TECHNICAL REPORT N. 4 2008
IMATI-GE / CNR
Consiglio Nazionale delle Ricerche

Year 2008, Number 4

Hierarchical Convex Approximation of 3D Shapes for Fast Region Selection

Marco Attene, Michela Mortara, Michela Spagnuolo and Bianca Falcidieno

IMATI-GE / CNR, Italy

Abstract

Given a 3D solid model S represented by a tetrahedral mesh, we describe a novel algorithm to compute a hierarchy of convex polyhedra that tightly enclose S. The hierarchy can be browsed at interactive speed on a modern PC and it is useful for implementing an intuitive feature selection paradigm for 3D editing environments.
Convex parts often coincide with perceptually relevant shape components and, for their identification, existing methods rely on the boundary surface only. In contrast, we show that the notion of part concavity can be expressed and implemented more intuitively and efficiently by exploiting a tetrahedrization of the shape volume.
The method proposed is completely automatic, and generates a tree of convex polyhedra in which the root is the convex hull of the whole shape, and the leaves are the tetrahedra of the input mesh. The algorithm proceeds bottom-up by hierarchically clustering tetrahedra into nearly convex aggregations, and the whole process is significantly fast. We prove that, in the average case, for a mesh of n tetrahedra $O(n \log^2 n)$ operations are sufficient to compute the whole tree.

1. Introduction

In the last decade, 3D content creation crossed the boundaries of specialized computer graphics sectors and is now presented to users that do not have expertise in sophisticated modeling techniques. Meshes and easy-to-use editing operators are key ingredients to support emerging shape modeling systems based on the so-called *modeling by composition* paradigm. In these systems it is possible to pick existing models from a repository and to interact with them by selecting and detaching portions of interest to compose a new object [FKS*04]. Hence the call for intuitive mechanisms for selecting regions of interest that can actively involve even non-expert users in the creation and re-use of 3D content.

Intuitiveness of part selection can be supported by organizing the object shape in a hierarchy of convex parts. Convex decompositions play a crucial role in shape perception and are correlated with the so-called "minima rule" that states that humans tend to perceive a shape as a composition of parts separated through lines of minimum curvature [HR84]. Hierarchical decompositions are also needed to provide the users with a view of the shape that reflects the level of detail of the various parts. Not by chance, recent approaches tend to decompose the shape in parts that conform to the concept of convexity [LA07, KJS07, CS07, KT03].

Though the notion of convex component is intrinsically related to volumes, existing approaches to convex decomposition have been developed for surface meshes [KJS07,LA07]. Instead, describing solid objects explicitly by tetrahedral meshes (or, for short, *tet meshes*) has a number of advantages when the object is being analyzed (section 3.1) and edited (section 4.2); moreover, some useful characteristics (i.e. number of cavities, local feature size, ...) can be computed much more easily if the shape is represented through a tet mesh. Producing accurate tet meshes has been a difficult issue for a long time, but recent advances in mesh generation [ACSYD05, SG05] make it possible to convert traditional surface meshes to tetrahedral meshes, even if the input is particularly complex. Moreover, the extra-complexity introduced by the representation of the "inner parts" is no longer an issue, as modern computers have enough resources to manage such information.

Summarizing, convex-based decomposition and hierarchical shape representation can provide a valid support to intuitive selection systems, and in this paper we propose a novel paradigm that couples all these aspects through an original

use of tetrahedral meshes. The main contributions of the paper are the following:

- A formal, intuitive and effective definition of concavity for volumetric parts (section 3.1);
- A hierarchical shape representation in terms of convex approximations, along with an efficient, easy to be implemented and completely automatic algorithm to convert a tetrahedral mesh to its hierarchical representation (section 3.2);
- A novel interaction paradigm to select regions of interest to be processed (section 4).

Through our hierarchical representation (Figure 1) users may easily select 3D regions which have a visually salient shape, independently of the scale; in their turn, the features selected can be deformed, edited or composed within extremely user-friendly interaction paradigms.

2. Related Work

We analyze related works according to the three main contributions of the paper (new definition of concavity for volume meshes; hierarchical approximation algorithm; interactive selection mechanism). Note that, even if our hierarchical approximation induces a segmentation of the shape, shape segmentation is not the main focus of this work. For recent surveys on this topic we point the reader to [Sha06, APP*07].

Part concavity - Convex regions of "natural" shapes, such as human bodies, animals or proteins, are captured by direct or indirect applications of the minima rule. Direct methods such as [KT03, KT05, ZL05] use geodesic distances to guide the segmentation, and negative curvature is taken into account to define the boundaries of the segments. Indirect approaches look for convex regions that, in their turn, are typically separated by curves placed in negative curvature areas. An example of such a method is presented in [CDST95], where the authors prove that computing an exact convex decomposition is NP-hard. This intrinsic difficulty motivated the algorithm presented in [LA07], where the surface is decomposed in just "approximately convex" regions through a top-down approach: after having established the concavity of a part (i.e. the maximum distance from its convex hull), the algorithm decomposes it if such a concavity exceeds a given threshold. Similar results are achieved in [KJS07], where surface meshes are segmented in nearly convex patches through a variational approach that minimizes the overall concavity of the patches, but in this case the concavity has been defined as the average distance of the region from its convex hull. In their work, the authors recommend to bias this metric with additional factors to force patch compactness and to compensate for boundary jaggedness.

Hierarchical clustering - The clustering approach our algorithm is based on is an extension of the hierarchical face clustering (HFC) described in [GWH01] for triangle meshes. The basic idea in the HFC approach is to merge neighboring triangles into representative clusters; A cluster is a connected set of triangles, not necessarily simply connected, with some properties that justify their aggregation. In [GWH01], for example, clusters have the property that their triangles can be effectively approximated by fitting planes computed through principal component analysis, and the *cost* of merging a set of triangles into a single representative cluster is the integral L^2 distance of its vertices from the fitting plane. By defining the cost as the minimum of the L^2 distances of the vertices from a set of fitting primitives, in [AFS06] the hierarchical clustering is used to extract useful features from CAD models.

Region selection - In an easy-to-use 3D modeling system, users should be able to specify the desired selections even without prior expertise (intuitiveness) and with a minimal amout of interaction and system latency (efficiency); also, the modeler should provide all the means to quickly specify all the selections the user can think of (flexibility). Though a number of segmentation methods [Sha06, APP*07] can be used to derive selections, this approach is not really efficient and flexible (to define a single region the whole shape needs to be segmented with proper parameters). More flexibility can be achieved by combining the results from several segmentations [ARSF07], but this is even less efficient and not always intuitive. Defining regions of interest by drawing their bounding curves would be definitely more intuitive, but unfortunately in 3D this operation involves a number of problems such as rotating the scene, and zooming in and out to show the portions of the surface to draw on: this class includes some systems that allow the user to select a sequence of vertices on the surface mesh and cut along the shortest paths between them [ZSH00], and other systems that allow the user to just roughly sketch some cutting lines which are automatically aligned to the closest seams of the surface [LL02]. Finally, some approaches try to minimize the human interaction by completing easily sketchable *strokes* [FKS*04]; nevertheless, when the feature to be selected is bounded by several curves or contains cavities, these approaches still require a lot of human intervention.

The results described in this paper can be used to design a system in which the selection of regions is both efficient and intuitive thanks to the underlying hierarchical representation of the shape.

3. Hierarchical Convex Approximation

Having convex parts organized in a hierarchical structure is important to realize user-friendly and effective systems for region selection. Therefore, we target the creation of a hierarchy that contains virtually all the convex features of the shape at all the scales, and that can be browsed by the user to find "interesting" features. Also, we argue that the conversion of an existing shape to such a hierarchical represen-

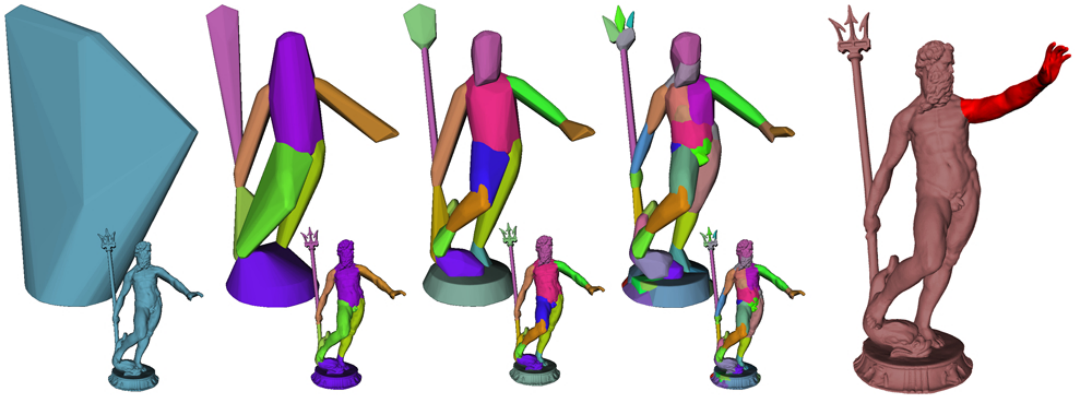

Figure 1: *A hierarchy of approximating convex polyhedra for the Neptune model (courtesy of AIM@SHAPE's Shape Repository) produced by our algorithm. On the right, the hierarchy was used to interactively select and deform a feature.*

tation should be fast and fully automatic, so that it can be transparently included in the loading stage. Hence, we have implemented an efficient algorithm that takes a tetrahedral mesh as input and computes a hierarchical convex approximation. Our algorithm is an extension of the HFC approach to tet meshes where the cost is a measure of the concavity of the cluster being created. In other words, the algorithm strives to merge tetrahedra so that the resulting clusters are as convex as possible. At any point of the algorithm, the clusters can be approximated by their convex hulls to obtain a simplified representation of the shape (Figure 1).

3.1. Definition of Concavity for solid models

As pointed out in [LA07], few methods have been proposed to measure the concavity of polygons [LA06] while for polyhedra no standard definition exists, and recent works use either the maximum [LA07] or the average [KJS07] distance from the convex hull. In the variational partitioning proposed in [KJS07], biasing the distance from the convex hull with additional factors is necessary to keep the surface patches compact and bounded by smooth lines. The importance of a bias term can be seen in the example of an egg broken in two pieces; the surface patch representing each piece has null distance from its convex hull, and this does not depend on the jaggedness of the patch's boundary (Fig. 2(a)). Thus, if smoother boundaries (Fig. 2(b)) are preferred, the distance from the convex hull must be biased. If we model the egg as a full solid (e.g. through a tet mesh), each of the two solid pieces is bounded by a closed surface (Fig. 2(c)), and its distance from the convex hull would be zero only if such bounding surface is convex. In other words, the sole minimization of the distance from the convex hull becomes sufficient to avoid too jagged connections between adjacent regions.

These considerations made us choose to represent the shape using a tetrahedral mesh instead of traditional surface

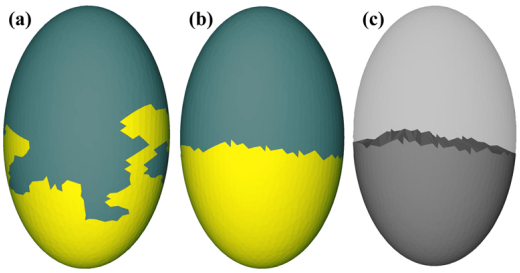

Figure 2: *An egg partitioned in two parts according to (a) the non-biased concavity measure described in [KJS07] and (b)(c) the volume-based concavity used here.*

meshes. In this setting, if S is a solid polyhedron an intuitive notion of concavity can be simply expressed as follows:

$$Concavity(S) = Volume(conv(S)) - Volume(S) \quad (1)$$

where $conv(S)$ denotes the convex hull of S.

Note that this expression is invariant under rigid transformations, but not under scaling. In order to achieve invariance under scaling, the notion of concavity could have been defined as the volume of the convex hull divided by the volume of the patch; this is the approach followed in [LCWK07] to find bounding ellipsoids in a variational setting. In our hierarchical framework, however, the proportionality of the concavity measure with the size is to be considered a positive aspect, as small shape features are expected to be merged as the clustering proceeds, even if they are nearly convex. In a human body, for example, fingers may be convex, but we do not want them to remain up to the coarsest segmentation; on the contrary, we would like that, at some point in the hierarchy, they are merged together with the palm to form a whole hand that can be processed as a single entity (Fig. 6).

3.2. Hierarchical Tet Clustering

Hereafter the hierarchical tet clustering (HTC) algorithm is described in detail, whereas it is assumed that concepts such as simplicial complexes and geometric graphs are familiar to the reader.

Let T be the set of tetrahedra of a manifold tetrahedral mesh M with boundary. The dual graph $D = (N,A)$ of M is defined as follows: each node of N corresponds to a tetrahedron of T, and there is an arc (dual edge) in A connecting two nodes in N if the corresponding tetrahedra in M share a facet. Now, if one considers each node of such a dual graph to represent a cluster (initially made of a single tetrahedron), merging two tetrahedra into a single representative cluster corresponds to contracting a dual edge into a single node, that is, the two nodes of the arc are identified and the adjacency relations are updated accordingly in the data structure (Figure 3). In the HTC approach, a priority queue is created in which all the dual edges are sorted based on the cost of their contraction. At each step, the dual edge with lowest cost is popped from the queue, it is contracted, and all the edges incident to the new representative node are updated, that is, their cost is re-computed and their position in the queue is updated accordingly. The method produces a hierarchy which can be represented by a binary tree of clusters in which the root is the whole tetrahedrization, and the leaves are the individual tetrahedra.

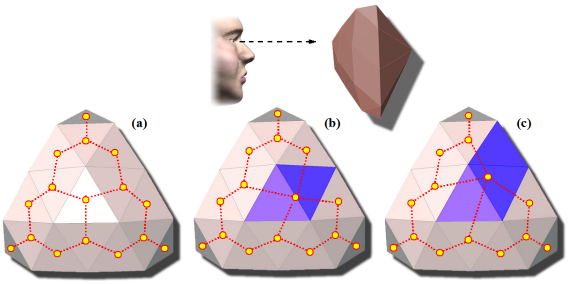

Figure 3: *A diamond-like shape modeled by a tetrahedral mesh. In (b) an arc of the dual graph was contracted, and the two tets corresponding to the arc end-points were marked as belonging to the same single cluster. In (c) another arc was contracted producing a resulting cluster made of three tets.*

The algorithm can be described by the pseudo-code in Listing 1, where we make use of an extended notion of concavity (Eqn. 1) referred to dual edges, and call $Concavity(e)$ the quantity $Concavity(S_1 \cup S_2)$, where S_1 and S_2 are the two clusters connected by the dual edge e. We remark that a cluster is the union of original tets, and not the approximating convex hull.

3.2.1. Tree Balancing

Clearly, $Concavity(S)$ is zero when S is perfectly convex, thus the use of such metric to sort the heap typically leads to several cases in which an arbitrary choice must be made between two dual edges of equal null cost. Instead of leaving the choice to the fate, we introduced an additional criterion that aims at keeping the resulting tree more balanced, with significant benefits both in terms of quality of the segmentation hierarchy (clusters are kept more compact) and in terms of computation time required to terminate. Specifically, in these ambiguous cases we prefer to merge clusters that are small and possibly of equal size (in terms of the number of tetrahedra that constitute them). Thus, we introduce a secondary cost term defined as follows:

$$C_2(e) = \frac{t_{1e}^2 + t_{2e}^2}{T^2} \tag{2}$$

where t_{1e} and t_{2e} are the number of tetrahedra constituting the clusters being merged through the contraction of e, while T is the total number of tetrahedra in the mesh. Since $0 \leq C_2 \leq 1$, we can combine the concavity with this secondary cost without overlap as follows:

$$Cost(e) = \begin{cases} Concavity(e) + 1, & \text{if } Concavity(e) > 0 \\ C_2(e), & \text{otherwise} \end{cases} \tag{3}$$

Note that the combined cost defined in Eqn. 3 does not impose any trade off between different requirements; the inclusion of C_2 is just a way of driving the selection of one of the possible orderings induced by the sole *Concavity*.

3.2.2. Complexity Analysis

After having run our HTC on more than 400 shapes, we have verified that the resulting binary trees are mostly balanced, hence we consider this as the average case and briefly analyze the algorithmic complexity of our clustering algorithm. For the sake of simplicity, we assume that the total number of tetrahedra is a power of 2, so that we can model the whole set of operations through a perfectly balanced binary tree.

Hereafter, we call t the total number of tetrahedra of the mesh being segmented, and e the number of dual edges. Also, we note that each tetrahedron can correspond to at most 4 dual edges, therefore the total number of dual edges e is bounded by $4t$ or, equivalently, $O(e) = O(t)$. To count the number of operations needed to construct it, we scan the hierarchy level by level from the leaves (level n in the tree) to the root (level 1). To construct level $n-1$ we need to compute the concavity of e potential clusters made of 2^1 tetrahedra each; For level $n-2$, we need to compute the concavity of $e/2$ potential clusters made of 2^2 tetrahedra each; and so on. In general, to compute level $n-i$ we need to compute $e/(2^{i-1})$ potential clusters made of 2^i tetrahedra each. The computation of the concavity is dominated by the computation of the convex hull which, for n points in 3D, costs $O(nlogn)$ operations [PH77]. Moreover, by construction, each cluster made of 2^i tetrahedra can have at most

Listing 1: *basic Hierarchical Tet Clustering algorithm*

```
HTC ( TetMesh T )
{
 1)     Create a dual vertex and a corresponding singleton cluster for each tetrahedron in T
 2)     Create a dual edge for each pair of tetrahedra sharing a facet
 3)     Associate a cost to each dual edge e /*Cost = Concavity(e)*/
 4)     Create a heap of all the dual edges sorted based on their associated cost
 5)     If the heap is empty terminate
 6)     pop the first edge from the heap (let it be e)
 7)     Contract e to a single dual vertex v and merge the corresponding clusters
 8)     Update the cost of all the edges incident to v and update their position in the heap
 9)     Go to 5
}
```

2^i+3 vertices, and computing the convex hull of 2^i+3 points costs $O(2^i log 2^i) = O(i2^i)$ operations. Hence, the total number of operations needed to construct the hierarchy is proportional to:

$$\sum_{i=1}^{n-1} \frac{e}{2^{i-1}} 2^i i = 2e \sum_{i=1}^{n-1} i = 2e \frac{n(n-1)}{2} \qquad (4)$$

By considering that e belongs to $O(t)$, and that $t = 2^n$, the above expression can be re-written as a function of the number of tetrahedra t as follows:

$$2e\frac{n(n-1)}{2} \in O(en^2) = O(2^n n^2) = O(t log^2 t) \qquad (5)$$

The above theoretical complexity reasonably matches the actual time required by our implementation to terminate on various examples (see Figure 10). With a similar analysis it can be proved that the complexity in the worst case (i.e. a completely unbalanced tree) becomes $O(t^3 logt)$.

3.2.3. Tree Pruning

Hierarchical clustering is a well-known technique for data mining, and is particularly useful when large sets of data have to be classified within an unspecified number of categories. Once the hierarchy is computed, it is often necessary to analyze it to produce a static, single-resolution partitioning of the dataset. One may choose to select a fixed number of clusters and stop the aggregation when such a number is reached, but this would be somehow against the philosophy of using a hierarchical setting, in which no preferred number of clusters exists; moreover, in these cases proper partitioners such as the k-means would probably perform a better job. Another solution is to select a maximum admissible cost and stop the algorithm when the next aggregation would exceed such a threshold error. This solution is widely used, but it relies on the assumption that the aggregation cost grows monotonically as the clustering proceeds; in this case, the resulting tree can be drawn within a coordinate frame so that the height of each node (i.e. its y coordinate) corresponds to the cost of aggregating its sub-clusters. Such a diagram, also called *dendrogram*, can be cut through a horizontal line at the desired height to produce a single-resolution partitioning (see Figure 5).

In our HTC algorithm the cost is not necessarily monotonic. Consider a 2D example in which an equilateral triangle t is composed of three sub-triangles t_1, t_2 and t_3 meeting at the barycentre of t (see Figure 4). In this case, the leaves of the binary tree represent the three sub-triangles t_i that, having a null approximation error (they are convex), would have $y = 0$. The first aggregation performed necessarily involves a pair $< t_i, t_j >$ which forms a non-convex shape (non-zero approximation error), thus the corresponding node would have $y > 0$. The root of the tree represents the whole shape that, being a triangle, has again a null approximation error.

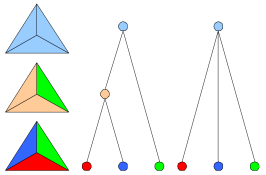

Figure 4: *Effect of the pruning on the tree created out of a simple shape.*

This behaviour reveals that the tree may contain spurious information which, instead of depending on the intrinsic structure of the data, is due to the greedy and binary nature of the algorithm used. This suggested us to look for methods to prune the binary trees so as to turn them to dendrograms. The method we have implemented starts from the root of the tree and recursively removes children having a bigger or equal cost, as described in the pseudo-code in Listing 2.

Notice that the output of this conversion algorithm is not necessarily a binary tree, but it better captures the underly-

ing structure of the set of tetrahedra being clustered. In the extreme case in which the initial shape is perfectly convex (and in this case only) the pruned tree may link all the leaves directly with the root.

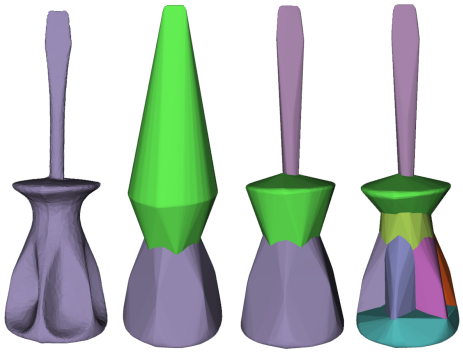

Figure 5: *Form left to right: the original "screwdriver" model (courtesy of AIM@SHAPE's Shape Repository) and three approximations computed by our algorithm after having cut the dendrogram.*

4. Interactive Region Selection and Applications

Clearly, our HTC generates a hierarchy of convex hulls that can be used to improve the performances of point location, proximity query and collision detection algorithms; the exploitation of bounding volume hierarchies in these contexts is described in [JTT01], while the particular benefits of using salient and convex decompositions are shown in [LWTH01, EL01]. Nonetheless, in our method parts are both salient and hierarchically structured, and this made us focus our experiments on the more promising scenario of shape editing. Thus, we describe how to exploit the hierarchy of convex polyhedra to select 3D features through an extremely simple and intuitive interaction. The cost that we plugged into the HTC algorithm, as well as the greedy and incremental nature of the algorithm itself, makes the resulting hierarchies contain virtually all the relevant convex features of the shape at all the scales. Thus, by browsing the hierarchy, it is likely that a user finds several "interesting" features.

Based on this premise, we have developed a novel interaction mechanism to select a convex part by simply clicking on a point of the shape's boundary, and by rotating the mouse wheel to select the size of the convex feature containing the point clicked. The computation of the hierarchy of convex polyhedra is performed once as a pre-processing step. Then, a mouse click on a point of the shape causes the selection of the tetrahedron containing the point clicked. Such a tetrahedron corresponds to a leaf of the cluster tree, and the rotation of the mouse wheel causes a motion upwards/downwards along the path connecting the selected leaf to the root of the tree. The nodes along this path correspond to clusters in the hierarchy; in particular, the path represents a sequence of clusters approximated by convex polyhedra of increasing size, with the property that each cluster properly contains its predecessor in the sequence.

In a human body model, for example, the user may click on a finger tip and rotate the mouse wheel to select among the finger tip itself, the whole finger, the hand, the arm and so on, up to the whole body (see Figure 6).

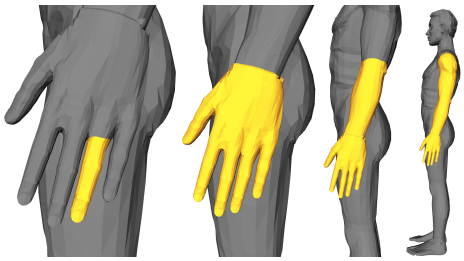

Figure 6: *Interactive selection of meaningful features on a human body. To browse these selections, the user must simply click on the finger tip and rotate the mouse wheel.*

Note that the boundary of a selection may be slightly jagged due to its alignment with the tets (Fig. 2 (b) and (c)). Normally, this can be considered as a sort of *aliasing* effect, and it can be ignored in applications such as deformation. However, if perfectly smooth boundaries are required, existing approaches may be used to "rectify" their curves on the surface [GFA04, LL02].

4.1. Deforming Shape Features

The deformation of free-form shapes, typically represented by surface meshes, is gaining more and more attention and several approaches to this problem have been proposed recently. In some cases, for example to animate virtual characters, skeleton-based skinning is particularly intuitive, and geometric tranformations of the skeleton's control points induce expected deformations of the shape. The most diffused and flexible paradigm, however, is based on the selection of a region of interest (ROI) and a handle attached to a point of the ROI [SA07]; by applying a rigid transformation to the handle, the ROI is deformed according to various criteria that depend on the specific method.

In the literature about mesh deformation, the problem of selecting a ROI typically relies on a manual drawing of bounding curves on the surface. While this can be easily done on 2D silhouettes, a 3D shape may be extremely complex and drawing the boundary of a ROI might become a rather time-consuming task.

We have experimented our selection engine as a tool for specifying ROIs for deformation of shape features. To do this, we have implemented the deformation method proposed in [SA07] and obtained particularly good results. Our modeling interaction is extremely simple: after clicking on

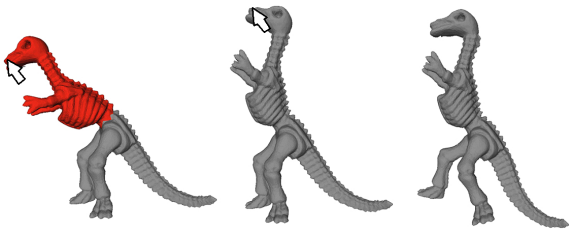

Figure 7: *Deformation of the Dinosaur model (courtesy of Cyberware). By clicking on the nose-tip and rotating the mouse wheel, we obtained the first ROI (left). Then, through a simple mouse drag we moved the nose-tip to its new position and obtained the first deformation (middle). Similarly, we performed few further interactions to deform the tail, the head and the legs (right). About 15 seconds were needed to perform all these operations.*

a surface point and rotating the mouse wheel to select a ROI, we just move the handle by dragging the mouse. In our framework the default position of the handle corresponds to the point clicked to select the ROI, but the user is also allowed to change its position by clicking on another point within the currently selected ROI. The mouse drag defines a 2D vector in screen coordinates that, when projected on the current view plane, is transformed to 3D coordinates in model space. The transformed vector defines the displacement of the handle (see Figure 7).

4.2. Copying and pasting 3D shapes

Exploiting the copy-and-paste mechanism to edit 3D objects is another example of challenging problem in which an effective region selection is fundamental.

Once again, we have experimented our selection engine as the basis for copying relevant shape parts to be used for subsequent modelling, and we have found that the benefit of using our method is twofold. First, convex features often match the intuitive notion of semantic feature for natural shapes, which means that in most cases there is no need to further edit the selection. Second, both the whole shape and its convex sub-parts are true solid models (and not surface patches), and pasting a copied part onto a "destination" shape turns out to be particularly easy, as it involves a simple union of solids and does not require complicated surface intersection tests, hole filling or blending operations.

In our prototype system the user can select a part and create a copy as a new connected component of the tetrahedrization; after an interactive placement of the new component to the desired position, the system computes the resulting simplicial complex by simply splitting intersecting simplices and removing possible duplications.

This copy and paste mechanism is useful in various application contexts, ranging from model completion (Figure 8, left) to shape design by feature composition (Figure 8, right).

5. Results and Discussion

Besides the examples presented in this paper, we have tested our algorithm on more than 400 models obtained by tetrahedrizing the surface meshes used in [VtH07] through a publicly available software called *tetgen* [SG05], and we have verified that our method is robust to both noise and non-uniform sampling density. Note that, since small features are aggregated first (it is a bottom-up approach), our algorithm is also *stable* in the sense that local modifications induce only local changes in the resulting hierarchy. We have observed that *tetgen* is very fast and inserts few additional vertices on the boundary. It is also possible to let *tetgen* insert internal vertices to build well-shaped tetrahedra; even in this case the HTC algorithm works well, and our experiments have shown that the features captured by the resulting hierarchies do not vary significantly. Clearly, if internal points are added the resulting hierarchy changes accordingly and might also include completely internal clusters; nonetheless, such additional information does not represent a problem and relevant convex features are still captured (Fig. 9). Thus, we have concluded that it is unnecessary to introduce internal vertices that would only slow down the computation. When tetrahedrizing a surface mesh made of n triangles, *tetgen* produces $1.78n$ tets (average value on the aformentioned 400 shape dataset). During the experiments with our prototype, we have verified that the strongest aspect of the HTC algorithm is the speed; timing results are summarized in Figure 10, where it is assumed that the input shape is "natively" represented as a tet mesh. If the input is given as a polygon mesh, a rough 20 % time increase must be considered to include *tetgen*'s conversion. Note that, normally, the polygon mesh to be converted must be closed to define an inner volume to be tetrahedrized. However, if surface holes are small and due to missing data, they may be patched prior to conversion through state-of-the-art hole filling algorithms. Instead, if the polygon mesh is not meant to define a volume (e.g. the surface of a car door) our algorithm is not appropriate.

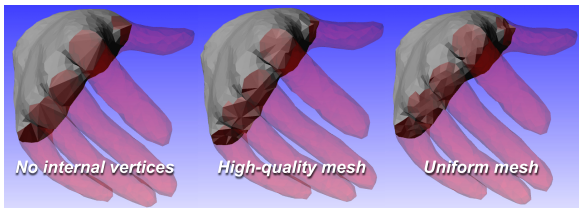

Figure 9: *The same convex features are captured independently of the kind of volume tetrahedrization.*

Algorithms which are close in spirit to the HTC presented here are descirbed in [KT03, ZL05, LA07]. Altough these methods work on surface meshes, their strategies are all

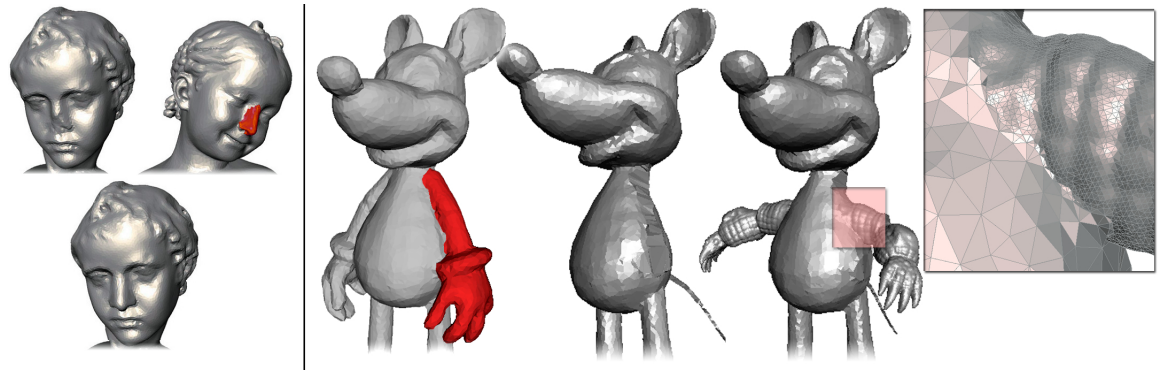

Figure 8: *Left - The original Eros model has a damaged nose; through our selection mechanism we have copied the nose from the Bimba model and pasted it onto Eros (both models are courtesy of AIM@SHAPE's Shape Repository). Right - The arms of the "Mickey" model have been selected and removed. After having selected the arms from another model, they have been placed in their final position and the system created the resulting connected tetrahedral mesh. Mickey model is courtesy of 3D Cafe.*

Listing 2: *Pseudo-code for turning a cost-labeled binary tree into a dendrogram. The outer "for" scans all the children of r, including the ccs moved from subtrees.*

```
procedure ToDendrogram (node r) {
  for each node c in children of r {
    /* This cycles over newly inserted children too */
    if ( c is not a leaf AND cost(c) ≥ cost(r) ) {
      for each node cc in children of c {
        add cc to children of r
        remove cc from children of c
      }
      remove c from children of r
    }
  }
  for each node c in children of r
    ToDendrogram (c)
}
```

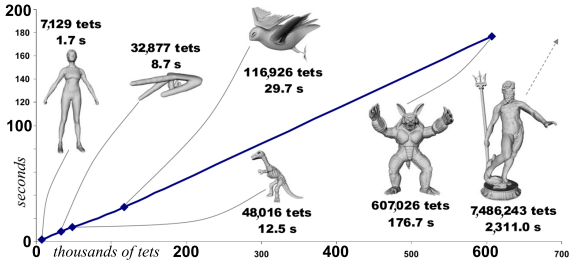

Figure 10: *Processing seconds for models of various tet count. Tests were run on an Intel Core 2 PC at 1.87 GHz with 2 Gb RAM and running Microsoft's Windows Vista.*

meant to identify visually-relevant parts, hence they are worth a comparison with our method. We started with [KT03] and verified that whereas this method produces good hierarchical results, it needs to compute the geodesic distance between all the pairs of vertices, and this introduces a quadratic term in the complexity. For big meshes, an approximate solution may be obtained through a prior mesh simplification, but this is rather complicated (it requires a *projection* of coarse faces onto the original mesh, which in turn requires a voxelization) and small convex features might be missed due to the simplification itself and to a non easy tuning of some parameters. With a comparable quality of the results (Fig. 12), our approach is substantially faster, easier to implement and it does not require any user-defined parameter. Then we compared our method with [LA07] and [ZL05] and we verified that our approach is more flexible by giving hierarchical results, and more practical by not requiring any parameter setting; also in these cases the resulting decompositions have a comparable quality (Fig. 12). Comparisons are summarized in Figure 11 in terms of properties which are desirable to perform region selection in an interactive environment (i.e. hierarchical results, no need to set parameters, speed). As a further general advantage, by working on tetrahedral meshes our approach can properly handle solid models with cavities that would be misinterpreted by surface-based methods.

Note that the HTC is a *greedy* algorithm and, as such, it is expected to produce sub-optimal results when compared with variational approaches; actually, such a comparison is not really fair, as the two approaches have different targets (hierarchical vs. globally optimal decomposition) which are mutually exclusive. An interesting discussion of this aspect was presented in [AFS06]. We have also compared our selection mechanism with the one proposed in [FKS*04]: in the best situation (i.e. selection of clearly visible regions bounded by

Method	Hierarchy	Parameters	Complexity ; Speed*	Notes
KT03	Yes	**3** (thresholds for termination)	$n^2 \log n$; 145.30 s	* Seconds needed for a 40k faces polyhedron.
LA07	No	**1** (concavity tolerance)	$n^2 \log n$; 38.02 s	** Results of our method on a tetrahedrized version of a 40k faces model. Timing not including the 2.81 seconds needed to perform the conversion.
ZL05	No	**1** (salience threshold)	$n \log n$; 12.76 s	
ours	Yes	None	$n \log^2 n$; 12.61 s**	

Legenda: *Desirable property* *Not really attractive* *Not desirable*

Figure 11: *Comparison with state-of-the-art algorithms in terms of properties which are desirable to perform region selection in an interactive environment (i.e. hierarchical results, no need to set parameters, speed).*

few curves) the two methods require a comparable amount of interaction, whereas in worse cases (see Fig. 13) our approach proved to be much more efficient.

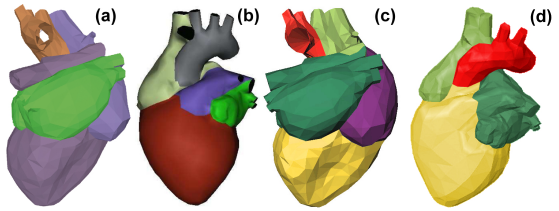

Figure 12: *Regions identified on a heart model by [KT03] (a), [ZL05] (b) and our method (c-d). In (c-d) surface holes were filled before converting from triangle to tet mesh.*

Being fully automatic, our algorithm can be run transparently by a solid modeler just after having loaded the tet mesh. On today's PCs, a model made of 22k tets can be processed in less than 5 seconds, which is acceptable and makes it possible to exploit intuitive editing paradigms such as those described in section 4. Note that in our prototype system the hierarchy is computed only once when the model is loaded. After a deformation, one may choose either to recompute it or to proceed with the same hierarchy of the original model, and there are good reasons for both the choices; re-computing the hierarchy maintains a one-to-one correspondence between the shape and its hierarchical decomposition, whereas maintaining the original hierarchy may preserve some "semantics" that was captured in the original pose (e.g. if an arm was originally bent and the user stretches it, a new hierarchy might "miss" the forearm).

5.1. Limitations

Currently, major weaknesses are mostly related to the editing paradigm proposed. Our deformation mechanism allows only to translate the handle, while in a complete modeling system rotations should be possible as well; also, the placement of pieces to be pasted must be performed manually by the user through interactive manipulators. As shown in [KJS07], however, in some cases it is possible to look for *compatible* parts to be exchanged and, for the sake of completeness, such a modeling metaphor should be imported into our setting.

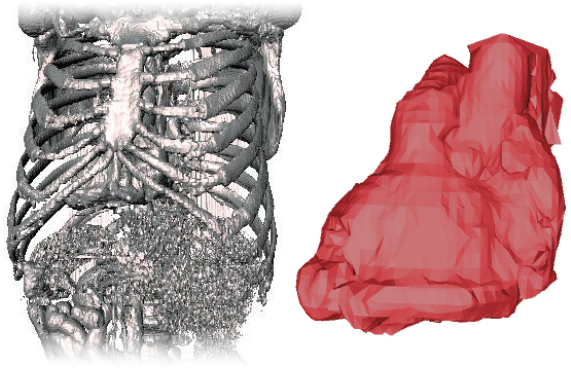

Figure 13: *The heart is connected with multiple vessels and tissues, and is tightly occluded by the chest, thus selecting it as described in [FKS*04] would be rather unpractical. Through our mechanism the selection required just a mouse click and a wheel rotation.*

6. Conclusions and future research

We have discussed the advantages of representing 3D shapes through hierarchical convex approximations, and have proposed an algorithm to convert an existing tetrahedral mesh to its hierarchical representation. Furthermore, we have observed that such a hierarchy contains virtually all the convex features of the shape at all the scales, and we have exploited this fact to design an innovative and intuitive interaction mechanism to browse and select these features for further processing.

Thanks to recent meshing algorithms, the long-standing problem of boundary-constrained tetrahedrization has been solved satisfactorily. In this paper we used tet meshes to propose an elegant and effective definition of part concavity and to ease the composition of parts, but we believe that this representation is well-suited for many other shape analysis and editing purposes, and we plan to investigate these possibilities in our future research. Further work will be mostly focused on the modeling paradigm, and solutions will be searched to face the weaknesses described here. Furthermore, applications of the HTC will be investigated in contexts such as shape retrieval and classification [DGG03].

Acknowledgements

This work is partly supported by the AIM@SHAPE EU NoE (IST Contract N. 506766), by the FOCUS-K3D EU CA (Contract N. ICT-2007.4.2) and by the FIRB - SHALOM project (Contract N. RBIN04HWR8). Thanks are due to all the members of the Shape Modeling Group of the CNR-IMATI-GE for their helpful discussions and suggestions.

References

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. *ACM Transactions on Graphics 24* (2005), 617–625.

[AFS06] ATTENE M., FALCIDIENO B., SPAGNUOLO M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (2006), 181–193.

[APP*07] AGATHOS A., PRATIKAKIS I., PERANTONIS S., SAPIDIS N., AZARIADIS P.: 3d mesh segmentation methodologies for cad applications. *Computer-Aided Design and Applications 4*, 6 (2007), 827–841.

[ARSF07] ATTENE M., ROBBIANO F., SPAGNUOLO M., FALCIDIENO B.: Part-based annotation of virtual 3d shapes. In *CW '07. International Conference on Cyberworlds, 2007.* (2007), pp. 427–436.

[CDST95] CHAZELLE B., DOBKIN D.-P., SHOURABOURA N., TAL A.: Strategies for polyhedral surface decomposition: An experimental study. In *Symp. Computat. Geometry* (1995), pp. 297–305.

[CS07] COHEN E., SINGH M.: Geometric determinants of shape segmentation: Tests using segment identification. *Vision Research 47* (2007), 2825–2840.

[DGG03] DEY T. K., GIESEN J., GOSWAMI S.: Shape segmentation and matching with flow discretization. In *WADS 03, LNCS 2748* (2003), pp. 25–36.

[EL01] EHMANN S., LIN M. C.: Fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum 20*, 3 (2001), 500–510.

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Transactions on Graphics 23*, 3 (2004), 652–663.

[GFA04] GUILLAUME L., FLORENT D., ATILLA B.: Curvature tensor based triangle mesh segmentation with boundary rectification. In *Computer Graphics International* (2004), pp. 10–17.

[GWH01] GARLAND M., WILLMOTT A., HECKBERT P.: Hierarchical face clustering on polygonal surfaces. In *ACM Symp. Interactive 3D graphics* (2001), pp. 49–58.

[HR84] HOFFMAN D., RICHARDS W.: Parts of recognition. *Cognition 18* (1984), 65–96.

[JTT01] JIMENEZ P., THOMAS F., TORRAS C.: 3d collision detection : a survey. *Computers and Graphics 25*, 2 (2001), 269–285.

[KJS07] KRAEVOY V., JULIUS D., SHEFFER A.: Shuffler: Modeling with interchangeable parts. *The Visual Computer (to appear)* (2007).

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics 22*, 3 (2003), 954–961.

[KT05] KATZ S., TAL A.: Segmentation using feature point and core extraction. *The Visual Computer 21*, 8-10 (2005), 865–875.

[LA06] LIEN J.-M., AMATO N.: Approximate convex decomposition of polygons. *Comput. Geom. Theory Appl. 35*, 1 (2006), 100–123.

[LA07] LIEN J.-M., AMATO N.: Approximate convex decomposition of polyhedra. In *Symposium on Solid and Physical Modeling* (2007), pp. 121–131.

[LCWK07] LU L., CHOI Y.-K., WANG W., KIM M.-S.: Variational 3d segmentation for bounding volume computation. *Computer Graphics Forum 26*, 3 (2007), 329–338.

[LL02] LEE Y., LEE S.: Geometric snakes for triangular meshes. *Comp. Graphics Forum 21*, 3 (2002), 229–238.

[LWTH01] LI X., WOON T., TAN T., HUANG Z.: Decomposing polygon meshes for interactive applications. In *ACM Symp. Interac. 3D Graphics* (2001), pp. 35–42.

[PH77] PREPARATA F., HONG S.: Convex hulls of finite sets of points in two and three dimensions. *CACM 22* (1977), 87–93.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modelling. In *Eurographics Symposium on Geometry Processing* (2007), pp. 109–116.

[SG05] SI H., GAERTNER K.: Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable* (2005), pp. 147–163.

[Sha06] SHAMIR A.: Segmentation and shape extraction of 3d boundary meshes. In *Eurographics 2006 State of the Art Reports* (2006), pp. 137–149.

[VtH07] VELTKAMP R. C., TER HAAR F. B.: *SHREC2007 3D Shape Retrieval Contest*. Tech. Rep. UU-CS-2007-015, Utrecht University, 2007.

[ZL05] ZHANG H., LIU R.: Mesh segmentation via recursive and visually salient spectral cuts. In *Proceedings of Vision, Modeling and Visualization* (2005), pp. 429–436.

[ZSH00] ZOECKLER M., STALLING D., HEGE H.: Fast and intuitive generation of geometric shape transitions. *The Visual Computer 16*, 5 (2000), 241–253.