Approximate Convex Decomposition of 3D Digital Object Surface Using Scalar Triple Product of Vectors

Somrita Saha [1], **Arindam Biswas**[2]
Department of Information Technology
Indian Institute of Engineering Science and Technology
Shibpur, India
[1]somrita.besu@gmail.com [2]barindam@gmail.com

Abstract

The work presents a method of approximate convex decomposition based on a convexity measure which is derived from the volume of the tetrahedron generated by the two edge-adjacent face triangles. Each convex surface is grown by considering the adjacent face triangles in a breadth first manner. Subsequently, tiny and insignificant convex regions are merged with the adjacent bigger region by which they are surrounded. The convex regions generated by the proposed algorithm reveal characteristic features of the shapes of different objects.

Introduction

Convex decomposition of digital objects is an active field of research since the last four decades. It aims at decomposing a 3D digital object into simpler units. An object decomposed into convex regions is more suitable for improved shape analysis, collision detection, skeleton extraction, pattern recognition, origami folding analysis and many such fields. This work presents a new approach for convex decomposition where concavity between two adjacent face triangles is measured by the volume of the tetrahedron generated by those two faces. Exact convex decomposition generates an unmanageable number of convex regions. On the contrary, an approximate decomposition renders meaningful and lesser number of regions which can be better exploited for further studies related to the shape of the object.

The journey began in late 70s by B. Chazelle and D. Dobkin with the attempt of decomposition of polygons into its convex parts, which deals decomposition in 2-dimension [1]. In 1981, Chazelle extended the work for polyhedra, where the algorithm is based on cutting the polyhedra along the notches or the edges which exhibit a reflex angle [2]. S. Svensson et al. presented an algorithm for nearly convex decomposition based on distance transform, in 2002 [9]. It identifies the significant voxels of the objects and group them into seeds, which are used to originate the parts of the object by applying the reverse and the constrained distance transformations. In 2004, J. M. Lien and N. M. Amato proposed a method of convex decomposition which proceeds by removing (resolving) the non-convex features in order of importance [5]. They presented another algorithm in 2007, which identifies the most concave feature(s) in each iteration and then partitions the polyhedron so that the concavity of the identified features is minimized [6]. A method of convex decomposition which computes a hierarchical segmentation of the mesh triangles by applying a set of topological decimation operations to its dual graph was proposed by K. Mamou et al. in 2009 [7]. Here, The decimation technique is based on a cost function evaluating the concavity and the shape of the detected clusters. In 2013, M. Ghosh et al. proposed a new algorithm based on a strategy for evaluating potential cuts that aims to reduce the relative concavity, rather than absolute concavity [4]. M. Muller et al.'s volumetric convex decomposition method is based on Voronoi diagram of the object, where nodes are placed on the mesh and, given the nodes, Voronoi decomposition of the bounding box of the mesh is computed, resulting in a set of convex shapes [8]. In 2018, Convex decomposition of static objects was extended to animated meshes by D. Thul et al., where an animated object is decomposed into temporally coherent approximately convex parts [10]. İ. Demir et al. [3] presented an algorithm for convex decomposition, which partitions the input triangle set into some disjoint triangle clusters, called search subspaces. They use heuristics to compute these search subspaces that contain partially similar parts of the model, if any. Finally, the segmentation is done keeping some criteria in view, such as concavity measure, surface angles, the balance between size and number of components.

Preliminaries

Definition 1 *Convex Decomposition: It is the process by which a large and complicated digital object is decomposed into a number of convex subsets, union of which results in the original object.*

Exact Convex Decomposition (ECD) generates a huge and unmanageable number of convex subsets of the object, which is not suitable for further utilisation. Instead of ECD, some approximation can be exploited to generate a reasonable number of convex regions which are also visually meaningful.

Definition 2 *Surface Decomposition: It is the process by which the surface of a 3D digital object is decomposed into a number of convex or concave subsets, union of which results in the surface of the original object.*

Our idea of convex decomposition is primarily based on a convexity measure guided by the volume of the tetrahedron generated by each two edge-adjacent face triangles of the input object. This volume can be computed with the scalar triple product or the mixed product of the vectors defining the tetrahedron. If the surface generated by the two faces is concave, then the tetrahedron has a positive volume. On the contrary, the volume of the tetrahedron will be negative if the corresponding faces of that surface are convex with respect to each other. If the faces are coplanar, then the volume will be zero. Fig. 1 illustrates the cases of convex and concave surfaces. The volume, v, of the tetrahedron is the scalar triple products of the vectors $\vec{a}, \vec{b}$, and $\vec{c}$. This products is mathematically denoted as

$$v = [\vec{a}, \vec{b}, \vec{c}] = \vec{a} \cdot (\vec{b} \times \vec{c})$$

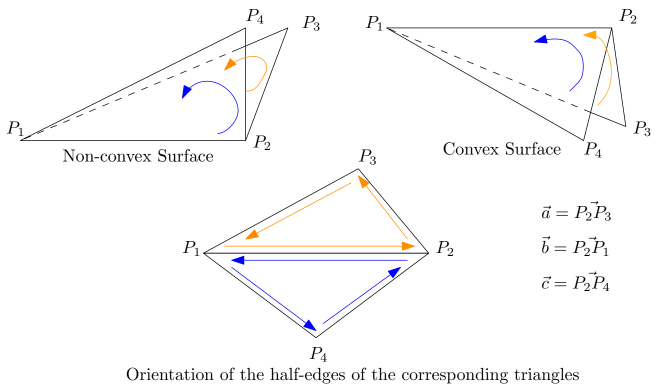


Figure 1: Scalar Triple Product

Specific Volume: Specific volume or volume per area, v_s, is given by the ratio of the volume, v, of an object and its projected area, a. Here, in this work, we will consider the volume, v, as the volume of the tetrahedron subtended by the two edge adjacent triangles $\triangle P_1P_2P_3$ and $\triangle P_1P_2P_4$, and the area, a, as the sum of there areas a_1 and a_2, respectively.

Definition 3 *ρ-convexity: Two edge adjacent face triangles are ρ-convex with respect to each other, if the specific volume, v_s, of the tetrahedron subtended by them is less than a predefined threshold ρ.*

In Fig. 1, if the area of the edge adjacent triangles $\triangle P_1P_2P_3$ and $\triangle P_1P_2P_4$ are a_1 and a_2, respectively, then the condition on which they will be convex to each other is

$$\frac{v}{(a_1 + a_2)} \leq \rho \tag{1}$$

where ρ is a predefined threshold on the volume of the tetrahedron given by the corresponding triangles.

The data structure used is Doubly Connected Edge List (DCEL). Apart from the basic structure of DCEL, some additional information are stored as a part of the DCEL for this work. For the object vertex, whether it is a part of any convex region's boundary, number of region boundaries it is part of are stored. For the edge object, additional information on destination vertex, twin half edge, new previous half edge, new next half edge, whether the edge is dropped, and the convex region it is a part of, are maintained. Finally, for the face objects, additional information on its norm, color code of the convex region it belongs to, area, and the base face corresponding to the convex region it is part of, are stored.

Convex Decomposition Procedure

The convex decomposition procedure considers water-tight objects as inputs. The surface of 3D digital objects is made of face triangles. Based on the changing curvature of the surface of the objects, the face triangles may have different norms. Our objective is to decompose the object surface into its convex subparts. Now, to discern the continuous convexity of the surface or a sudden concave region in the course of the investigation procedure, a face is checked with the three neighboring faces.

The procedure is based on breadth first search of the face list of the object, which is obtained from the .obj input data file. The face list is investigated starting from the very first face in the list. As we have considered watertight models only, each face will have exactly three edge-adjacent faces. The first face in the face list is considered as the base face of the first convex region to be generated. Each edge-adjacent face is taken into consideration while investigating the convexity of the surface.

The scalar triple product or the mixed product is calculated based on the corresponding vectors, as given in Fig. 1. This product is the volume of the tetrahedron generated by the two edge-adjacent faces. Theoretically, if this volume is negative, then the corresponding surface is convex. However, in our procedure, as we intend to generate approximate convex decomposition, the convexity checking criteria has been relaxed by a threshold, ρ. So, instead of checking whether the volume, v, is less than zero, it should suffice to check whether the specific volume, v_s is less than this threshold ρ, as given in equation 1. If the specific volume, v_s, is less than ρ, then the edge-adjacent face is considered to be convex with respect to the face it is being checked with and the face is enqueued for further investigation of convexity.

When all the three edge-adjacent faces are checked for convexity, a face is dequeued from the queue and its edge-adjacent faces are considered for investigation with respect to the dequeued face. This approach continues till the last face in the queue. Once the queue is exhausted, we get the first convex region of the object. Then the next unvisited face from the face list is taken into consideration and it is assigned as the base face of the next convex region to be generated. This process continues till the last face of the face list is investigated. As per the goal of the approximate decomposition, very small convex regions, having an area less than a predefined value, are included in the adjacent bigger convex region.

This algorithm can further be utilised for improved shape analysis. It can be used to set up a deep learning model of a set of synthetic surfaces. This model can be used to detect shapes which already exist in the training set of the model.

Algorithm 1: 3D-SURFACE-CONVEX-DECOMPOSITION

```
Input: Triangulated object in the form of DCEL($(D)$), $\rho$
Output: Modified DCEL ($D$) for the decomposed convex
        surface of the object
1  $L \leftarrow \emptyset, S \leftarrow \emptyset, i \leftarrow 0$
2  Initialise color of all faces to $\emptyset$
3  $b \leftarrow$ first face from the face list of $D$
4  $S \leftarrow b$        /* add $b$ to the set $S$ */
5  $b.color \leftarrow c[i]$, $t_0 \leftarrow b$
6  while ($facelist \neq \emptyset$) do
7      foreach edge-adjacent-face $t$ of $t_0$ do
8          if ($t.color = \emptyset$) then
9              $v \leftarrow 1/6 \times$
               $[t_0.\vec{edge}, [t_0.\vec{edge}].next, [[t_0.edge].\vec{twin}].previous]$
10             $v_s \leftarrow \frac{v}{t.area + t_0.area}$
11             if ($v_s \leq \rho$ ) then
12                 add $t$ to $S$
13                 $L[i] \leftarrow$ boundary edge of $S$
14                 $t.color \leftarrow c[i], t.base \leftarrow b$
15                 ENQUEUE($Q$, $t$)
16         else if ($t.color \neq \emptyset$) then
17             if ($t.color = c[i]$) then
18                 add $t$ to $S$
19                 $L[i] \leftarrow$ boundary edge of $S$
20     if ($Q = \emptyset$) then
21         $CR_i \leftarrow S, CR_i.id \leftarrow i, CR_i.color \leftarrow$
           $i, CR_i.base \leftarrow b, CR_i.feob \leftarrow L_i$
22         $b \leftarrow$ next unvisited face from face list, $t_0 \leftarrow b$
23         $i \leftarrow i + 1, S \leftarrow \emptyset$
24         $S \leftarrow b$, $b.color \leftarrow c[i]$
25     $t_0 \leftarrow$ DEQUEUE($Q$)
26 foreach $CR$ do
27     if $CR.area < \epsilon$ then
28         APPROXIMATE($CR.feob, CR.id$)
```

Procedure APPROXIMATE($feob, cr$)

```
1  $start \leftarrow feob, next \leftarrow 0, tw \leftarrow$
   $start.twin, face0 \leftarrow tw.face, flag \leftarrow 1$
2  while $temp \neq feob$ and
   $[[start.twin].face].color =$
   $[[[start.next].twin].face]color$ do
3      $temp \leftarrow start.next, start \leftarrow temp$
4  if $start \neq feob$ then
5      $CR.outerface \leftarrow$
       $[[[start.next].twin].face].CR$
6  else
7      $CR.outerface \leftarrow [[start.twin].face].CR$
8  $CR.color \leftarrow [CR.outerface].color$
9  $CR.base \leftarrow [CR.outerface].base$
10 merge $CR$ with $CR.outerface$
```

Algorithm 3D-SURFACE-CONVEX-DECOMPOSITION

The proposed algorithm 3D-SURFACE-CONVEX-DECOMPOSITION takes triangulated digital objects and a threshold value on the volume of the tetrahedron, ρ, as inputs and produces decomposed convex surface as output. It is based on Breadth First Search (BFS) traversal of the face list in the object file. Necessary initialisation operations are performed in Steps 1-5. The linked list L, to maintain one boundary edge of the set of triangles and the set S to contain the convex triangles with respect to the base b, are initialised to $\emptyset$ (Step 1). Color of all the faces are also initialised to $\emptyset$ (Step 2). The first face from the face list of the input object file is assigned to be the base face of the first convex region to be grown and is added to S(Step 3-4). In Step 5, a color is assigned to the base face b from the color array c and t_0 is assigned with b. Then, in the **while** loop (Steps 6-25), all the faces in the face list are considered for the checking of convexity. First, each of the three edge-adjacent faces of t_0 are checked for convexity or merging (Steps (7-18)). If the faces do not have any color yet (Step 8), then they are checked for convexity (Step 11) as per the equation 1 given in the section Preliminaries. The incident edge on t_0, the next edge of it, and the previous of the twin of the edge are the vectors, scalar triple product of which gives the volume of the tetrahedron subtended by t and t_0 (Step 9). The specific volume, v_s, is calculated by dividing v with the sum of areas of t and t_0 in Step 10. If t satisfies the condition in Step 11, i.e., v_s less than or equal to ρ, a predefined threshold on v_s, then the following operations are performed in Steps (12-15). Triangle t is added to the set, s, of convex triangles for this region as of now. A boundary edge of S is assigned to $L[i]$, the i^{th} node of of the linked list L. t is assigned with the current color $c[i]$ and the base face of t is marked as b. Also, t is enqueued for further investigation. In Steps (16-19), if the face triangle t has the same color as of t_0, then it is added to set S and Li is updated with one of the current boundary edges of S.

While triangle t is added to the set S of convex triangles pertaining to the current convex surface region, adjacent edges of t and S are dropped or marked obsolete and the next and previous edges of the remaining edges are reassigned by the modification of the half-edge structure, as given in Fig. 2.

Next, if the queue, Q, is exhausted (Step 20), then the first convex surface region of the object is produced. An instance of the DCEL object convex-region, CR, is constructed for this iteration. Region id, color, base face, and first edge on the boundary ($feob$, an anchor edge to traverse the boundary of the region) are initialised for CR with respect to the current iteration (Step 21). In Steps 22-24, preparation for the next region to be grown is done. The next unvisited face from the face list is identified and set as the base face, b, for the next region and t_0 is initialised to b. Iteration variable i is incremented and S is set to $\emptyset$. S is initialised to b. Color of b is assigned as $c[i]$.

If Q is not empty, then it is dequeued to get the next face

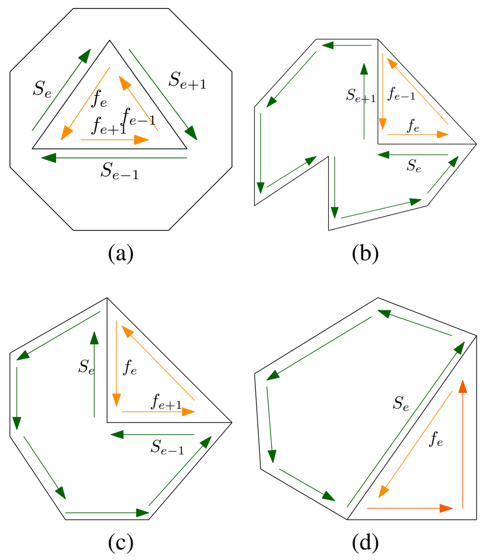


Figure 2: Adding a triangle, f, with a set, S, of triangles of the regions ((a) Three common pairs of half-edges between S and f, (b) - (c) Two common pairs of half-edges between S and f, (d) One common pair of half-edges between S and f)

t_0 for the further investigation (Step 25). The **while** loop, (Steps (6-25)), is repeated till the last face in the face list.

Once the last face of the face list is checked, the algorithm yields i number of convex surfaces. Each of the convex region is checked for its boundary regions and merged with one of the boundary regions if its area is less than a predefined threshold ϵ (Step (26-28)). This is performed through a called procedure APPROXIMATE. First edge on the boundary of the convex surface region ($feob$) and the convex region id are passed as parameters to the procedure APPROXIMATE.

Procedure APPROXIMATE performs the merging of small regions with one of its boundary regions which has an area bigger than that. Area of a triangle can be computed from the vertices of each triangle obtained from the object file. And, the area of a convex region can be computed from the sum of the areas of the triangles which are part of that region. Now, the procedure starts with some assignment operations in Step 1. First edge on the boundary ($feob$) is assigned as the starting edge, $start$, for the traversal of the boundary of the regions cr. The edge $feob$ serves as an anchor edge to traverse the boundary of cr. The next edge of $start$, $next$, is initialised to 0. The twin edge of $start$ is assigned as tw and the face incident on the edge tw is assigned as $face0$. A flag, $flag$, which indicates that the region cr is completely enclosed by some other region, is set to 1. ϵ is assigned a certain constant value to denote the threshold on the area of the convex region cr. Starting from $feob$, the boundary of the region cr is traversed by accessing the next boundary edges, one after another, in the **while** loop of Steps 2-3. The loop executes till the next edge on the boundary is $feob$ and the each two consecutive faces are of same color, i.e., in other words, all the boundary faces in course of traversal are having same color. After the completion of the while loop, if the next edge, $start$, is not $feob$, then we can say that the region cr is not completely surrounded by some other region, i.e., cr has more than one regions in its boundary (Step 4). In that case, the outer region is set as the region corresponding to $face1$, which has a different color (Step 5). Otherwise, if the next edge, $start$, is not $feob$, then the region cr is completely enclosed by a single outer region. Hence, the outer region of cr is set as the region corresponding to $face0$ (Step 6-7). Once the outer region is identified, the inner region, cr's color is assigned as the color of the outer region of it and the base face of the inner region is assigned as the base face of the outer region. Finally it is merged with the outer region by some modification (the shared edges on the boundary of the regions are marked as obsolete) of the DCEL data structure (Steps 8-10).

Complexity

Time Complexity

The algorithm 1, 3D-SURFACE-DECOMPOSITION, checks the convexity of all the faces in the face list. Each face triangle of the original input object is enqueued a maximum of three times as the triangle is edge-adjacent to only three other triangles. In the loop Steps (7-25), the main operations are enqueue, dequeue, convexity checking, and some assignment operations, which take constant amount of time. There are some merging operations which are also linear with respect to time, as they involve manipulation of edges. Procedure APPROXIMATE is called in Steps (28-29) for convex regions with a very small area (area less than ϵ, a threshold on area). The number of convex regions will be much less than the number of faces in the original input file. Moreover, the regions with a very small area will be even more smaller in number. Also, this number will depend on the type of object we are working on. So, this number of iterations will be output sensitive. In the procedure APPROXIMATE, all are assignment operations, except a **while** loop, which iterates for the number of edges on the boundary of the convex region. As these are small regions, the boundaries will be short, too. Hence, the overall time complexity of the algorithm is bounded by $O(n)$, n being the number of faces in the face list of the original input object.

Space Complexity

The algorithm works on an input object represented by the DCEL data structure, which has a space complexity of $O(n)$. The DCEL data structure is augmented by adding a new structure for the convex regions and some extra attributes to the already existing structures of face, half-edge, and vertex, as par the necessity of the proposed algorithm. This modification does not increase its size. A separate linked list L is maintained, each node of which stores the first edge ($feob$, it works as an anchor edge for traversing the boundary) on the boundary of a convex region. Linked list L has a size less than $O(n)$. Hence, the overall space complexity of the algorithm 3D-SURFACE-CONVEX-DECOMPOSITION is $O(n)$.

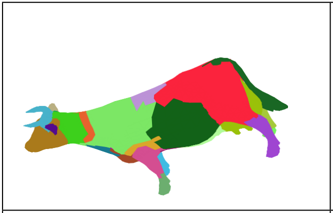
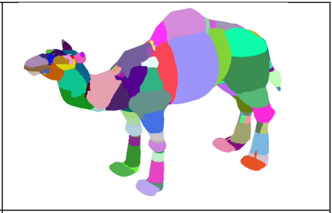
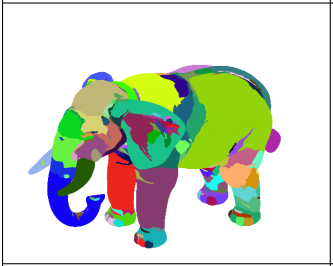
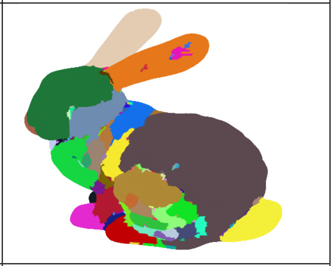
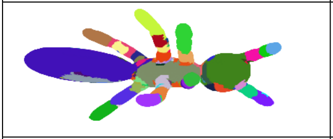
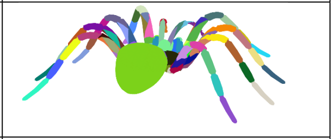
Results

The algorithm 3D-SURFACE-CONVEX-DECOMPOSITION has been implemented using the programming language C and the output files are plotted in the language Python to render the final images. The algorithm is applied on 8 3D input objects having triangulated surface. The table 1 contains the output results obtained for ant, camel, elephant, bunny, cow, fox, spider, and a foot bone. The threshold for the volume of the tetrahedron, ρ, is chosen based on the type of the input objects. For each object, the value of ρ and the number of convex regions, $\#CR$, generated for that value of ρ, are given. The value of ρ depends on the type of the object and the value of $\#CR$ will be determined by the value of ρ.

Now, as we look into the output images, in table 1, for the cow and the camel, all the convex regions of the objects have distinct colors. For some regions, for example, the portion below the knee joint of the camel's hind limbs, the elephant's ear, and the bunny's hind limb, a cluster of small convex regions can be seen. This is due to the very fast changing curvature of the surface in those regions. Similar visual experience happens for some portions of the ant, too. For example, one or more patches of different colors can be found in the abdominal portion of the ant due to sudden change of the curvature of that portion.

Table 2 contains the output images of the object bunny for four different ρ values which are $0.01, 0.04, 0.6$, and 0.08. From the table, it is evident that the number of convex regions, $\#CR$, decreases as the value of ρ increases. This is obvious because with increasing value of ρ, the convexity checking condition becomes more relaxed. So, many faces cluster under one convex surface though their curvatures vary widely. Here, the value of ρ should be chosen wisely so that the output image resemble the original input at its best. For our case, $\rho = 0.06$ results in the best output where each convex region can be identified separately. The two ears, the torso, the tail and the limbs have different colors. Whereas, for $\rho = 0.01$, the object is over-decomposed, i.e., it generates a huge number of convex surfaces. On the contrary, a value $\rho = 0.08$ does not decompose the object enough, so the torso and the tail fall in same convex surface, which is not visually convincing. Hence, the bottom-left figure of the table 2 is the more acceptable decomposition out of these four figures.

Table 1: Convex Decomposition of some objects

	
$\rho = 0.0125, \#CR = 23$	$\rho = 0.00005, \#CR = 126$
	
$\rho = 0.012, \#CR = 381$	$\rho = 0.06, \#CR = 306$
	
$\rho = 0.031, \#CR = 99$	$\rho = 0.00015, \#CR = 119$

Conclusion

This work is primarily focused on the convex decomposition of the surface of 3D digital objects. The proposed algorithm can distinguish between different regions of an input with triangulated surface. This decomposition is done based on a vector mathematics, the scalar triple product of the relevant edge vectors of the two adjacent faces. So, the criterion for checking the convexity is very simple and fast. The outputs can be exploited for shape recognition of different objects. For example, we can take a set of synthetic surfaces and triangulate them to set up a deep learning model. The synthetic models can be used as the training set. Then, given an input object, this trained set can be exploited to determine the curved surfaces of the input. The training of the synthetic models may be time consuming but, once the training is done, it can be utilised for easy detection of different types of shapes.

References

[1] Chazelle, B.; and Dobkin, D. 1979. Decomposing a polygon into its convex parts. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, 38–48.

[2] Chazelle, B. M. 1981. Convex decompositions of polyhedra. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, 70–79.

[3] Demir, İ.; Aliaga, D. G.; and Benes, B. 2018. Near-convex decomposition and layering for efficient 3D printing. *Additive manufacturing*, 21: 383–394.

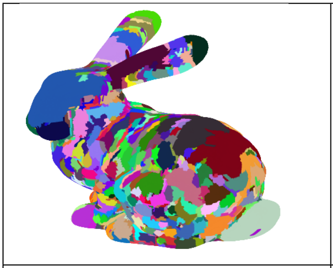
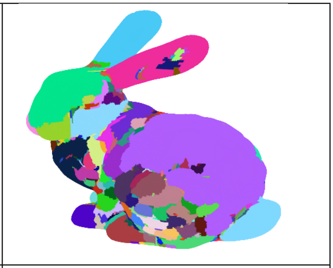
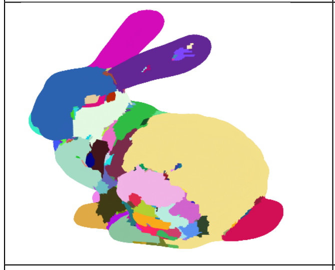
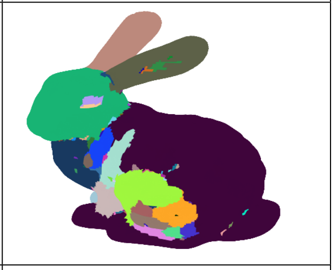
[4] Ghosh, M.; Amato, N. M.; Lu, Y.; and Lien, J.-M. 2013. Fast approximate convex decomposition using relative concavity. *Computer-Aided Design*, 45(2): 494–504.

[5] Lien, J.-M.; and Amato, N. M. 2004. Approximate convex decomposition. In *Proceedings of the twentieth annual symposium on Computational geometry*, 457–458.

[6] Lien, J.-M.; and Amato, N. M. 2007. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, 121–131.

[7] Mamou, K.; and Ghorbel, F. 2009. A simple and efficient approach for 3D mesh approximate convex de-

Table 2: Convex Decomposition of some objects

	
$\rho = 0.01, \#CR = 1069$	$\rho = 0.04, \#CR = 583$
	
$\rho = 0.06, \#CR = 306$	$\rho = 0.08, \#CR = 220$

composition. In *2009 16th IEEE international conference on image processing (ICIP)*, 3501–3504. IEEE.

[8] Müller, M.; Chentanez, N.; and Kim, T.-Y. 2013. Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Transactions on Graphics (TOG)*, 32(4): 1–10.

[9] Svensson, S.; and Di Baja, G. S. 2002. Using distance transforms to decompose 3D discrete objects. *Image and Vision Computing*, 20(8): 529–540.

[10] Thul, D.; Ladickỳ, L.; Jeong, S.; and Pollefeys, M. 2018. Approximate convex decomposition and transfer for animated meshes. *ACM Transactions on Graphics (TOG)*, 37(6): 1–10.