Abstract

Visualisation, assembling/disassembling simulation, reverse engineering, or finite element analyses are so many application fields where a polyhedral simplification can be mandatory. In most of the configurations, such an operation must preserve the significant shapes of the original model. This paper addresses the way these simplifications can be performed while preserving character lines, i.e. those lines which affect significantly the object visual appearance. The proposed simplification process is based on a vertex removal algorithm that uses a heterogeneous map of sizes computed with information extracted from 2D images of the object. The more the vertices are far from the character lines and the more the simplification is important. This method reduces the computational time since there is no need to compute neither the curvature nor the saliency at each vertex. In addition, a first attempt to enrich a polyhedral model with semantic information extracted from images is thus proposed.

1. Introduction

Today, the amount of data that are needed for the design of products is in constant increase. For example, the complete digital mock-up (DMU) of an airplane Airbus A400M represents about fifteen tera-octets of data. Among these data, those relative to the definition of the product geometry represent about ten tera-octets for the CAD models and about two tera-octets for the polyhedral models. Consequently, all these information cannot be used simultaneously. Depending on the application field, e.g. visualisation of the geometric models, assembling/disassembling simulation or finite element analyses, the DMU must be filtered and adapted as fast as possible. This is even truer when the DMU is shared between several actors located on multiple sites. In this paper, we concentrate on the polyhedral models that are widely used all along the product design process (e.g. visualization, simulation or manufacturing) as well as on the models and methods to simplify them. Even if today the graphic cards can display several tens of millions of triangles, the simultaneous visualization of several hundreds of millions of triangles is not yet possible (four hundred millions of triangles for the complete polyhedral description of the airplane Airbus A400M). Moreover, it is obviously useless both to try to display thousands of triangles with a single screen pixel and to try to display hidden parts of the visualization model. This assertion is even truer when the data have to be transferred in real-time through Internet. But the simplification needs also exist in simulation where the complexity of the computations performed on the polyhedral models has to be optimized. This is the case when simulating the assembling/disassembling of a product or for finite element analyses where solely some simplified representations can be treated in reasonable computation times. All these arguments in favour of the development of polyhedral simplification techniques remain true in reverse engineering, for the reconstruction of architectural edifices, in medical visualization, in image synthesis, for movie special effects and video games.

The simplification of a polyhedral model consists in reducing the number of triangles according to a set of user-specified criteria. Depending on the application domain, it can be interesting either to preserve the object overall shape, or to minimize the variation of its volume, or to produce equilateral triangles, or to simplify according to a combination of these criteria. Concerning the simplifications that preserve the shapes of an object, there are those which take care of the so-called character lines. Character lines correspond to those curves that affect significantly the object visual appearance [1]. They impress a kind of visual signature enabling the distinguishing between several objects and this independently of the underlying geometric models. Therefore, having a simplification technique that could preserve these key elements would help the definition and maintenance of a multi-resolution DMU structured around the concept of features. Here, the features would correspond to
high level entities built on top of the character lines and giving a meaning to a set of triangles [2].

![Shapes and character lines](image)

**Fig. 1:** Shapes can be considered as a set of triangles on a polyhedral model (a), as a set of pixels in a photo (b), or as a set of pixels in a filtered image (c). Character lines can also be identified on all these models.

Actually, the concept of feature is close to the notion of shape, the latter being more generic in the sense that it corresponds to a multi-dimensional media characterized by a visual appearance in a space of one, two, three or more dimensions [3]. As a consequence, a part of a polyhedral model, an area of photography or some vectors of a filtered image can be considered as shapes. Similarly, the character lines of an object can be highlighted either on its polyhedral model, or on its photo, or on one of its filtered versions (fig. 1). In this paper, we take advantage of this bringing together to propose a polyhedral simplification algorithm that uses at best the available shape representations. More precisely, the polyhedral simplification is driven by a heterogeneous map of tolerances that preserves the details in the surrounding of the character lines extracted from 2D images of the object. These images can either come from photos of the real object or from screenshots of the displayed model. Using such a process, the character lines extraction is smoothed and the computation times are reduced.

The paper is organized as follows. The section 2 summarises and compares the existing simplification techniques. Section 3 recalls the principle and the main advantages of the adopted simplification algorithm. The proposed image-based polyhedral simplification algorithm is then presented and detailed in section 4. The section 5 treats an example and discusses the results. The last section concludes the paper and discusses the perspectives in other application domains.

2. Related work

The state-of-the-art on the polyhedral model simplification techniques has been divided in two parts: one concerning the polyhedral simplification mechanisms and the other relative to the approaches that try to preserve shape features.

### 2.1. Polyhedral simplification mechanisms

Basically, the polyhedral simplification mechanisms can be classified in three categories: the adaptive subdivision, the geometry removal and the sampling algorithms [4].

**Adaptive subdivision** algorithms subdivide recursively a simple basic model until it approximates the original one according to a user-specified quality criterion. Such algorithms are well adapted to the simplification of polyhedral models whose basic model is well-known or easy to find. For example, a terrain model will have a rectangle as starting basic model.

**Geometry removal** algorithms iteratively remove faces and remesh the deleted areas. The face removal operations can come from either a vertex removal operation or an edge contraction. The process is repeated until the simplified model reaches a target degree of approximation. These algorithms are interesting on extremely complex models which require drastic simplifications. Most of them do not authorize vertex or face removal operations that would induce topological changes.

**Sampling** algorithms sample the original model with either some points distributed randomly on the outer surface or with a voxels overlaid 3D grid. This technique is among the most sophisticated one but it is hard to implement. It suits well the simplification of smooth organic models with no sharp edges but it does not succeed in sampling efficiently highly-frequency features.

### 2.2. Features-preserving simplification approaches

Since many researches have been undertaken to try to simplify polyhedral models while preserving shape characteristics, solely the most relevant ones will be presented. They can be classified in two main categories: the geometric and the perceptual approaches. Each of them is based on one of the three mechanisms presented in section 2.1.

Among the geometric approaches, some make use of a quadric error metric to assign costs to the edges. These costs are then used to drive an edge contraction iterative algorithm [5]. At each step, the lowest-cost edge is contracted if the costs of the surrounding edges stand acceptable after the simplification. Other methods try to preserve the mesh features while using a curvature tensor estimation [6] or a centre-surround operator on Gaussian-weighted mean curvature [7]. All these methods require the computation of local geometric quantities on the whole polyhedron.

The perceptual approaches choose the elements to be simplified while using criteria like visibility [8], luminance [9], or texture [10] variations between the original and the simplified models. Some of them use
images and optical sensors to compute the simplified models. For example, the cost of an edge collapse can be determined by performing the collapse, rendering the corresponding model from multiple views and computing the luminance mean-square error between pixels of the original model’s screenshots and pixels of the simplified model’s screenshots [9].

Unfortunately, most of these approaches still suffer from drawbacks and notably in terms of computation time. Actually, curvature-based methods take a long time to compute the discrete curvature at each vertex of the polyhedron to simplify. Moreover, curvature estimators may fail in detecting the most relevant features of a polyhedron. It has been proved that the curvature does not pay attention to “regional importance”, that is, for instance a thin excrecence of high-curvature within a huge low-curvature area will be considered as an important shape even if it corresponds to an irrelevant feature [7]. The perceptual methods, i.e. those based on the comparison between screenshots of the original and simplified models, spend too much time to iteratively compute the simplifications, to render the simplified models and to compare the original and final screenshots. This is particularly true when dealing with polyhedral models made of several hundreds of millions of triangles. Actually, since you need to display the huge amount of triangles several times before simplifying it, this method is clearly not adapted to our requirements. Furthermore, these methods are unable to use photographs taken with a digital camera, which can be of interest in the context of architectural reconstruction.

Based on these analyses, we propose a simplification approach that overcomes the previously highlighted drawbacks. As explained in the introduction, our method considers shapes as being high-level entities identifiable on polyhedral models as well as on images. More precisely, the algorithm is driven by a set of character lines extracted from images of the original object whose polyhedral model has to be simplified. The images can either be photographs of the object or simple screenshots. The simplification process is fast since it requires neither the computation of curvatures nor saliencies. Actually, the detection of the character lines uses digital images that already correspond to simplified and smoothed versions of the shape. Each file to be analyzed will never exceed five mega-octets, which can be compared to the giga or tera-octets that the polyhedron may represent. Our approach is comparable to the one of [11] which drives a classical simplification process with geometric error constraints that are specified locally on the polyhedron. In their case, the error constraints are led by the user sensibility who specifies the areas of the polyhedron that have to be simplified more than others. Here, these information are extracted from images.

3. Adopted polyhedral simplification engine

The adopted simplification engine uses a vertex removal iterative process driven by two geometric criteria [12,13]: an error area criterion which imposes the simplified model to remain inside an envelope of user-specified error spheres initially associated to the vertices; and an optional discrete curvature criterion used to control the remeshing operation. The algorithm can be decomposed as follows:

- **Initializations:**
  - initialization of the error spheres for each vertex. The radii are not necessarily equal the one with the others to enable more or less decimation in particular areas. It results in a so-called map of sizes.
  - initialization of the lists of dependencies to the map of sizes. For each face of the polyhedron, a list is created and contains the error spheres that intersect with this face. In case of a triangle, this list contains at least three error spheres.
  - initialization of the ordered list of vertices \( L_v \), that have to be simplified. All the vertices are initially tagged as “removable”.

- **Loop on all the removable vertices contained in \( L_v \) – Simplification of \( V_i \):**
  - creation of a list \( L_z \) of error spheres that gathers together the lists of dependencies associated to the faces connected to \( V_i \). This list contains naturally the error sphere \( Z_i \) associated to \( V_i \) (fig. 2.a);
  - remove the vertex \( V_i \) as well as the faces connected to it;
  - fill in the hole while using the discrete curvature criterion which tends to preserve the curvature distribution of the initial polyhedron (fig. 2.b). Using such an option, the filling process connects first the vertices that have the highest discrete curvatures;
  - test if the result is acceptable (fig. 2.b):
    - if all the error spheres of the list \( L_z \) intersect at least one newly inserted face, the simplification is acceptable with respect to the error area criterion. New lists of dependencies can be assigned to the new faces. Since the produced configuration may affect some previously tested vertices, the set of vertices surrounding \( V_i \) are tagged as “removable”; otherwise, the simplification is not acceptable and the vertex \( V_i \) is tagged as “not removable”. Both the polyhedron and the lists of dependencies of the previous iteration are kept.

- **End of the simplification:**
  - the process stops when no more vertices of \( L_v \) are removable.
The rest of the document:

- the simplification algorithm will not anymore compute the discrete absolute curvature during the remeshing step. The remeshing will be driven by an equilaterality criterion (contribution 3).

4. Image-based polyhedral simplification algorithm

The main steps of the proposed image-based polyhedral simplification algorithm are depicted on the figure 5. The process starts with both a fine polyhedron and one or several image(s) of the object. The images can either be simple screenshots or photographs of the physical object. The first step is relative to the extraction and vectorization of the contours inside the images. This aims at revealing the character lines of the object. To enable a mapping between the identified character lines and the 3D polyhedron, one or several calibration(s) of the camera(s) are mandatory. Once the image(s) and the polyhedron are in the same reference frame, the contours can be projected onto the polyhedron. The simplification is then run. It is driven by a heterogeneous map of sizes computed from the identified character lines.

One of the advantages of such a linear process lies in the fact that it is modular and that each module can easily be bypassed to any existing tools and/or algorithms. It results that the most appropriate tools can always be used to perform each specific task.

4.1. Contours extraction

Most of the contours detection methods are based on the assumption that a contour occurs where there is a fast change in the intensity function I(x,y) of the 2D image (fig. 4.a and 4.b). Therefore, contours can be extracted by convolving the intensity function with a filter that approximates a first or a second order derivative operator. They correspond to the peaks (resp. the zero-crossings) in the convolution output when using a filter that approximate the first (resp. second) order derivative operator (fig. 4.c to 4.d).

\[
\sum_{i=-1}^{1} \sum_{j=-1}^{1} \alpha_{ij} I(x+i, y+j) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \beta_{ij} \frac{\partial I}{\partial x} |_{y=y_i} + \sum_{i=-1}^{1} \sum_{j=-1}^{1} \gamma_{ij} \frac{\partial^2 I}{\partial x^2} |_{y=y_i}
\]

The convoluted intensity function is defined for each pixel (x,y) of the image, with a filter approximated by a matrix F of dimension 3x3 is defined as follows (fig. 4.e):

\[
\text{conv}(I, F)(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} I(x-1+i, y-1+j)F(i,j)
\]
In order to remove the noise in the image, a median smoothing filter is applied in a pre-processing step. This filter is approximated by a matrix $M$ of dimension $3 \times 3$ with its nine elements equal to $1/9$. It assigns to each pixel the median value of the grey levels of the nine surrounding pixels. The median smoothed intensity function is then:

$$I_{\text{med}}(x, y) = \text{conv}(I, M)(x, y)$$  \hspace{1cm} (2)

In two dimensions, the first and second order derivative operators respectively correspond to the Gradient and Laplacian operators. The filtering based on a Gradient operator seems to yield a clearer correlation between peaks and contours than the filtering using a Laplacian operator. Moreover, the result of a Laplacian filtering is more affected by the presence of noise, whereas the result of a Gradient filtering enhances the depth of the detected contours (fig. 6.a to 6.c). The filter adopted for the detection of the character lines is the Sobel filter that uses a Gradient operator (fig. 6.d). Since it requires a noiseless image, its use will always be preceded by the application of a median smoothing filter. The Sobel filter [14] detects separately the horizontal and vertical fast changes and gives both the magnitude of the fast variations as well as their directions:

$$I_{\text{sobel}}(x, y) = \sqrt{\text{conv}(I_x, H)(x, y)^2 + \text{conv}(I_y, V)(x, y)^2}$$  \hspace{1cm} (3)

$$f_{\text{sobel}}(x, y) = \tan^{-1}\left(\frac{\text{conv}(I_y, V)(x, y)}{\text{conv}(I_x, H)(x, y)}\right)$$  \hspace{1cm} (4)

with $H = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ and $V = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$.

The horizontal and vertical matrices $H$ and $V$ come from the approximation of the first order derivatives with second order centred finite differences. To finalise the filtering and thus obtain a binary black and white image of the fast variations in the intensity function, a threshold is applied (fig. 6.d). Thus, in our application we do not use the information concerning the direction of the fast changes.

4.2. Contours vectorization

Two categories of vectorization algorithms exist: thinning-based methods and non-thinning ones [15]. Thinning-based methods usually use an iterative boundary erosion process to remove pixels until a one pixel width skeleton remains. However, these techniques are slower than the non-thinning approaches. Among the later, the Orthogonal Zig-Zag (OZZ) algorithm [16] has been adopted for its simplicity and its efficiency which is due to a sparse sampling of the image. It is based on tracking a one pixel width path within the foreground area (black pixels) of an image, turning orthogonally each time the path hits the boundary of the area (fig. 7). This algorithm partially overcomes the problems that may arise at the intersections and junctions between foreground areas. It also preserves the line width. The main disadvantage concerns its low capacity to approximate arcs, the neighbouring vectors being often either in intersection or disconnected.

Fig. 5: Principle of the image-based polyhedral simplification algorithm

Fig. 6: Screenshot of the rocker arm (a) respectively filtered with the Gradient (b), the Laplacian (c) and the Sobel (d) filters.
in a camera without taking into account some real optical phenomena such as distortion [18,19]. It is an optimization of the OZZ algorithm. Rather than visiting all the pixels of a foreground area at least once, only a selected subset of the medial axis points is visited. The result is a polyline smoothed by a polygonal approximation algorithm [17] to remove redundant points.

4.3. Camera calibration

To retrieve the relationships between points of the 2D filtered image and points of the 3D polyhedron, a camera calibration step is required. Calibrating a camera consists in finding the position, the orientation and the focal distance of a camera in a global reference frame containing the 3D polyhedron and the 2D image (fig. 8). The adopted camera model is the ideal pinhole camera model which is based on the Euclidian geometry. This model formalises the image formation process which occurs in a camera without taking into account some real optical phenomena such as distortion [18,19]. It is defined by an optical centre C, the orientation of the camera local reference frame (C,C_x,C_y,C_z) in the global reference frame of the scene (O_x,y,z), and a focal distance f = Cc where c is the centre of the 2D image (fig. 8).

The camera owns intrinsic as well as extrinsic parameters that the calibration process must estimate. Let (x_i,y_i,z_i) be the coordinates of a point P_i of the 3D scene and (u_i,v_i,w_i) the homogeneous coordinates of its projection p_i onto the image plane. The calibration aims at computing the matrix M such as for each couple of points (p_i,P_i):

\[
\begin{bmatrix}
  u_i \\
  v_i \\
  w_i
\end{bmatrix}
= M
\begin{bmatrix}
  x_i \\
  y_i \\
  z_i
\end{bmatrix}
\]

with

\[
M = \begin{bmatrix}
  m_{11} & m_{12} & m_{13} & m_{14} \\
  m_{21} & m_{22} & m_{23} & m_{24} \\
  m_{31} & m_{32} & m_{33} & m_{34}
\end{bmatrix}
\] (5)

The writing of the equation (5) for n couples of points (p_i,P_i) leads to the resolution of a homogeneous system of 2n equations with solely twelve unknowns (the m_ij coefficients). In order to get a non trivial solution, an additional constraint must be imposed. In our implementation, we use the Faugeras and Toscani’s constraint [20] which imposes that m_{11}^2 + m_{22}^2 + m_{33}^2 = 1, thus facilitating the resolution of the equations system. The whole system is then solved while using a least squares minimization. Since the system to be solved is composed of 2n+1 equations and twelve unknowns, at least six couples of 2D/3D points are mandatory, each of them giving two equations. Of course, the more the number of couples is important, the more the calibration results are accurate. Moreover, specific constraints have to be fixed to avoid degenerated configurations. For instance, all the configurations that would make M become noninvertible as taking the 3D points on the same plane or on a twisted wire must be avoided.

Finally, one can notice that the camera calibration step is optional when dealing with screenshots or georeferenced images since in both cases the parameters of the camera are already known.

4.4. Contours projection

Once the camera is calibrated, the image contours can be mapped onto the 3D polyhedron. Depending on the way the initial polyhedral model has been obtained, two edges projection algorithms might be of interest. In case of a tessellated model, the sharp edges already exist and the problem is to identify which existing vertices match the image contours (fig. 9.a). There is no need to insert new vertices. In case of a scanned model, the edges of the reconstructed polyhedron do not necessarily match the image contours (fig. 9.b and c). Thus, the insertion of new vertices is required to avoid aliasing effects (fig. 9.d). The projection of a contour properly saying is performed in two steps:

- the contour end points p_1 and p_2 are projected in two points P_1 and P_2 of the polyhedron. This is performed while computing the intersections of the rays (Cp_1) and (Cp_2), going from the optical center C to the end points, with the faces of the polyhedron [22]. In case of a scanned model, i.e. in case new vertices must be inserted, different cases must be considered to avoid numerical instabilities: if the intersection occurs close to an existing vertex, the projected point is move...
to that vertex; if the intersection is close to an edge, this edge is split in two at the intersection; otherwise the face is subdivided in three new faces.

- the computation of the intersection of the plane \((C_{p1}p_2)\) with the polyhedron [22]. If required, the insertion of the new edges is performed while paying attention to produce a local triangulation which is still a Delaunay one. In most cases, an error sphere is just created at the 3D position of the intersection. Hence, as the simplified model has to be contained by the hull of the error spheres, the character lines are preserved.

Finally, one can notice that in some cases the projection may never exist. It occurs when the identified contours are on the object boundary inside the image. Effectively, the ray 1 of the figure 8 may never reach the polyhedron if there is a significant error in the camera calibration. The solution lies in the use of another image taken from a more adapted point of view.

4.5. Map of sizes computation and polyhedron simplification

Before running the simplification properly saying, the heterogeneous map of error spheres has to be built in such a way that the more the vertices are far from the character lines and the more the decimation is important. The idea is to perform a distribution of the error spheres so that their radius decreases when the geodesic distance between a vertex and a character line decreases (fig. 10.a).

To simulate such behaviour while avoiding the computation of the complex and time consuming geodesic distances, a specific filtering process has been designed. The idea is to start by filtering several times the 2D image thus giving rise to several contours \(I_1, I_3, I_5, \ldots, I_{n-2}, I_n\) having a contour thickness increasing respectively from 1, 3, 5 ..., n-2 to n pixels. The images \(I_k\), for \(k \in [3..n]\), are then subtracted from the images \(I_{n-2}\) thus producing the new contours \(\Gamma'_{k-2}\) (figures 10.b to fig. 10.f with \(n = 5\)).

Once this is performed, the projection algorithm developed in section 4.4 is run with the images \(\Gamma_k, k \in [1..(n-1)/2]\). For each point obtained by projection of a black pixel contained in the image \(\Gamma_k\), an error sphere of radius \(r(k)\) is created. The evolution of \(r(k)\) follows the evolution of a sigmoid function defined by four user-specified parameters: the radius of the smallest sphere \(r_1\), the number of filtering \(n\) which corresponds to the width of the sigmoid, the difference between the radius of the biggest sphere \(r_{(n-1)/2}\) and \(r_1\) which represents the height of the sigmoid, the position of the extrema of the derivative \(r'(k)\) as well as its value (fig. 10.g) which represents the velocity in the evolution between the smallest and biggest spheres.

Once the map of sizes is computed, we use our prototype software to perform the simplification as explained in the section 3.

5. Results and discussion

The whole image-based simplification process (fig. 5) has been applied to the polyhedral model of a scanned climbing hold (courtesy Tomoadour). The figure 11 shows the various steps relative to the detection of the character lines inside a screenshot of the displayed model.

Once the character lines have been identified the projection on the initial 3D model (fig. 12.a) is
run. It results in a non uniform map of error spheres that is used to drive the simplification process. Here, the ordering of the vertices candidates to the removal operation has been performed according to the size of the spheres. The resulting polyhedron is composed of 2628 faces (fig. 12.b).

Fig. 12: Initial climbing hold model (a). Results of simplification using either the equilateral remeshing (b) or the curvature variation preservation remeshing (c).

The figure 12.b is obtained while using an ordering of the vertices as well as a remeshing algorithm based on a discrete curvature estimator [13]. All the radii of the spheres have been initialized with a single value such that the resulting polyhedron has exactly 2628 faces. Consequently, one can notice that for a given number of remaining faces, the quality of the character lines preservation is much better using our new image-based simplification process. In addition, the automatic generation of the map of error spheres is far quicker than if it would be made manually and we do not have to pay the huge cost of a curvature calculus. But the map of sizes computation could be even faster with the use of bounding boxes for the projection.

Finally, the two remeshing techniques are compared using either the new imaged-based simplification approach (fig. 13.a and 13.c) or a curvature-based simplification (fig. 13.b and 13.d). Here again, for the same number of simplified triangles and whatever the remeshing technique is, the character lines are much better preserved when using our imaged-based algorithm.

6. Conclusion and future works

In this paper, a complete and modular image-based polyhedral simplification algorithm has been proposed and validated. It aims at preserving the so-called character lines during the decimation process. The simplification properly saying is based on a vertex removal algorithm driven by a heterogeneous map of error spheres whose hull defines the boundary inside which the polyhedron has to stay. The size of the spheres is computed automatically while using a newly developed image filtering method to approximate the geodesic distance between the vertices and the character lines that are extracted from 2D filtered images.

Even if the achieved results prove the efficiency of our overall workflow, there are still many possible improvements for each module. One of them concerns the reduction of the calibration errors. These errors are due to the inaccuracy the user has when coupling the pixels of the 2D image with the 3D points of the polyhedron. To overcome this limit, we would like to investigate the use of methods for self-calibration or auto-calibration which does not require the specification of correspondences but which use several images [21]. An other field of improvement concerns the contours projection step that could be faster while using an octree of bounding boxes.

Until now, in mechanical engineering, the images were mainly used to complete the description of the object (DMU enrichment) but almost never as a source of information that could help the processing of the geometric models. In this sense, the proposed approach open the way to new applications, and not only in the polyhedral simplification context. For example, in reverse engineering, the use of photographs could help the filling of holes which often result from the scanning and reconstruction processes. In some cases, it will be easier to take a photography of an object than to move the laser scanner in order to reach the desired area of the object (e.g. the roof of a high building). New application could also be imagined in the field of features extraction and mesh segmentation.

Such results and perspectives fit well the objectives of the Aim@Shape European Network of Excellence [3]. Given this context, two additional results can be emphasized. First; the proposed approach is a first attempt to the definition of a multi-representation model composed of polyhedron and images. Second, it is also a first attempt to the enrichment of a geometric model with semantic information extracted from images.

7. References


Remeshing with curvature variation preservation

Remeshing with the equilaterality criterion

Fig. 13: Comparisons between the curvature variation preservation remeshing (a,b) and the equilaterality criterion remeshing (c,d) using either the new imaged-based simplification approach (a,c) or the curvature-based approach (b,d).


