Volume 31 (2012), Number 8 pp. 2516–2526 COMPUTER GRAPHICS *forum*

Smart Scribbles for Sketch Segmentation

G. Noris[†1,2], D. Sýkora[3], A. Shamir[4], S. Coros[1], B. Whited[5], M. Simmons[5], A. Hornung[1], M. Gross[1,2], and R. Sumner[1]

[1]Disney Research Zürich, [2]ETH Zürich, CGL, [3]CTU in Prague, FEE,
[4]The Interdisciplinary Center, [5]Walt Disney Animation Studios

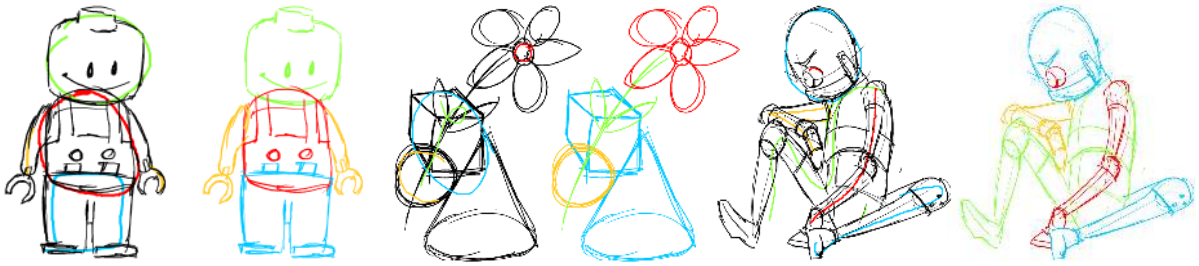

Figure 1: *Sketch segmentation: For each example pair, Scribbles on the left produce the segmentation on the right.*

Abstract

We present Smart Scribbles*—a new scribble-based interface for user-guided segmentation of digital sketchy drawings. In contrast to previous approaches based on simple selection strategies, Smart Scribbles exploits richer geometric and temporal information, resulting in a more intuitive segmentation interface. We introduce a novel energy minimization formulation in which both geometric and temporal information from digital input devices is used to define stroke-to-stroke and scribble-to-stroke relationships. Although the minimization of this energy is, in general, a NP-hard problem, we use a simple heuristic that leads to a good approximation and permits an interactive system able to produce accurate labelings even for cluttered sketchy drawings. We demonstrate the power of our technique in several practical scenarios such as sketch editing, as-rigid-as-possible deformation and registration, and on-the-fly labeling based on pre-classified guidelines.*

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.4]: Graphics Utilities—Graphics editors, Computer Graphics [I.3.6]: Methodology and Techniques—Interaction techniques, Image Processing and Computer Vision [I.5.3]: Clustering—Similarity measures

Keywords: digital sketches, interactive segmentation, scribble-based user interface, energy minimization

1. Introduction

Sketchy drawings are prevalent across a wide range of applications and domains. In early development phases, rough drawings are used, for example, for concept art in product design, and for storyboards in animation environments, and are favored both for the speed of generation, and the expressiveness of the results. A sketchy style also has a place in finished art – providing a level of visual richness not found in "clean" line representations, i.e. drawings constructed from crisp, distinct outlines and minimal interior detail.

Modern digital devices and graphics software solutions offer powerful stylization, deformation, morphing, and animation capabilities for 2D drawings. However, in order to perform these high-level tasks, a certain degree of under-

† chino@disneyresearch.com

standing of the content of the drawing is required. This is a challenging problem due to the significant gap between the ability of a human to discern structure in a drawing and the capability of an algorithm to derive it from low level stroke information. This is true even for clean line drawings, and most existing approaches rely on the presence of a human user to provide sufficient information to guide the task.

The problem of extracting structure from drawings becomes substantially more difficult for sketchy input, and this is one reason it is far less common to find a consistently sketchy style in full-length animations or automatic support for sketchy input in high-level editing packages. One important category of drawing abstraction is segmenting the drawing into logical parts. To-date, there is no efficient method available for automatic segmentation in this domain. In contexts where a breakdown of the drawing is required, segmentation is typically achieved by design: the drawings are created in different layers, one for each logical component. This approach is too limiting in practice: it requires a priori knowledge of the use of the drawing, is cumbersome (especially when different tasks require different segmentations), and is an error-prone process, even for experienced artists.

We seek a semi-automated solution to segmenting sketchy drawings that is fast enough for interactive use, but also predictable and easy to use – making it accessible to even the most novice user.

To this end, we propose the concept of *Smart Scribbles* as an accurate and simple way for the user to specify semantically meaningful stroke clusters within a drawing. In contrast to previous methods that use scribbles as positional constraints for various image editing tasks [BJ01,LLW04,AP08, SDC09b], our formulation considers more detailed geometric (position, orientation, curvature) and temporal information (time of creation) when analyzing stroke-to-stroke and Scribble-to-stroke relationships. In addition, we introduce the concept of locality control as a way of conveniently trading off the Scribbles' areas of influence for accuracy. This allows our system to produce desired results with minimal user intervention even for cluttered sketches.

We evaluate our approach on a collection of digitally drawn sketches of varying complexity, and demonstrate its application to various tasks including sketch editing and as-rigid-as-possible (ARAP) deformation and registration. As our solution is fast to compute, our method enables tight integration of these tasks within an interactive digital drawing session.

2. Related Work

Relevant prior art can be divided into three main categories: sketch labeling interfaces, scribble-based image segmentation, and classification of vector fields in scientific visualization.

User-guided labeling of strokes in hand-drawn images plays a central role in many sketch-based editing systems. In Lank et al. [LS05], the authors present an approach for inferring user intent from the local velocities, accelerations and curvatures of the selection lasso. More recently, Wolin et al. [WSA07] presented a technique for labeling groups of strokes from a vectorized sketch where the system attempts to automatically fragment continuous strokes into logical pieces to assist the user. Both of these techniques ultimately utilize a region-based selection approach. ScanScribe, a system developed by Saund and colleagues [SFLM04], presents the user with an intuitive selection paradigm that allows for the creation of objects from collections of pixels and supports further grouping into composite objects. The system is able to automatically segment the image into basic primitives, such as linear curve fragments, and then group them into more complex objects, such as rectangles, using a fragment alignment metric (or by finding perceptually closed paths as proposed in [Sau03]). Two limitations of this automatic technique are 1) limited complexity of objects detected by the system and 2) the inability to handle sketchy overlapping curve fragments, thus requiring more traditional and tedious lasso/selection-box methods for more complex drawings.

The approach presented in this paper leverages previous works on interactive image segmentation in order to optimize the labeling process based on user scribbles. Boykov et al. [BJ01] developed such an approach based on graph cuts for segmenting images and finding optimal boundaries between objects. In [LLW04], Levin and colleagues present a similar framework based on a least-squares optimization for colorizing gray-scale images by roughly labeling regions with colored scribbles. More recently, An and Pellacini [AP08] developed an interactive energy minimization framework for propagating color edits to similar regions throughout the image. Our approach is most similar to LazyBrush [SDC09b], a graph cut based system for the selection of regions in sketchy drawings. The main difference is that this system cannot provide the labeling of the strokes that bound each painted region. From this point of view, our framework can be seen as a generalization of LazyBrush, since it extracts meaningful boundaries first, and then builds regions inferred from those boundaries. Because this process removes clutter from the input drawing, it greatly improves the accuracy of selection and reduces the amount of user interaction needed to obtain clean results.

Our approach also bears some resemblance to sketch-based clustering of vector fields in scientific visualization [WWYM10]. Here the aim is to allow the user to sketch 2D curves and use them as a query to retrieve 3D field lines whose view-dependent 2D projection is most similar to the input sketch. The curvature along the sketched input is used to measure the similarity between the input and projected curves using the edit distance [WF74]. In our approach, curvature is also used to distinguish between different shapes. However, the main advantage of our work is that we formu-

late an energy minimization problem where, in addition to shape similarity, we also take proximity, orientation, temporal information, and smoothness of the final labeling into account. As a result, our system can produce reasonable clustering even in cases when the shape of the input sketch is very rough or incomplete.

3. Method

The method we present allows users to intuitively segment digital sketches into semantically meaningful regions. The input to our framework consists of a digitally hand-drawn sketch and a small set of rough *Scribbles*. The input sketch is composed of a set of *strokes*, which are piecewise linear curves represented by sets of 2D vertices recorded from a digital input device such as a tablet. For each vertex of a stroke, we additionally store its time of creation. This helps us to differentiate strokes which are spatially close but are drawn at different moments in time.

The input Scribbles are special strokes that indicate the user's intent to segment a particular portion of the drawing. Two criteria related to the Scribble primitives are critical in order to ensure a useful and intuitive system. First, Scribbles should not have to closely follow the target region. However, if desired, the user should be able to precisely select localized regions. We call this property *locality control*. The second criterion specifies that the time of creation of the Scribble should not influence the segmentation results.

We observe that generally speaking, processing strokes as a whole is very difficult. A single stroke can be arbitrarily complex: it can cross or overlap with itself multiple times, and/or it can densely cover an area the artist wished to fill in. For this reason, we break strokes and Scribbles into linear *segments* by densely resampling the input. Any property defined locally over the stroke can easily be transferred to the segments.

The remainder of this section describes in detail each of the steps used by our method

We formulate the task of sketch clustering as an optimization problem, where the goal is to label each stroke in a way that minimizes an energy function. The concept of our design is depicted in Fig. 2 and the remainder of this section describes in detail each of the steps used by our method. The energy function is defined in Section 3.1. It relies on a smoothness and data term which are described in Sections 3.1.1 and 3.1.2, respectively. In Section 3.2 we discuss the minimization method used to compute the final solution to the stroke labeling.

3.1. Energy function

The input to our method consists of a set of stroke segments S and a set of Scribble segments R associated with a set of labels L. The goal is to find a labeling, i.e., an assignment ϕ of the labels in L to every segment in S, that minimizes the following energy function E:

$$E(\phi) = \sum_{i,j \in S} V_{i,j}(\phi_i, \phi_j) + \lambda \sum_{i \in S} D_i(\phi_i) \qquad (1)$$

where $V_{i,j}$ is a smoothness term that captures the cost of the labeling with respect to the similarity between two stroke segments i and j. The data term D_i measures the affinity between Scribbles and strokes. The parameter λ controls the relative influence of the smoothness and data terms.

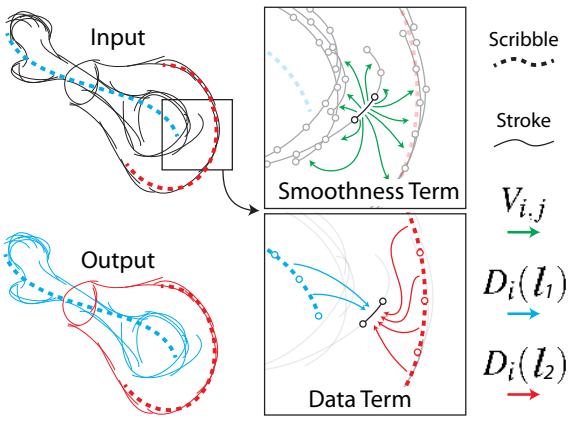

Figure 2: *Energy definition overview. The input consists of a set of strokes (black) and Scribbles (red and blue dotted lines). The output consists of a labeling of all strokes (the labeling is indicated here by the red/blue color assignment to the strokes in the output).* ***Smoothness Term:*** *For a segment i and a neighbor segment j, $V_{i,j}$ expresses the energy of assigning a different label to i and j, based on how similar they are.* ***Data Term:*** *Given a labeling $\phi_i = l_*$ (assigning label l_* to segment i), $D_i(\phi_i)$ expresses the energy of the labeling, which is a function of the similarity of segment i to all Scribbles associated with l_*.*

3.1.1. Smoothness Term

The smoothness term is defined as:

$$V_{i,j}(\phi_i, \phi_j) = \prod_{g \in G} \delta(g(i,j), \sigma_g) \qquad (2)$$

when $\phi_i \neq \phi_j$, otherwise it is zero. G is a set of similarity terms:

$$\begin{aligned} prox(i,j) &= ||\vec{p}_j - \vec{p}_i|| \\ dir(i,j) &= 1 - |\vec{d}_i \cdot \vec{d}_j| \\ curv(i,j) &= 1 - min(c_i, c_j)/max(c_i, c_j) \\ time(i,j) &= |t_j - t_i| \end{aligned}$$

where i and j are two segments, and p, d, c and t are the position, direction, radius of curvature, and time of creation associated with each segment. The fall-off function δ is defined as:

$$\delta(g(i,j),\sigma_g) = \exp\left(-\frac{g(i,j)^2}{\sigma_g^2}\right) \quad (3)$$

3.1.2. Data Term

The data term is defined as:

$$D_i(\phi_i) = 1 - \max_{r \in R(\phi_i)} A(i,r) \quad (4)$$

where $R(\phi_i)$ denotes a set of Scribble segments r with label ϕ_i. The affinity $A(i,r)$ is defined as:

$$A(i,r) = \prod_{g \in G_{data}} \delta(g(i,r),\sigma_g) \quad (5)$$

Here, as with the smoothness term, we measure the similarity between segments rather than strokes. However, as Scribbles have no associated time information, we reduce the set of similarity terms to $G_{data} = \{prox, dir, curv\} \subset G$. Additionally, we alter the definition of curvature to become oriented: $curv(i,j) = ||\vec{c}_i - \vec{c}_j||$. This allows extra control in separating curves with the same curvature but different orientation (e.g. the tangled lines in Fig. 5).

One of our main goals is to allow users, if desired, to have precise local control over the strokes that get affected by each Scribble. To illustrate this, we consider a scenario where the user draws a single Scribble, as shown in Fig. 3a. In this case, because no concurrent label exists, all strokes are selected. This behavior, though reasonable, is not in line with a user's expectations of having local control.

To address this, we introduce an artificial background label $b \in L$, in addition to the labels prescribed by the user. This new label has a constant influence on each stroke segment i regardless of the existence of any particular user-defined Scribble, i.e., $A(i,b) = B$, where B is a threshold to override the influence of distant Scribbles. The background label therefore serves as a lower bound for computing the *max* component in the data term (4).

Furthermore, we control the locality of each Scribble r by modifying its proximity fall-off δ (3) as follows:

$$\delta(prox(i,r),\sigma_{prox}) = \frac{1}{\sigma_{prox}} \exp\left(-\frac{prox(i,r)^2}{\sigma_{prox}^2}\right). \quad (6)$$

Here σ_{prox} follows the desired locality (i.e., is large for global influence and small for local influence) and the normalization term $1/\sigma_{prox}$ ensures the integral over the fall-off function stays equal for different values of σ_{prox} (i.e., amplitude is high for small values and low for large ones). In other words, the overall energy remains constant, while its spatial spread is controlled. When σ_{prox} becomes very low, the response of the fall-off function (6) for distant stroke segments also becomes very low and can therefore be easily overridden when computing the *max* value in (4) as illustrated in Fig. 3b-f.

There are several possible ways to control the parameter σ_{prox}. One natural way is to use the speed of the Scribble based on the experimentally demonstrated linear relationship between speed and perceived locality [AZ97]:

$$W = \frac{\beta \cdot L}{T - \alpha}. \quad (7)$$

Here W is the selection radius, L is length of the Scribble, T is time spent on drawing it, and α and β are empirically measured constants. This rule was used to control the selection locality in systems having limited modality [LS05]. Since the spatial spread of the fall-off function (6) grows linearly with the increasing σ_{prox} we can set $\sigma_{prox} = W/2$. Alternatively, one could consider the use of pen-pressure, or—in the case of binary modality—a simple key toggle to switch between two locality values.

3.2. Optimization method

As shown in [BVZ98], minimizing the energy function defined in Equation 1 is equivalent to solving a multi-way cut on a specific weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{S, L\}$ is a set of vertices and $\mathcal{E} = \{\mathcal{E}_s, \mathcal{E}_l\}$ is a set of edges (See Fig. 4). The graph vertices $\mathcal{V}$ consist of stroke segments S and label terminals L. Each stroke segment $i \in S$ is connected to all other stroke segments $j \in S - \{i\}$ via edges $\mathcal{E}_{i,j}$ having weight $w_{i,j}$ equal to the smoothness term $V_{i,j}$ when $\phi_i \neq \phi_j$. In addition, auxiliary edges $\mathcal{E}_{i,l}$ connect stroke segments $i \in S$ to label terminals $l \in L$. Each $\mathcal{E}_{i,l}$ has weight $w_{i,l} = \lambda(1 - D_i(l))$, where λ is the parameter defined in Equation 1.

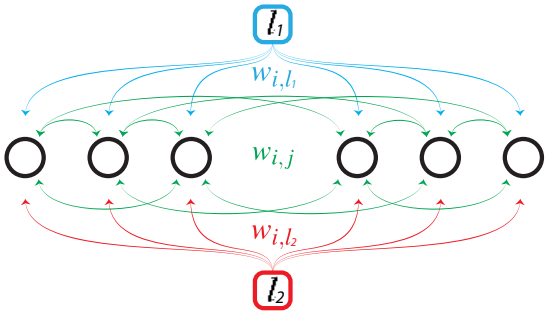

Figure 4: *Graph Construction. Stroke segments are shown as black circles. Terminal labels (in this example l_1 and l_2) are shown as colored squares. The graph edges $w_{i,j}$ reflect the smoothness terms $V_{i,j}$ between the stroke segments $i, j \in S$, while the data terms $D_i(l)$ for stroke segment $i \in S$ and label $l \in L$ are captured by the weights $w_{i,l}$.*

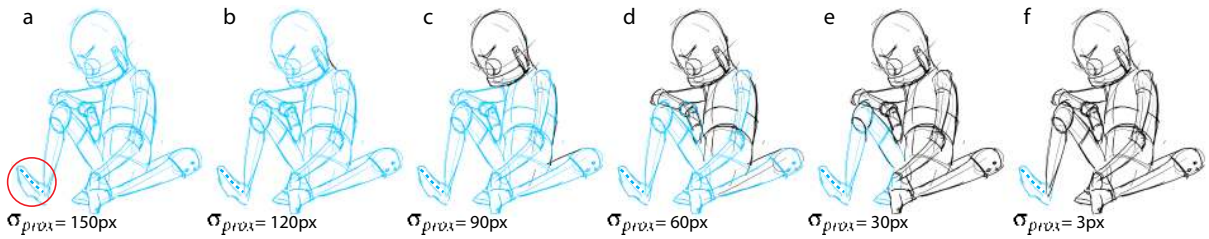

Figure 3: *The effect of the locality control by varying σ_{prox}: A blue Scribble is drawn on the foot (circled in red). On the right, the value of σ_{prox} is progressively decreased. Notice how the selection becomes progressively more local as the influence of the blue label gets overruled by the background label (shown in black).*

The multi-way cut problem with two terminals is equivalent to a max-flow/min-cut problem for which efficient polynomial algorithms exist [BK04]. However, for three or more terminals the problem is NP-hard [DJP*92]. To obtain a good approximate solution we use a simple divide-and-conquer heuristic previously proposed in [SDC09b] to gradually simplify the N-terminal problem into a sequence of $N-1$ binary max-flow/min-cut sub-problems. This approach provides results similar to more advanced techniques (such as α-expansion or α/β-swap [BVZ01]), but is significantly faster and therefore better suited for interactive applications.

4. Results

We demonstrate the effectiveness of our algorithm on a variety of input sketches. All results were generated using the parameters in Table 1.

Fig. 5 shows a collection of simple input sketches and Scribbles, together with the color-coded stroke labeling output by our system. These results show that desirable sketch segmentations can be obtained using very different scribbling strategies. We note that the input Scribbles do not have to closely match the sketch in order for our algorithm to work well—approximate similarity in terms of position, orientation and curvature is sufficient.

Figs. 1 and 6 show results from more complex input sketches. To correctly segment these images, users typically start with rough, fast strokes, and then refine the output locally using slower, more accurate strokes. Our method robustly handles scenarios where strokes that are close together and almost parallel belong semantically to different regions (as shown on the waiter's legs and snake and pole example in Fig. 6). In these cases, the time metric plays an important role in the labeling process.

Our framework does not require artists to draw the input sketches in any particular manner. It is possible that strokes representing the same region can be drawn at very different moments in time. This happens, for instance, when artists first draw silhouettes for the whole scene, and then proceed to refine the drawing. This can diminish the advantage of taking timing into account in the similarity metric. Correct segmentations can still be obtained, but more Scribbles may be required. Alternatively the similarity metric can be adjusted to apply a smaller weight to the time parameter, or it can be removed as is done for the Scribble metric.

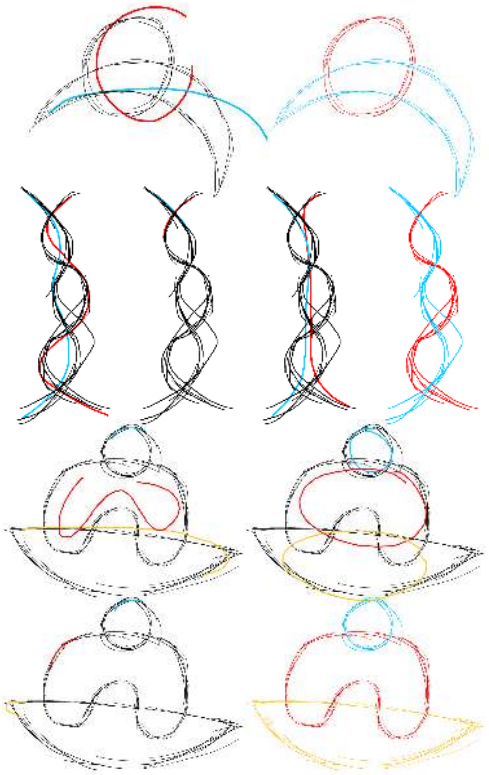

Figure 5: *Results for simple sketches: several different inputs produce the same segmentation.*

4.1. User Study

In order to test the efficiency and ease of use of our method, we conducted a user study comparing Smart Scribbles to our implementation of several commonly-used selection tools,

Figure 6: *Example Results: For each example, the colored Scribbles are shown on the input drawing and the adjacent image shows the resulting color-coded labelings.*

Parameter	Value	Unit
λ	4	
$\sigma_{prox\ smooth}$	100	px
$\sigma_{dir\ smooth}$	0.5	
$\sigma_{time\ smooth}$	1000	ms
$\sigma_{curv\ smooth}$	0.1	
$\sigma_{prox\ data}$	[10 ; 90]	px
$\sigma_{dir\ data}$	0.1	
$\sigma_{curv\ data}$	0.25	
B	0.0001	
artboard width	1200	px
artboard heigth	1200	px

Table 1: *Parameter settings for the user study and all examples in this paper.*

drawing	speed-up	$t(df)$	p-value
combo	1.23x	−1.8798	0.07847
snake	1.53x	−2.4807	0.02461
skull	1.83x	−8.3931	0.00000
house	1.85x	−5.2488	0.00001
abstract	2.36x	−5.5759	0.00003
characters	1.65x	−3.8896	0.00118

Table 2: *Median speed-ups and results of paired t-tests comparing times spent on labeling different drawings using Smart Scribbles and common selection tools.*

namely point, box, and lasso (these tools are typically included in professional vector graphics software such as Adobe Illustrator or Inkscape). This section includes an overview of the study results.

Our user study had 35 participants (8 female and 27 male with ages ranging from 18 to 62). Subjects had no prior experience using Smart Scribbles, and varying levels of proficiency (from none to expert level) with the professional software packages.

Participants were asked to use the different tools to match given labelings on four different drawings of varying complexity (see Fig. 7). The system recorded the time taken to complete the tasks and the mouse mileage, as well as the accuracy of the final labeling. Overall distributions of times and mouse mileage measured during the experiment are depicted in Fig. 8. There is a notable performance gain (1 : 23x to 2 : 36x median speed-up) when comparing Smart Scribbles to the common selection tools. Paired t-tests (see Table 2) indicate that this gain is statistically significant ($p <$ 0.005) for 4 out of 6 drawings. The lower confidence level for the snake and combo drawings is reflected by notable intersection between interquartile ranges of box-and-whisker plots in Fig. 8. Although the median speed-up is apparent, the advantage of Smart Scribbles is not as convincing in this case. The main reason here could be the relatively low complexity of these examples.

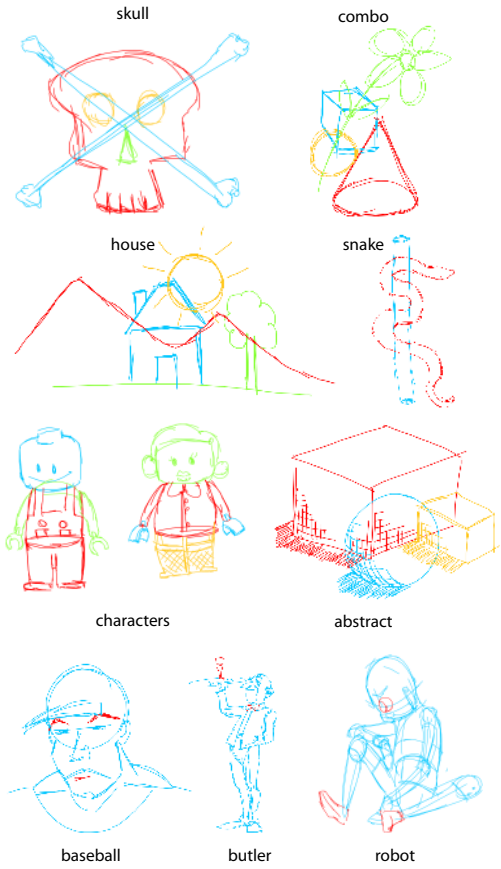

Figure 7: *Given drawings and labelings from the user study.*

In addition, participants were presented with different ways of controlling the locality. We tested the linear relationship proposed in [AZ97], as well as a simple binary modality associated with the extreme values of the parameter $\sigma_{prox\ data}$ as shown in Table 1. When asked about their experience, a majority (31) preferred the binary switching approach. We believe this is due to two reasons. First, in previous work, stroke speed had a direct associated visual feedback. This cannot easily be done with our Scribbles, as the result of the selection is not strictly bounded by a spatial radius. A simpler, more explicit interface may therefore be more appropriate. Secondly, a majority of the participants (27) indicated that they do not want to be forced to draw Scribbles slowly.

5. Applications

The labeling produced by our approach can be utilized to generate input to perform region as well as stroke segmentation (see Fig. 10a). Once the labels for the segments of each stroke are computed, we can automatically obtain an area mask of the enclosed region using the LazyBrush [SDC09b]

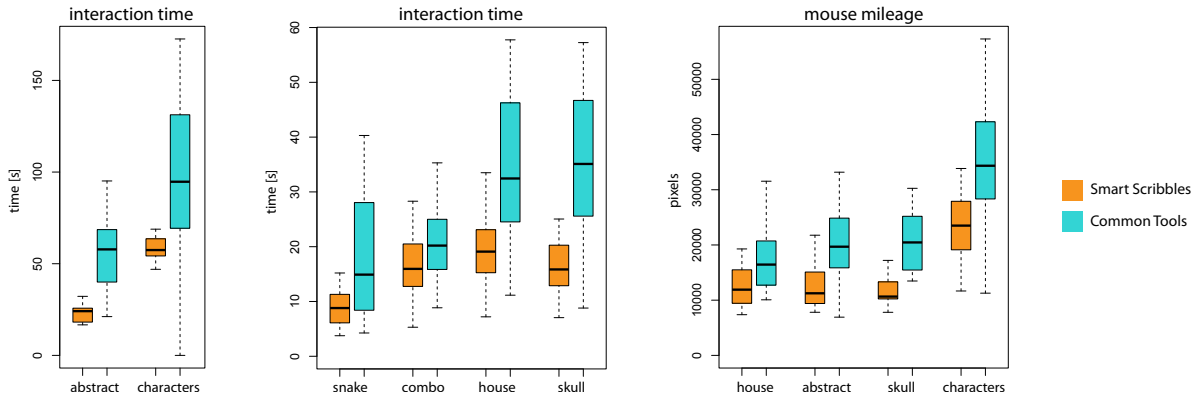

Figure 8: *Interaction times and mouse mileage of participants for different drawings using Smart Scribbles (orange) and common selection tools (blue).*

algorithm (see Fig. 9). To this end, we first render all segments assigned to a specific label to a raster image. This image is used both as an input gray-scale image (Fig. 9a) and as foreground soft scribbles (blue in Fig. 9b) for input to LazyBrush. In addition, we use a default background hard scribble around the image boundary (red in Fig. 9b). Given this input, LazyBrush produces the desired area mask (Fig. 9c). As compared to other naive methods (like convex-hull or flood-fill), this approach works with concave regions and is robust to small gaps.

For more complex sketches, the user may need to specify additional Scribbles (Fig. 9d) to classify interior strokes and use them as additional background soft scribbles for LazyBrush (Fig. 9e). These new scribbles enable the area computation method to produce masks that contain holes (Fig. 9f).

We note that similar masks (Fig. 9i) can be produced with the original LazyBrush algorithm directly. However, the area segmentation alone is not sufficient to provide a labeling of the individual strokes, because strokes at the boundaries between different area masks cannot be consistently assigned to one mask or another (Fig. 9g). Moreover, with the original LazyBrush algorithm, the user must be more careful, since the optimization only takes into account the position of the scribbles. In contrast, our framework also considers orientation, curvature, and time (compare Fig. 9d and h).

The ability to easily label both strokes and areas empowers a large variety of applications. One can easily alter the individual drawing style for all strokes that have the same label. It is also possible to accurately separate the different parts of a sketch, specify their depth ordering [SSJ*10], and then deform them independently using ARAP shape deformation techniques [IMH05] (see Fig. 10b). These operations can help, for instance, in the context of image registration [SDC09a] to produce better alignment.

When artists start a drawing, they typically begin with a simple, high-level sketch that depicts a set of of primitive shapes (see examples and references in [GIZ09]) that are called volume or scaffold lines. If available, we can use these aiding structures as Scribbles to segment the final detailed sketch (see Fig. 10c). This would let the artist focus on drawing without having to switch between different brushes. ARAP deformation, for instance, could then be used to correct the shape of semantically meaningful sketch regions.

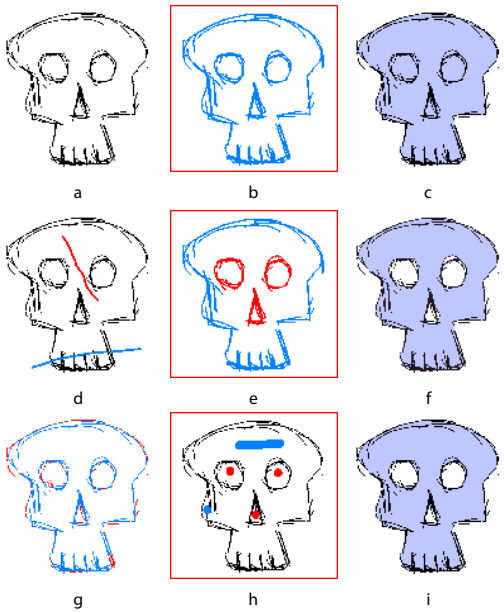

Figure 9: *Area mask computation: strokes of an input sketch (a) can be used as LazyBrush soft scribbles (b) to automatically fill the drawing (c). Additional Scribbles (d) can be used to segment the strokes (e) for better control of the paint fill (f). Using the original LazyBrush algorithm (h) to paint the figure also produces a good result (i), however, the strokes cannot be classified based on the painting alone (g).*

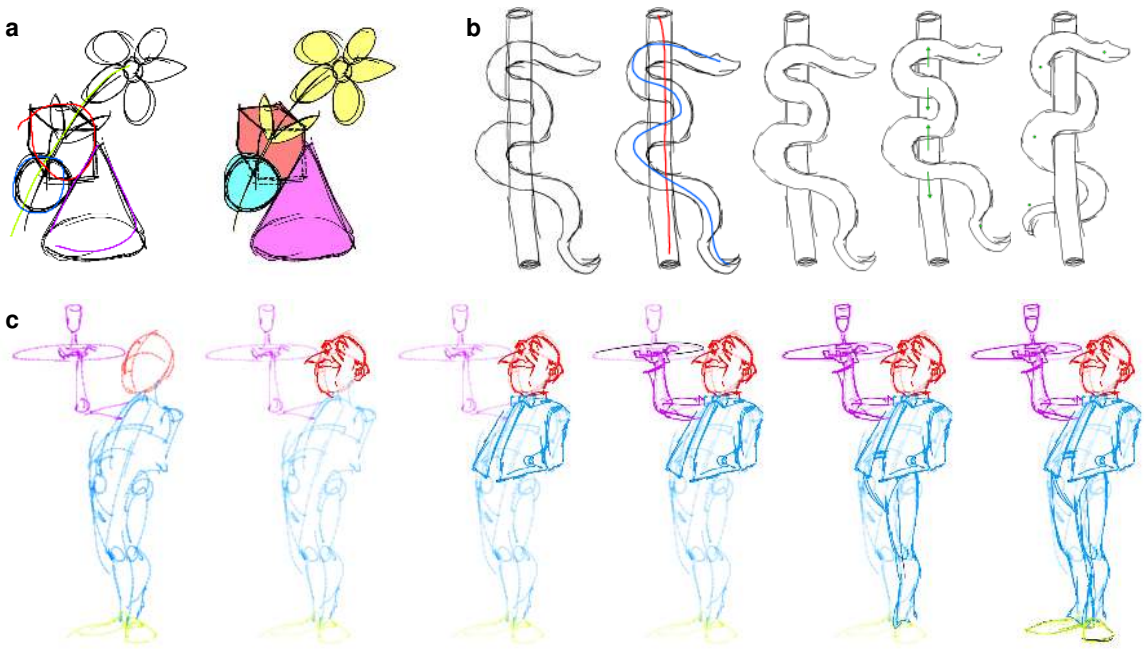

Figure 10: *Applications. Opaquing of the segmented clusters (a), ARAP deformation and opaquing with depth inequalities (b), on-the-fly labeling (c).*

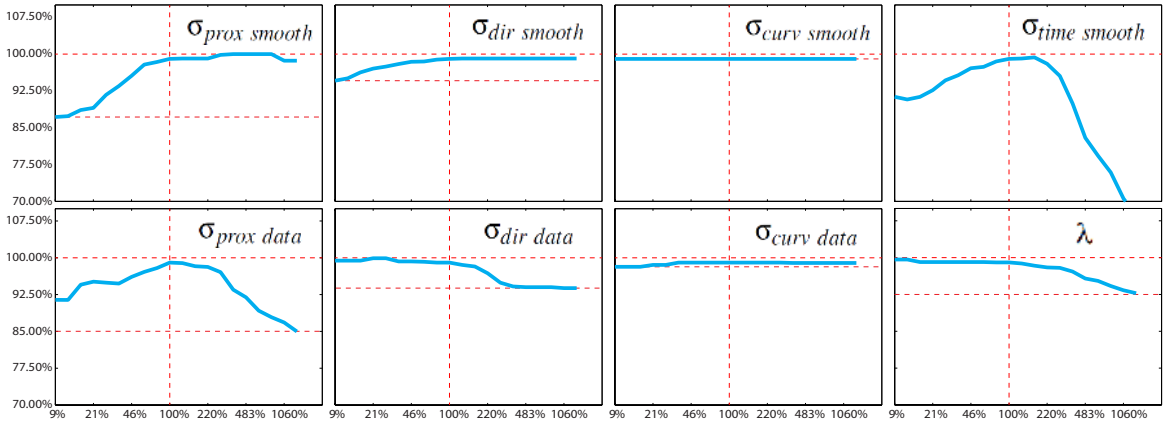

Figure 11: *Single parameter perturbation. Given a database of 8 drawings, each with 5 different sets of Scribbles drawn to match a desired segmentation, we measure the segmentation accuracy obtained with perturbations of the empirically chosen settings of Table 1. In each graph, the horizontal axis shows the multiplication factor for one of the parameters in exponential scale. The vertical axis shows how the correctness of the segmentation evolves. The graphs show that the system is mostly sensitive to time and proximity information, while direction and curvature have less influence.*

6. Limitations and Future Work

The selection of good parameters for the similarity terms and the energy function requires some effort. As can be observed in the parameter sensitivity graphs in Fig. 11, the system is robust when parameters are perturbed one at a time. This is due to the correlation that exists between the similarity terms. However, it is possible that the perturbation of multiple parameters can lead to significant changes in the result. We also tested the benefit of including the stroke creation time in the stroke analysis. After removing this information from the similarity terms, and re-tuning the remaining parameters, we achieved the results shown in Fig. 12. In our experience, omitting this temporal information reduces the effectiveness of our method, as overlapping strokes require

more effort to be separated. In the future we plan to develop a system that allows automatic parameter tuning based on a database of ground truth data.

Although we aim to produce accurate labeling with minimal user effort, detailed selection is necessary when ambiguities exist. One such ambiguous case occurs when an object is occluded by another object and parallel strokes from each are very close together or even overlap. In this case, only the time constraint can provide a distinctive metric to obtain correct labeling. However, when the time is not available or when the user does not preserve temporal coherency of strokes, our approach requires additional user guidance.

The proposed graph-cut energy minimization strategy is generally very fast and produces the labeling at interactive rates. However, in the worst case, when a large number of stokes are close to each other as defined by our similarity measure, the number of edges in the graph can grow quadratically with the number of strokes and the computation can become prohibitively slow (see Fig. 13). The problem can be alleviated by subsampling the strokes and processing disconnected components individually. Another problem is related to the non-polynomial complexity of the core max-flow algorithm [BK04]. In certain situations where the cost of the minimal cut is very high and the graph topology is complex, the number of augmentation paths can grow very quickly along with the computation time. This issue can be solved by a recently proposed incremental breadth-first search solution [GHK*11] that works in polynomial time and is typically notably faster than [BK04].

The use of previously labeled drawings as Scribbles offers another avenue for future work. These annotations could be used on-the-fly to label new sketches as they are created, thus simplifying further interactions. This approach could be used, for instance, as an extension to the recently presented ShadowDraw system [LZC11], by augmenting each sketch in the database with Scribbles. In this way the segmentation could be provided automatically as new drawings are created. A similar use case arises in the context of sketchy animations where image registration [SDC09a] can be used to transfer already labeled strokes and treat them as Scribbles for the next frame. This can help, for instance, to better control temporal noise [NSC*11]. Scribbles could also potentially be used to improve the accuracy of drawing simplification methods [GDS04,BTS05,SC08], as a typical problem with the current, fully automatic, approaches is that they do not take into account any semantic information such as provided by our approach.

7. Conclusions

We have presented Smart Scribbles, a scribble-based interface for sketch segmentation. Our method is fast, supports multi-label segmentation, and acts as an enabling technology for a variety of applications in the context of drawing, editing, and animation.

In the long term, we envision a next-generation drawing application, where drawing, editing, and animation are tightly integrated, and where the simplicity of the interaction is the key. This work represents a step in this direction; a bridge between classic drawing and digital editing.

Acknowledgements

We would like to thank Adam Sporka for help with the user study, Maurizio Nitti for creating some of the drawings, and all anonymous reviewers for their insightful comments and suggestions.

References

[AP08] AN X., PELLACINI F.: AppProp: All-pairs appearance-space edit propagation. *ACM Transactions on Graphics 27*, 3 (2008), 40. 2517

[AZ97] ACCOT J., ZHAI S.: Beyond Fitts' law: Models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1997), pp. 295–302. 2519, 2522

[BJ01] BOYKOV Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings of Internation Conference on Computer Vision* (2001), pp. 105–112. 2517

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9 (2004), 1124–1137. 2520, 2525

[BTS05] BARLA P., THOLLOT J., SILLION F. X.: Geometric clustering for line drawing simplification. In *In Proceedings of the Eurographics Symposium on Rendering* (2005), pp. 183–192. 2525

[BVZ98] BOYKOV Y., VEKSLER O., ZABIH R.: Markov random fields with efficient approximations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (1998), pp. 648–655. 2519

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (2001), 1222–1239. 2520

[DJP*92] DAHLHAUS E., JOHNSON D. S., PAPADIMITRIOU C. H., SEYMOUR P. D., YANNAKAKIS M.: The complexity of multiway cuts. In *Proceedings of ACM Symposium on Theory of Computing* (1992), pp. 241–251. 2520

[GDS04] GRABLI S., DURAND F., SILLION F. X.: Density measure for line-drawing simplification. In *Proceedigns of Pacific Conference on Computer Graphics and Applications* (2004), pp. 309–318. 2525

[GHK*11] GOLDBERG A. V., HED S., KAPLAN H., TARJAN R. E., WERNECK R. F. F.: Maximum flows by incremental breadth-first search. In *ESA* (2011), pp. 457–468. 2525

[GIZ09] GINGOLD Y., IGARASHI T., ZORIN D.: Structured annotations for 2D-to-3D modeling. *ACM Transactions on Graphics 28*, 5 (2009), 148. 2523

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics 24*, 3 (2005), 1134–1141. 2523

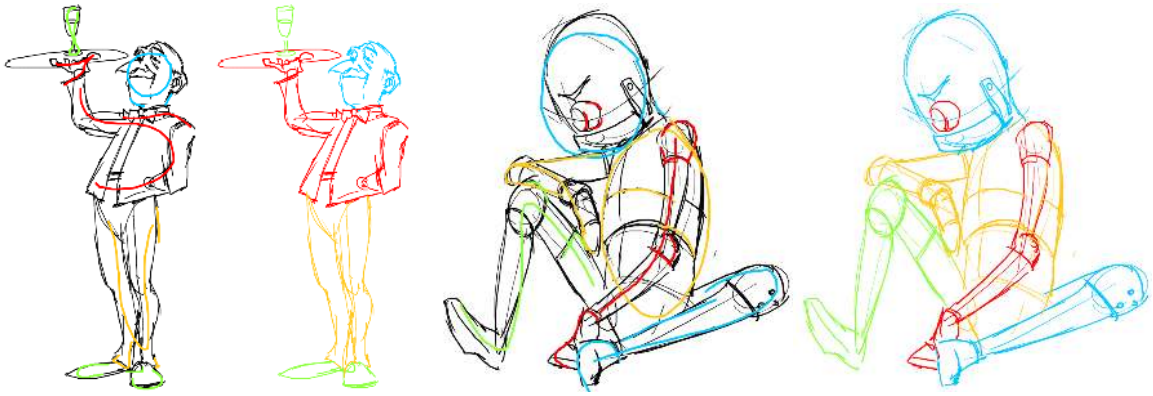

Figure 12: *No-Time-Test: this image shows how the system works when no temporal information is used. Notice how the cluttered regions require more Scribbles to produce a proper segmentation.*

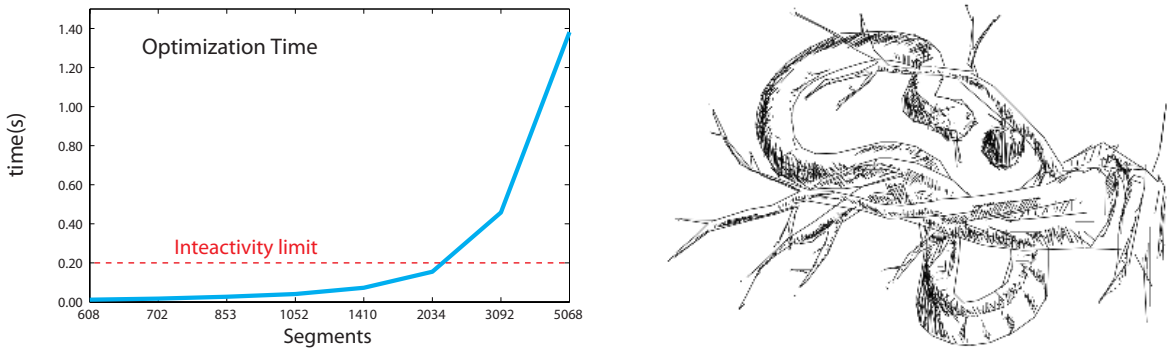

Figure 13: *Performance Limit. The graph shows the computation time for the optimization in our implementation. The data was generated by progressively subsampling a complex drawing. Assuming a interactivity limit of* 0.2 *seconds, our implementation can optimize the labeling for up to 2000 segments. The corresponding subsampled drawing is shown on the right.*

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Transactions on Graphics 23*, 3 (2004), 689–694. 2517

[LS05] LANK E., SAUND E.: Sloppy selection: Providing an accurate interpretation of imprecise selection gestures. *Computers & Graphics 29*, 4 (2005), 490–500. 2517, 2519

[LZC11] LEE Y. J., ZITNICK C. L., COHEN M. F.: ShadowDraw: real-time user guidance for freehand drawing. *ACM Transactions on Graphics 30* (2011), 27. 2525

[NSC*11] NORIS G., SÝKORA D., COROS S., WHITED B., SIMMONS M., HORNUNG A., GROSS M., SUMNER R.: Temporal noise control for sketchy animation. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering* (2011), pp. 93–98. 2525

[Sau03] SAUND E.: Finding perceptually closed paths in sketches and drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 4 (2003), 475–491. 2517

[SC08] SHESH A., CHEN B.: Efficient and dynamic simplification of line drawings. *Computer Graphics Forum 27*, 2 (2008), 537–545. 2525

[SDC09a] SÝKORA D., DINGLIANA J., COLLINS S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering* (2009), pp. 25–33. 2523, 2525

[SDC09b] SÝKORA D., DINGLIANA J., COLLINS S.: LazyBrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum 28*, 2 (2009), 599–608. 2517, 2520, 2522

[SFLM04] SAUND E., FLEET D., LARNER D., MAHONEY J.: Perceptually-supported image editing of text and graphics. *ACM Transactions on Graphics 23*, 3 (2004), 728–728. 2517

[SSJ*10] SÝKORA D., SEDLACEK D., JINCHAO S., DINGLIANA J., COLLINS S.: Adding depth to cartoons using sparse depth (in)equalities. *Computer Graphics Forum 29*, 2 (2010), 615–623. 2523

[WF74] WAGNER R., FISCHER M.: The string-to-string correction problem. *Journal of the ACM 21*, 1 (1974), 168–173. 2517

[WSA07] WOLIN A., SMITH D., ALVARADO C.: A pen-based tool for efficient labeling of 2D sketches. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2007), pp. 67–74. 2517

[WWYM10] WEI J., WANG C., YU H., MA K.-L.: A sketch-based interface for classifying and visualizing vector fields. In *Proceedings of IEEE Pacific Visualization Symposium* (2010), pp. 129–136. 2517