Dual-Space Decomposition of 2D Complex Shapes *

Guilin Liu
gliu2@gmu.edu

Zhonghua Xi
zxi@gmu.edu

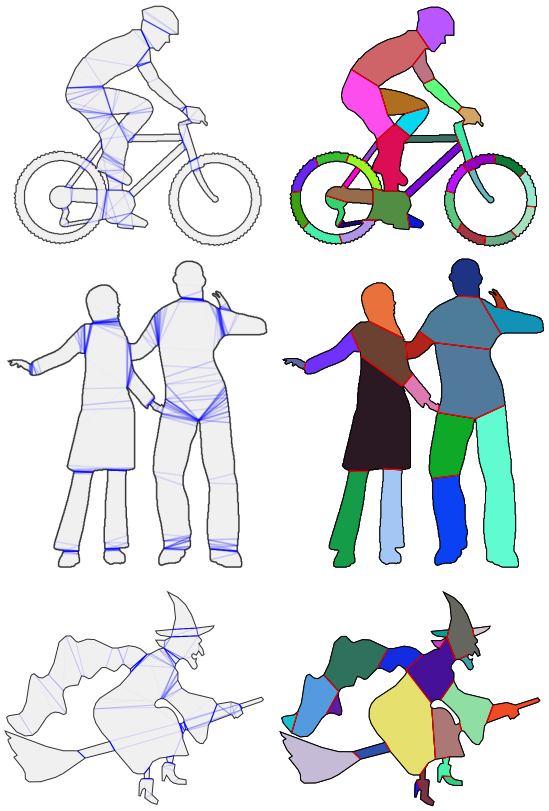
Jyh-Ming Lien†
jmlien@cs.gmu.edu

Abstract

While techniques that segment shapes into visually meaningful parts have generated impressive results, these techniques also have only focused on relatively simple shapes, such as those composed of a single object either without holes or with few simple holes. In many applications, shapes created from images can contain many overlapping objects and holes. These holes may come from sensor noise, may have important parts of the shape or may be arbitrarily complex. These complexities that appear in real-world 2D shapes can pose grand challenges to the existing part segmentation methods. In this paper, we propose a new decomposition method, called Dual-space Decomposition *that handles complex 2D shapes by recognizing the importance of holes and classifying holes as either topological noise or structurally important features. Our method creates a nearly convex decomposition of a given shape by segmenting both the polygon itself and its complementary. We compare our results to segmentation produced by non-expert human subjects. Based on two evaluation methods, we show that this new decomposition method creates statistically similar to those produced by human subjects.*

1. Introduction

Decomposing two-dimensional (2D) shapes into functional and visually meaningful parts is a fundamental process in human vision [12, 15, 4, 5, 1, 30]. Many segmentation and decomposition techniques have been proposed to mimic this process computationally [6, 16, 31, 2]. While these techniques (usually known as *part segmentation*) generate impressive results, they also have only focused on relatively simple shapes, such as those composed of a single object either without holes [30] or with few simple holes [22, 25, 29]. Even when holes are explicitly considered, existing techniques usually decompose the shape so that no holes are left [22, 25, 29]. In many situations, shapes are created from images (e.g., silhouettes) of overlapping objects and can also contain many undesired holes due to sensor noise. In other cases, these holes can be important part of the shape and arbitrary complex. These complex shapes, such as those shown in Figures 1, 6, and 7 pose grand challenges to the existing part segmentation methods.



(a) Human (b) DuDe

Figure 1. Examples of decompositions from human and dual-space decomposition (DuDe).

Another major limitation in the existing part segmentation methods is the lack of well-defined criteria and benchmark for quality comparisons. Unlike image segmentation, comparison between part segmentation methods is usually done visually and evaluated based on the size of the final decompositions, i.e. smaller decomposition is better.

*This work is supported in part by NSF IIS-096053, CNS-1205260, EFRI-1240459, and AFOSR FA9550-12-1-0238.

[2]All authors are with the Department of Computer Science, George Mason University, Fairfax, VA, USA, 22030

The most widely used MPEG 7 image set [21] consists of only shapes with single boundary. This limitation is further hindered by the lack of public domain implementations of many existing methods.

To address the aforementioned issues, in this paper, we propose a new decomposition method, called *Dual-space Decomposition* (or simply DUDE). DUDE is designed to handle complex 2D shapes that may be composed of overlapping objects with significant number of holes. DUDE accomplishes this by recognizing the importance of these holes by classifying holes as either topological noise and structurally important features. DUDE is developed based on the idea of nearly convex parts. Strategies that decompose shapes into nearly convex components have been proposed recently in the communities such as computer vision, pattern recognition and graphics [22, 33, 25, 29]. In order to recognize the importance features from the external and hole boundaries, DUDE creates the nearly convex decomposition of the shapes by segmenting both positive and negative regions of the shape (the polygon itself and its complementary).

In this paper, we propose an efficient dual-space method to decompose a non-convex polygon into several approximate convex parts, similar to human decomposition result which will be explained in following sections. Our method is designed to handle shapes with complex holes. To quantitatively evaluate DUDE, we take an initial step to collected more than 3800 segmentation from 142 non-expert using 72 polygons (shown in Figure 6). Using two statistics-based evaluation methods described in Section 5, we show that DUDE generates segmentation closer to man-made segmentation than existing methods.

2. Related Work

Polygon decomposition has been extensively studied theoretically and experimentally in areas including computer graphics, computational geometry [19], computer vision and pattern recognition.

In computational geometry, researchers are traditionally interested in creating decompositions subject to some optimization criteria, such as a minimum number of convex components [7, 13, 18, 8, 20]. Most of these problems are shown to be NP-hard [23, 18, 24]. More recently, several methods have been proposed to partition at salient features of a polygon. Tănase and Veltkamp [32] decompose a polygon based on the events that occur during the construction of a straight-line skeleton. Dey et al. [11] partition a polygon into collections of Delaunay triangles of points sampled from the polygon boundary. Lien and Amato [22] partition a polygon into *approximately convex* components. Their method reveals significant shape structures by recursively resolving the most concave features until the concavity of every component is below some user specified threshold. Wan [33] extends [22] to incorporate both concavity and curvatures and prevent over segmentation by avoid cuts inside pockets. Liu et al. [26] proposed the idea of α-decomposition that use persistence analysis of features obtained from the continuous convolution [3] between the polygon and a disc.

In pattern recognition and computer vision, shape decomposition is usually a step toward shape recognition. For instance, Siddiqi and Kimia [30] use curvature and region information to identify *limbs* and *necks* of a polygon and use them to perform decomposition. Recently, Liu et al. [25, 34] and Ren et al. [29] have been proposed to improve [22] to create fewer and more natural nearly convex shapes. Both methods [25, 29] use mutex pairs to enforce the concavity constraint. Points p_1 and p_2 form a mutex pair if their straight line connection is not completely inside the given shape. Their focus is on separating all mutex pairs with concavity-based weights larger than a user-specified threshold. Liu et al. [25] used linear programming to compute decomposition with minimum cost. Short cut rule is the only rule considered for evaluating the cost of a cut. Ren et al. [29] applied a dynamic subgradient-based branch-and-bound search strategy to get minimum number of cuts. Other than short cut rule, [29] also took minima rule into consideration which is that a cut resolving at positions with greater negative curvatures is preferred. Similarly, Juengling and Mitchell [17] formulate decomposition of a polygon as an optimization problem and applies dynamic programming to find the optimal subset of cuts from all possible cuts. The objective functions used for optimization favors short cuts that create dihedral angles close to π. Mi and DeCarlo [27] propose to decompose shape into elliptical regions glued by a hyperbolic patches.

An important requirement in shape decomposition is its robustness to boundary noise. Several of these methods require pre-processing (e.g., model simplification [17, 32]) or post-processing (e.g., merging over-partitioned components [11, 27]) due to boundary noise. Other methods [22, 25, 29] are designed to tolerate these noise. However, as far as we know, no existing approaches focused on handling topological noise that appear quite commonly in polygons generated from images with significant noise and overlapping objects.

3. Concavity

Concavity and the process of measuring it play the key role in our method (DUDE). To ease the concavity measuring process, we represent 2D shapes using polygons. Let us first define some notations used throughout the paper. A polygon P is represented by a set of n boundaries $\{P_0, P_1, \ldots, P_{n-1}\}$, where P_0 is the external boundary and $P_{k>0}$ are boundaries of *holes*. Each boundary consists of an ordered set of vertices $\{p_i\}$ which defines a set of edges. Each edge starts at vertex p_i as $e_i = \overline{p_i p_{i+1}}$ and has two

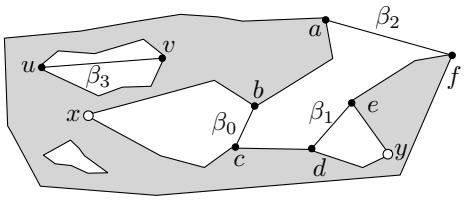


Figure 2. Bridges β_0, β_1, β_2, and β_3 and a concave feature x. The end points of β_3, u and v, are the antipodal vertices of the hole. Bridges β_0 and β_1 are the children of β_2.

associated vectors: the vector $\overrightarrow{v_i} = \overrightarrow{p_i\, p_{i+1}}$ and the outward normal $\overrightarrow{n_i}$. Because DUDE performs decomposition in both areas enclosed by P and, $\overline{P}$, the complement of P, it is worth noting that $\overline{P}$ can be obtained by reversing the ordering of $\{p_i\}$. In many cases, polygons are usually extracted from images by detecting the silhouettes, thus usually containing many geometric and topological noises.

Because DUDE relies heavily on the measure of concavity, let us now define the concept of bridge and pocket.

Definition 1. *A **bridge** β of a given polygon P is a segment $\beta = \overline{vu}$ connecting two points v and u on the boundary ∂P of P from the space exterior to P. More specifically, a segment $\overline{vu}$ is a bridge of P if and only if $v, u \in \partial P$ and the open set of $\overline{vu}$ is in the complement $\overline{P}$ of P, i.e. $\overline{vu}^\circ \subset \overline{P}$.*

Therefore, a bridge cannot enter P or intersect the boundary of P except at its end points. Examples of bridge are shown in Fig. 2. Note that, unlike bridges defined [22], this definition of bridge can be inside the convex hull of P.

Definition 2. *A **pocket** ρ of a bridge $\beta = \overline{vu}$ is a subset of the boundary ∂P connecting v and u so that the region enclosed by β and ρ is completely in $\overline{P}$.*

Intuitively, when traversing the boundary of P, a bridge can be viewed as a short cut over its pocket. For example, the pocket of the bridge β_0 in Fig. 2 is a polyline between vertices b and c via x.

The relationship between the bridge and pocket gives us an intuitive way to define concavity. For example, in the simplest case, the concavity of vertices in the pocket is simply the straight line distances to the bridge. A complete definition of concavity will be provided in Section 4 for the cases that the pocket is embedded in other pockets.

4. Dual Space Decomposition of Polygons

In dual-space decomposition (DUDE), the input polygon P is decomposed based on the decomposition of the complement $\overline{P}$ of P (*dual-space*). Algorithm 1 outlines the idea. The algorithm starts by determining the bridges and pockets from the convex hull $CH(P)$ of P. Algorithm 1 then proceeds by decomposing the regions $\overline{P_i}$ enclosed between the bridge β_i and pocket ρ_i pair. The cuts generated by these recursively calls DUAL-DECOMP($\overline{P_i}$, τ) are, by definition, bridges of P. Concavity is then measured, important concave features are identified using the bridges from $CH(P)$ and the cuts for $\overline{P_i}$. Finally, P is decomposed using these concave features. In the rest of this section, we will discuss each of these steps in detail. Figure 3 illustrates $\overline{P}$ is decomposed to find concave features of P.

Algorithm 1 Dual-Space Decomposition

```
1: procedure DUAL-DECOMP(P, τ)
2:     Compute convex hull CH(P) of P
3:     Determine bridges β = ∂CH(P) \ ∂P
4:     Determine pockets ρ from β
5:     for all P̄_i enclosed by ρ_i ∈ ρ and β_i ∈ β do
6:         κ = κ ∪ DUAL-DECOMP(P̄_i, τ)
7:     Measure concavity using β ∪ κ          ▷ see Section 4.1
8:     c = FEATURES(β ∪ κ, τ)                 ▷ see Section 4.2
9:     return CUT(P, c)                       ▷ see Section 4.3
```

4.1. Bridge Hierarchy

The bridges β are obtained from two sources: (1) the convex hull boundary (line 3 in Algorithm 1) and (2) the decomposition of the pockets (line 5 in Algorithm 1). First, we note that these bridges form a hierarchy. More specifically, we say that a bridge β is the parent of β' if the pocket ρ of β is contained in the pocket ρ' of β'. For example, in Fig. 2, bridges β_0 and β_1 are both kids of β_2. It is not difficult to show that this relationship of all bridges can be uniquely determined. That is, when two pockets overlap, one must enclose the other one. As a consequence, we can state the following theorem.

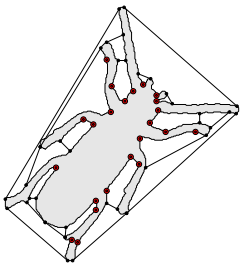


Figure 3. Decomposed $\overline{P}$.

Theorem 3. *The hierarchical relationship of bridges determined in Algorithm 1 must form a tree structure.*

As a result, each vertex of P has a unique path to the convex hull boundary of P. The length of this path determines the concavity of the vertex.

Note that a hole boundary is a pocket associated with only bridges from the decomposition. When a hole is nearly convex, there is no bridge associated with the hole. Therefore, a pseudo bridge is created by connecting the *antipodal pair*, i.e., two farthest apart vertices, of the hole (see β_3 in Figure 2). When a hole is decomposed by DUDE, bridges are formed from the cuts. Given k bridges, we can form k bridge hierarchies by using each cut as the root. We choose the hierarchy with lowest depth to measure the concavity of vertices in the hole.

Our objective in measuring the concavity is to identify pockets that have large intolerable concavity. Intuitively, the concavity of a pocket is the largest concavity of its vertices.

Definition 4. *For a pocket ρ without children, we define:*

$$\text{concavity}(\rho) = \max_{v' \in \rho} (\text{dist}(v', \beta)) \ . \tag{1}$$

Otherwise, concavity(ρ) is defined as:

$$\max_{\rho' \in C(\rho)} (\text{concavity}(\rho') + \text{dist}(\beta', \beta)) \ , \tag{2}$$

where $C(\rho)$ is the children of ρ, β' is the bridge of ρ', and $\text{dist}(x, y)$ is the shortest Euclidean distance between objects x' and y.

We say that ρ an *intolerable pocket* if concavity(ρ) is greater than τ. Finally, we would like to point out that this hierarchical definition of concavity provides better accuracy and efficiency over the traditional concavity measurement computed directly between vertices and the bridge using either the (fast but inaccurate) straight-line distance or the (accurate but slow) shortest-path distance [22].

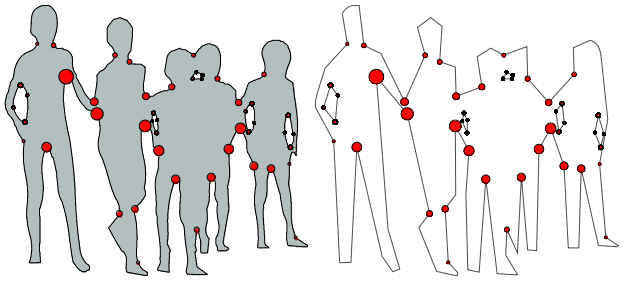
4.2. Detect Intolerable Concave Features

Given an intolerable pocket ρ either from the external boundary or a hole boundary of P and the complement polygon $\overline{P}$ enclosed by ρ and its associated bridge, we can determine the intolerable concave features of ρ by finding the smallest CH' approximation of the convex hull $CH(\overline{P})$ of $\overline{P}$ such that the distance from CH' to $CH(\overline{P})$ is smaller than τ. Using the idea similar to the variational shape approximation [10], we then obtain an optimal approximation CH'_i of $CH(\overline{P})$ with i vertices in iteration i. We start from two vertices, i.e. $i = 2$, in the approximation and iteratively add a vertex to the approximation to obtain a better approximation until the two-way Hausdorff distance between CH'_i of $CH(\overline{P})$ is less than τ. The vertices in CH'_i are the intolerable concave features. An example of the intolerable concave features is shown in Figure 4(a).

Because the complement polygon $\overline{P}$ of ρ must be nearly convex (otherwise $\overline{P}$ is decomposed), it is provable that all the intolerable concave vertices of ρ must be on the convex hull $CH(\overline{P})$ of $\overline{P}$ [14]. In addition, by determining the smallest approximation of $CH(\overline{P})$ with bounded Hausdorff distance, the method above provides the minimum number of intolerable concave features .

4.3. Decompose at Intolerable Concave Features

In this section, we describe how the intolerable concave features of a given polygon P are connected into *cuts* that segment P into nearly convex components. Our method starts off by determining the *resolvers* of each concave feature. A resolver of a concave feature v is a set of diagonals



(a) (b)

Figure 4. (a) Polygon with identified concave features. Size of the circle indicates the significance of the feature. (b) Simplified polygon using the concave features.

of P that can locally reduce the concavity of v to τ or less. Once all resolvers from P are identified and evaluated, our method then proceeds to determine a set of resolvers that maximizes the total scores subject to the constraints that no conflicting resolvers are selected.

In the rest of this section, we will discuss how a resolver is defined and identified in Section 4.3.1. We then address the optimization problem by solving *0-1 integer linear programming* in Section 4.3.2.

4.3.1 Resolvers of Intolerable Concave Features

A resolver of an intolerable concave feature v consists of a set of diagonals incident to v. We say that a diagonal d is valid for a given concave feature v if (1) d is in the interior of P and (2) the end points of d belong to different pockets.

Given a polygon P, valid diagonals are determined using the Constrained Delaunay Triangulation (CDT) of both P and the simplified polygon $\tilde{P}$. The simplified polygon $\tilde{P}$ is composed of intolerable concave features and the vertices between two consecutive bridges. Essentially, $\tilde{P}$ is P with all pocket vertices replaced by the intolerable concave features. An example of $\tilde{P}$ is shown in Figure 4(b). To find all valid diagonals, we first compute the $\text{CDT}_{\tilde{P}}$ of $\tilde{P}$. Let $\text{D}_{\tilde{P}} \subset \text{CDT}_{\tilde{P}}$ be a set of valid diagonals incident to concave feature of P. We then compute the CDT_P using the edges of P and the diagonals in $\text{D}_{\tilde{P}}$ as constraints. Let $\text{D}_P \subset \text{CDT}_P$ be a set of valid diagonals incident to concave feature of P. The valid diagonals $\text{D}_{\tilde{P}} \cup \text{D}_P$ are candidates for resolvers.

We say that a set of valid diagonals D can locally resolve a concave feature v if the *local residual concavities* at v are all less than τ. These local residual concavities at v due to D can be computed using only D and the edges of $\tilde{P}$ incident to v. Then, a resolver $\mathcal{R}$ of v is simply a *minimum set of valid diagonals* that can resolve v. An example of resolvers for vertices v and x is shown in Fig. 5.

The resolvers from a vertex v are mutually exclusive because a single resolver, by definition, can resolve v. As a consequence, no two resolvers of v should be selected in

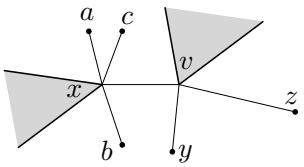


Figure 5. Resolvers of vertices x and v. The resolvers of vertex v include $\{\overline{vx}, \overline{vz}\}$ and $\{\overline{vy}\}$. The resolver $\{\overline{vx}, \overline{vz}\}$ conflicts with the resolver $\{\overline{vy}\}$ and all resolves of x.

the final decomposition. To ease our discussion, we define the complement $\overline{\mathcal{R}_i}$ of a resolver $\mathcal{R}_i$ be a set of mutually exclusive (conflicting) resolvers. For example, in Fig. 5, the complement of resolver $\{\overline{vy}\}$ (for vertex v) is the resolver $\{\overline{vx}, \overline{vz}\}$. See the caption of Fig. 5.

4.3.2 Final Cut Selection

Cut selection in dual decomposition is an optimization problem that maximizes the total score of the selected cuts subject to the constraints that no conflicting resolvers are selected and each concave feature requires a resolver. We will now show that this optimization problem can be solved using 0-1 integer linear programming.

Let $\mathcal{R}$ be a resolver and $\overline{\mathcal{R}}$ be the conflicting resolvers of $\mathcal{R}$. The score $S(\mathcal{R})$ of a resolver $\mathcal{R}$ is a function of (1) the number of features resolved by $\mathcal{R}$, (2) the concavity of resolved features and (3) the total length of $\mathcal{R}$. Our goal is to select a subset $\mathcal{K}$ of all resolvers $\mathbf{R}$ so that $\sum_{\mathcal{R}\in\mathcal{K}} S(\mathcal{R})$ is maximized subject to the following constraints (1) $||\mathcal{K}|| \leq n$ and (2) $\mathcal{K} \cap \overline{\mathcal{R}_i}$ is empty for all $\mathcal{R}_i \in \mathcal{K}$. To formulate this problem as a 0-1 integer linear programming, we let $\mathbf{R}$ be a vector of all resolvers and $\mathbf{x}$ be a 0-1 vector of size $||\mathbf{R}||$. Then the final cuts are obtained by solving the following linear programming problem:

$$\text{maximize} \qquad S(\mathbf{R}^{\mathrm{T}}\mathbf{x}) \tag{3}$$

$$\text{subject to} \qquad \sum_{x_i\in\mathbf{x}} x_i \leq n\ , \tag{4}$$

$$\text{and} \quad x_i + x_j \in \{0,1\},\ \forall(\mathcal{R}_j \in \overline{\mathcal{R}_i})\ , \tag{5}$$

where n is the number of intolerable concave features.

5. Experimental Results and Evaluation

We have implemented DUDE in C++. To evaluate DUDE, we use 72 shapes shown in Fig. 6. To provide a baseline for comparison, we collected 3818 segmentation from 142 human subjects. Many shapes are from the widely used MPEG7 database. All the images from MPEG7 contain only a single boundary. To test the polygons with many holes, we add synthesized noise to MPEG7 images and use silhouette images, such as trees, human crowd, and bikes. In

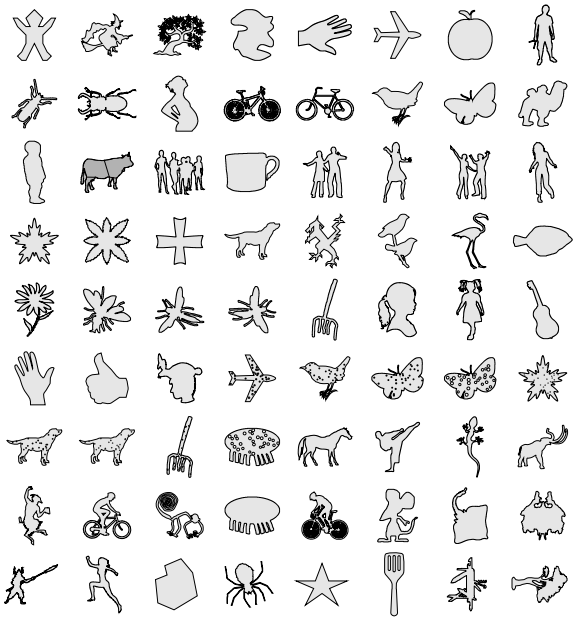


Figure 6. Shapes used in our experiments

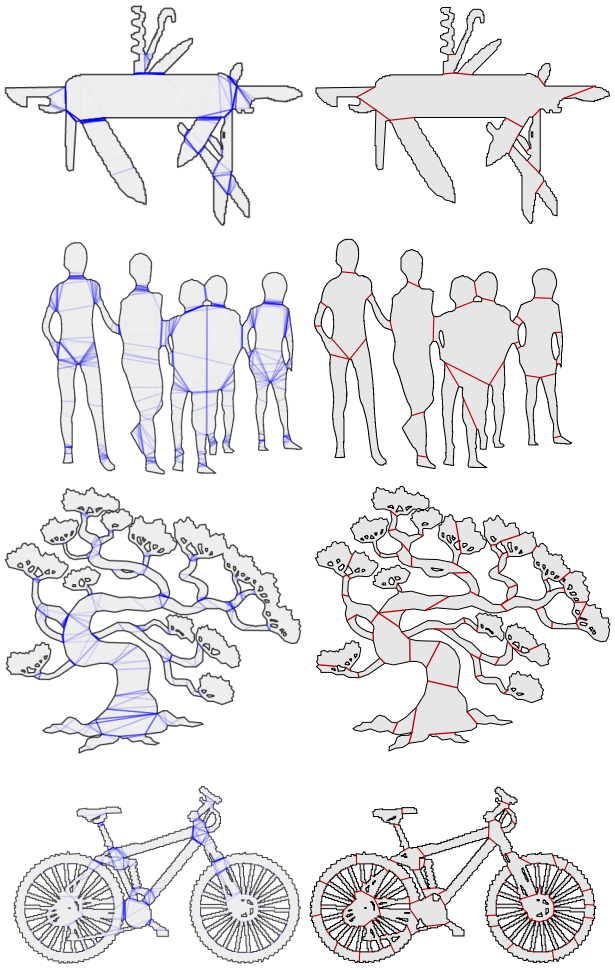
all experiments, the value of concavity tolerance τ is fixed for both DUDE , MNCD [29] and ACD [22]. The segmentation collected from human, DUDE, MNCD and ACD are then compared using two statistics-based evaluation methods (detailed in Section 5.2). The average running time for DUDE is less than 2 seconds on a regular laptop. Example output can be found in Figures 7 and 8. Complete human segmentation results and collecting software tools are available to the public at *http://masc.cs.gmu.edu*.

5.1. Collect Human Segmentation

We used Amazon Mechanical Turk to collect human segmentation. Non-expert users were asked to draw lines to segment the displayed polygon into *meaningful* parts via a web-based interface. Several good and bad examples were shown before the user starts. Constraints were added to prevent user making segments outside or intersecting with the polygon or other existing cuts. There were in total 142 people participated, and we obtained 3818 valid segmentation. On average, each user spent around 90 seconds to segment a polygon, and made about 8.12 cuts for a polygon. To visualize the segmentation, we overlay user created segments, each of which is drawn translucent therefore the most frequently picked segments are darker. Examples of human segmentation can be found in Figures 7 and 8.

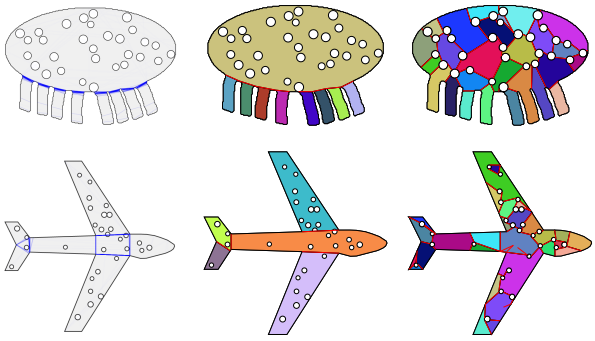
5.2. Evaluate Part Segmentation

Although few works focusing on evaluating part segmentation, evaluating image segmentation has been widely investigated (see a survey in [9]). In this section, we propose two statistical methods. One is based on *Rand Index* (RI)



(a) Human (b) DUDE

Figure 7. Differences between human segmentation and DUDE.



(a) Human (b) DUDE (c) ACD

Figure 8. Decomposition results of the aggregated human cuts, DUDE and ACD.

[28] and the other one is called *Cluster Coverage* (CC).

To perform RI evaluation, we first obtained the representative human segments via k-means clustering for each polygon. Here k is determined by the mean number of cuts created by human for a given polygon.

Next, we overlay the decomposition with a 200×200 regular grid tightly bounding the polygon P. A cell is considered as valid if it has more than half of its area inside P. For a given segmentation, each component in the segmentation is assigned a unique index and the cells enclosed by this component are assigned the same index number. RI evaluation of two segmentations (of the same polygon) is measured by the likelihood that each pair of cells is in the same component of two segmentations [28]. More specifically, let S_1 and S_2 be two segmentations, and s_i^1 and s_i^2 be the indices of cell i in S_1 and S_2, respectively. Then their RI is defined as:

$$\mathrm{RI}(S_1, S_2) = \binom{2}{N}^{-1} \sum_{i,j,i<j} \left[C_{ij}P_{ij} + (1-C_{ij})(1-P_{ij})\right],$$

where N is the number of cells inside the original polygon, $C_{ij} = 1$ if $s_i^1 = s_j^1$, $P_{ij} = 1$ if $s_i^2 = s_j^2$, $C_{ij}P_{ij} = 1$ if cell i and cell j have the same index in both S_1 and S_2, and, finally, $(1 - C_{ij})(1 - P_{ij}) = 1$ indicates the cell i and j have different indices.

A major drawback of the RI-based evaluation is that the size of the representative cuts can affect the RI value dramatically. Therefore, we propose another evaluation method called *Cluster Coverage*. A Cluster Coverage (CC) value between a segmentation S and clusters C of segmentation created by the human subjects is determined in the following way:

$$\mathrm{CC}(S, C) = \frac{\sum_i^k W_i f_i}{\sum_i^k W_i},$$

where W_i is the number of elements in C_i and f_i is an indicator function that returns one if $\exists S_j \in C_i$.

The RI and CC values in Table 1 are obtained from the evaluations between segmentations created by a given method and the representative human segmentations over all the polygons shown in Fig. 6. Therefore, for both RI and CC evaluations, higher score indicates higher similarity to the representative human segmentations. Human vs. Human comparison is used as the baseline for our comparison. From Table 1, we see that the average RI value of DUDE is only slightly below that of Human, while higher than those from MNCD and ACD. This means DUDE provides segmentations closer to the aggregated human segmentations. The average CC value of Human is only slightly below that of DUDE while the standard deviation of DUDE is slightly higher. The CC evaluation again confirms that DUDE provides segmentation closer to the aggregated human segmentation than MNCD and ACD.

5.3. Compare with CSD and MNCD

Both CSD [25] and MNCD [29] generate candidate cuts and mutex pairs using Reeb graphs from multiple rotations.

Table 1. Mean values of Rand Index (RI) and Cluster Coverage (CC) and their standard deviations SDV_{RI} and SDV_{CC}.

Method	Human	DUDE	MNCD	ACD
RI	0.73	0.68	0.46	0.39
SDV_{RI}	0.12	0.21	0.17	0.36
CC	0.67	0.77	0.73	0.40
SDV_{CC}	0.094	0.15	0.26	0.38

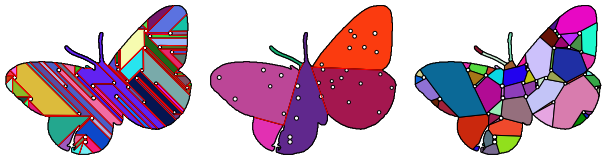


Figure 9. Decomposition result of MNCD and DUDE. From left to right: MNCD ($\tau = 0.1$), DUDE ($\tau = 0.1$), DUDE ($\tau = 0.01$).

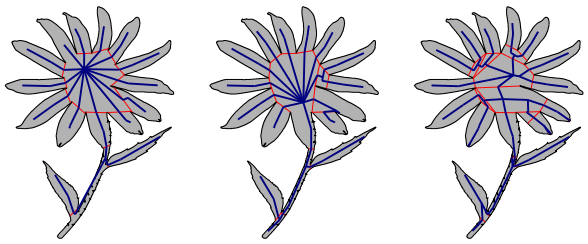
The major difference between DUDE and these methods is that DUDE can handle holes (that appear frequently in shapes of many overlapping objects) much more naturally. Figure 9 shows the difference between the decompositions of MNCD and DUDE. Consequently, as we have also seen from Table 1, DUDE can generate more similar results than MNCD compared to human segmentations. The reasons of lower RI and CC scores may also be coming from the fact that there are insufficient optimal candidate cuts in their methods that are generated from intersection of scanning lines and polygon boundary. Moreover, for polygon with complex holes, MNCD requires parameters to balance between the visual naturalness and the number of cuts (while DUDE requires only the value τ). The naturalness may be sensitive to the local noise. Finding good values for these parameters is usually not easy.

6. Conclusion and Discussion

In this paper, we proposed a new method: *Dual-space Decomposition* (DUDE) to do 2D shape decomposition, which has applications in region/part-based recognition, skeleton-extraction (Fig. 10), *etc*. DUDE is designed to segment shape composed overlapping objects with a significant number of holes. Using two evaluation methods: Rand Index and Cluster Coverage, we show that segmentation created by DUDE is statistically more similar to the man-made segmentation than those created by other methods.

Since DUDE is developed purely based on the geometric properties, its ability to segment meaningful part is limited. For example, our data shows that people are good at segmenting shapes composed of humans and animals. When a figure contains human shape and other non-human objects, people prefer to separate human from the models.

Although there has been many methods recently proposed in segmenting 3D shapes, 2D part segmentation has its own difficulty because 2D shapes tend to overlap thus



(a) Human (b) DUDE (c) ACD

Figure 10. Skeleton extracted from different methods

create ambiguity. When (either human or non-human) shapes overlap, people also tend to segment each individual object or human from the group, even when the cut is long (e.g., the crowd shape in Figure 7). Moreover, different people also assign the overlapping area to different parts. More people tend to assign the overlapping area to an object if it has larger ratio of non-overlapping parts in the whole shape rather than other objects that share this overlapping area. For example, in the bike polygon in Figure 7, when the frame has overlapping area with wheel, more people tend to assign the area to wheel instead of frame.

References

[1] D. Attali, P. Bertolino, and A. Montanvert. Using polyballs to approximate shapes and skeletons. In *ICPR94*, pages 626–628, 1994.

[2] O. E. Badawy and M. Kamel. Shape representation using concavity graphs. *ICPR*, 3:461–464, 2002.

[3] E. Behar and J.-M. Lien. Dynamic minkowski sums under scaling. *Computer-Aided Design*, Nov. 2012.

[4] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[5] G. Borgefors and G. S. di Baja. Methods for hierarchical analysis of concavities. In *Proceedings of the Conference on Pattern Recognition (ICPR)*, volume 3, pages 171–175, 1992.

[6] G. Borgefors and G. S. di Baja. Analyzing nonconvex 2d and 3d patterns. *Computer Vision and Image Understanding*, 63(1):145–157, 1996.

[7] B. Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 339–349, 1982.

[8] B. Chazelle and D. P. Dobkin. Optimal convex decompositions. In G. T. Toussaint, editor, *Computational Geometry*, pages 63–133. North-Holland, Amsterdam, Netherlands, 1985.

[9] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3d mesh segmentation. *ACM Transactions on Graphics (TOG)*, 28(3):73, 2009.

[10] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914, 2004.

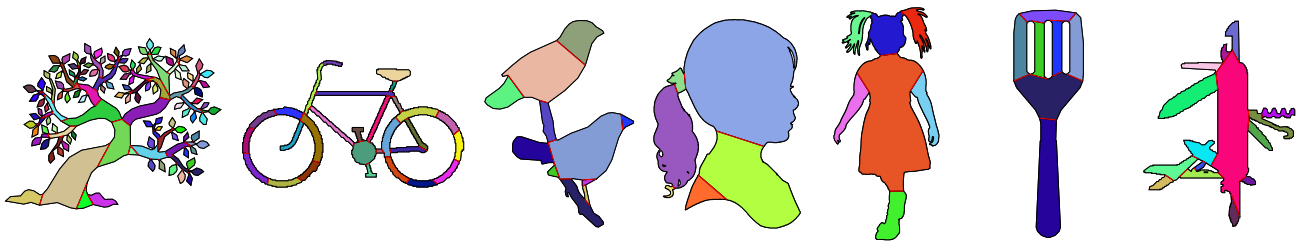


Figure 11. Decomposition results of DUDE for polygons with many holes.

[11] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In *Proc. Workshop on Algorithms and Data Structures*, pages 25–36, 2003.

[12] H. Y. F. Feng and T. Pavlidis. Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition. *IEEE Trans. Comput.*, 24:636–650, June 1975.

[13] D. H. Greene. The decomposition of polygons into convex parts. In F. P. Preparata, editor, *Computational Geometry*, volume 1 of *Adv. Comput. Res.*, pages 235–259. JAI Press, Greenwich, Conn., 1983.

[14] J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line simplification algorithm. In *Proc. 5th Internat. Sympos. Spatial Data Handling*, pages 134–143, 1992.

[15] D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[16] D. Hoffman and M. Singh. Salience of visual parts. *Cognition*, 63:29–78, 1997.

[17] R. Juengling and M. Mitchell. Combinatorial shape decomposition. In *Proceedings of the 3rd international conference on Advances in visual computing-Volume Part II*, pages 183–192. Springer-Verlag, 2007.

[18] J. M. Keil. Decomposing a polygon into simpler components. *SIAM J. Comput.*, 14:799–817, 1985.

[19] J. M. Keil. Polygon decomposition. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 491–518. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[20] M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. In M. Soss, editor, *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 54–55, Montréal, Québec, Canada, 1998. School of Computer Science, McGill University.

[21] L. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 424–429. IEEE, 2000.

[22] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polygons. *Comput. Geom. Theory Appl.*, 35(1):100–123, 2006.

[23] A. Lingas. The power of non-rectilinear holes. In *Proc. 9th Internat. Colloq. Automata Lang. Program.*, volume 140 of *Lecture Notes Comput. Sci.*, pages 369–383. Springer-Verlag, 1982.

[24] A. Lingas, R. Pinter, R. Rivest, and A. Shamir. Minimum edge length partitioning of rectilinear polygons. In *Proc. 20th Allerton Conf. Commun. Control Comput.*, pages 53–63, 1982.

[25] H. Liu, W. Liu, and L. Latecki. Convex shape decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 97 –104, june 2010.

[26] Y. Lu, J.-M. Lien, M. Ghosh, and N. M. Amato. α-decomposition of polygons. *Computer & Graphics*, SMI 12 special issue, 2012.

[27] X. Mi and D. DeCarlo. Separating parts from 2d shapes using relatability. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[28] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[29] Z. Ren, J. Yuan, C. Li, and W. Liu. Minimum near-convex decomposition for robust shape representation. In *Proc. of ICCV*, 2011.

[30] K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):239–251, 1995.

[31] M. Singh, G. Seyranian, and D. Hoffma. Parsing silhouettes: The short-cut rule. *Perception & Psychophysics*, 61:636–660, 1999.

[32] M. Tănase and R. C. Veltkamp. Polygon decomposition based on the straight line skeleton. In *Proceedings of the nineteenth conference on Computational geometry (SoCG)*, pages 58–67. ACM Press, 2003.

[33] L. Wan. Parts-based 2d shape decomposition by convex hull. In *Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on*, pages 89–95. IEEE, 2009.

[34] W. L. Zhou Ren, Junsong Yuan. Minimum near-convex shape decomposition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 35, pages 2546–2552, 2013.