

# Convex Shape Decomposition

Hairong Liu, Wenyu Liu

Huazhong University of Science and Technology, P.R. China

lhrbss@gmail.com, liuwuy@hust.edu.cn

Login Jan Latecki

Temple University, USA

latecki@temple.edu

## Abstract

*In this paper, we propose a new shape decomposition method, called convex shape decomposition. We formalize the convex decomposition problem as an integer linear programming problem, and obtain approximate optimal solution by minimizing the total cost of decomposition under some concavity constraints, using linear programming method. Our method is based on Morse theory and combines information from multiple Morse functions. The obtained decomposition provides a compact representation, both geometrical and topological, of original object. Our experiments show that such representation is very useful in many applications.*

## 1. Introduction

Just like image segmentation in image processing, shape decomposition has been considered as a fundamental problem in shape related areas, such as computer vision, computer graphics and scientific data visualization. After decomposition, many operators, which cannot be or too complicated to be applied on original objects, can be applied on decomposed parts easily and efficiently. Thus, shape decomposition is very useful in shape analysis, shape matching, topology extraction, collision detection and other geometric processing methods employing divide-and-conquer strategies. The aim of this paper is to propose a new shape decomposition method, called *convex shape decomposition*, denoted by *CSD*. This method can decompose an object into approximately convex parts at the minimal cost. Its advantages include:

- It is suitable for arbitrary objects. In theory, this method can decompose objects of arbitrary dimension, although for very high dimension objects, the complexity is high. For 2D and 3D objects, such decomposition is fast.
- The decomposition is achieved by minimizing the total cost of decomposition under some concavity constraints, which guarantees to be globally optimal in many situations.

- It can naturally deal with objects with holes.

First, we give a mathematical definition of the decomposition.

**Definition 1.** For an object  $O$  of dimension  $n$ , a cut is a nonempty connected component of the intersection of a  $(n - 1)$ -dimensional hyperplane with  $O$ .

We observe that a cut satisfies two criteria: 1) it lies entirely within the object  $O$ ; 2) its border points are also the border points of object  $O$ . Obviously, to decompose an object into  $q$  parts, we need at least  $q - 1$  cuts which do not intersect with each other.

A cut  $C$  can be assigned a positive real number, denoted by  $\text{Cost}(C)$ , as its cost. In the same way, a part  $P$  can be assigned a positive real number which represents the degree of its concavity, denoted by  $\text{Concavity}(P)$ .

Formally, the decomposition can be described as follows: for an object  $O$ , divide it into  $q$  parts using  $m$  cuts such that the total cost of  $m$  cuts is minimal and the concavity measure of every decomposed part is no more than a threshold  $\varepsilon$ .

$$\min \sum_{i=1}^m \text{Cost}(C_i) \quad (1)$$

where  $O = \bigcup_{i=1}^q P_i$ ,  $P_i \cap P_j = \emptyset$  when  $i \neq j$  and  $\text{Concavity}(P_i) \leq \varepsilon$  for all  $i$ .

Such decomposition is very useful, at least in two applications:

- **Shape Representation.** After decomposition, since every decomposed part is approximately convex, it can be approximately represented by its convex hull; thus, a compact representation of original object is obtained. Such representation captures not only all the important topological information, but also all the important geometric information of original object. Obviously, such representation facilitates many applications, such as collision detection, motion planning and inner distance computation.
- **Topology Extraction.** After decomposition, if we regard each part as a node and two nodes have an edge if and only if they are adjacent, a graph, called **convex graph**, is obtained. Compared with Reeb Graph

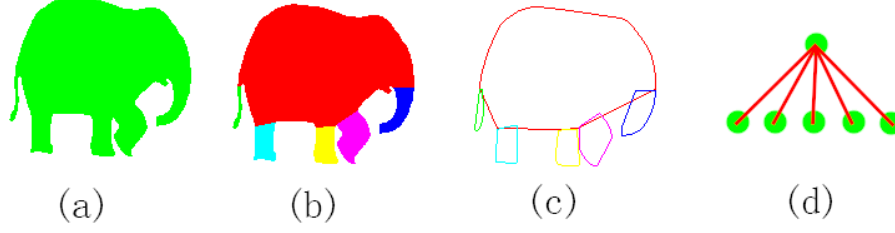


Figure 1. Illustration of shape decomposition and representation.

[3], convex graph has many advantages: it is unique and captures all important topological information of the object. Convex graph is a topological descriptor; based on it, we can also extract the approximate skeleton of the object.

In Fig.1, (b) is a convex decomposition of (a); (c) is an approximation of (a), with every decomposed part is substituted by its convex hull, it captures nearly all important information of (a); (d) is the convex graph of (a), which represents the topological relation of parts.

We now present our main idea.

If an object is convex, then it contains all line segments between each pair of its inner points. This fact indicates us to consider the relation between inner points of an object.

**Definition 2.** For two points,  $p_1$  and  $p_2$ , in the object  $O$ , if  $O$  does not contain the line segment between  $p_1$  and  $p_2$ ,  $p_1$  and  $p_2$  is called a **mutex pair**, denoted by  $p_1 \approx p_2$ .

According to Def.2, in Fig.2,  $p_1 \approx p_2$  and  $p_1 \approx p_4$ . Obviously, after convex decomposition,  $p_1$  and  $p_2$  cannot be in the same part. Thus, mutex pairs are in fact constraints of convex decomposition. However, convex decomposition usually results in representations with an unmanageable number of parts, thus we seek to decompose an object into approximately convex parts, which usually provide similar benefits as convex parts. For approximately convex decomposition,  $p_1 \approx p_2$  and  $p_1 \approx p_4$  are obviously different:  $p_1$  and  $p_2$  are more likely to belong to different parts than  $p_1$  and  $p_4$ . For a mutex pair formed by  $a$  and  $b$ , a weight is assigned to it to measure this, denoted by  $\text{Concavity}(a, b)$ . Note that if the object  $O$  contains the line segment between  $a$  and  $b$ ,  $a$  and  $b$  can be considered to form a mutex pair with weight 0. In Fig.2,  $\text{Concavity}(p_1, p_2) > \text{Concavity}(p_1, p_4) > \text{Concavity}(p_1, p_3) = 0$ .

For an object  $O$ , its  $\varepsilon$ -mutex set, denoted by  $M_\varepsilon(O)$ , is the set of all mutex pairs whose weight is not smaller than  $\varepsilon$ . By ignoring mutex pairs with small weights,  $M_\varepsilon(O)$  provides the constraints for approximately convex decomposition. In the same way, for an object  $O$ , all possible cuts form a set, called **candidate cut set**, denoted by  $C(O)$ . Since each candidate cut in  $C(O)$  can satisfy some mutex pairs in  $M_\varepsilon(O)$ , to decompose an object, we just need to select some cuts from  $C(O)$  to satisfy all mutex pairs in  $M_\varepsilon(O)$ .

Supposing there are  $n$  candidate cuts in candidate cut set,  $C(O) = \{\text{cut}_1 \dots \text{cut}_n\}$ , and  $m$  mutex pairs in  $\varepsilon$ -mutex set,  $M_\varepsilon(O) = \{\text{mp}_1 \dots \text{mp}_m\}$ . The index set of final cuts is  $I$ , which means that if  $i \in I$ , then  $\text{cut}_i$  is a final cut. Let us assign a binary variable  $x_i$  to  $\text{cut}_i$  where  $x_i = \begin{cases} 1 & i \in I \\ 0 & i \notin I \end{cases}$ .

For every candidate cut in  $C(O)$ , say,  $\text{cut}_i$ , the mutex pairs it satisfies form a subset of  $M_\varepsilon(O)$ , denoted by  $S_i$ . In this way, we obtain  $n$  subsets of  $M_\varepsilon(O)$ . For a mutex pair in  $M_\varepsilon(O)$ , say,  $\text{mp}_i$ , in all cuts that can satisfy it, at least one cut index must be in set  $I$ ; thus, it implies a constraint:

$$\sum_{j=1}^n a_{ij} x_j \geq 1 \quad a_{ij} = \begin{cases} 0 & \text{mp}_i \notin S_j \\ 1 & \text{mp}_i \in S_j \end{cases} \quad (2)$$

Since there are  $m$  mutex pairs, we have  $m$  constraints.

Let us denote  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ ,  $\mathbf{c} = (\text{cost}(\text{cut}_1), \text{cost}(\text{cut}_2), \dots, \text{cost}(\text{cut}_n))^T$ ,  $\mathbf{A} = \{a_{ij} | i = 1 \dots m, j = 1 \dots n\}$ ,  $\mathbf{1} = (1, 1, \dots, 1)^T$ , the formula (3) translates problem (1) into a integer linear programming problem [16]:

$$\min \mathbf{c}^T \mathbf{x} \quad \mathbf{A} \mathbf{x} \geq \mathbf{1} \quad x_i \in \{0, 1\} \quad (3)$$

It is NP-complete; however, we can relax our constraint on  $x_i$  and let  $0 \leq x_i \leq 1$ . By such relaxation, the problem is now a linear programming problem [16], which is expressed as follows:

$$\min \mathbf{c}^T \mathbf{x} \quad \mathbf{A} \mathbf{x} \geq \mathbf{1} \quad 0 \leq x_i \leq 1 \quad (4)$$

By solving (4), we can get an approximate optimal solution of (3).  $x_i$  can be regarded as the possibility of choosing  $\text{cut}_i$ . After (4) is solved, we can iteratively select the cut which has the highest possibility until all the mutex pairs in  $M_\varepsilon(O)$  are satisfied. For an object  $O$ ,  $M_\varepsilon(O)$  and  $C(O)$  are very large, thus unpractical to deal with. However, many mutex pairs in  $M_\varepsilon(O)$  are redundant, in other words, when some mutex pairs are satisfied, other mutex pairs are automatically satisfied. In Section 3.3, we will show how to eliminate these redundant mutex pairs and obtain a very small set,  $M_\varepsilon(O)$ . As for  $C(O)$ , if our aim is to find rough position of the cuts, we just need a small subset of it which

ensures to satisfy all the mutex pairs in  $M_\varepsilon(O)$ . Based on a rough cut, we can adjust it according to the local structure of the object to get precise cuts. Thus, in our method, we in fact deal with small  $M_\varepsilon(O)$  and  $C(O)$ . In Section 3, a systematic method to build  $M_\varepsilon(O)$  and  $C(O)$  based on Morse theory is introduced. Of course, we can also add cuts found by other methods to  $C(O)$ . The related literature

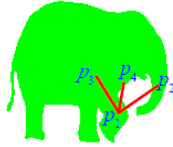


Figure 2. The relations of inner points of an object

is reviewed in Section 2. In Section 4, some experimental results are demonstrated.

## 2. Literature Review

Shape segmentation is an important step toward shape analysis and understanding [1, 17]. A variety of applications in computer vision and computer graphics could benefit from preprocessing the shape using an efficient and reliable decomposition method, such as shape simplification [2], collision detection [10], and skeleton extraction [7].

Most of shape decomposition methods fall into two large categories. One category is to decompose an object into meaningful parts; the other category is to decompose an object into certain kind of geometric primitives.

The accepted notion of meaningful part relies on human perception and thus has no accurate definition. However, there are some perception rules from cognitive science which is the basis for meaningful decomposition. As pointed out by Hoffman [6], the human visual system perceives region boundaries at negative minima of principal curvature, or concave creases—this observation is known as the *minima rule*. The *minima rule* is an elegant theory that defines a framework for how human perception might decompose an object into its constituent parts. However, *minima rule* just define boundary points at which to parse, but does not tell how to use these points to cut shapes, and therefore does not tell what the parts are. *Short cut rule*, proposed by Singh, Seyranian and Hoffman [18], is its complement. It states that, other things being equal, human vision prefers to use the shortest possible cuts to decompose shapes into parts. Most of decomposition methods are based on these two rules. Wu et al. [19] proposed to decompose 3D meshes using a simulated electrical charge distribution, which in fact decomposes along the deep surface concavities. Latecki and Lakaemper [8] proposed a method to decompose 2D shapes using discrete contour evolution. In their paper, they used convexity rule, which is very similar to *minima rule*.

For the second category, there are many geometric primitives. Erickson and Harpeled [4] proposed a method to cut a surface into disks. Mortara et al. tried to decompose a 3D object into blowing bubbles [14]. At the same time, they proposed another method which decomposes a 3D object into tubular parts [15]. The most popular primitive is convex part. This is because many algorithms perform more efficiently on convex objects than on non-convex objects. Thus, convex decomposition is widely used in many areas, such as collision detection [10] and motion planning [9]. However, for complex objects, convex decomposition is time-consuming and usually results in an unmanageable number of parts. Thus, some researchers try to decompose an object into approximate convex parts. Lien and Amato proposed methods to decompose polygons and polyhedrons into approximately convex parts [11, 12]. They reported that approximate convex decomposition usually results in significantly smaller number of parts and can be computed more efficiently. For some applications, the ability to consider only important features may not only be more efficient, but may lead to improved results. Thus, approximate convex decomposition is very important, especially in computer vision, since we need to ignore some small details. However, little research has been done in this field.

The aim of our method is to decompose an object into approximate convex parts by minimal cost. Thus, it falls into the second category. But, it can decompose many objects into meaningful parts. This is because convexity plays an important role in human perception [17]. Nearly all the methods analyzed above, whether decomposing them into convex parts or meaningful parts, are specially designed to some representation methods, such as polygon, polyhedral and point set. However, our method is suitable for all these representation methods. At the same time, our method forms the convex decomposition problem as an integer linear programming problem. To our best knowledge, this is the first such formalization, and it is very important, since it can guarantee our solution is optimal in many situations. For example, it usually results in smallest number of cuts. Another important advantage of our method is that it is not limited by dimension of objects. In theory, it can decompose objects of arbitrary dimension. For high dimensional data analysis and visualization, our method is based on Reeb graph [5] and can be considered to be an extension of Reeb graph. Thus, it may be very useful for high dimensional data.

## 3. Computing Mutex Pairs and Candidate Cuts by Morse Theory

In the procedure of translating the decomposition problem (1) into liner programming problem (4), the central problem is how to build  $M_\varepsilon(O)$  and  $C(O)$  systematically

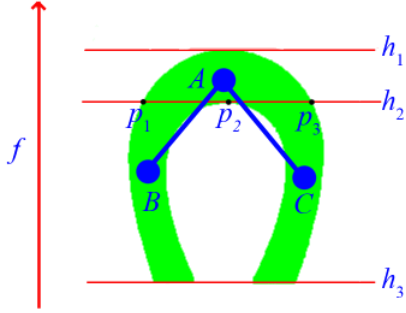


Figure 3. The Reeb Graph of a horseshoe

and efficiently. Our method is inspired by Reeb graph, which has a solid mathematical foundation in Morse theory. Simply speaking, for a manifold, it has two basic operations. First, constructs a Morse function  $f : M \rightarrow R$ , where  $R$  is the set of real numbers, which can be considered to be a projection from higher dimension to one dimensional manifold; second, contract the connected components of the level sets  $f^{-1}$  to points.

Fig.3 illustrates an object and its Reeb graph (blue nodes and edges). The Morse function  $f$  is constructed as follows: for every point  $p$  in this object,  $f(p)$  is the height of point  $p$ , thus called **height function**. The Reeb graph is determined by the changes in the number of connected components of  $f^{-1}$ . In Fig.3, the Reeb graph has three nodes and each corresponds to one part of the object, which reflects partial topological information of the object.

Note that a Reeb graph provides both part of  $M_\varepsilon(O)$  and part of  $C(O)$ . In Fig.3, every point in part  $B$  and every point in part  $C$  form a mutex pair; at the same time, line segment  $p_1p_2$  and line segment  $p_2p_3$  are two candidate cuts, both can satisfy the mutex pairs formed by the points in part  $B$  and part  $C$ . To obtain  $M_\varepsilon(O)$  and  $C(O)$ , a natural idea is to utilize multiple Reeb graphs. Alg. 1 illustrates the overall framework of our algorithm.

### 3.1. Morse functions

For an object  $O$  with dimension  $n$ , in order to detect the concave parts as effective as possible, we choose to evenly, or approximately evenly, sample half of the unit  $n - 1$  dimensional sphere to obtain  $t$  directions,  $\{d_i | i = 1 \dots t\}$ , with each  $d_i$  is an  $n$ -dimensional unit vector,  $d_i = (d_{ij} | j = 1 \dots n)$ . Corresponding Morse functions are then constructed as follows:

$$f_i(p_j) = \langle p_j, d_i \rangle \quad j = 1, \dots, N \quad (5)$$

$\langle \cdot \rangle$  is inner product, which is in fact a projection on the direction vector  $d_i$ . Obviously, every Morse function is in essence a projection from  $n$  dimension to 1D, in this procedure, information is inevitably lost. However, as the

---

### Algorithm 1: Convex shape decomposition

---

1. Compute multiple Morse functions;
  2. **for each Morse function do**
    - 1) Construct Reeb Graph;
    - 2) Compute Mutex pairs and add them into Mutex Set;
    - 3) Compute candidate cuts and add them into Candidate Cut Set;
  - end**
  3. **for each cut in Candidate Cut Set do**
    - for and each mutex pair in Mutex Set do**
      - Check whether the cut satisfy the mutex pair
  - end**
  - end**
  4. Solve the linear programming problem (4);
  5. Obtain final cuts;
- 

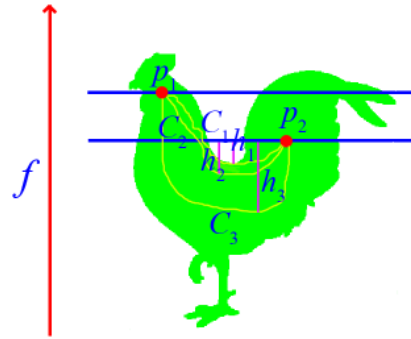


Figure 4. The concavity measure of two points

number of Morse functions increases, the lost of information quickly decreases.

### 3.2. Concavity Measure

Concavity measure is fundamental in approximate convex decomposition method. We now give a precise definition of concavity measure  $\varepsilon$  in the viewpoint of Morse functions. For the simplicity, we first consider the situations that the object  $O$  has no holes. For objects with holes, there is just a little difference, see Section 3.5.

**Definition 3.** For two points  $p_1$  and  $p_2$  in the object  $O$ , a curve connecting  $p_1$  and  $p_2$  within  $O$  is called a **path** between  $p_1$  and  $p_2$ . All the paths connecting  $p_1$  and  $p_2$  form a set, denoted by  $C(p_1, p_2)$ .

Obviously, the number of paths in  $C(p_1, p_2)$  may be infinite. For each path, a concavity measure depending on  $f$  is assigned: for a point  $p$  on path  $C$  connecting  $p_1$  and  $p_2$ , in the Morse function  $f$ , suppose  $f(p_1) \geq f(p_2)$ , we define:

$$g_f(p) = \begin{cases} f(p) - f(p_1) & f(p) \geq f(p_1) \\ 0 & f(p_2) < f(p) < f(p_1) \\ f(p_2) - f(p) & f(p) \leq f(p_2) \end{cases} \quad (6)$$

$$g_f(C) = \max_{p \in C} g_f(p) \quad (7)$$

In Fig.4,  $C_1$ ,  $C_2$  and  $C_3$  are all paths between  $p_1$  and  $p_2$ ,  $g(C_1) = h_1$ ,  $g(C_2) = h_2$  and  $g(C_3) = h_3$ .

**Definition 4.** The concave measure of a path  $C$  connecting  $p_1$  and  $p_2$ , denoted by  $\text{Concavity}(C)$ , is the maximum of  $g_f(C)$  among all  $f$ .

$$\text{Concavity}(C) = \max_f g_f(C) \quad (8)$$

By considering all paths connecting  $p_1$  and  $p_2$ , we can define the concavity measure for points  $p_1$  and  $p_2$ .

**Definition 5.** The concave measure of points  $p_1$  and  $p_2$ , denoted by  $\text{Concavity}(p_1, p_2)$ , is the minimum of the concavity measure of all paths connecting points  $p_1$  and  $p_2$ .

$$\text{Concavity}(p_1, p_2) = \min_{C \in C(p_1, p_2)} \text{Concavity}(C) \quad (9)$$

Based on the concave measure of two points, we can define the concave measure of a part  $P$ .

**Definition 6.** The concavity measure of a part  $P$ , denoted by  $\text{Concavity}(P)$ , is the maximum of the concavity measures of arbitrary two points in  $P$ . That is:

$$\text{Concavity}(P) = \max_{p_1 \in P, p_2 \in P} \text{Concavity}(p_1, p_2) \quad (10)$$

From Reeb graph, it is easy to obtain the paths with minimal concavity measure, which make it possible to efficiently decompose objects.

### 3.3. Mutex Pairs and Candidate Cuts

For an object  $O$ ,  $M_\varepsilon(O)$  contains the mutex pairs whose weight is not smaller than  $\varepsilon$ . Obviously,  $M_\varepsilon(O)$  is too large. To reduce the complexity, instead of dealing with the mutex pairs formed by points, we consider the mutex pairs formed by two point sets, which leads to much smaller  $M_\varepsilon(O)$ .

**Definition 7.** For two point sets  $A$  and  $B$  without intersection, if there is a mutex pair formed by a point from  $A$  and a point from  $B$ , set  $A$  and  $B$  is called a mutex pair, and is denoted by  $A \approx B$ .

Note that a point can be regarded as a set with one point, thus, mutex pair of point sets in fact includes mutex pair of points. Especially, two points are considered to be a mutex pair with weight 0 even if there is a line segment between them within the object. This means for point set  $A$  and  $B$ , if  $A \cap B = \emptyset$ , then  $A \approx B$ . Just like the concavity measure of points, we can define the concavity measure of point sets. For a point set, it has two concave measures.

**Definition 8.** For two point set,  $A$  and  $B$ , if  $A \approx B$ , the maximal (minimal) concavity measure of  $A$  and  $B$ , denoted by  $M(A, B)$  ( $m(A, B)$ ), is the maximum (minimum) of the concavity measure of mutex pairs of points from  $A$  and points from  $B$ .

$$M(A, B) = \max_{p_1 \in A, p_2 \in B} \text{Concavity}(p_1, p_2) \quad (11)$$

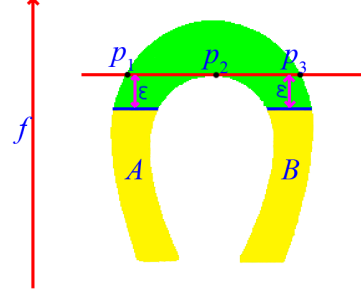


Figure 5. Mutex pair of regions and Candidate cuts that can separate it

$$m(A, B) = \min_{p_1 \in A, p_2 \in B} \text{Concavity}(p_1, p_2) \quad (12)$$

These two measures are very useful. If  $M(A, B) < \varepsilon$ , this mutex pair can be ignored; if  $m(A, B) \geq \varepsilon$ , every pair of points from  $A$  and  $B$  forms a mutex pair whose weight is not smaller than  $\varepsilon$ .

Now our aim is to find point sets  $A$  and  $B$  with  $m(A, B) \geq \varepsilon$ , and also find candidate cuts that can separate them. Note that regions are all point sets. In our implementation, we in fact use the mutex pair of regions.

Given the Reeb graph constructed from  $f$ , such task is very easy. Since Reeb graph is determined by the changes in the number of connected components of  $f^{-1}$ , two connected components (regions) determined by function value range  $[f_1, f_2]$ , in other words, two connected components formed by points belonging to  $f^{-1}([f_1, f_2])$ , is a mutex pair. At the same time, the cuts between adjacent nodes of Reeb graph can separate these mutex pairs, thus are all candidate cuts.

In Fig.5, two sets (regions) in yellow,  $A$  and  $B$ , form a mutex pair,  $A \approx B$ . At the same time, two candidate cuts,  $p_1 p_2$  and  $p_2 p_3$ , both can separate  $A$  and  $B$ . Obviously,  $m(A, B) = \varepsilon$ .

By combing all the mutex pairs and candidate cuts found from different Reeb graphs,  $M_\varepsilon(O)$  and  $C(O)$  are built.  $M_\varepsilon(O)$  contains mutex pairs of regions instead of mutex pairs of points, which greatly reduces its size. Since the candidate cuts guarantee to satisfy the mutex pairs constructed from the same Reeb graph,  $C(O)$  guarantees to satisfy  $M_\varepsilon(O)$ , which means that linear programming problem (4) is always solvable.

Obviously, from Reeb graphs, we just can obtain some basic cuts. By selecting some of them, we can decompose the object, but the positions of these cuts are usually not precise, since we only consider Morse functions in a small number of directions. By increasing the number of Morse functions, more precise cuts can be obtained, but it also increases the complexity of the method. The results can be improved in two stages: first, when building  $C(O)$ , more cuts found by other methods can be added; second, when

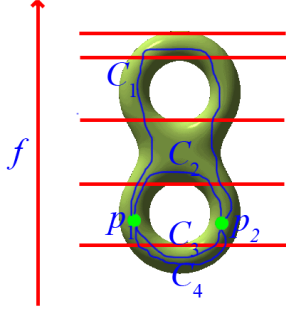


Figure 6. Paths between two points for an object with holes

final cuts are chosen by (4), we can adjust their positions according to the local structure of the shape. However, since these methods usually need some heuristic rules depending on the shapes and no heuristic rule is suitable for all shapes. In our experiments, we don't use these methods and just demonstrate the rough results of decomposition.

### 3.4. Cost of cuts

Since our aim is to minimize the total cost, we prefer cuts with small cost. Thus, cost is the criteria to select cuts. No matter how cost is computed, our method guarantees that the obtained parts are all approximate convex. In our experimental results in Section 4, we only use the short cut rule, which means the cost of a cut is its norm. For a 2D cut, its cost is its length and for a 3D cut, its cost is the area of the cut.

### 3.5. Object with holes

For two points  $p_1$  and  $p_2$  in a object  $O$  with no holes, if a cut segments a path in  $C(p_1, p_2)$ , then it segments all paths in  $C(p_1, p_2)$ . Thus, to satisfy the mutex pair formed by  $p_1$  and  $p_2$ , we just need to select one cut; however, for objects with holes, we usually need to select more than one cut.

Fig.6 illustrates four paths between  $p_1$  and  $p_2$ ,  $C_1$ ,  $C_2$ ,  $C_3$  and  $C_4$ . Obviously, a cut segmenting  $C_3$  cannot guarantee to segment  $C_1$ ; however, a cut segmenting  $C_3$  can guarantee to segment  $C_4$ .

For two paths  $C_1$  and  $C_2$  in  $C(p_1, p_2)$ , if a cut segmenting  $C_1$  can guarantee to segment  $C_2$  and vice versa,  $C_1$  and  $C_2$  is regarded as equivalent, denoted by  $C_1 \sim C_2$ . Such relation forms an equivalence relation on  $C(p_1, p_2)$ . The quotient set of  $C(p_1, p_2)$  by  $\sim$  is  $C(p_1, p_2)/\sim$ . Each element in  $C(p_1, p_2)/\sim$  contributes one constraints for (4), thus, the number of constraints contributed by mutex pair  $p_1 \approx p_2$  is the norm of quotient set  $C(p_1, p_2)/\sim$ . We can interpret it as point  $p_1$  and  $p_2$  form multiple mutex pairs from different paths.

## 4. Experiments

Our method has two parameters,  $t$  and  $\varepsilon$ .  $t$  is the number of Morse functions,  $\varepsilon$  is the threshold of concavity measure.

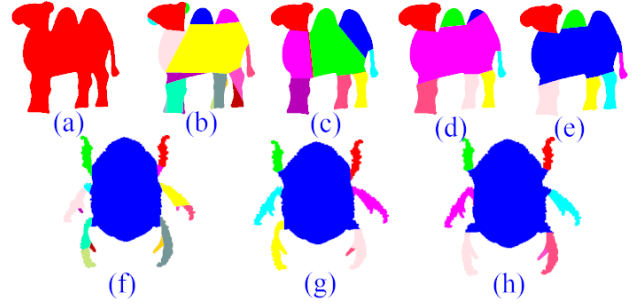


Figure 7. The decomposed results of a camel and a beetle at different concavity threshold

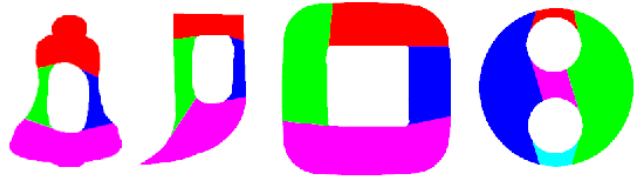


Figure 8. The decomposed results of objects with holes

The size of two central sets,  $M_\varepsilon(O)$  and  $C(O)$ , are both propositional to  $t$ . The most time-consuming part is constructing multiple Morse functions, which is  $O(tn \log(n))$ ,  $n$  is the number of basic elements of an object. For example, for point sets,  $n$  is the number of points; for meshes,  $n$  is the number of triangles.

In our experiments,  $t$  is small, for 2D objects,  $t = 16$ , for 3D objects,  $t = 33$ ; increasing  $t$  can improve the precision of the cuts, but also increases the complexity.  $\varepsilon$  depends on the concave regions we want to ignore, the smaller  $\varepsilon$  is, the larger the number of decomposed parts is.

In Fig.7, the first row demonstrates the decomposed results on a camel shape at  $\varepsilon = 0.03R$ ,  $\varepsilon = 0.06R$ ,  $\varepsilon = 0.1R$ ,  $\varepsilon = 0.14R$ , respectively; the second row demonstrates the decomposed results a beetle shape at  $\varepsilon = 0.03R$ ,  $\varepsilon = 0.06R$ ,  $\varepsilon = 0.12R$ , respectively. Obviously, the results of decomposition greatly depend on  $\varepsilon$ . As  $\varepsilon$  increases, the number of parts decreases; at the same time, the precision of some cuts decreases. This is because  $\varepsilon$  is in fact a concavity tolerance of the decomposed parts, large  $\varepsilon$  means large error.

Fig.8 demonstrates the decomposed results on objects with holes. There is no limit on the number of holes. To partition two parts, more holes usually mean more cuts.

Although the goal of our method is to decompose an object into approximately convex parts at the minimal cost, our method can also be used to decompose most of objects into perceptual parts. This is because convexity plays a very important role in human perception. Fig.9 offers an explicit comparison with the method proposed by Xiaofeng and DeCarlo [13], which is specially designed to decompose 2D object into meaningful parts. Row A is the decomposed

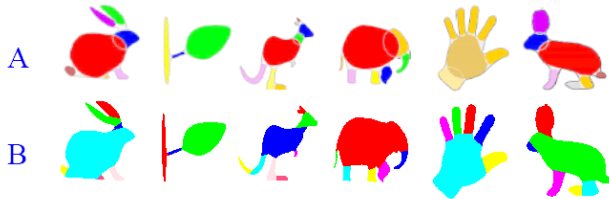


Figure 9. Row *A* demonstrates the results in [13]; Row *B* demonstrates the results of our method

results using their method; Row *B* is the results using our method. Since some perceptual parts have large concavity, our method will decompose them into multiple parts, for example, the leg and tail of the kangaroo in row *B*.

Fig.10 compares our methods with Reeb graph (column *B*). The problem with Reeb graph is that it can just capture partial information of an object. Since our method utilizes multiple Reeb graphs, thus, more information, especially all important information is preserved. We observe that no Reeb graph theory exists that allows for combination of multiple Reeb graphs. Column *D* illustrates the convex graph obtained by our method. In Fig.10, column *A* contains five shapes from MPEG-7 shape database. Column *B* illustrates their Reeb graphs, using height functions along vertical direction as Morse functions. Column *C* shows the decomposition results by our method, red lines are the cuts. Column *D* illustrates the convex graphs of these shapes. According to (4), when the costs of all cuts are nearly identical, we seek for a minimal number of cuts. The second image (fork) illustrates such situation. There are just four cuts; the second branch and the center part are in one part.

Fig.11 compares the approximate convex results of our method with the method proposed by Jyh-Ming Lien [11]. The second row shows the decomposed results of our method. Both methods can limit the concavity of the decomposed parts, although the definitions of concavity are different. The advantage of our method is that it can guarantee the number of the cuts is minimal.

Fig.12 demonstrates more 2D decomposed shapes from MPEG-7 shape database. For some objects, we can decompose them into meaningful parts; but in many situations, it will decompose a meaningful part into many approximate convex sub-parts.

Fig.13 demonstrates some decomposed 3D shapes. Most of the obtained parts seem meaningful. However, in the human model, the body and a leg belong to the same part; this is because the aim of our method is to decompose an object into approximate convex parts, it cannot guarantee that all decomposed parts are meaningful.

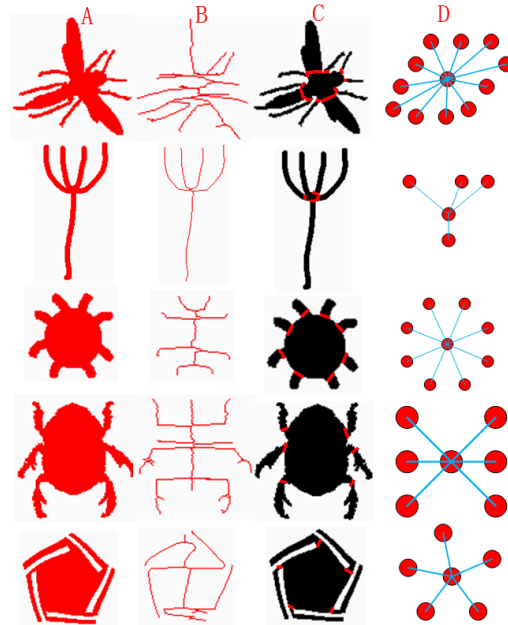


Figure 10. Reeb graphs and convex graphs. Column *A* contains five shapes from MPEG-7 shape database. Column *B* illustrates their Reeb graphs, using height functions along vertical direction as Morse functions. Column *C* shows the decomposition results by our method, red lines are the final cuts. Column *D* illustrates the convex graphs of these shapes

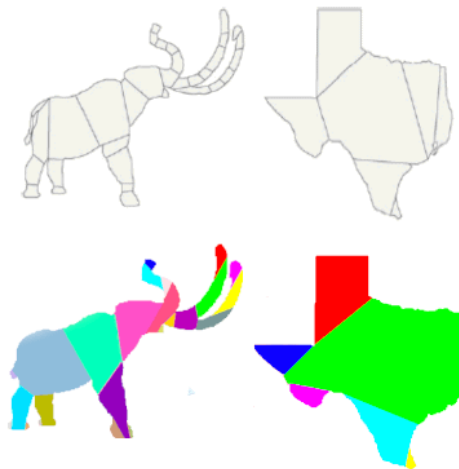


Figure 11. The results of approximate convex decomposition. The first row is the results in [11] and the second row is the result of our method

## 5. Conclusion

In this paper, we propose a novel method that can decompose an object into approximately convex parts. The decomposition is achieved by minimizing total cost of the cuts under some concavity constraints. Thus, it usually results in the number of decomposed parts is minimal. Our

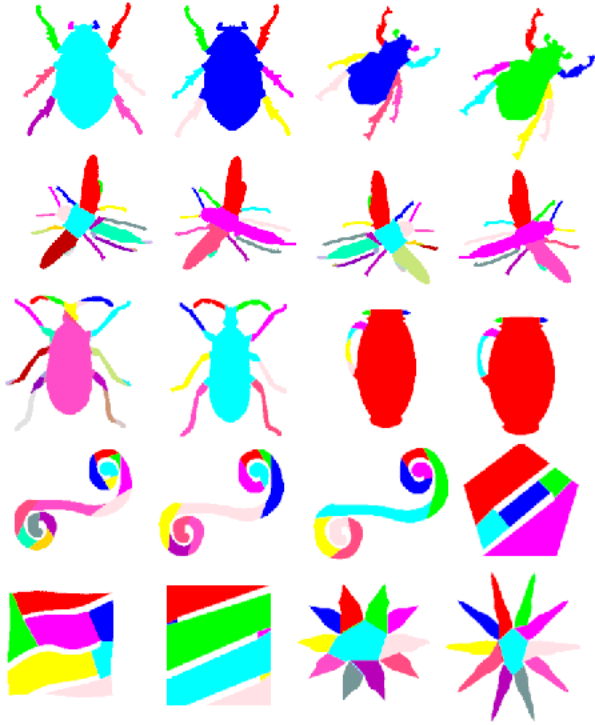


Figure 12. Decomposition of some shapes from MPEG-7 shape database



Figure 13. Decomposition of some 3D models

method is based on Morse theory and it combines the information from multiple Morse functions. After decomposition, we obtain a compact and efficient representation which contains nearly all important geometrical and topological information of original object. Such representation is very useful and we demonstrate its applications in some graphics and vision tasks.

## 6. Acknowledgement

This work is supported by National Science Foundation of China, under Grant no. 60873127 and Grant no. 60903096.

## References

[1] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological review*, 94(2):115–147, 1987. 3

[2] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *International Conference on Computer Graphics and Interactive Techniques*, pages 905–914, 2004. 3

[3] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. *Discrete and Computational Geometry*, 32(2):231–244, 2004. 2

[4] J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *Discrete and Computational Geometry*, 31(1):37–59, 2004. 3

[5] A. Fomenko and T. Kunii. *Topological modeling for visualization*. 1997. 3

[6] D. Hoffman and M. Singh. Saliency of visual parts. *Cognition*, 63(1):29–78, 1997. 3

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

[8] L. Latecki and R. Lakamper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73(3):441–454, 1999. 3

[9] J. Latombe, P. Svestka, M. Overmars, L. Kavraki, and L. Kavraki. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE International Conference on Robotics and Automation*, 1994. 3

[10] X. Li, T. Woon, T. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 35–42, 2001. 3

[11] J. Lien and N. Amato. Approximate convex decomposition of polygons. *Computational Geometry: Theory and Applications*, 35(1-2):100–123, 2006. 3, 7

[12] J. Lien and N. Amato. Approximate convex decomposition of polyhedra. In *ACM symposium on Solid and physical modeling*, 2007. 3

[13] X. Mi and D. DeCarlo. Separating parts from 2D shapes using relatability. In *International Conference on Computer Vision*, 2007. 6, 7

[14] M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, 2003. 3

[15] M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *ACM symposium on Solid modeling and applications*, 2004. 3

[16] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons Inc, 1998. 2

[17] K. Siddiqi and B. Kimia. Parts of visual form: Computational aspects. *Space*, 1(3), 1995. 3

[18] M. Singh, G. Seyranian, and D. Hoffman. Parsing silhouettes: The short-cut rule. *Perception and Psychophysics*, 61(4):636–660, 1999. 3

[19] K. Wu and M. Levine. 3D part segmentation using simulated electrical charge distributions. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(11):1223–1235, 1997. 3