Send Orders for Reprints to reprints@benthamscience.ae

390 *The Open Cybernetics & Systemics Journal,* 2015, *9,* 390-405

Open Access

New Framework for Decomposing a Polygon with Zero or More Holes

Zhou Hongguang* and Wang Guozhao

Institute of Computer Graphics and Image Processing, Department of Mathematics, Zhejiang University, Hangzhou, Zhejiang, 310027, P.R. China

Abstract: In this paper, we propose a unified framework for decomposition of a polygon with or without holes, enlightened by approximate convex decomposition. We transform target polygon containing holes into the polygon without holes by merging holes into external boundary, thus only need to partition the polygon without holes. Several constraints are enforced to ensure generate more visually meaningful decompositions, more elegant final components than that of other methods. Moreover, we estimate tolerance δ from geometry information of target polygon to obtain a naturally visual decomposition, a polished hierarchical representation. Our approach computes a decomposition of a polygon, containing zero or more holes, with n vertices and r notches in O(nr) time. In contrast, exact convex decomposition is NP-hard, or if the polygon has no holes, takes much more time. Many experimental results and applications are presented to show the superiority, applicability and flexibility of our approach.

Keywords: approximate convex decomposition, constraints, hierarchical representation, holes, naturally visual decomposition, Polygon, tolerance δ,

1. INTRODUCTION

Shape is the essence of many geometric problems. Many geometric problems have simpler and faster solutions on such a restricted type of polygon, so the strategy of solving these problems for general polygons is to partition them into simpler parts, solve the problem on each part and combine the partial solutions [1], therefore, decomposition is a technique commonly used to break complex models into submodels which are easier to process [2]. There is more emphasis on the optimality of the decomposition due to polygon decomposition often serves as a preprocessing step for many geometric algorithms. Besides, polygon decomposition has applications in networking area [3, 4].

In fact, many algorithms perform more efficiently on convex objects than on non-convex objects [5], so convex decomposition, which partitions the model into convex components has applications in many areas including: pattern recognition [6], Minkowski sum computation [7], motion planning [8], computer graphics, and origami folding [9]. Unfortunately, exact convex decomposition can be costly to construct and can result in a representation with an unmanageable number of components. For instance, the problem is NP-hard for polygons with holes by using exact convex decomposition [10].

Recently, [5] proposed a new approach called approximate convex decomposition (ACD) to decompose polygon. ACD is to identify and resolve the concave features of polygon in order of importance, while to preserve the less significant concave features, finally to obtain approximately convex pieces. The decomposing results using ACD are both significantly smaller and can be computed more efficiently due to ACD provides a mechanism to focus on key features, and ignore less significant features. However, in [5], ACD has not been applied to decomposition of polygon containing holes. Moreover, there are some drawbacks for ACD approach itself, such as, the results of ACD are inaesthetic, not sensuous, as a whole; the final partition could not resolve the visually noticeable features well, oftentimes. so how to successfully apply the ACD strategy to the decomposition of polygon containing holes and how to eliminate the problems generated by ACD approach have become two important issues.

In this paper, we introduce a novel approach and framework for decomposing a polygon with or without holes. Our approach does this task by applying the ACD strategy to the target polygon with or without holes neatly to construct a unified processing framework, while adding variety of effective constrained measures to the ACD strategy to improve the quality of the partition. Accordingly, firstly, for target polygon with holes, we convert it into a new polygon without holes by some merging techniques; for target polygon without holes, we don't change it; secondly, we handle the polygon decomposition using the ACD strategy together with various constraints; finally, we recursively partition the target polygon until all remaining components have concavity less than tolerance δ to obtain a naturally visual, elegant decomposition, where δ is estimated from geometry information of corresponding target polygon.

The contributions of our approach are summarized in the following:

*Address correspondence to this author at the department of Mathematics, Zhejiang University, Hangzhou, Zhejiang, 310027, P.R. China;
Tel: 13588745973; E-mail: zhgzju@gmail.com

1874-110X/15 2015 Bentham Open

●**Unified framework:** Our approach provides a unified framework for decomposing any simple polygon, with or without holes, especially, can handle the partition of polygon containing holes.

●**Visually meaningful results:** Our approach could guarantee generating more visually meaningful, more elegant final components by adding variety of constrains in our framework.

●**Estimated tolerance:** Our approach estimates tolerance δ by analyzing geometry information of target polygon, and subsequent self-adaptive adjustment to obtain a naturally visual decomposition, a polished hierarchical representation.

●**Fast and efficient:** Our approach is fast and efficient as it has only O(*nr*) time complexity, where *n* and *r* are the number of vertices and notches of target polygon with or without holes, respectively.

Besides, we explore the potential applications of our approach in various aspects.

2. RELATED WORK

Polygon decomposition is a well-established research area in Computational Geometry. There is a substantial body of literature that focuses on developing efficient algorithms for decomposing a polygon containing zero or several holes with the smallest number of a particular type of sub-polygons [11].

Convex partition approaches can be classified according to the following criteria: (1) Input polygon: simple, holes with or without; (2) Decomposition method: additional Steiner points allowed or disallowed; (3) Output partition properties: minimum number of components, shortest internal length. When Blum introduced the medial axis [12] in 1967, a partition at its branching points was also suggested. The medial axis is region-based and can be defined as the locus of centers of maximally inscribed disks. It can capture important visual cues of the shape, such as symmetry and complexity. However, its drawbacks are its sensitivity to noise and it may require extensive pre-processing or post-processing. A different approach [13] to using skeletons for shape decomposition, associates to the medial axis a weighted graph, called axial shape graph, the weights are seizing both local and global information about the shape.

As for the case with holes, Narkhede and Manocha [14] implemented the algorithm [15] for triangulating polygons without holes and extended the code to handle holes. Recently, Held [16] has also presented an algorithm for triangulating polygons with holes. So if we eliminate inessential diagonals of the triangulations obtained by both algorithms, we can obtain convex decompositions. For polygons with holes, the convex decomposition is NP-hard for both the minimum components criterion [17] and the shortest internal length criterion [18].

Recently, some approaches have been presented to decompose at salient features of a polygon. Siddiqi and Kimia [19] use curvature and region information to identify limbs and necks of a polygon and use them to do partition. Dey [20] decomposes a polygon into stable manifolds which are collections of Delaunay triangles of sampled points on the polygon boundary. Although these methods focus on visually significant features, these approaches require pre-processing or post-processing to adjust for boundary noise. Moreover, Lien and Amato [5] proposes ACD strategy. However, this method has many drawbacks and limitations.

3. PRELIMINARIES

We define H_p as the convex hull of a polygon *P*, and *P* is convex if $P = H_p$. Vertices of P which are not vertices of H_p are notches, i.e., notches have internal angles greater than 180^o.

4. OUR FRAMEWORK FOR DECOMPOSING A POLYGON WITH ZERO OR MORE HOLES

4.1. Our Framework

Inspired by ACD strategy, we develop a new technique and framework for decomposing a polygon with zero or more holes. Especially, our framework can handle the decomposition of polygon containing holes, thus not only expand the scope of polygon decomposition extensively, but also promote the application of polygon decomposition greatly. In a few words, we explore more effective, more robust, more unified approach based on ACD strategy actively.

Our strategy shown in Fig. (**1**) is that: firstly, for target polygon with holes, we merge all holes into its external boundary so as to combine them into a single entity without holes; for target polygon without holes, we do nothing; secondly, we use ACD strategy in company with variety of constrained measures to do polygon decomposition; finally, we recursively decompose the polygon until the concavity of all remaining components is less than tolerance δ to eventually attain a naturally visual, elegant decomposition, where δ is a specified parameter denoting the non-concavity tolerance and it is assessed from geometry information of the target polygon beforehand.

4.2. Merging holes into external boundary

For target polygon without holes, we would decompose it directly. However, for target polygon containing holes, the partition problem is far less understood, since this polygon is containing holes, holes P_i are separated from external boundary P_0, and holes P_i are isolated each other. Many previous methods can not deal with the decomposition of polygon containing holes successfully.

Fortunately, we develop a new technique for the decomposition of polygon containing holes. In brief, it is that: before the formal decomposition, we use connecting lines to

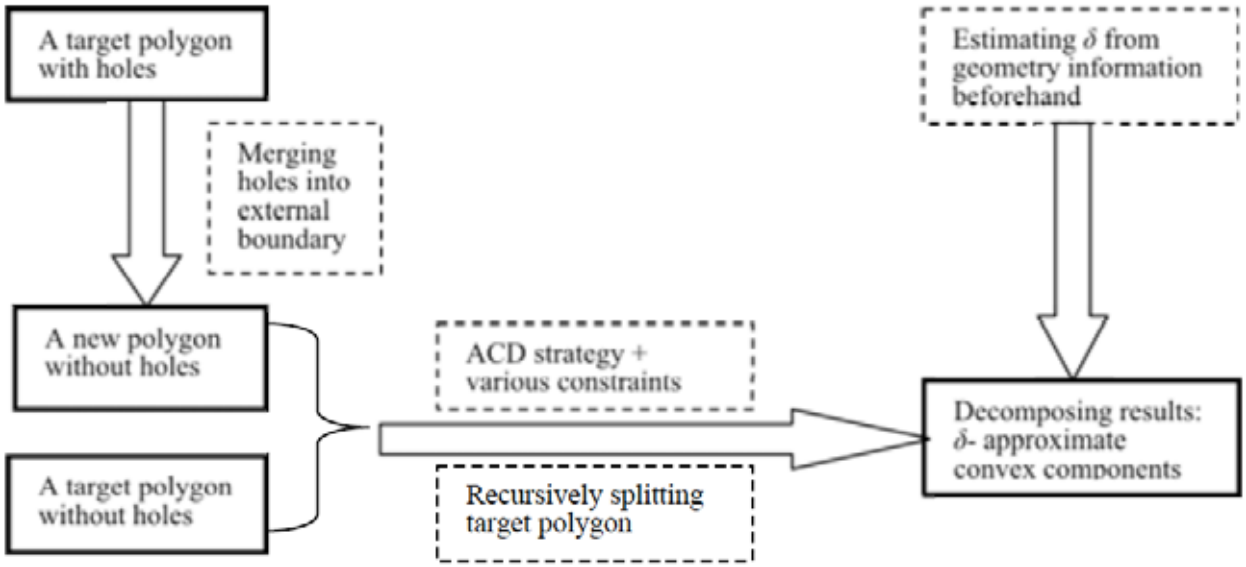


Fig. (1). The pipeline of our framework.

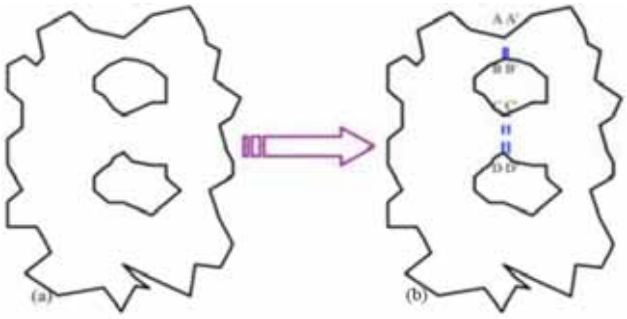


Fig. (2). The merging process: (**a**) Target polygon containing two holes; (**b**) A new polygon without holes, the blue broken lines denote the connecting lines, A and A', B and B', C and C', D and D' are the same point, respectively. The corresponding path is **A-B-C-D- D'- C'- B'- A'.**

merge holes into external boundary in order to compose a new polygon without holes; then, naturally, we could decompose this new polygon conveniently, see Fig. (**2**).

Apparently, the goal of merging process is to convert the target polygon containing holes into a new polygon without holes, thus the holes need to combine with external boundary uniformly and in order; and then we need to split this new polygon at the notable features, hence the connecting lines require revealing the largest possible concavity features of holes at the same time. To meet the above requirements, the merging process, i.e., computing the connecting lines should follow the following important principles:

- **Merging in an organized way:**

Firstly, holes are divided into several groups; secondly, we use the unidirectional connecting lines to string every hole in each group to form a chain, respectively; finally, every chain is linked with the external boundary. In the process, different chains do not intersect with each other.

- **Revealing the external concave features:**

The connecting lines, which connect one hole with another hole or external boundary are desired to be as shorter as possible.

- **Revealing the interior concave features:**

The antipodal pair of each hole, which represents the diameter of hole, can reflect the concave features of hole effectively. So we need to compute the antipodal pair of each hole, and then use the connecting lines to connect corresponding antipodal pair of one hole with corresponding antipodal pair of another hole or external boundary.

Here, we compute shortest paths between all pairs of vertices in the mesh generated by triangulating each hole, and choose a pair whose length of shortest path is maximal as the antipodal pair. During the process, the mesh can be generated in linear time, and the available pair can be found in linear time too, therefore our method is simple and fast.

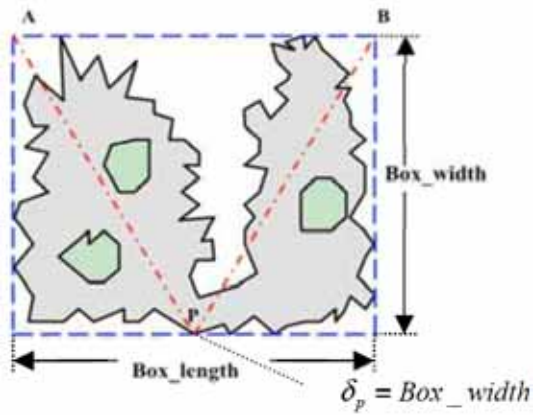


Fig. (3). Dotted blue box of a rectangle is the encasing box of the target polygon.

It is worthwhile to note that the connecting lines themselves are a part of all splitting lines, from the point of decomposition. Therefore, our approach for merging holes into external boundary is heuristic, smart, and simple.

4.3. Measuring concavity

Here, the measurement of boundary concavity (including hole's boundary) in our framework is listed as following:

$$\text{Concavity}(P) = \max_{x \in \partial P} \{\text{concavity}(x)\}; \quad (1)$$

If $x \in \partial P_0$**, then**

Compute concavity(x), using H-Concavity [5];

$$\text{Concavity}(P_0) = \max_{x \in \partial P_0} \{\text{concavity}(x)\}; \quad (2)$$

If $x \in \partial P_{i>0}$**, then**

$$\text{Concavity}(P_i) = \text{concavity}(P_0) + \text{concavity}(cx) + \text{dist}(P_i, cx) + \text{dist}(e_i, \text{cw}(e_i)). \quad (3)$$

$$\text{Concavity}(x) = \text{concavity}(P_0) + \text{concavity}(cx) + \text{dist}(P_i, cx) + \text{dist}(e_i, x); \quad (4)$$

(P_i: hole, P_0: the external boundary, e_i, cw(e_i): the antipodal pair points of hole P_i, cx: the connecting point with P_i, if $cx \in \partial P_0$, concavity (cx) is known, else $cx \in \partial P_{j>0}$, concavity (cx) = concavity (P_j), dist (e_i, x): distance of the shortest path from e_i to x in hole P_i, and e_i is one of the antipodal pair points, which is near to P_0)

4.4. Selection of Non-Concavity Tolerance δ

In our framework, δ needs to be setted as a logical value beforehand, since δ is a specified parameter denoting the non-concavity tolerance of the application. Here, we propose a method which is estimation with apriority combined with self-adaptive adjustment to obtain the optimal numerical value of δ.

4.4.1. Estimation with Apriority

Firstly, we give the estimated expression about tolerance δ with apriority, see Fig. **(3)**

$$\delta = \alpha \, \mathbf{gmin}(\textbf{\textit{Box\_length, Box\_width}}); \quad (5)$$

Box_length (Box_width): The length (the width) of the encasing box which envelops the polygon (target polygon without holes) or the external polygon (target polygon containing holes);

α**:** The tunable parameter, $0 < \alpha \leq 1$

From Fig. **(3)**, since encasing box could evaluate the global geometry information of polygon well, $\delta_P = Box\_width$ denotes the concavity of vertex P related to its associated bridge AB, which is an edge of the encasing box. From visual perception, vertex P which is related to bridge AB is too visually salient to be resolved, so δ_P is upper bound of tolerance δ; On the other hand, local geometry information of polygon, including angle and length ratio of adjacent edges, is more complicated, then we use α ($0 < \alpha \leq 1$) as a tunable parameter to adjust δ.

4.4.2. Self-adaptive Adjustment

Next, we should consider user needs and applications to do self-adaptive adjustment of δ.

(1) If user feels that the degree of decomposing is too weak or too strong, we divide or multiply the initial value of δ by 2 to attain the new tolerance δ, afterwards, according to the new tolerance δ, we do polygon decomposition renewedly. We repeat the above steps monotonously until we meet user requirements.

(2) During (1) process, if user considers that the visible concave features is resolved too little or too much, we divide or multiply the current value of δ by 2 to attain the new tolerance δ, after that, we do polygon decomposition renewedly by the new tolerance δ, but user realizes that the situation reversals, i.e. the visible concave features is resolved too much or too little, so we update δ by setting the average

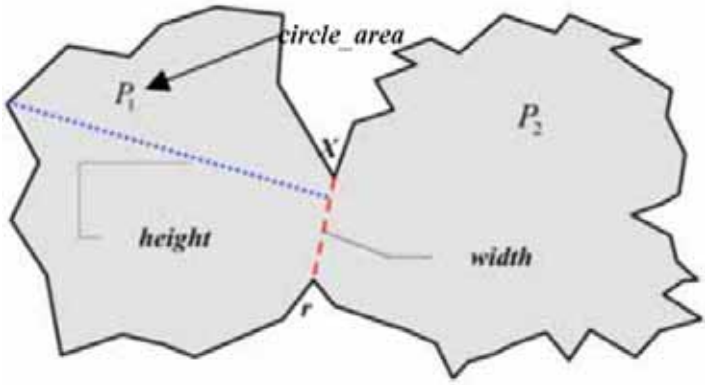


Fig. (4). Several parameters in this sketch map.

value of the old(current) tolerance δ and the new tolerance δ. Then, we do polygon decomposition newly by the updated tolerance δ, judging the partition situation constantly, setting the average value as the new tolerance δ constantly, and decomposing the target polygon constantly until we meet user requirements.

In this way, through the above two successive steps, we finally attain the best tolerance δ.

In addition, for the same target polygon with or without holes, when we get the best tolerance δ, δ could be divided or multiplied by 2 repeatedly to achieve a polished hierarchical representation which produces a series of increasingly or decreasingly convex decompositions.

4.5. Various Constrained Measures

Then, we start to do formal partition for this uniform polygon P without holes. We define concavity (P) = concavity (X) = $\max_{x\in\partial P}\{\text{concavity}(x)\}$. If concavity ($P$) is less than tolerance δ, then P is the final result, not do partition; else, we need to find another point of P, and compose a cutting line with X to decompose polygon P, called **Resolve (*P, X*).**

In [5], the process of **Resolve (*P, X*)** consists of the following steps: (1) Selecting all possible visible diagonals for the resolving notch X as candidates; (2) Picking the diagonal $\overline{XR}$ with the highest score, by using the following function expression (6) to score the candidates; (3) Using diagonal $\overline{XR}$ to decompose P. This method could decompose the target polygon P at the noticeable features and conform with human recognition because it consider the visible diagonal, large concavity and short diagonal as criterions, but it doesn't emphasize the constraints on the shape, so the resulting components would be narrow, saw-toothed, nonlogical and the final partition would be lack of aesthetic sensibility.

Hereby, to solve the above possible serious problem, we need to take various potent constrained measures. In this way, we decide to use the **Resolve** step of ACD strategy cooperated with various potent constraints to do decomposition. Our principle is that we need to resolve all noticeable, salient notches, while remaining some inconspicuous notches; on this basis, we try our best to assure that the resulting components have graceful shape and exquisite configuration**.**

$$f(r,X)=\frac{1+\alpha\times concavity(r)}{\beta\times dist(r,X)}$$ X: the resolving notch; $r\in\partial P_{i\geq 0}$, α=0.1, β=1. (6)

4.5.1. Various Constraints with Shape

From the heuristic inspired by human perception, we put forward variety of constraints to make the resulting components have polished appearance and get an improved version.

Firstly, after selecting all possible visible diagonals, we should filtrate candidates by checking the indexes of the resolving notch X and one candidate r. i.e., if the interval of indexes of r and X is less than 4, then r would be deleted from candidates. This step is used to eliminate the sharp and narrow area of resulting components.

Secondly, we further filtrate the remaining candidates.

Setting some parameters in advance, please see Fig. (**4**).

The resolving notch X and any one candidate r decompose polygon P into P_1, P_2, here we consider P_1 as a smaller sub-polygon.

***height*:** the maximum straight-line distance from vertices of P_1 to the diagonal $\overline{Xr}$;

***width*:** the length of the diagonal $\overline{Xr}$;

***area: height*×*width; circle_area*:** the area of P_1**;**

***fill: circle_area / area; aspect: width / height*;**

Through many experiments, we prescribe that if ***fill*** is more than 0.8 or ***aspect*** is less than 0.3, then r is remained, otherwise, r is deleted. This step is used to eliminate the saw-toothed region or the salient region of resulting components.

Thirdly, after using the function (6) to pick up the final candidate R, we require to do a significant estimation to

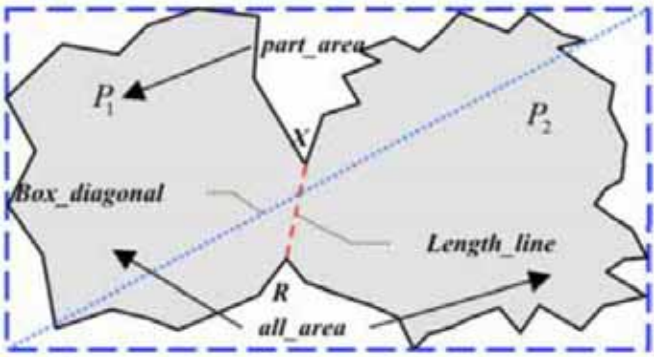


Fig. (5). Several parameters in this sketch map.

finally determine whether R is an appropriate candidate to compose a cutting line $\overline{XR}$ to decompose polygon P.

Setting some parameters in advance, please see Fig. (**5**).

The resolving notch X and the final candidate R decompose polygon P into P_1, P_2 , here we consider P_1 as a smaller sub-polygon.

***Length_line*:** the length of the diagonal $\overline{XR}$;

***Box_diagonal*:** the length of diagonal of the encasing box which envelops polygon P;

***rate: Length_line / Box_diagonal*;**

***part_area*:** the area of P_1; ***all_area*:** the area of polygon P;

***ratio: part_area / all_area*;**

By the way of many experiments, we perform the estimation that if ***rate*** is more than 0.17 and ***ratio*** is less than 0.1, then the final candidate R is invalid, the process of decomposition is halted, polygon P is returned. Otherwise, R is valid, and polygon P is decomposed by the diagonal $\overline{XR}$. This step is used to assure the decomposing components are moderate, not too big or too small. More formally, the visually salient features are more related to the short cutting line, so if $\overline{XR}$ is short enough, the decomposition by $\overline{XR}$ is also permitted.

By carrying out the above three constraints together with the **Resolve** step of ACD strategy, we eventually obtain a proper cutting line $\overline{XR}$ to decompose polygon P into two resulting components, denoted as P_1, P_2.

4.5.2. Subsequent Reparative Step

It is worthwhile to point out that there are some failed cases where there is no valid candidate, then decomposition will be halted, our method might not succeed in the desired decomposition. So this is an issue which we need to solve.

Actually, there are very few such failed cases in the experimental examples shown in the paper, since we specify the appropriate parameters in the constraints based on much decomposing experiments. If such failed cases occur, we remove all the constraints and only use **Resolve** step of ACD strategy to do decomposition forcibly until the concavity of all remaining components is less than δ to maintain our goal. Although we could not guarantee the quality shape of every resulting piece, we could ensure the whole quality shape of resulting components effectively.

Hence, by carrying out the above partition, our method is not only able to eliminate the problem of possibly producing the abnormal final components, which are sharp, narrow, saw-toothed, salient, oversize, too small, but also ensure that the resulting decomposition would have refined shape and exquisite configuration.

4.6. Recursively Splitting the Target Polygon

Finally, we need to decompose the resulting components P_1, P_2 further and recursively until all remaining components have concavity less than given tolerance δ.

However, this recursion is more difficult due to the target polygon P may contain more holes or no holes, the decomposing components P_1, P_2 may contain more holes or no holes too, and manifest themselves as variety of complicated structures. Therefore, we should take several potent techniques to do the recursion effectually.

When target polygon P is containing no holes, we only take the direct recursion for all new components, see in Fig.6

But when target polygon P is containing more holes, specifically, the recursion itself is divided into two cases:

(1) The resulting components (including P_1, P_2 and the new target polygon generated by splitting previous target polygon) are polygons without holes, we only take the direct recursion for all new components too, see in Fig. (**6**).

(2) The resulting components (including P_1, P_2 and the new target polygon generated by splitting previous target polygon) are also polygons containing holes, then the recursion is more complicated, since this resulting components

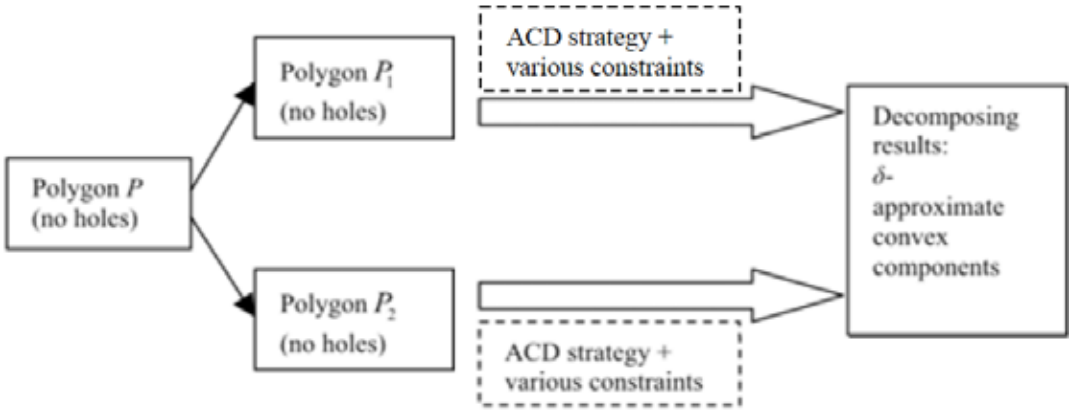


Fig. (6). Recursively splitting the target polygon without holes.

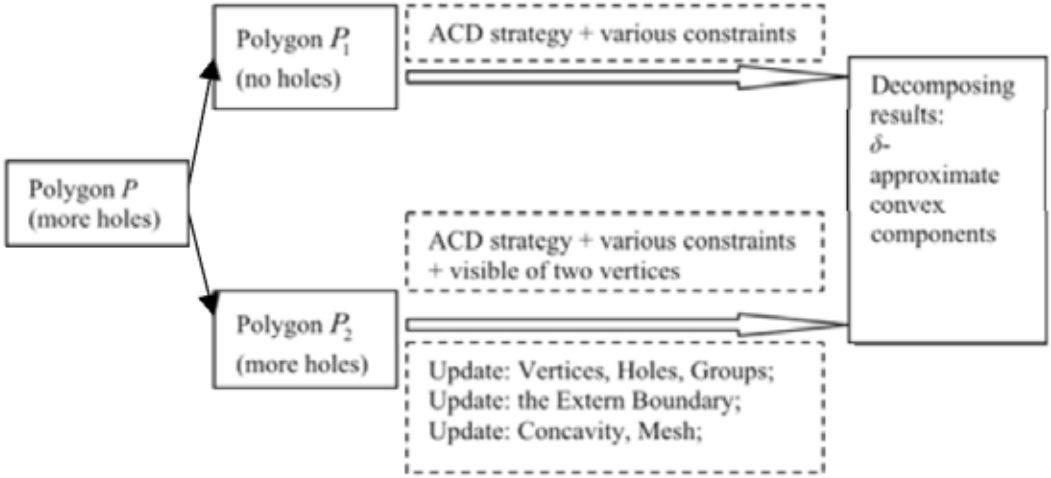


Fig. (7). Recursively splitting the target polygon containing holes.

are composed of holes, connecting lines and external polygon, not generic simple polygons. To account for this, we adopt a simple and useful scheme to control the recursive partition of all new components availably. In this scheme, we use two successive procedures recursively.

Update all the useful information of target polygon: we need to update the vertices, the holes, the groups of holes, the extern boundary, the concavity of new target polygon (call P_1, P_2 as new target polygons), the structural mesh constructed by holes, extern boundary of the new target polygon, and all connecting lines of the new target polygon in turn. The vertices of new target polygon are naturally obtained by using the cutting line. Then, since the structure of the groups of holes and the geometry information of each hole maybe changed, we recompute the vertices and the antipodal pair of each hole, in addition, we adjust the composing of groups of holes, the structural mesh and all connecting lines. And it is noted that the new target polygon is composed of the groups of holes and the extern boundary, so the vertices of the extern boundary are also naturally obtained. We use the measure of concavity (see **Sec 4.3**) to recompute the concavity of all vertices. Now, we acquire all necessary new information to do the formal partition.

Do the partition: we use ACD strategy cooperated with various constraints to decompose the new target polygon into two components. It is noted that we have updated mesh and all connecting lines in the previous procedure, so we utilize them to determine the visible of two vertices in the new target polygon in this procedure.

We apply this two successive procedures recursively to new target polygon produced by splitting previous target polygon until all remaining components have concavity less than given tolerance δ.

In fact, when target polygon P is containing more holes, the resulting components would be polygons without holes, or with holes, therefore the whole recursion is composed of the above two cases of recursion, one is mixed with the other, as shown in Fig. (**7**).

In conclusion, for any target polygon containing holes or no holes, by taking the recursion which is a significant and last part of unified framework, we ultimately obtain all the beautiful final components.

4.7. Algorithm Steps

We summarize the steps of our approach as following:

Input: A polygon *P* with or without holes.

Output: A more visual natural, elegant partition of *P*, $\{C_i\}$, such that $\forall$i=1,2,3,..., concavity(C_i) $\leq \delta$.

Step 1. Estimate tolerance δ from geometry information of polygon *P*.

Step 2. If polygon *P* contains more holes, transform it into a new polygon without holes using merging techniques. If polygon *P* contains no holes, don't change it.

Step 3. Measure concavity of every point on ∂P using proposed concavity measures.

Step 4. Choose a point *X* as a witness of the concavity of *P*, concavity (*P*) = concavity (*X*) = $\max_{x\in\partial P_i}\{\text{concavity}(x)\}$, i=0, 1, 2…n.

Step 5. If concavity (*P*) $\leq \delta$, then polygon *P* is the final result; else, we do Resolve (*P*, *X*).

Step 6. Add various constraints to Resolve (*P*, *X*) in ACD strategy to decompose polygon *P* into two resulting components P_1, P_2.

Step 7. Analyze the features of the resulting components according to corresponding cutting line.

Step 8. Update all the useful information of the resulting components.

Step 9. Repeat Step 4 to Step 8 recursively until all the final components have concavity less than δ, then we get $\{C_i\}$, concavity(C_i) $\leq \delta$.

4.8. Analysis and Discussion

In our approach, we make the best of the advantage of ACD strategy which is that it only deal with salient feature notches and keep less significant feature notches, meanwhile, we add several more useful constraints to the process of resolving feature points, thus our resulting decomposition represents more visually meaningful components, more elegant components well than that of ACD approach [5], especially than that of exact convex decomposition.

Moreover, in our resulting decomposition, the number of pieces is greatly less than that of whatever exact convex decomposition, regardless of emphasizing minimizing number, due to our method uses ACD strategy which provides a mechanism to focus on key features and ignore unimportant features. Compared with ACD approach [5], the number of pieces generated by our approach is also less, since we take extra helpful constraints to remove many problems produced by ACD approach itself.

5. COMPLEXITY ANALYSIS

Theorem: Let $\{C_i\}$, i = 1,…, *m*, be a δ-approximate convex decomposition of a polygon *P* with *n* vertices, *r* notches and *k* holes (*k* = 0,1,2…). Polygon *P* can be decomposed into $\{C_i\}$ in O (*nr*) time using our framework.

Proof:

(1) When *k* = 0, i.e., polygon *P* has no holes.

For each iteration, we compute the convex hull of *P* and the concavity of *P* in linear time in the number vertices of *P* [21]. In order to compute the concavity of *P*, we need to pair up the bridges and pockets which demands O(*n*) time, compute the distance from the pockets to the bridges and measure concavity(*P*) using H-Concavity, which takes O(*n*) time. Moreover, **Resolve** step splits *P* into C_1 and C_2 in O(*n*) time, and the several constraints take O(*n*) time, too. Hence, each iteration takes O(*n*) time in our framework.

If the resulting decomposition has m components, the total number of iterations is *m*-1; after we split *P* into C_1 and C_2 each time, at most three new vertices are created. So the total time required for our framework decomposing polygon *P* without holes is O(*n*+(*n*+3)+…+(*n*+3×(*m*-2))) = O(*n*×(*m*-1) + 3× $\frac{(m-1)\times(m-2)}{2}$) = O(*nm*+ m^2).

(2) When *k* > 0, i.e., polygon *P* has some holes.

Firstly, we estimate the concavity of a hole locally using its principal axis (O(*n*) time) or its medial axis (Linear time) or shortest paths (Linear time), afterwards, we add a diagonal between the vertex with the maximum concavity and its closest vertex on ∂P (O(*n*) time). At most three new vertices are created when each hole connects to ∂P_0. Hence, resolving k holes, which equals to join into a single entity without holes, takes O (*nk*+ k^2) time.

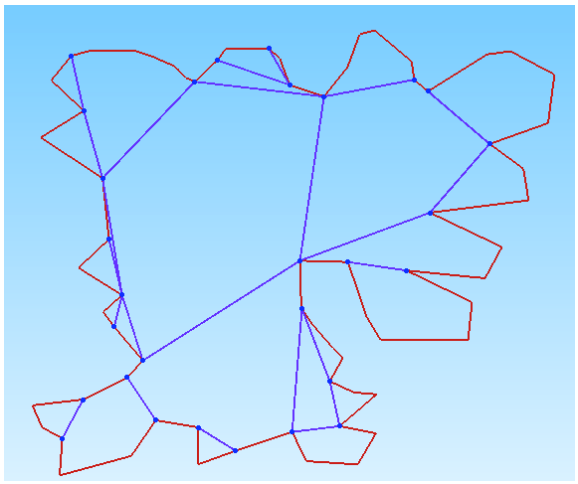
In an analogous manner, for each iteration, the process take O(*n*) time, too. Hence, the time required for formal decomposition is O(*n*+(*n*+3)+…+(*n*+3×(*m*-2))) = O(*n*×(*m*-1)+3× $\frac{(m-1)\times(m-2)}{2}$) = O(*nm*+ m^2). Therefore, the total time required for our framework decomposing polygon *P* containing holes is O(*nm*+ m^2)+ O (*nk*+ k^2) time.

Accordingly, we consider two cases (*k* =0; *k*>0) together to come to a conclusion: Polygon *P* with *n* vertices, *r* notches and *k* holes (*k* = 0,1,2…) can be decomposed into $\{C_i\}$ in O(*nm*+ m^2) + O (*nk*+ k^2) time. Since *m*≤*r*+1 , *k*<*r*,

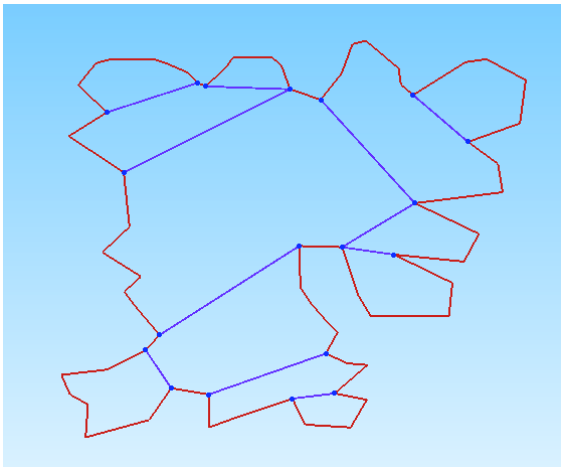
O (*nm*+ m^2)+ O (*nk*+ k^2) = O (*n*(*m*+*k*) + m^2 + k^2) =

O (*nr*+ r^2), because *r* < *n*, O (*nr*+ r^2) = O (*nr*). Hence, using our framework to do the partition of polygon *P* with holes or without holes takes O (*nr*) time. Q.E.D.

Our framework has O (*nr*) time complexity as the same as that of ACD approach [5]. In contrast to other methods, such as Green's approach [22] has O (r^2n^2) time complexity, Keil's approach [18] has O($r^2n^2 \log n$) time complexi-



(**a**)



(**b**)

Fig. (8). Comparison of decomposing results: (**a**) The decomposing result using the ACD approach; (**b**) The decomposing result using our approach.

ty, our approach could be more efficient to generate more naturally visual partitions than other decomposing methods.

6. EXPERIMENTAL RESULTS AND APPLICATIONS

We will show some examples illustrating the superiority, flexibility and applicability of our approach. All the examples presented in this paper were made on a 2.67 GHz Pentium IV computer with 2GB memory.

6.1. Decomposing a Polygon Without Holes

Compared with ACD approach [5]**,** our approach makes many significant improvements, so as to can generate more naturally visual decomposition, more graceful and beautiful final components.

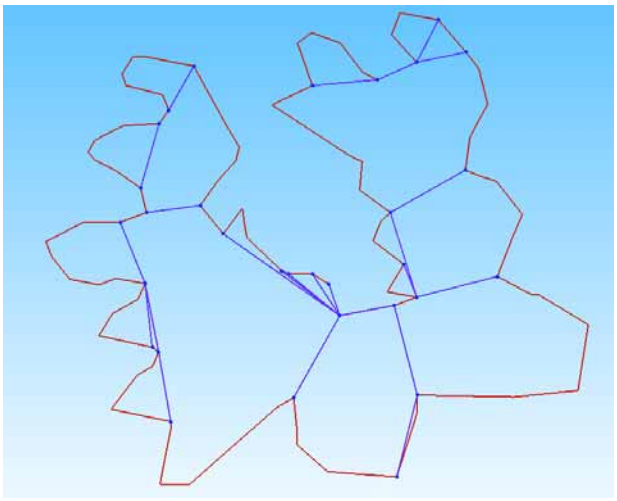
Fig. (**8a**) shows the decomposition of this fish polygon by using the ACD approach**,** the partition using our approach is shown in Fig. (**8b**). As we can see, the result using our approach in Fig. (**8b**) has better visual quality than that in Fig. (**8a**), e.g., some of the final components shown in Fig. (**8a**) are long, narrow, or serrate, not harmonious, while the components shown in Fig. (**8b**) are elegant and beautiful.

We present another comparison to demonstrate the superiority and effectiveness of our method for decomposing a polygon without holes, see Fig. (**9**).

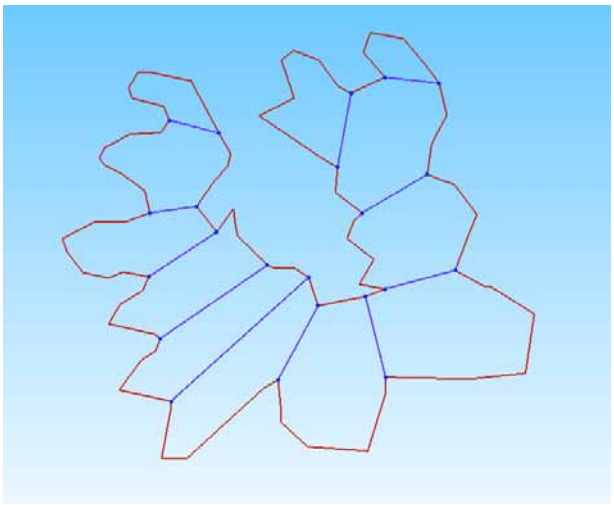
6.2. Decomposing a Polygon Containing Holes

The most important property of our framework is that it can partition a polygon with holes into “approximately convex” components availably and quickly. Our approach is suitable for decomposing rich variety of polygons with arbitrary many holes, regardless of the complexity of the input.

The result of Fig. (**10a**) shows that our approach can be used to decompose whatever complex polygon with arbitrary

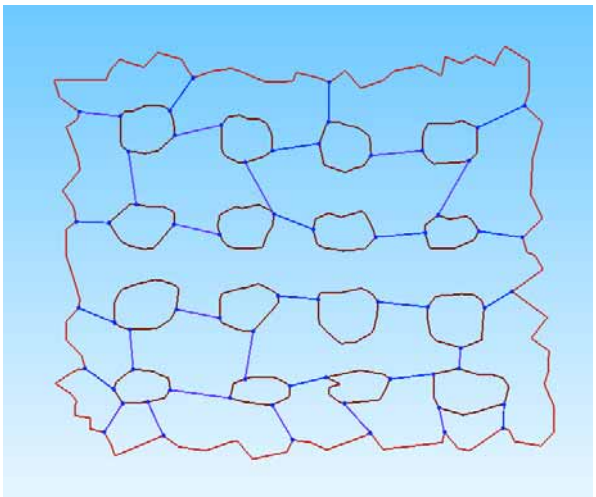


(**a**)



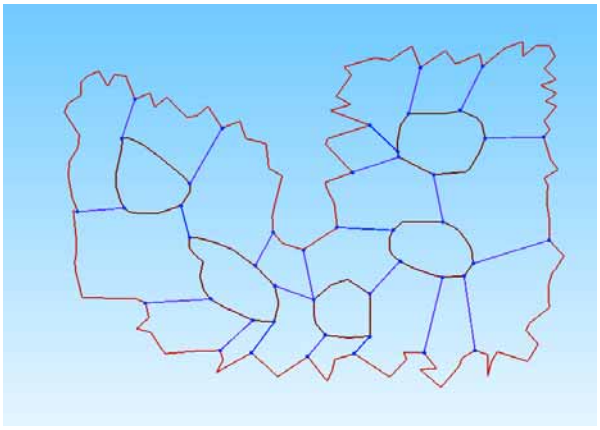
(**b**)

Fig. (9). Comparison of decomposing results: (**a**) The decomposing result using the ACD approach; (**b**) The decomposing result using our approach.



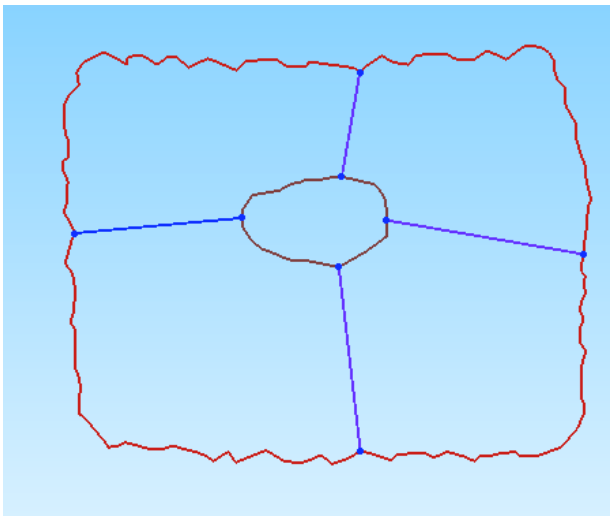
(**a**)

Fig. (**10**). Contd…

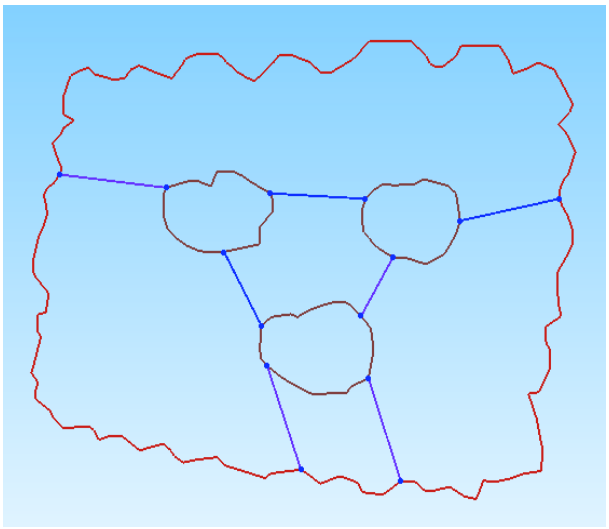


(**b**)

Fig. (10). Decomposing a polygon with holes using our approach: (**a**) A polygon denoted by red lines containing 16 holes denoted by brown lines is decomposed; (**b**) A polygon denoted by red lines containing five holes denoted by brown lines is decomposed.

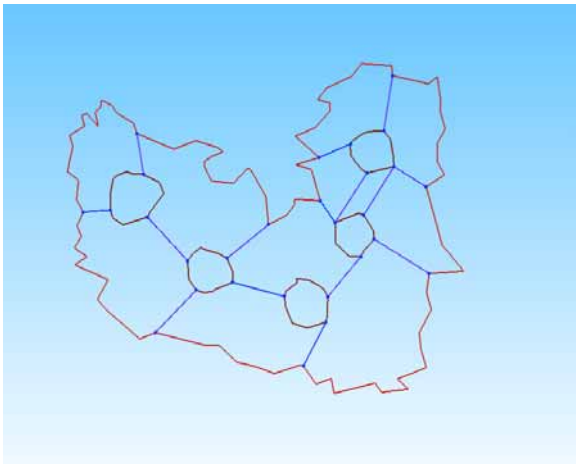


(**a**)

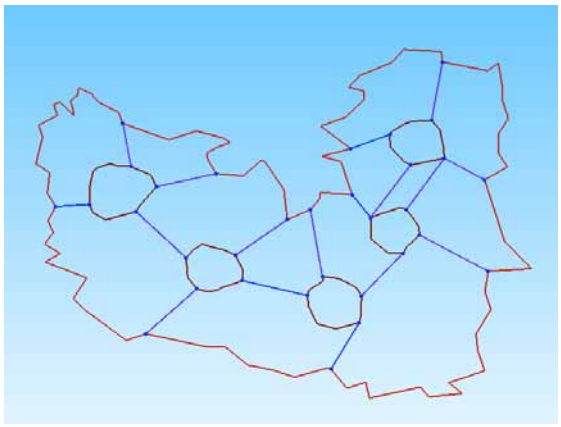


(**b**)

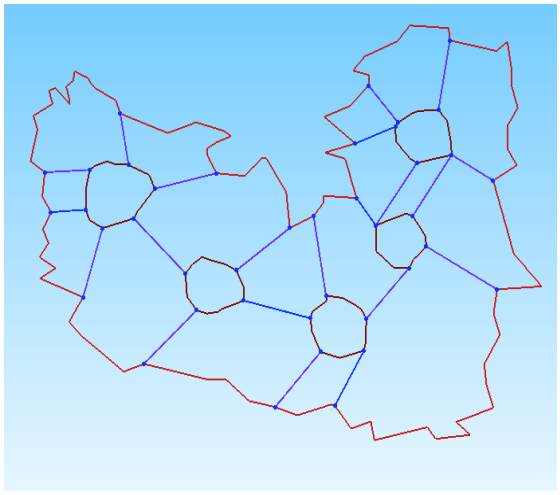
Fig. (11). Decomposing a polygon with holes using our approach: (**a**) A polygon denoted by red lines containing one hole denoted by brown lines is decomposed; (**b**) A polygon denoted by red lines containing three holes denoted by brown lines is decomposed.



(**a**)

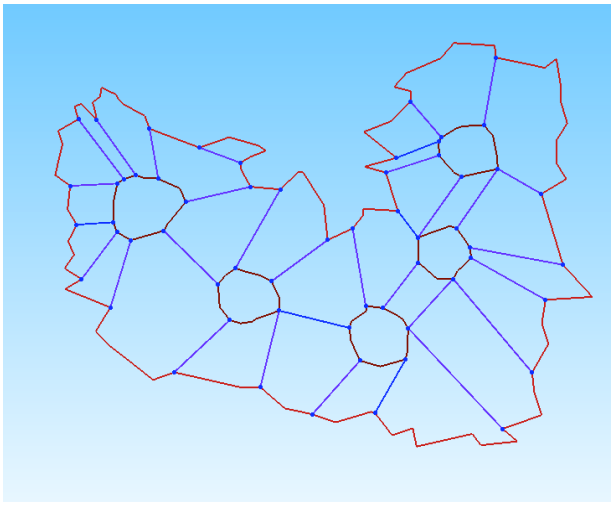


(**b**)



(**c**)

Fig. (**12**). Contd…



(**d**)

Fig. (12). The hierarchical representation for decomposing a polygon with five holes, tolerance δ decreases from (**a**) to (**d**): (**a**) δ= 0.1; (**b**) δ= 0.08; (**c**) δ= 0.05; (**d**) δ= 0.02.

Table 1. Lists the summary information for the decomposing models shown in this paper. As we can see, our approach achieves a good combination of speed, decomposed quality, and desirable robustness.

Model	*# Vertices*	*# Notches*	*#Holes*	*# Resulting components*	*Running time (s)*
Fig.8a	253	121	0	25	0.06
Fig.8b	253	121	0	12	0.05
Fig.9a	173	86	0	22	0.3
Fig.9b	173	86	0	12	0.08
Fig.10a	603	337	16	18	50
Fig.10b	402	180	5	19	12
Fig.11a	386	170	1	4	3
Fig.11b	530	267	3	5	40

number of holes. Fig. (**10b**) shows another decomposing example using our approach, we could observe that the result reflects visually important features, such as some small serrations in the peripheral boundary marked by red lines are remained, not decomposed, while several sharp concave features are decomposed.

Figs. (**11a, b**) demonstrate the applicability and superiority of our approach for decomposing a polygon with many holes. Note that a and b are two armor plate models containing one hole, three holes respectively, the decomposing results show that our approach could generate the elegant, visual meaningful, harmonious partitions.

Table **1**. Summary information for models studied in the paper. The front four rows show the comparison between the decomposing results using ACD approach [5] and the decomposing results using our approach; the behind four rows show the partitions of polygon with holes using our method.

6.3. Hierarchical Representations

Our framework sets a set of tolerances δ to attain an elegant hierarchical representation and high visual quality decompositions, as shown in Fig. **(12)**.

6.4. Decomposing 2D Graphs

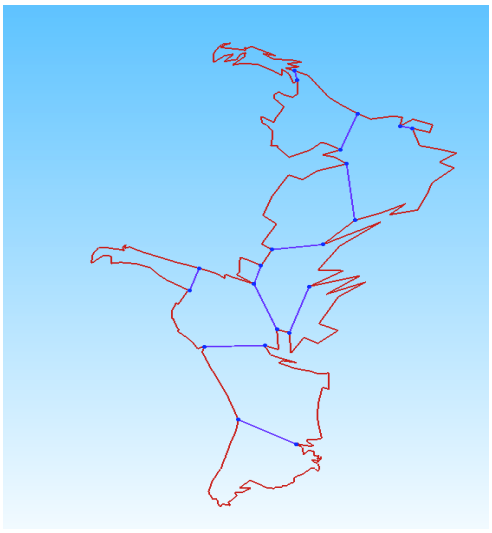
Besides, our framework could be used to decompose 2D graphs. Fig. (**13**) shows three examples of decomposing 2D graphs, the results using our framework are visual meaningful, beautiful, and reflecting the concave features naturally.

6.5. Decomposing Surfaces

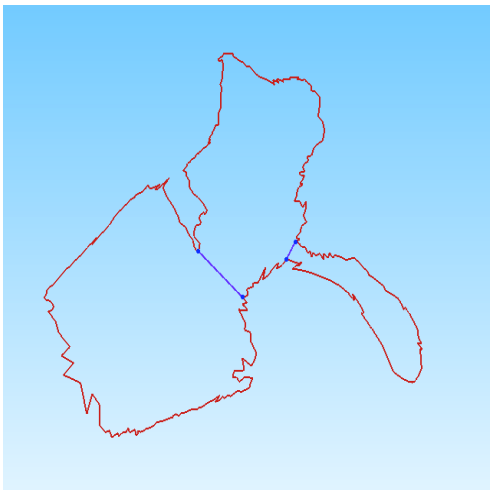
Here, we try to apply our framework to decomposing 3D surfaces. The results shown in Fig. (**14**) demonstrate the applicability of our framework.

CONCLUSION

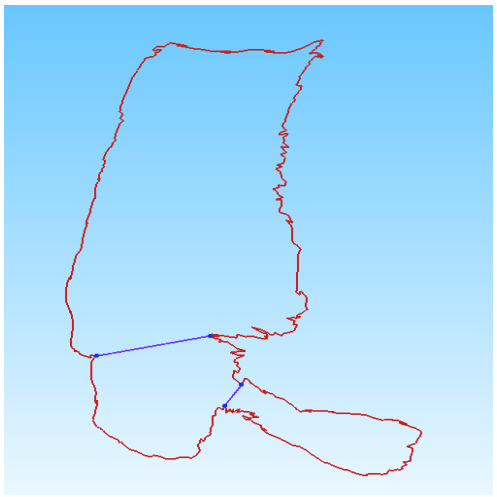
A unified framework for decomposing any simple polygon, with or without holes is presented in this paper. The target polygon containing holes is transformed into a polygon without holes by combining the external polygon and isolated holes into a single entity, and then we only need to handle the decomposition of polygon without holes. Our approach is inspired by ACD strategy; while variety of constraints is integrated into our approach to assure produce



(**a**)

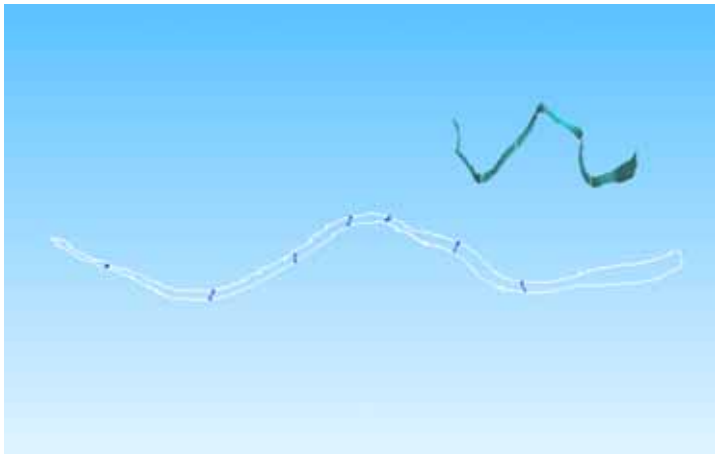


(**b**)



(**c**)

Fig. (13). Decomposing 2D graphs: (**a**) Decomposing a 2D horse graph using our framework; (**b**) and (**c**) Decomposing a 2D graph using our framework. Observe the decomposition in details in the area of branches and the sharp concave features.



(**a**)

(**b**)

Fig. (14). Decomposing surfaces: (**a**), (**b**) Two vascular surfaces are decomposed using our framework.

more visually meaningful partitions, and more beautiful final components.

Our algorithm has only O (*nr*) time complexity, so it is fast and effective. In addition, an important feature of our approach is that we could estimate proper tolerance δ from target polygonal geometry information to generate a naturally visual decomposition, farther more, to attain the polished hierarchical representation. Many experimental results have been presented to show the applicability and flexibility of our approach.

Decomposition plays fundamental and critical roles in many fields. Certainly, there is still a lot of work remaining to be done. We hope to find some effective decomposing methods which are related to the physical properties of geometric models.

ABOUT THE AUTHORS

First Author: Zhou Hongguang, studying for PhDs in Zhejiang University. The author's major is Applied Mathematics.

Second Author: Wang Guozhao, Professor and PhD supervisor of Zhejiang University. His main research interests include compute graphics, computer aided geometric design. 36 papers received by EI or SCI had been published.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (No. 61272300).

REFERENCES

[1] M. Tănase, and R.C. Veltkamp, "Polygon decomposition based on the straight line skeleton," *In: Proceedings of the 19th Conference on Computational Geometry (SoCG2003)*, New York, USA, 2003, pp. 58-67.

[2] J.-M. Lien, "*Approximate convex decomosition and its applications*", PhD Dissertation, Texas A&M University, College Station, Texas: USA, 2006.

[3] A. Kröller, S. Fekete, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction for large sensor networks," In: *Proceedings of the 17th Annual ACM-SIAM Symposium*

on Discrete Algorithms (SODA 2006), Miami, Florida, USA, 2006, pp.1000-1009.

[4] X.-J. Zhu, R. Sarkar, and J. Gao, "Shape Segmentation and Applications in Sensor Networks," *The 26th IEEE International Conference on Computer Communications*, Anchorage, Alaska, USA, 2007, pp. 1838-1846.

[5] J.-M. Lien, and N.-M. Amato, "Approximate convex decomposition of polygons," *Computational Geometry*, vol. 35, pp. 100-123, 2006.

[6] H. Feng, and T. Pavlidis, "Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition," *IEEE Transactions on Computers*, vol. C-24, pp. 636-650, 1975.

[7] P. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of minkowski sums," *Computational Geometry*, vol. 21, pp. 39-61, 2002.

[8] S. Hert, and V.J. Lumelsky, "Polygon area decomposition for multiple-robot workspace division," *International Journal of Computational Geometry and Applications*, vol. 8, pp. 437-466, 1998.

[9] E. Demaine, M. Demaine, and J. Mitchell, "Folding flat silhouettes and wrapping polyhedral packages: New results in computational origami," *Computational Geometry*, vol. 16, pp. 3-21, 2000.

[10] A. Lingas, "The power of non-rectilinear holes," In: *Proceedings of the 9th International Colloquium on Automata Languages and Programing*, Berlin, 1982, pp. 369-383.

[11] J.M. Keil, *Polygon Decomposition*, Elsevier Science Publishers B.V., North-Holland, Amsterdam, 2000, pp.491-518.

[12] H. Blum, *A Transformation for Extracting New Descriptors of Shape*, Cambridge, MIT Press, 1967, pp. 362-380.

[13] M. Simmons, and C. Séquin, "2D shape decomposition and the automatic generation of hierarchical representations," *International Journal of Shape Modeling*, vol. 4, pp. 63-78, 1998.

[14] A. Narkhede, and D. Manocha, *Fast Polygon Triangulation Based on Seidel's Algorithm*, Academic Press, 1995, pp.394-397.

[15] R. Seidel, " A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons," *Computational Geometry: Theory and Applications*, vol. 1, pp. 51-64, 1991.

[16] M. Held, "FIST: Fast industrial-strength triangulation of gons," *Algorithmica*, vol.30, pp.563-596, 2001.

[17] A. Lingas, R. Pinter, R. Rivest, and A. Shamir, "Minimum edge length partitioning of rectilinear polygons," *In: Proceedings of the 20th Allerton Conference Communications Control Computing*, USA, 1982, pp.53–63.

[18] J.M. Keil, "Decomposing a polygon into simpler components," *SIAM Journal on Computing*, vol. 14, pp. 799-817, 1985.

[19] K. Siddiqi, and B.B. Kimia, "Parts of visual form: Computational aspects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 239-251, 1995.

[20] T. Dey, J. Giesen, and S. Goswami, "Shape segmentation and matching with flow discretization," *The 8th International Workshop on Algorithms and Data Structures (WADS 2003)*, Ottawa, Ontario, Canada, 2003, pp. 25-36.

[21] D. McCallum, and D. Avis, "A linear algorithm for finding the convex hull of a simple polygon," *Information Processing Letters*, vol. 9, pp. 201-206, 1979.

[22] D. Greene, "The decomposition of polygons into convex parts," *Computational Geometry*, vol. 1, pp. 235-259, 1983.

Received: October 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014