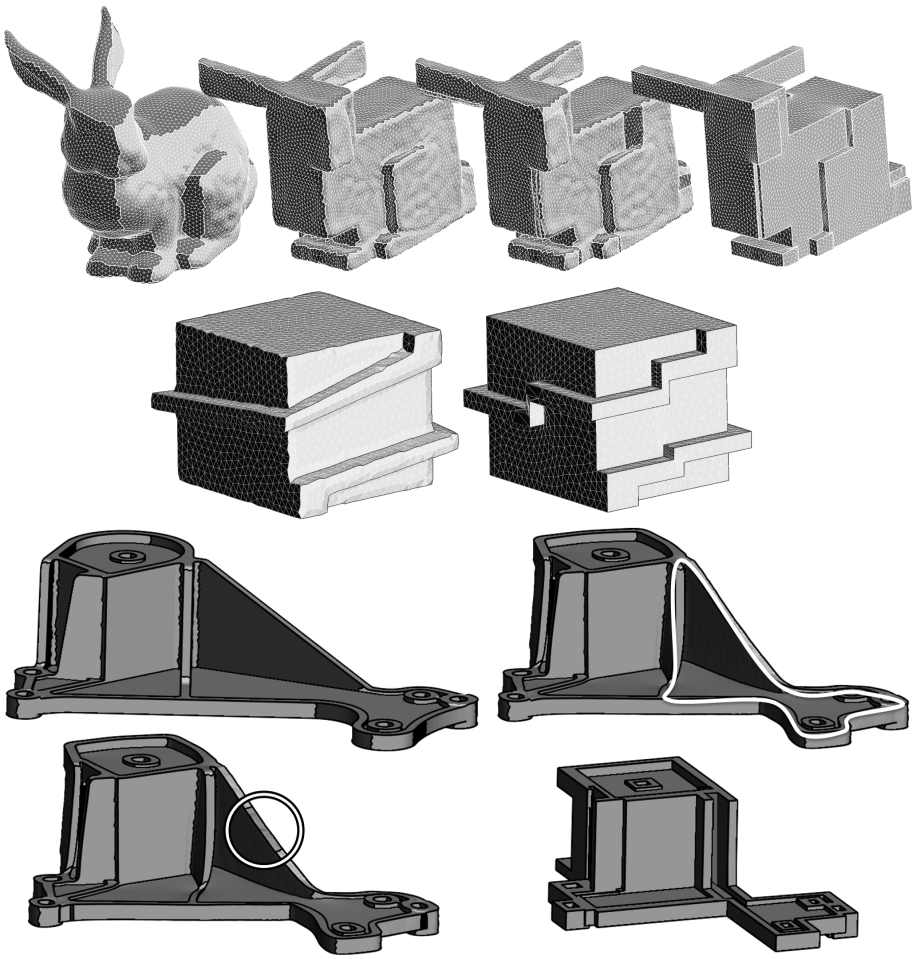

Fig. 17. A scanned mesh and an artifical example of a screw.

be dependant on the current axis to be solved. We present difficult cases of degenerated quads that requires several passes of our algorithm to be solved. We also discuss the problem of possible volumic foldovers that could be created by our algorithm.

10.1 Characterization of target objects

This paper is devoted to remesh smooth objects like bones, petroleum reservoirs, etc. These objects have no or few hard edges, making it possible to change the mapping of the polycube edges on the surface. We can also process many CAD/CAM models, but having a regular (distorted) grid inside the volume makes it hard to obtain nice results even on simple objects such

as prism, pyramids, etc. Indeed, these objects can not be nicely approximated by polycubes.

10.2 Why doing an axis dependant quad decomposition ?

As explained in Section 5, the decompositon of meta-faces into quads is axis dependant. It is motivated by two reasons:

- *avoid T-junctions to generate impossible steps* A T-junction splits a quad edge into two. If this quad edge lies on an horizontal meta-face, it will generate two "'step"' variables able to create different step positions. If the quad strip crossed by the step ends at both extremities with such T-junction, the step to create may have no valid position.(Fig. 18—Left)
- *avoid to generate an over constrained system* Every single T-junction could be avoided by iteratively splitting the quad having a splitted edge, but this process will not converge.

Our process cuts vertical faces only in the vertical direction to avoid T-junctions between vertical meta-faces, and propagate these T-junction on the horizontal meta-faces to prevent having edge split in these faces. The remaining T-junctions adds a vertex in the middle of an horizontal edge of a vertical quad. As illustrated in Fig. 18—Right, these reamining T-junction can not produce impossible situations, and only requires to introduce a new slack variable to prevent foldovers.

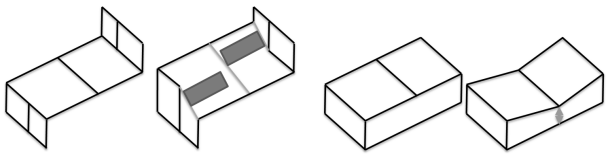

Fig. 18. A T-junction having a double edge on an horizontal meta-face may generate a set of non horizontal edges that cannot be crossed by a single step (Left). However, if the double edge of the T-junction is associated to another meta-face, it is sufficient to add a slack variable to prevent foldovers in the vertical meta-face (Right).

10.3 Degenerate quads hard to fix

When it is required to create a step on a quad strip that includes a degenerated quad, the situation becomes hard to solve (see Fig. 19). We detect such cases and try to solve other dimensions first. We did not see any case and we were not able to produce an example where all dimensions are locked at the same time.

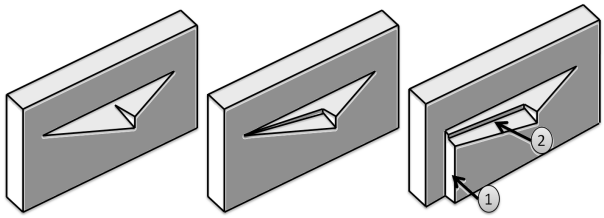

Fig. 19. Dead lock for solving the first dimension (Left). Creating a step would create two new degenerated quads, including one in the current dimension to be solved (Middle). Solving the other dimension first (Right—1) makes it possible to solve the current dimension ((Right—2)).

10.4 Prevent surface self intersections

The current results ensure that the boundary surface can be deformed to satisfy the normal constraints. This does not prevent volumic foldovers. It should be possible to detect the bounday intersections, and place slack variables to avoid such situations. In practice, such situations appear in some CAD/CAM models when the object thickness is very low. However, detection of conflicting meta-faces is beyond the scope of this paper.

It is interesting to notice that introducing such slack variables in our system should even be able to create steps if needed, as illustrated in Fig.20.

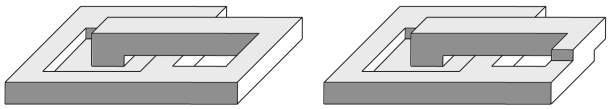

Fig. 20. A deformation that will allign the faces of the left object would push the hole under the opposite face, creating a volume foldover. Preventing such situation would require to create a step (Right).

Conclusion

Automatic generation of polycubes is a challenging problem and affecting the wished polycube normal to the original surface prior to constructing the polycube is a promizing idea. In this work, we have presented some limits of this approach as well as a solution to resolve the surfacic foldovers issues. It makes it possible to convert many challenging surfaces into polycubes such as Escher style polycubes, screws, or situation involving partial shrink of the polycube face (degenerated quads). The polycube construction could be made more robust by dealing with the volumic foldovers, and provide nicer results by both

pre-processing (better smooth object deformation) and post-processings (optimizing the mapping between polycube and surface, use a better constrained deformation, apply suitable local hex mesh operations).

References

1. Tarini, Marco and Hormann, Kai and Cignoni, Paolo and Montani, Claudio, PolyCube-Maps (2004) ACM Trans. Graph.
2. Ying He, Hongyu Wang, Chi-Wing Fu, Hong Qin A divide-and-conquer approach for automatic polycube map construction (2009) Computers & Graphics
3. Lin, Juncong and Jin, Xiaogang and Fan, Zhengwen and Wang, Charlie C. L. Automatic polycube-maps (2008) Proceedings of the 5th international conference on Advances in geometric modeling and processing
4. Shenghua Wan, Zhao Yin, Kang Zhang, Hongchao Zhang, Xin Li A topology-preserving optimization algorithm for polycube mapping (2011) Computers and Graphics
5. Shuchu Han, Jiazhi Xia, Ying He Hexahedral shell mesh construction via volumetric polycube map (2010) Symposium on Solid and Physical Modeling
6. Nicolas Ray, B Vallet, WC Li, B Lévy N-Symmetry Direction Fields on Surfaces of Arbitrary Genus (2006) ACM Trans Graph
7. Ju, Tao and Schaefer, Scott and Warren, Joe Mean value coordinates for closed triangular meshes (2005) SIGGRAPH '05
8. Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun Designing Quadrangulations with Discrete Harmonic Forms (2006) ACM/EG Symposium on Geometry Processing
9. F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust. Computing a canonical polygonal schema of an orientable triangulated surface (2001) ACM SoCG , pages 8089.
10. Wan Chiu Li, Bruno Lévy and Jean-Claude Paul Mesh Editing with an Embedded Network of Curves (2005) IEEE International Conference on Shape Modeling and Applications
11. J. Gregson, A. Sheffer, E. Zhang (2011) All-Hex Mesh Generation via Volumetric PolyCube Deformation Computer Graphics Forum (Proc. SGP)