Free Hand-Drawn Sketch Segmentation

Zhenbang Sun[1,⋆], Changhu Wang[2], Liqing Zhang[1], and Lei Zhang[2]

[1] Brain-Like Computing Lab, Shanghai Jiao Tong University, P.R. China
[2] Microsoft Research Asia

Abstract. In this paper, we study the problem of how to segment a freehand sketch at the object level. By carefully considering the basic principles of human perceptual organization, a real-time solution is presented to automatically segment a user's sketch during his/her drawing. First, a graph-based sketch segmentation algorithm is proposed to segment a cluttered sketch into multiple parts based on the factor of *proximity*. Then, to improve the ability of detecting semantically meaningful objects, a semantic-based approach is introduced to simulate the *past experience* in the perceptual system by leveraging a web-scale clipart database. Finally, other important factors learnt from past experience, such as *similarity*, *symmetry*, *direction*, and *closure*, are also taken into account to make the approach more robust and practical. The proposed sketch segmentation framework has ability to handle complex sketches with overlapped objects. Extensive experimental results show the effectiveness of the proposed framework and algorithms.

1 Introduction

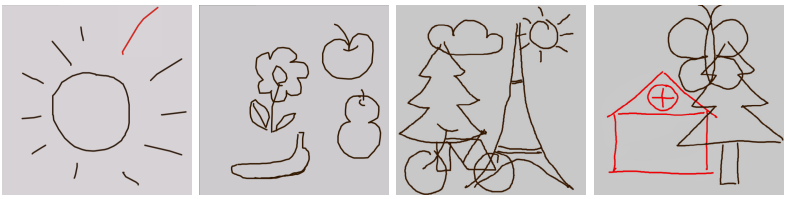
In Marr's primal sketch theory [1], the importance of a sketch in the early processing of human visual information has been particularly emphasized. Sketch analysis not only is a fundamental problem in human perception, but also plays an important role in human-computer interaction. For example, a deeper understanding to a hand-drawing on a tablet or a gesture before a Kinect camera, is highly desired for a more natural communication between human and computers.

However, although it looks fairly easy for human to recognize a sketch as shown in Fig. 1, it still presents a great challenge for a computer to recognize. Existing work on sketch/shape retrieval [2, 3] and recognition [4, 5] is usually based on the assumption that the sketch/shape query only contains a single object, and seldom considers a complex sketch/shape without segmentation. As shown in Fig. 1(a), without any prior information, it is not easy for a computer to judge whether the sketch is a sun or a circle enclosed by some sticks.

In this work, we study the sketch segmentation problem, which is a primary step for further sketch analysis and other computer vision tasks. In spite of continuous efforts in the last decade, most existing work about the so-called "sketch segmentation" [6–8] mainly targets at segmenting a sketch at the stroke level rather than at the object level, and partitions each stroke into basic geometric

⋆ Zhenbang Sun performed this work while being an intern at Microsoft Research Asia.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part I, LNCS 7572, pp. 626–639, 2012.



(a) (b) (c) (d)

Fig. 1. Example sketches. Although it looks fairly easy for human to recognize a sketch, it still presents a great challenge for a computer to recognize it.

components, such as straight lines, circles, and arcs. [7] and [8] leveraged the direction and curvature of strokes to detect the vertices of basic components. [6] transformed the problem of partitioning strokes into the operation of fitting an implicit function to the strokes. These works were all adopted as the first step of domain-specific sketch understanding [9, 10].

According to human perception evidences in Gestalt theory [11], two levels of factors should be considered in sketch segmentation: low-level perception and high-level knowledge. As for the low-level aspect, *proximity* [11] is the most basic factor for perceptual grouping. When a person is asked to segment the sketch in Fig. 1(b), even without any prior knowledge to fruits or flowers, he/she can also successfully separate these objects, since they are far from each other. In the meantime, the high-level knowledge plays a crucial role in human perceptual grouping. For example, a person can easily recognize the bike from the background in Fig. 1(c). Besides the knowledge to a specific object such as the car, some common knowledge learnt from past experience can also help human make decisions in the face of unfamiliar objects. For example, as shown in Fig. 1(a), if we add another ray (in red) to a sun, it could be easily predicted that the newly added curve is only one part of the sun, because it does not change too much to the sun. Another example is shown in Fig. 1(d). It is very likely that the candidate (in red) is an object, because of its *symmetry*.

In this paper, a novel sketch segmentation framework is proposed by leveraging both the low-level perception and high-level knowledge. First, the *proximity-based sketch segmentation* is proposed, in which the distance between strokes is used to measure the *proximity*. Then, the *semantic-based sketch segmentation* is presented, in which the high-level knowledge comes from a large-scale clipart database. The basic idea is to ask the knowledge base to confirm whether a group of strokes is an object in "memory" or not. Since the process of the segmentation (select a group of strokes as the query) and recognition (confirm whether it is a meaningful object) is generally a chicken-egg problem, a *robust entropy descent merging* algorithm is proposed to judge whether the group of strokes represent an object, followed by a *greedy backward segmentation* algorithm for segmentation. Furthermore, to make the algorithms more robust and efficient, this framework incorporates some of the past experience into intuitive rules such as *symmetry* and *closure* by designing a classifier for segmentation. Moreover, the particular

consideration to the drawing order of strokes in the algorithms, makes it possible to conduct real-time analysis during a user's drawing.

As far as we know, this is the first work to segment a freehand sketch at the object level. Various experiments are performed to show the effectiveness and efficiency of the proposed framework.

2 Object-Level Sketch Segmentation

In this section, we first introduce the proximity-based segmentation. Then the semantic-based segmentation is presented in detail, followed by a merging strategy guided by intuitive clues. Finally, the sketch segmentation framework is developed based on the two levels of perceptual factors.

2.1 Proximity-Based Sketch Segmentation

In Gestalt theory, *proximity* and *similarity* are the most predominant factors to influence the natural groupings of human beings: a person tends to group together nearby items with similar appearances.

These two factors have been widely used in image segmentation tasks [12–15]. However, different from image segmentation, in which multiple clues such as intensity and color could be leveraged, in sketch segmentation, a sketch is usually lack of such information. Thus, we only use the factor of *proximity* to conduct the segmentation in the stroke level rather than pixel level. Moreover, the graph segmentation strategy should also be adapted for stroke elements.

In the following part, the Efficient Graph-based Image Segmentation (*EGIS*) algorithm [12] is first briefly introduced, based on which a sketch segmentation algorithm is then presented.

Efficient Graph-Based Image Segmentation. Let $G = (V, E)$ denote an undirected graph to represent an image, in which the vertices $v \in V$ represent the pixels in the image. Each edge $(v_i, v_j) = e \in E$ has a weight $w(v_i, v_j)$ to indicate the dissimilarity between pixels. A segmentation is to partition V into segments $\{S\}$, in which each segment S corresponds to a subgraph $G' = (S, E')$ where $E' \subseteq E$.

In *EGIS*, the *internal difference* of a segment S is defined as the largest weight in the minimum spanning tree $MST(S, E')$ of the subgraph $G' = (S, E')$, i.e. $Int(S) = \max_{e \in MST(S,E')} w(e)$.

Then, the *minimal internal difference* of S_1 and S_2 is also defined by appending a tolerance $\tau(S)$ for each segment S:

$$MInt(S_1, S_2) = \min(Int(S_1) + \tau(S_1), Int(S_2) + \tau(S_2)), \tag{1}$$

in which $\tau(S) = k/|S|$, where $|S|$ denotes the size of S and k is a parameter.

The *segment distance* between S_1 and S_2 is defined as the minimal edge weight between the two segments:

$$Dist(S_1, S_2) = \min_{v_i \in S_1, v_j \in S_2, (v_i, v_j) \in E} w(v_i, v_j). \tag{2}$$

Fig. 2. Illustrations of the problems in the *MEGIS* algorithm. Middle column: the input sketches. Left column: segmentation results of the proposed proximity-based algorithm (*Proximity*). Right column: segmentation results of the *MEGIS* algorithm.

Therefore, the criterion to merge two segments is that, if $Dist(S_1, S_2) \leq MInt(S_1, S_2)$, these two segments will be merged together.

The algorithm begins from initializing each pixel as a segment. Then two segments with the smallest *segment distance* will be merged if satisfying the merging criterion. If successful, repeat the merging process for the segments with the next smallest *segment distance*. The algorithm will stop if two segments cannot be merged. Then, the remaining segments are the results of the segmentation.

Graph Construction for Sketch Segmentation. In the sketch segmentation task, the vertices $v \in V$ represent the strokes of the drawing, and the weight between two strokes $w(v_i, v_j)$ is just the minimal distance between them.

Based on the graph-based representation of the sketch, the *EGIS* algorithm is naturally adapted for solving the problem of sketch segmentation, which is called as the *modified EGIS* (*MEGIS*) algorithm. Notice that the $|S|$ in the tolerance in Eqn. 1 is defined as the total length of the strokes in the segment S divided by the maximal side of the sketch panel.

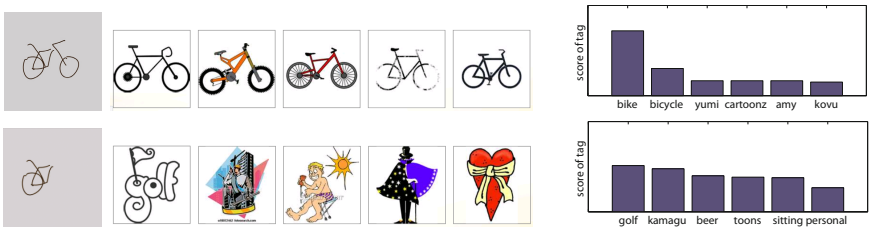
Modified Graph Segmentation Criterion. The segmentation results of the *MEGIS* algorithm are still far from satisfactory. As shown in Fig. 2, the algorithm tends to separate a short stroke from an object (a), and also fails when some components are enclosed by a major component in the same object (b).

The first problem is caused by equally treating the two segments when calculating the *minimal internal difference* in Eqn. 1. Actually, in sketch segmentation, a short stroke is probably a part of a large segment nearby regardless of some distance between them. Thus, Eqn. 1 is modified to:

$$MInt(S_1, S_2) = \min(Int(S_1), Int(S_2)) + \max(\tau(S_1), \tau(S_2)), \tag{3}$$

which relaxes the merging threshold if one segment is very small: τ will be large for a short stroke.

The second problem results from the change of the graph representation. Different from the simple spacial relationship between two pixels in image segmentation, the proximity between two strokes is more complex. As shown in Fig. 2(b), the algorithm should merge two segments together if one is predicted to be "on" or enclosed by the other one. Thus, when segmenting the graph, the *enclosure correlation* between two segments is also considered:



(a) (b) (c)

Fig. 3. Top results of the sketch-based image search from one-million clipart image database. (a) The hand-drawn sketches. (b) Top image search results of the sketch queries. (c) Word distributions of the top results: unbalanced for a meaningful sketch, while more flatter for a meaningless sketch.

$$Corr(S_1, S_2) = \frac{|C_{S_1} \cap C_{S_2}|}{\min\{|C_{S_1}|, |C_{S_2}|\}}, \tag{4}$$

in which C_S is the convex hull of the segment S and $|C_S|$ is its area. $C_{S_1} \cap C_{S_2}$ means the overlap region between C_1 and C_2.

This item is also added to the *minimal internal difference* to ensure that highly overlapped segments will be merged together:

$$MInt(S_1, S_2) = \min(Int(S_1), Int(S_2)) + \max(\tau(S_1), \tau(S_2)) + \lambda Corr(S_1, S_2). \tag{5}$$

The modified algorithm is named as *proximity-based sketch segmentation*. It is used as the first step in our segmentation framework to segment separable sketches. To separate overlapped parts into meaningful objects, further analysis beyond low-level perception is needed.

2.2 Semantic-Based Sketch Segmentation

The high-level knowledge is necessary for human to separate an object from a complex background. To simulate this process, one million clipart images crawled from the web is adopted as a knowledge base. Then, a *robust entropy descent merging* algorithm is proposed to decide whether to merge two segments. Finally, a *greedy backward segmentation* algorithm is introduced for sketch segmentation.

The Knowledge Base. We collected one million clipart images from the web, and preserved the textual information of each image, such as image title and surrounding texts. Most of these images were created by human, and thus their contours have a similar style to the hand-drawn sketches.

We follow the technology of Edgel Index [2, 16] and build a sketch-based clipart image search engine [17] to explore this knowledge base. For a sketch segment submitted to this engine, the most similar images to the sketch will be returned, as shown in Fig. 3. For details please refer to [2, 17].

Robust Entropy Descent Merging. From Fig. 3 we can see that, for a meaningful sketch, the top results are usually relevant images; whereas for a meaningless sketch, the objects in the top results usually become diverse. In the view of information theory, the information entropy of a meaningful object is usually lower than that of a non-object. This motivates us to leverage the entropy of the concepts in the top results to decide whether to merge two segments.

The Sketch Entropy
Assume that for a sketch '$Sketch$', there are N images $\{I_1, I_2, \ldots, I_N\}$ returned from the engine. Let $W = \{w_1, w_2, \ldots, w_M\}$ be M unique words in the textual information of the N images. We use the following equation to calculate the word distribution of the sketch:

$$Pr(w) = \frac{Score(w|Sketch)}{\sum_{w \in W} Score(w|Sketch)}, \tag{6}$$

in which the score between a word w and the sketch is given by:

$$Score(w|Sketch) = \sum_{n=1}^{N} \delta(w, I_n) \times Sim(Sketch, I_n), \tag{7}$$

where $\delta(w, I_n)$ is 1, if the word w appears in the description of image I_n; and 0, otherwise. $Sim(Sketch, I_n)$ is the similarity between the sketch and image I_n, which is provided by the search engine.

Thus, the *sketch entropy* is defined as:

$$H = \sum_{w \in W} -Pr(w) \log Pr(w). \tag{8}$$

To further illustrate the difference between object entropies and non-object entropies, we compared the entropies of 100 object/non-object pairs[1]. As shown in Fig. 4(a), most entropies of objects are lower than those of non-objects.

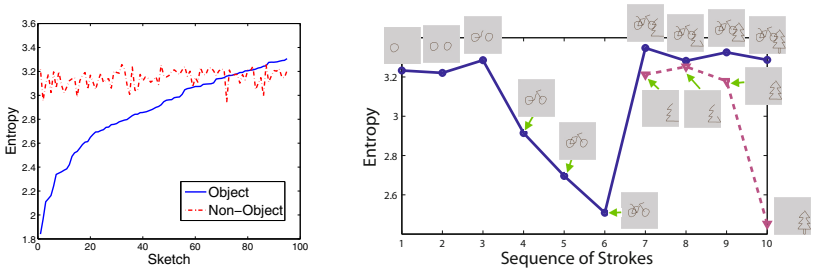
Fig. 4(b) illustrates the entropy change when drawing a bicycle and a tree. We can see that, the entropies of the bicycle and the tree reached local minimums.

Robust Entropy Descent Merging
Thus, we leverage sketch entropy to judge which segment is more meaningful. For sketch segmentation, we can merge two segments S_1 and S_2, if their entropies are larger than that of the merged sketch, i.e. $H(S_1 \cup S_2) < \min\{H(S_1), H(S_2)\}$.

In practice, adding a short stroke to an object might cause small entropy fluctuation. To be more robust, we also take into account the difference of word distributions. First, the KL-divergence[2] between the word distributions of two sketches is defined as:

[1] The 100 sketches of objects were randomly sampled from the Sketch500 database (see the experiments), and the 100 non-object sketches were collected by randomly adding or deleting some strokes based on the above 100 sketches of objects.

[2] The KL-divergence is not symmetric. Thus we use $Sketch_2$ to represent a newer stroke than $Sketch_1$ for differentiation.



(a) (b)

Fig. 4. The entropy of sketches. (a) The entropy comparison between 100 object/non-object pairs. (b) The entropy curve of all steps when drawing a bicycle and a tree. The entropies of the tree with different strokes are also illustrated.

$$D_{KL}(Sketch_1, Sketch_2) = \sum_{w \in W_1} Pr_1(w) \log \frac{Pr_1(w)}{Pr_2(w)}. \tag{9}$$

Then, the reciprocal of the KL-divergence is added to the merging criterion. It ensures that a small stroke will not make the sketch entropy increase if the word distribution does not change too much. That is, S_1 and S_2 will be merged together, if $H(S_1 \cup S_2) < \min\{H(S_1) + \frac{\beta}{D_{KL}(S_1, S_1 \cup S_2)}, H(S_2) + \frac{\beta}{D_{KL}(S_2, S_1 \cup S_2)}\}$. Here β is set to 0.3 empirically.

Greedy Backward Segmentation Algorithm. Based on the merging strategy, a greedy algorithm is proposed to segment a sketch into meaningful objects.

The order of the drawing is leveraged in the segmentation process. The basic idea is that, for each newly drawn stroke, we first check whether it could be merged with all existing segments to be one segment. If unsuccessful, we will exclusive the oldest segment, and check again, until the merging criterion is reached. The detailed algorithm is listed in Algorithm 1.

In the worst case, the algorithmic complexity is $\mathcal{O}(n^3)$, where n is the stroke number of the drawing. This needs to frequently query the knowledge base, and thus is inefficient. In order to reduce the querying time and make the segmentation algorithm more practical, we leverage some intuitive clues learnt from the past experience to help make decisions before querying the knowledge base.

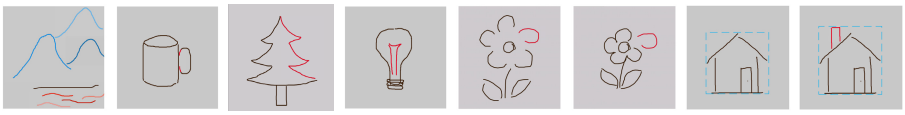
2.3 Classification Based on Intuitive Clues

Without any prior knowledge, human beings can still have some feelings about whether an unfamiliar part is an object, or whether two parts belong to one object. Psychological research [11] pointed out that some common rules learnt from past experience could intuitively help human in the perceptual grouping.

Fig. 5 illustrates some examples of this kind of factors. (a) *Similarity*: similar strokes of 'mountains' or 'water' are likely to be merged together; (b) *Continuity of Direction*: nearby strokes with a continuous direction tend to belong to the same object; (c) *Symmetry*: a symmetric sketch is likely to be an object; (d)

Algorithm 1. The Greedy Backward Segmentation Algorithm

Input The set of strokes ordered by the drawing sequence, s_i, $i = 1, ..., n$.
Output Ensemble of segmentations, $\mathcal{S} = \{S_j, j = 1, ..., m\}$.
Process
Set $\mathcal{S} = \emptyset$, $m = 0$.
For each stroke s_i, $i = 0, ..., n$,
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_i\}$, $m \leftarrow m + 1$. For the last j segments $\mathcal{S}_j = \{S_{m-j+1}, ...S_m\}$,
    let C_j denote a segment with all strokes in $\mathcal{S}_j$.
    For $j = m, ..., 1$,
        For each segment $S_k \in \mathcal{S}_j$,
            If $H(C_j) < H(S_k) + \frac{\lambda}{D_{KL}(S_k, C_j)}$,
                $\mathcal{S} \leftarrow \{S_1, ..., S_{m-j}\} \cup \{C_j\}$. Go to 14.
            End
        End
    End
End
Return $\mathcal{S}$.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 5. Example sketches. Some common clues learnt from past experience can help human in the perceptual grouping: (a) similarity, (b) continuity of direction, (c) symmetry, (d) closure, (e-f) compactness, (g-h) change of bounding box.

Closure: if one part is enclosed by another part, they might be merged together; (e-f) *Compactness*: a part of an object should not change too much the compactness of the object; (g-h) *Change of Bounding Box*: if a newly drawn stroke does not bring too much information to the original object, it tends to be one part of that object.

Based on the psychological studies in [11] and our observations, eight features are defined to quantitatively measure these intuitive clues. Then, a *random forests regression* algorithm is leveraged to predict whether and how to merge the newly drawn stroke. We assume that the inputs are a newly drawn stroke s and an existing segment $S = \{\hat{s}_1, ..., \hat{s}_m\}$. Besides, let S_{new} denote the new segment after merging s and S.

Similarity. First, the stroke s is equally divided into 2 sub-strokes, denoted by s_1 and s_2. Second, we calculate the orientation of each sub-stroke by fitting a line to it, denoted by θ_1 and θ_2. Then a feature of s is obtained: $\alpha_1 = \theta_1 - \theta_2$. In a similar way, we equally divided s into 4, 8, 16 sub-strokes, and get another 3, 7, 15 features respectively. Thus, a 26-dimension feature vector α_s is obtained for s. Then, the final similarity feature between the stroke s and the segment S is defined as $f_{sim} = \min_{\hat{s} \in S} ||\alpha_{\hat{s}} - \alpha_s||_2$.

Continuity of Direction. Let p denote the first point of the newly drawn stroke s, and $\hat{p}$ denote the closest point to p in the segment S. Then, the orientation difference of the strokes at these two points is used as the feature: $f_{direction} = \theta_p - \theta_{\hat{p}}$.

Symmetry. First, we compute the centroid of S_{new}. Then, the sketch is horizontally flipped to S'_{new} centered by the centroid. Finally, we obtain the feature f_{symH} by calculating the Chamfer distance [18] between S_{new} and S'_{new}. Similarly, by vertically and centrosymmetrically flipping the sketch, we can obtain another two features f_{symV} and f_{symC}. We use the minimal value of the three features as the final symmetry feature f_{sym}.

Closure. The *enclosure correlation* defined in Eqn. 4 is used here: $f_{closure} = Corr(S, s)$.

Compactness. The *internal difference*, which was introduced in Section 2.1, is used to measure the compactness of a segment. Thus, the compactness feature is defined as: $f_{compact} = \frac{Dist(S,S_{new})}{Int(S)}$, in which $Dist(S, S_{new})$ is the *segment distance* defined in Eqn. 2, and $Int(S)$ is the internal difference.

Change of Stroke Length, Convex Hull, and Bounding Box. Let $|S|$ and $|S_{new}|$ denote the total length of the strokes in segment S and S_{new}. Then, the change of stroke length is given by: $f_{length} = |S_{new}|/|S|$. Similarly, let $|C|$ and $|B|$ denote the areas of convex hull and bounding box of S. Then we can obtain the other two features in a similar way: $f_{convex} = |C_{new}|/|C|$, and $f_{bounding} = |B_{new}|/|B|$.

Learning for Segmentation Given a segment and a newly drawn stroke, we first extract the above features. Then, a Random Forests Regressor is trained to predict whether to merge or not.

In practice, for a newly draw stroke, there might be multiple segments available to merge. In this case, we use this classifier to classify each segment. If more than one segments are qualified to merge, we consider the order of drawing and select the latest segment to merge with the newly drawn stroke.

2.4 The Sketch Segmentation Framework

Fig. 6 shows the flowchart of the proposed sketch segmentation framework. This framework can automatically obtain segmentation results in real time during a user's drawing. As shown in Fig. 6, when the user adds a new stroke s, the proximity-based segmentation is first used to segment all existing strokes. Assume s belongs to a segment S. Then, the intuition-based classifier is used to check whether s can be merged to S or any sub-segments[3] in S. If yes, the current task is finished, and the system will wait for the next drawing. Otherwise, it will turn to the semantic-based method to decide whether to merge.

[3] In each segment S obtained by proximity-based segmentation, there might be multiple segments such as S_1 and S_2 in S. The two segments can be separated by the semantic-based method when analyzing a previous stroke, in spite of not separated by proximity-based step. This segmentation information was recorded and will be leveraged when analyzing a newly drawn stroke.

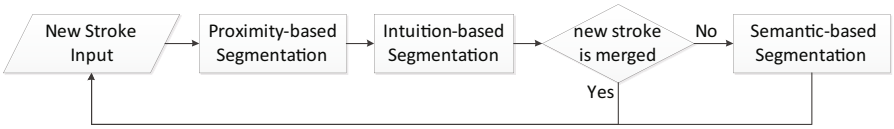


Fig. 6. Flowchart of the sketch segmentation framework

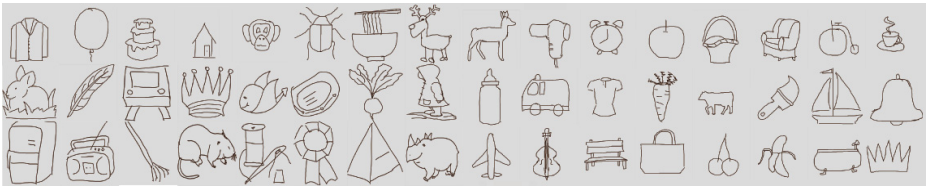


Fig. 7. Example hand-drawn sketches in the Sketch-500 dataset

It should be noted that, the proximity-based segmentation is indispensable in this framework. Besides the effect of reducing the query space in semantic-based segmentation, it can help a lot if a user does not draw objects one by one, since it does not leverage the order of drawing. For example, if a user draw objects S_1 and S_2 in turn, and then add another stroke s to S_1. The proximity-based method will group S_1 and s together. Thus, it greatly reduces the risk of leveraging the order of drawing in the semantic-based segmentation.

3 Experimental Results

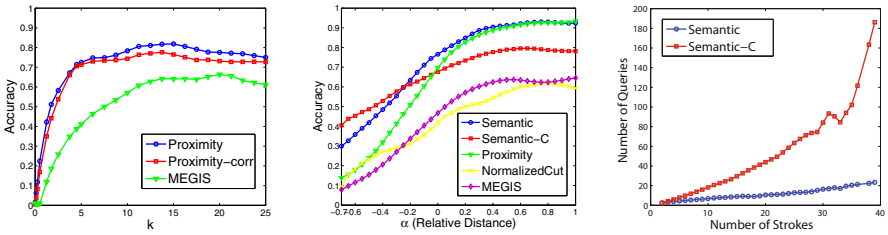
3.1 Experiment Setup

The Sketch-500 [4] data set was used in this experiment. Sketch-500 contains 500 hand-drawn sketches, each of which corresponds to a meaningful object selected from 1000 frequently-used non-abstract nouns. The drawing sequences were also recorded in Sketch-500. Example sketches are shown in Fig. 7.

The *relative distance* α is introduced as a parameter for composing simple sketches in Sketch-500 to complex sketches for training/testing. It is defined as $\alpha = \frac{Dist(S_1,S_2)}{\min(d_{S_1},d_{S_2})}$, where $Dist(S_1, S_2)$ is the minimal distance between two simple sketches, and d_S is the diagonal length of the bounding box of sketch S. If there is some overlap between the two sketches, $Dist(S_1, S_2)$ becomes negative and measures the width of the overlap region. For a constructed sketch composed of N simple sketches, α will be set to be the average of all *relative distances* between simple sketches. The *accuracy*, which is the number of correctly segmented objects divided by N, is used as the measurement.

100 sketches from Sketch-500 were selected to randomly compose 2,000 sketches using $\alpha \in [-0.7, 1]$ and $N = 2$ for training the random forests. The other 400 sketches in Sketch-500 were used to randomly compose 2,000 sketches for testing.

We first evaluate the proximity-based segmentation algorithm, followed by the experiments related to the semantic-based framework.



(a) (b) (c)

Fig. 8. (a) Performance comparison for proximity-base sketch segmentation. (b) Performance comparison of different segmentation algorithms under different α. (c) The times of querying the knowledge base for each sketch with different number of strokes.

3.2 Proximity-Based Sketch Segmentation

Three algorithms were compared, i.e. 1) the *MEGIS* algorithm, 2) the proximity-based segmentation algorithm (*Proximity*), and 3) the *Proximity* algorithm without considering the enclosure correlation (*Proximity-corr*). Since the parameter k in the *minimal internal distance* in Eqn. 1 is very crucial to control the sizes of segments, we compared the algorithms for different k. 2,000 testing sketches were randomly composed using $\alpha \in [-0.1, 1.0]$.

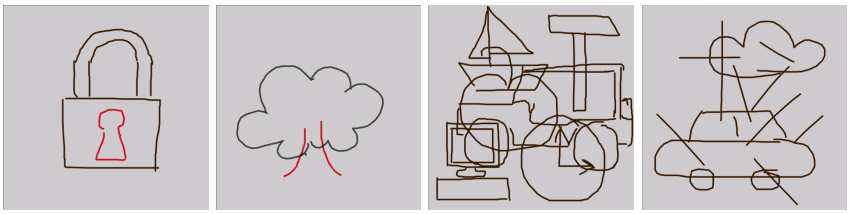
The average accuracies of different algorithms are shown in Fig. 8(a). We can see that both of *Proximity-corr* and *Proximity* greatly outperform *MEGIS*, which shows the effectiveness of the two major modifications in the graph segmentation criterion. It is also an evidence that image segmentation algorithms are inappropriate to directly use for solving the sketch segmentation problem.

In the rest of the experiments, k was set to be 15.

3.3 Semantic-Based Sketch Segmentation

We extensively evaluate the proposed image segmentation framework (*Semantic*) in this experiment. The *Proximity* algorithm, and the *Semantic* algorithm without intuitive classification (*Semantic-C*) are also compared.

Algorithm Comparison. Fig. 8(b) shows the average accuracies of different algorithms. Each testing image is composed of two single objects ($N = 2$). From Fig. 8(b) we can draw the following conclusions. First, the performances of all algorithms consistently increase as the distance between objects being larger. Second, *Proximity* performs quite well when objects are separable ($\alpha > 0$), but significantly drops if there is some overlap ($\alpha < 0$). This is reasonable since only the proximity factor is used. Third, both of *Semantic-C* and *Semantic* perform much better than *Proximity* when $\alpha < 0$, owning to the leveraging of high-level knowledge. *Semantic* is a little worse than *Semantic-C* when there are significant overlaps, since some intuitive features such as *closure* and *change of convex hull* sometimes do not work for highly overlapped objects. However, highly overlapping of objects seldom appears in freehand drawings, which makes *Semantic*



(a) (b) (c) (d)

Fig. 9. Example sketches for analysis: (a) a lock, (b) a tree, (c) a sketch composed with six objects, and (d) a sketch with noisy line segments.

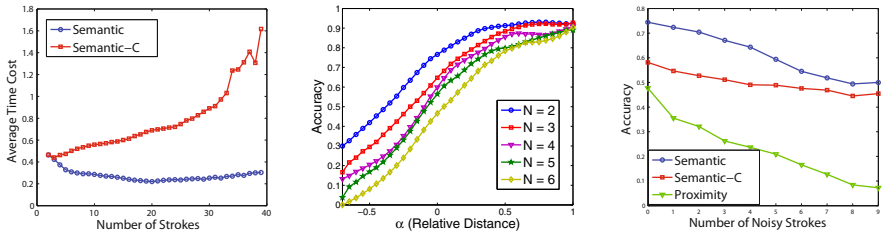
still quite useful for real cases. Fourth, when $\alpha > 0$, *Semantic-C* performs worse than the other methods. The reason is that it might over-segment some sketches. For example, the algorithm wrongly recognized the sketch without the red part in Fig. 9(a) as a 'bag', and Fig. 9(b) as a 'cloud'. However, these errors could be corrected by the intuitive features such as *symmetry* or *change of bounding box* in the *Semantic* algorithm. We also show the performances of *MEGIS* and *Normalized Cut* [13][4] in Fig. 8(b), which are worse than the proposed methods.

Efficiency. *Semantic* avoids frequently querying the knowledge base, and thus is much more efficient than *Semantic-C*. Fig. 8(c) shows the querying times of the two algorithms as the number of strokes n changes. We can see that, the complexity has been reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$ by *Semantic*. We conducted the experiments on an Intel Xeon 2.4GHz QuadCore server with 16GB memory. Fig. 10(a) shows the average time cost for each newly drawn stroke. We can see that the segmentation speed of *Semantic-C* significantly drops as the user draws more strokes. However, the speed of *Semantic* is almost constant, i.e., smaller than 0.3 second. This makes the segmentation system have the ability of real-time segmenting a user's sketch during his/her drawing.

Add More Objects. If adding more objects to the testing sketches, as illustrated in Fig. 9(c), the problem becomes more challenging. Fig. 10(b) shows that the performance is decreasing when the sketches become more complex. This is reasonable since it is still very challenging for human to segment and recognize all the objects in Fig. 9(c). Another reason comes from the strict evaluation strategy. In our evaluation, an object cannot be deemed as correctly segmented unless a segment contains and only contains all the strokes in the object. This strategy makes it more difficult to segment complex sketches.

Add Noises. To evaluate the robustness of the proposed algorithms, we added noisy line segments to each testing sketch with random length and at random position, as shown in Fig. 9(d). The testing sketches were composed using $\alpha \in [-0.7, 1]$ and $N = 2$.

[4] The graph was constructed in the same way as *MEGIS*. The number of segments was determined by the number of eigenvalues larger than 0.7.



(a) (b) (c)

Fig. 10. (a) The average time cost for each newly drawn stroke. (b) Performance of the *Semantic* algorithm with different numbers of objects in the testing sketches. (c) Performance comparison under different numbers of noisy strokes.

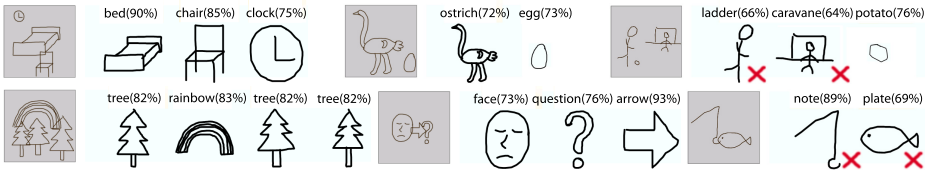


Fig. 11. Examples of segmentation results. The labeled word and its probability are also automatically provided for each segment.

Fig. 10(c) shows the comparison results of different algorithms. We can see that, as the number of noisy strokes increases, the performance of *Proximity* drops faster than the other two algorithms. This is because noisy strokes might indirectly connect different objects and thus affect the *proximity* factor. *Semantic-C* is most robust to noises owning to the leveraging of high-level knowledge. Since the intuitive clues might be affected by the noises, *Semantic* is not so robust as *Semantic-C*. However, it still consistently outperforms *Semantic-C*.

Recognize the Sketch. The output segments can be naturally labeled in the segmentation process. We can use the word with the highest $Score(w|Sketch)$ (Eqn. 7) as the label of a segment. Some examples are shown in Fig. 11. We can see that, for some overlapped objects, the results are still acceptable. Sketch recognition is beyond the scope of this work, and please refer to [4] for the sketch recognition work.

4 Conclusions

In this paper, we studied the problem of sketch segmentation, which might be one of the most fundamental problems in sketch-related research. A sketch segmentation framework was developed to support real-time analysis during a user's

drawing. The proposed sketch segmentation framework took into account the basic factors in human perceptual organization, the effectiveness of which has been shown in the experiments.

Acknowledgments. The work of Zhenbang Sun and Liqing Zhang was partially supported by the National Natural Science Foundation of China (Grant No. 90920014, 91120305) and NSFC-JSPS international exchange program (Grant No. 61111140019).

References

1. Marr, D.: Early processing of visual information. Philosophical Transactions of the Royal Society of London. B, Biological Sciences (1976)
2. Cao, Y., Wang, C., Zhang, L., Zhang, L.: Edgel index for large-scale sketch-based image search. In: CVPR (2011)
3. Bronstein, A., Bronstein, M., Guibas, L., Ovsjanikov, M.: Shape google: Geometric words and expressions for invariant shape retrieval. TOG (2011)
4. Sun, Z., Wang, C., Zhang, L., Zhang, L.: Query-adaptive shape topic mining for hand-drawn sketch recognition. In: ACM Multimedia (2012)
5. Temlyakov, A., Munsell, B., Waggoner, J., Wang, S.: Two perceptually motivated strategies for shape classification. In: CVPR (2010)
6. Pu, J., Gur, D.: Automated freehand sketch segmentation using radial basis functions. In: Computer-Aided Design (2009)
7. Sezgin, T., Stahovich, T., Davis, R.: Sketch based interfaces: Early processing for sketch understanding. In: ACM SIGGRAPH (2006)
8. Sezgin, T.: Feature point detection and curve approximation for early processing of freehand sketches (Masters thesis, Massachusetts Institute of Technology)
9. Hammond, T., Davis, R.: Ladder, a sketching language for user interface developers. Computers & Graphics (2005)
10. Sezgin, T., Davis, R.: Hmm-based efficient sketch recognition. In: ACM IUI (2005)
11. Wertheimer, M.: Laws of organization in perceptual forms. A source book of Gestalt psychology (1938)
12. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. IJCV (2004)
13. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI (2000)
14. Boykov, Y., Funka-Lea, G.: Graph cuts and efficient nd image segmentation. IJCV (2006)
15. Carreira, J., Sminchisescu, C.: Constrained parametric min-cuts for automatic object segmentation. In: CVPR (2010)
16. Cao, Y., Wang, H., Wang, C., Li, Z., Zhang, L., Zhang, L.: Mindfinder: Interactive sketch-based image search on millions of images. In: ACM Multimedia (2010)
17. Wang, C., Zhang, J., Yang, B., Zhang, L.: Sketch2cartoon: composing cartoon images by sketching. In: ACM Multimedia (2011)
18. Borgefors, G.: Hierarchical chamfer matching: A parametric edge matching algorithm. PAMI (1988)