EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2010)
M. Alexa and E. Do (Editors)

Stroke Extraction and Classification for Mesh Inflation

L. Olsen† and F.F. Samavati

Department of Computer Science
University of Calgary

Abstract

We provide a method for extracting and classifying stroke segments from a line drawing or sketch with the goal of producing perceptually-valid output in the context of mesh inflation. This is important as processing freehand sketch input is a fundamental task in sketch-based interfaces, yet many systems bypass the problem by forcing simplified, unnatural drawing patterns. Our stroke extraction combines contour tracing with feature-preserving post-processing. The extracted strokes are classified according to the objects and regions in the sketch: object and region boundaries, interior features, and suggestive lines. The outcome of this classification is demonstrated with examples in feature-sensitive mesh inflation.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Interaction Techniques—[I.3.5]: Surface Representations—Image Processing and Computer Vision [I.4.6]: Edge and feature detection—

1. Introduction

Sketch-based interfaces for 3D modeling (SBIM) are often not "sketch-based" in the strictest sense, but simply offer freehand input. That is, user are expected to draw freeform lines in a particular fashion, instead of sketching freely as they would on paper. The way in which an object is drawn (the number and order of pen strokes) makes little difference in our perception of the object – for instance, both sketches in Fig. 1 are easily recognized as the same object. However, few systems support unconstrained sketching.

Sketch-based mesh inflation tools in particular tend toward a simple single-stroke input for creating an initial mesh, followed by a "sketch-rotate-sketch" workflow (using the terminology of [GIZ09]). While effective, this approach may not facilitate the exploratory and evolutionary aspects of sketching.

In 2D sketch-based applications, there has been a focus on supporting natural sketching [BCF*07, RH08]. Our goal in this work is to adapt some of these ideas to 3D SBIM. The main distinction is that 2D sketch processing is typically concerned with sketch *recognition*, while 3D applications are focused on *reconstruction*. This requires a method tailored to the needs of 3D modeling – to determine "this stroke is on an object boundary", rather than "this stroke looks like a circle."

Figure 1: *Idealized sketches (left; 1 stroke) are more commonly accepted as input than unconstrained sketches (right; 32 strokes).*

In mesh inflation systems such as FiberMesh [NISA07], a sketch is usually interpreted as the boundary contour of a smooth surface. This interpretation, which is consistent with the 'visual rules' described by Hoffman [Hof00], is straightforward for single-stroke input. When the sketch has multiple unstructured strokes, however, we must extract structural information, such as: Where is the boundary of the object? Which lines are inside an object, connoting interior features? Are there stray lines that merely hint at an object or feature?

In order to answer these questions while supporting un-

† olsenl@cpsc.ucalgary.ca

© The Eurographics Association 2010.

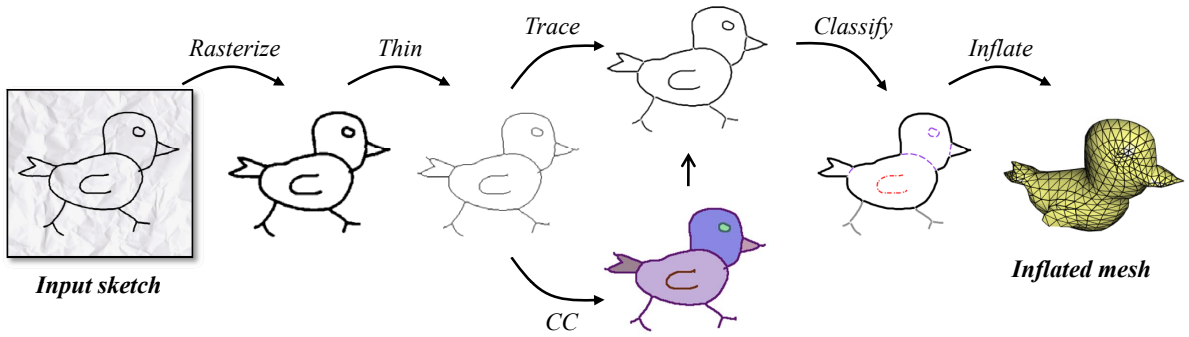


Figure 2: *System overview: a given input sketch is rasterized to a binary image, which is then thinned and traced to extract stroke segments. Connected components (CC) of the thinned image are then used to classify the extracted strokes.*

constrained sketching, we propose a 2-stage sketch processing system (see Fig. 2). First, the input is rasterized and traced to extract contiguous stroke segments (Sec. 2). Second, these segments are classified according to their relative positions and containment within the regions defined by them (Sec. 3). After classification, the strokes can be used to inflate a mesh.

Our main contributions are: a robust stroke extraction method that can operate on both off- and on-line sketch input, such that when online information is available, the algorithm can be tuned to the input nature to better handle 'real' sketch input; and, a novel stroke classification based on region adjacency that is tailored to the needs of mesh inflation applications.

The proposed method is also suitable as a pre-process for a variety of SBIM systems. For example, if only boundary strokes are of interest [IMT99, NISA07] these can be identified and any other strokes can be ignored. Other systems that support non-boundary strokes [OS10] can utilize them to add detail to the output. The stroke connectivity can also be examined to find important patterns such as T-junctions (a ternary branch point with two aligned branches), that are useful in some systems [KH06].

1.1. Related Work

Stroke Extraction. Bartolo et al. [BCF*07] use an image-based approach for stroke extraction, based on multi-scale Gabor filtering to identify salient features, followed by Kalman filtering to vectorize the paths. This method is nicely able to detect gaps between lines (and thus reduce unwanted stroke blending). Rajan & Hammond [RH08] also use an image-based contour tracing approach in a sketch recognition application, predicting the drawing sequence at ambiguous points where the strokes overlap by choosing the minimum angle deviation relative to the incoming direction.

Stroke-based approaches have also been explored. Simhon & Dudek [SD04c] propose a learning-based system for separating domain-relevant strokes from extraneous ones. Sezgin & Davis [SD04a] perform point-based 'thinning' by fitting regression lines to local clusters of the input strokes' sample points. Ku et al. [KQW06] distinguish between straight and curved strokes, then fit lines and conic curves to local stroke clusters. Aoyama & Yamaguchi [AY07] extract single strokes from a set of candidates via criteria such as the stroke with the highest pen pressure, or by combining several lines by weighted averaging. Kara & Shimada [KS07] extract an ordered point sequence from multiple unordered strokes by projecting the points onto the first principal component axis. Pusch et al. [PSNW07] use a similar approach, combined with adaptive space subdivision to handle closed or looping strokes.

While stroke-based methods allow for the use of auxiliary information such as speed and pressure, we argue (as has been argued before [FMK*03, BCF*07]) that an image-based approach is better able to mimic human perception, and also deal with scanned input if necessary. In paper sketches the drawing style can be manifested in perceptually-important ways; for example, pressure variations are manifested as darker or lighter pen marks.

Stroke Classification. After extracting a usable stroke representation from the input, the system *classifies* it according to a set of objects or commands. Olsen et al. [OSSJ09] refer to sketch-based systems as either *evocative* or *constructive* [OSSJ09], differing in how the input is classified.

In evocative systems, the goal is to recognize the input as one of several known objects or templates. For example, strokes can be classified as instances of shape primitives, such as lines, arcs, ellipses, squares, and so on [SD04b]. More complex objects can be recognized as a particular arrangement of primitives [HD05]. Gesture-based interfaces must consider a form of the same problem: matching a time-ordered set of points to a known command. Wobbrock et al.'s $1 classifier [WWL07] compares two point sets by re-

© The Eurographics Association 2010.

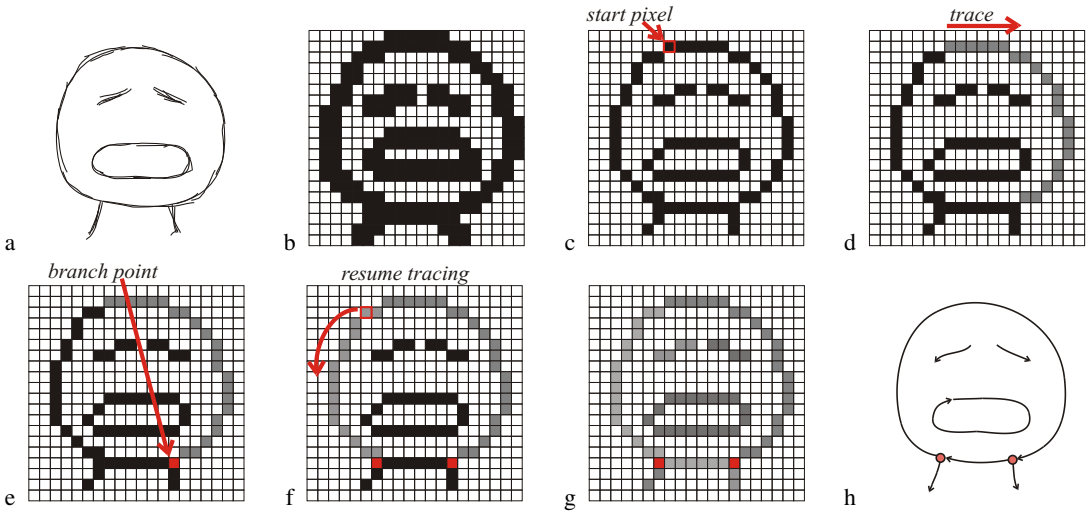


Figure 3: *Tracing algorithm: (a) the input sketch is (b) rasterized and (c) thinned; (d-e)tracing starts at the top-left foreground pixel, and continues until the line terminates or a branch point is found; (f-g) tracing continues in scanline fashion until all strokes and branches are found; (h) the extracted strokes and branch nodes.*

sampling each to the same number of points, normalizing the rotation, scale, and position, and then computing the point-wise Euclidean distance. Olsen et al. [OSS07] propose a similar classification method based on counting the number of segments in a number of angular ranges.

Mesh creation applications are constructive, in that the goal is not to recognize the input (except perhaps in 3D search applications, where the sketch is used to retrieve a similar-looking model [FMK*03]), but rather to construct a model that matches the sketch via a set of rules or modeling operations. For example, CAD-type systems use rules such as 'three lines meeting at a point define a vertex' [LS96], while a mesh inflation system might assume that the sketch defines the boundary contour of a smooth object [IMT99,NISA07]. In most cases, the classification is implicit. For example, in Cherlin et al. [CSSJ05], two strokes define a rotational-blend surface, and adding a third stroke defines the cross-section. Similarly, in FiberMesh [NISA07] the first stroke entered is interpreted as a boundary contour, and subsequent strokes are drawn directly on the object to define features. An example of explicit classification is the SmoothSketch system [KH06], in which the sketch is examined to find *T-junctions* and *cusps* and infer hidden contours.

In this work, we propose a classification system tailored to freeform meshing applications. From an unordered set of strokes, we extract the set of closed regions to classify strokes as being on an object boundary, inside an object, and so on. This allows us to create a smooth mesh from the boundary contour, and also create interior features from the other strokes.

2. Stroke Extraction

In this section, we describe our approach to extracting a set of strokes from an unconstrained sketch. The source image could come from a scanned sketch, but in our research we have used a digitizing tablet device. Our method is similar to previous work based on thinning and contour tracing [RH08]. The main contributions are: the identification of branch points; sharp feature recovery; and, use of on-line sketching information to improve accuracy.

The algorithm is focused on line drawings without shading cues. Thus in the following description, we assume a binary image with white background (paper) pixels and black foreground (ink) pixels; edge detection, thresholding, and other image processing techniques can be applied to attain such an image if necessary [SS01].

Figure 3 depicts the steps of our extraction algorithm. The first step (a-b) is rasterizing the input strokes (that is, drawing them to an image). An important factor in the extraction's success is the *width* w of the strokes – if drawn too thin, then perceptual connections may not be made, but drawing too thick can conversely result in unwanted connections or region closing. (In Sec. 4, we describe how the nature of the input can be used to control w.)

Varying the size $W \times H$ of the raster image changes the relative stroke widths (eg. a width of 4 pixels in a 256×256 raster is equivalent to a width-8 stroke in a 512×512 raster). Changing the raster size has a more prominent impact on running time, since the complexity of scanline algorithms is $O(W \cdot H)$; however, small rasters are less able to resolve small-scale sketch features.

© The Eurographics Association 2010.

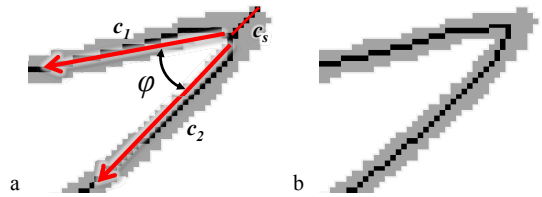


Figure 4: *Sharp features: (a) characterized by a ternary branch point with an acute angle ϕ between the longer branches; (b) after restoring the sharp feature. (The image before thinning shown in gray; thinned image in black.)*

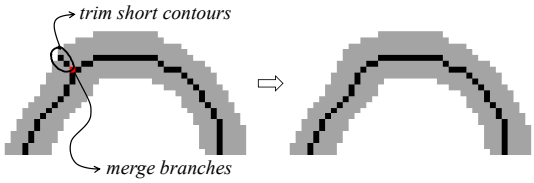
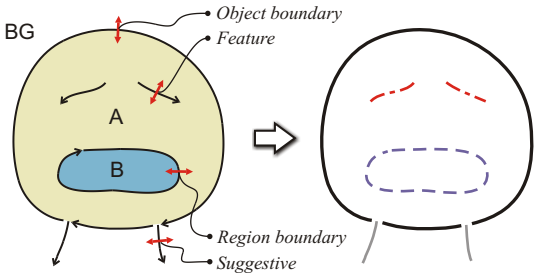


Figure 5: *Post-processing: the extracted strokes are cleaned up by trimming spurious short lines and merging branch points with only 2 connected strokes.*



Class	Style	# BG	# Non-BG
Object boundary		1	1
Region boundary		0	2
Feature		0	1
Suggestive		1	0

Table 1: *Stroke classification is based on the number and type of adjacent image regions.*

After rasterizing, a binary image is generated by applying a Gaussian blur (to soften small-scale noise) and then thresholding the image. Morphological thinning [SS01] is then applied until each line is only a single pixel wide (Fig. 3c); the number of thinning iterations is proportional to w.

We use a modification of the standard contour tracing algorithm [Pav81]. In the algorithm, the image I to be traced is paired with a label image L of the same size; all pixels are initially unlabeled. The goal is to assign a label to all foreground pixels of I such that connected pixels (lines) have the same label. Starting from an unlabeled foreground pixel (Fig. 3c), tracing proceeds by advancing to the first unlabeled (counter-)clockwise foreground neighbor until a termination condition is met – returning to the start pixel, a pixel with no unlabeled neighbors, or a branch point (Fig. 3c-d).

When the algorithm reaches a pixel with multiple foreground neighbors (Fig. 3e), instead of continuing in the default direction, we mark that pixel as a *branch point* and terminate the active line. Tracing then resumes at the next unmarked foreground pixel until the entire image has been scanned (Fig. 3f-g). After tracing, the set of extracted strokes and branch points can be thought of as a graph, with the former as edges and the latter as nodes.

2.1. Post-processing

An unfortunate effect of morphological thinning is erosion around sharp features. When drawing a sharp corner the pen path overlaps, resulting in a thicker ink region – after thinning, this variation results in an unwanted branch point with a short line attached (see Fig. 4a). In most cases (when w is small) the effect is not prominent; however, for very sharp features, or very messy sketches where w must be higher (see Sec. 4), the erosion results in incorrect sketch topology (an extra branch point). Fortunately, these artifacts can be easily identified both visually and algorithmically (Fig. 4a).

We restore sharp features by identifying branch points with three connected strokes (c_1, c_2, and c_s), such that the shortest stroke c_s has a length close to the pen width, and the angle between the other two strokes is acute. When these conditions are satisfied, we discard c_s and then extend c_1 and c_2 to pass through the midpoint of c_s (Fig. 4b).

After tracing all the lines and restoring sharp features, some additional post-processing is done to refine the output. First, very short strokes (with length on the order of w) arise in regions where the rasterized line thickness is not uniform; after thinning, the non-uniformity results in a spurious branch that can be trimmed (Fig. 5). Second, branch points with only two connected strokes – as a result of sharp feature restoration and stroke trimming – can be discarded, and the connected strokes merged.

3. Stroke Classification

After extracting strokes from the sketch, a mesh inflation system needs to how the strokes are perceived as an object. This should consider not just where the strokes are placed, but also the regions they define. For example, consider the sketch in Fig. 3: the two strokes defining the 'eyes' are meaningless if seen alone, but in the context of the other strokes – contained within the head, near the mouth – we can perceive them as features of a larger object. Thus our classification must consider concepts such as containment and adjacency.

© The Eurographics Association 2010.

To do this, we use the connected components (CC) labeling algorithm [SS01] on the rasterized stroke image, which assigns a unique label to contiguous image regions. This approach is similar to [OS10], but our classification is more generic and complete, able to handle sketches with overlapping and crossing strokes when coupled with robust stroke extraction. A stroke-based method could also be used, but the CC algorithm fits well in an image-based approach with complexity dependent on the raster size, not the number of strokes or regions. An example labeling is shown in Table 1.

Given that strokes are terminated at branch points, each stroke must be adjacent to two regions – it could not be adjacent to more regions, as any point where 3 or more regions meet would be a branch point. Thus it is sufficient to check the regions adjacent to a stroke at any point along it; we use the stroke's midpoint.

In a CC labeling, there are only two region types: background (BG), and interior (INT; labeled *A*, *B*, ... in the figures). Since each stroke is adjacent to two regions, the possible adjacency arrangements are: BG-BG, BG-INT, and INT-INT. In the last case, the interior region can be the same, or two different regions. Thus there are a total of four possible arrangements, or classes. As illustrated in Table 1, we have named these classes as *object boundary* (BG-INT), *region boundary* (INT1-INT2), *interior feature* (INT-INT), or *suggestive* (BG-BG).

In the example in Table 1, the sketch has three regions: *A*, *B*, and background *BG*. The circular 'head' strokes are adjacent to *BG* and *A*, and so are classified as object boundaries. The 'eyes' are each adjacent to only *A*, and so are features. The 'mouth' stroke is adjacent to both *A* and *B*, defining a region boundary, while the 'neck' strokes are adjacent to only *BG* and so are only suggestive.

Class information is useful in a mesh-inflation application. The object boundary indicates the region to be filled with mesh faces, while region boundaries and features can be used to create a constrained triangulation in which edges follow internal lines. Region boundaries, when closed and not adjacent to any object boundary, define potential holes in the object. Suggestive lines may or may not be useful to a particular application; for example, they could be replaced by "pipes" in 3D, or aligned pairs could be used to construct rotational blending surfaces [CSSJ05].

In addition to stroke-region adjacency, region-region adjacency can provide insight into object structure, such as the location of holes or the presence of multiple objects. Consider Fig. 6: region *A* is adjacent to both *BG* and *B*, while *B* is only adjacent to *A*; thus *B* is contained within *A*. All of the non-BG regions are connected, and thus define a single object. In general, any region not adjacent to BG is a potential hole, and each connected set of non-BG regions define a unique object in the sketch.

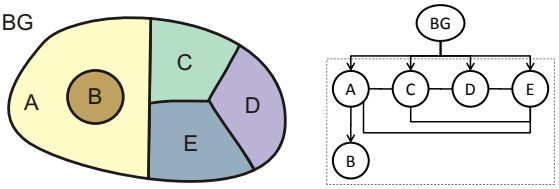


Figure 6: *A region hierarchy, constructed from the stroke-region adjacency information, is useful for identifying potential holes in an object.*

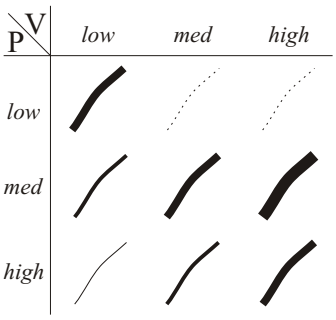


Figure 7: *Parameter tuning based on the input stroke's pressure (***P***) and velocity (***V***); dashed lines indicate strokes that are not rasterized (useful as guidelines).*

4. Parameter Tuning

The stroke extraction algorithm previously described can work well in an offline environment, i.e. when only the stroke image is available. In an online usage scenario where the user is creating their sketch within a computer application, additional information (speed, pressure, orientation) may be available. This information can allow us to tweak our algorithm to attain better results.

For example, it has been observed that artists will often sketch iteratively, at first making light, hasty strokes to define rough boundaries and guidelines, then tracing these rough lines with harder, more deliberate strokes [SIJ*07]. The viewer naturally perceives the harder (and thus darker and perhaps thicker) strokes as being more important, and uses these lines to understand the sketch.

Using this observation of artistic drawing tendencies, we assert that when the stroke speed and pressure are available, they can be somehow mapped to perceptual concepts such as "importance" and "trustworthiness." Strokes with very low pressure or high speed can be de-emphasized or ignored, while strokes with high pressure or low speed can be trusted as truly capturing the artist's intent.

In our implementation, we map each stroke's average pressure and speed to the rasterized width *w*. Pressure (**P**) and velocity (**V**) are quantized to 3 levels: low, medium, and high. The intuition is that low pressure and high velocity cor-

© The Eurographics Association 2010.

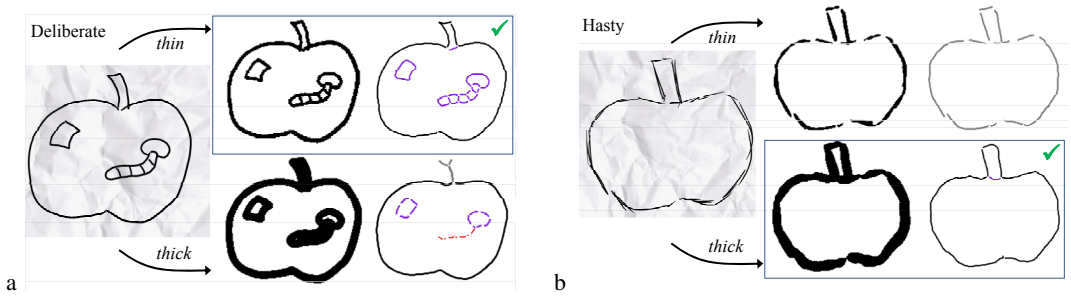


Figure 8: *Parameter tuning: (a) for deliberate sketches, thin rasterization (top) is best; (b) for hasty sketches, thicker lines (bottom) that overlap and blend together lead to better results. Reversing the settings can cause unwanted blending (eg. the stem in (a-bottom)) or misclassification (b-top).*

relate to 'hasty' strokes, while high pressure and low velocity correlate to 'deliberate' strokes. Thus, in the former case a larger width w should be used to create perceptual connections and overlaps, while in the latter case a narrower w can be used to preserve the original input. This style-to-width mapping is illustrated in Fig. 7. Note that the width adjustment does not change what the user sees – it only happens internally for the purposes of stroke extraction.

Figure 8 shows an example of two apples sketched in different styles. After parameter tuning, the deliberate sketch is rasterized with thinner strokes to preserve the fine details, while the hasty sketch's strokes are drawn thicker to blend together. In each case, the strokes are extracted and classified successfully. Reversing the settings (thin lines in the hasty sketch, and vice versa) can lead to misclassification and unwanted stroke blending.

5. Results & Discussion

Figure 9 shows some example sketches gathered from a number of artist and non-artist sources, along with the result of extraction and classification. The line style for each stroke class is as listed in Table 1. It can be seen that the algorithm produces a clean and accurate set of classified strokes.

The classified strokes are used to create a 3D mesh, via a 3-step inflation process [OS10]. First, a planar mesh is constructed from the constrained Delaunay triangulation of all non-suggestive input strokes, with critical points on the strokes serving as Delaunay vertices. Then, the planar mesh is subdivided and vertices are displaced to align with the strokes. Finally, a smooth 3D mesh is created by displacing vertices away from the plane, with boundary vertices fixed and interior vertices displaced proportional to their distance to the boundary. Suggestive lines have been replaced with cylindrical pipes.

This creates meshes whose edges and vertices follow all sketched lines, not only on the boundary (Figure 10). This allows for feature-based editing – such as raising the fish's eyes, and creating a hole in the guitar body – without costly re-meshing operations.

	Strokes		Time (s)		
Name	In	Out	Thin (Iters)	Trace	CC
Apple	43	3	0.266 (7)	0.005	0.016
Snake	13	15	0.765 (5)	0.063	0.031
Face	55	37	0.765 (5)	0.063	0.031
Cow	20	28	1.109 (7)	0.094	0.047
Hand	51	44	0.610 (4)	0.063	0.047
Turtle	87	48	1.890 (12)	0.063	0.047

Table 2: *Timing results for the sketches in Fig. 8b and Fig. 9. The raster size is* 512^2 *for each, except the apple (*256^2*).*

Discussion. A marked benefit of an image-based approach is that the complexity is related to image resolution, rather than the number of input strokes. This can be seen in the timing results listed in Table 2. The time required to trace the strokes and extract the connected components varies with image size, generally requiring 50-80 milliseconds. Morphological thinning takes the bulk of the algorithm's time, typically around 0.5 seconds depending on the number of iterations required (which in turn depends on the width of the rasterized strokes). Stroke classification requires only two pixel lookups per stroke, and effectively required zero time in our test cases. By reducing the raster size, the running time would be suitable for devices with limited computing power (at the expense of precision).

Our implementation is sub-optimal in a couple of ways that could be remedied for an end-user application. First, the algorithm is implemented in C# with an external library for thinning; replacing this with a native C implementation would streamline the most time-consuming component. Second, the *Trace* and *CC* algorithms are implemented as separate passes over the image, but could be combined into a single pass [CC03].

There are both method-specific and fundamental limitations to stroke extraction. In the former case, morphological thinning tends to introduce spurious and perceptually-unimportant short branches where the line thickness varies. In particular, this happens where multiple strokes overlap and at corners (Fig. 11a). The occurrence of these branches

© The Eurographics Association 2010.

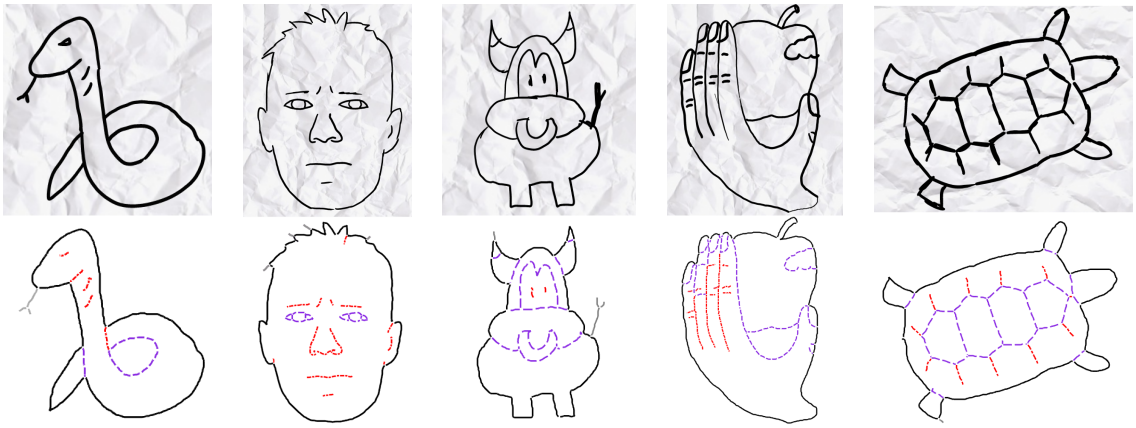


Figure 9: *Extraction and classification results from our system (number of input strokes in parentheses): snake (13), face (55), cow (20), hand (51), turtle (87).*

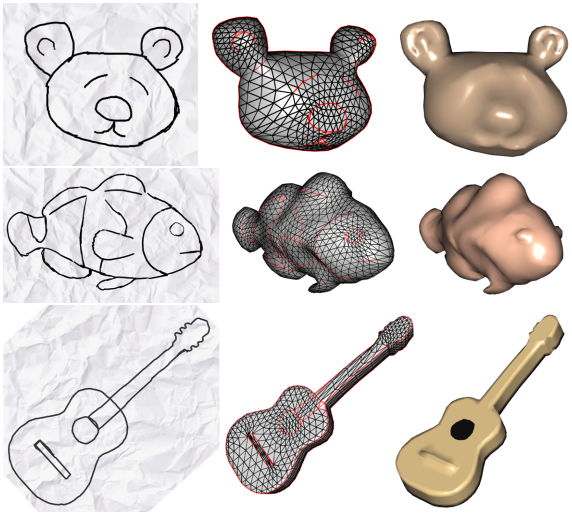


Figure 10: *Meshing results: (left) input sketch; (middle) the generated mesh; (right) after some feature-based editing.*

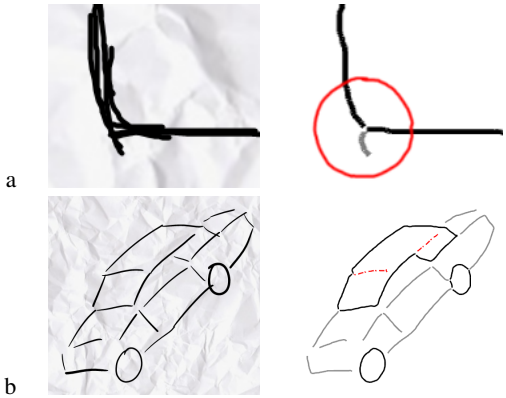


Figure 11: *(a) Even after processing, some short branches may persist; (b) When the sketch has large inter-stroke gaps, region boundaries may be missed because the strokes do not define closed regions.*

is reduced when the initial stroke rasterization has thin lines to begin with. However, 'messy' sketches often require a thicker rendering to induce the perceptual connections and crossings after thinning. We try to strike a balance by characterizing each input stroke's trustworthiness during online sketching, but classification errors can still occur with very 'messy' sketches (Fig 11b).

This also illustrates an inherent limitation of sketch processing. While most human observers would recognize the sketch as an automobile and be able to label regions such as the tires, hood, and fenders, this is not explicitly conveyed by the sketch. It could represent any number of things, such as a collection of chopsticks and plates. Our interpretation of the image depends on our vast shape knowledge. Bringing this knowledge to bear on the problem of sketch processing is a gargantuan task (though there has been work in that direction [YSvdP05]). Our position is that a system that respects the user's explicit stroke structure is more reliable than a system that attempts to infer, perhaps incorrectly, implied features.

6. Conclusion & Future Work

We have proposed a method for extracting and classifying perceptually-meaningful strokes from either a scanned line drawing or an unordered collection of user strokes. Salient strokes are extracted via an image-based thin-and-trace approach, with branch points identified and used to construct a stroke graph. We then classify each extracted stroke based on the number and type of adjacent regions.

The ability to process and classify complex sketches for inflation can be a great benefit for sketch-based modeling,

© The Eurographics Association 2010.

as it allows artists to construct their sketch 'on the paper' before transitioning to 3D for further editing.

In the future, the parameter tuning could be improved to handle a broader range of sketches. For offline sketches, it would be beneficial to characterize how a stroke was drawn – in terms of velocity and pressure – based on its appearance. Sketches with large inter-stroke gaps can also be problematic, as well as those with 'implicit' stroke crossings. Perhaps some combination of multi-scale extraction and line extrapolation could be used to handle such cases. As well, a formal user study should be conducted to evaluate the 'perceptual' correctness of our methods.

Artifacts caused by thinning – unwanted branching and erosion of features – can be troublesome for very complex sketches, and alternative methods to stroke extraction should be explored. Extending the stroke extraction beyond line drawings, to sketches with hatching or shading marks, is also an interesting problem worth considering.

Acknowledgments This research was supported by the National Science and Engineering Research Council (NSERC) and GRAND NCE of Canada, and the Informatics Circle of Research Excellence (iCore) of Alberta.

References

[AY07] AOYAMA H., YAMAGUCHI H.: *Sketch Based Modeling System*, vol. 4563/2007 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2007, pp. 421–430. 2

[BCF*07] BARTOLO A., CAMILLERI K. P., FABRI S. G., BORG J. C., FARRUGIA P. J.: Scribbles to vectors: preparation of scribble drawings for cad interpretation. In *Proc. of Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM '07)* (2007), pp. 123–130. 1, 2

[CC03] CHANG F., CHEN C.-J.: A component-labeling algorithm using contour tracing technique. In *Int. Conf. on Document Analysis and Recognition (ICDAR'03)* (2003), vol. 2. 6

[CSSJ05] CHERLIN J. J., SAMAVATI F., SOUSA M. C., JORGE J. A.: Sketch-based modeling with few strokes. In *Proc. of Spring Conference on Computer Graphics (SCCG '05)* (2005), pp. 137–145. 3, 5

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3d models. *ACM Trans. Graph. (Proc. of SIGGRAPH '03) 22*, 1 (2003), 83–105. 2, 3

[GIZ09] GINGOLD Y., IGARASHI T., ZORIN D.: Structured annotations for 2d-to-3d modeling. In *Proc. of SIGGRAPH Asia 2009* (2009), pp. 1–9. 1

[HD05] HAMMOND T., DAVIS R.: Ladder, a sketching language for user interface developers. *Elsevier Computers and Graphics 28* (2005), 518–532. 2

[Hof00] HOFFMAN D. D.: *Visual Intelligence: How We Create What We See*. W. W. Norton & Company, 2000. 1

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3d freeform design. In *Proc. of SIGGRAPH'99* (1999). 2, 3

[KH06] KARPENKO O. A., HUGHES J. F.: Smoothsketch: 3d free-form shapes from complex sketches. In *Proc. of SIGGRAPH '06* (2006), pp. 589–598. 2, 3

[KQW06] KU D., QIN S., WRIGHT D.: Interpretation of overtracing freehand sketching for geometric shapes. In *Proc. of Int. Conf. on Computer Graphics, Visualization and Computer Vision (WSCG '06)* (2006), pp. 263–270. 2

[KS07] KARA L. B., SHIMADAĂ K.: Sketch-based 3d shape creation for industrial styling design. *IEEE Computer Graphics and Applications 27*, 1 (2007), 60–71. 2

[LS96] LIPSON H., SHPITALNI M.: Optimization-based reconstruction of a 3d object from a single freehand line drawing. *Computer-Aided Design 28*, 8 (1996), 651–663. 3

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: Designing freeform surfaces with 3d curves. In *ACM Transactions on Graphics (Proc. of SIGGRAPH '07)* (2007), ACM Press. 1, 2, 3

[OS10] OLSEN L., SAMAVATI F. F.: Image-assisted modeling from sketches. In *Proc. of Graphics Interface (GI '10)* (2010). 2, 5, 6

[OSS07] OLSEN L., SAMAVATI F., SOUSA M. C.: Fast stroke matching by angle quantization. In *Proc. of the First Intl. Conference on Immersive Telecommunications (ImmersCom 2007)* (2007). 3

[OSSJ09] OLSEN L., SAMAVATI F., SOUSA M., JORGE J.: Sketch-based modeling: A survey. *Computers & Graphics 33* (2009), 85–103. 2

[Pav81] PAVLIDIS T.: *Algorithms for Graphics and Image Processing*, 1 ed. Computer Sciences Press, 1981. 4

[PSNW07] PUSCH R., SAMAVATI F., NASRI A., WYVILL B.: Improving the sketch-based interface: Forming curves from many small strokes. *The Visual Computer 23*, 9-11 (2007), 955–962. 2

[RH08] RAJAN P., HAMMOND T.: From paper to machine: Extracting strokes from images for use in sketch recognition. In *Proc. of Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM '08)* (2008). 1, 2, 3

[SD04a] SEZGIN T. M., DAVIS R.: Handling overtraced strokes in hand-drawn sketches. In *Making Pen-Based Interaction Intelligent and Natural* (2004). 2

[SD04b] SEZGIN T. M., DAVIS R.: Scale-space based feature point detection for digital ink. In *Making Pen-Based Interaction Intelligent and Natural* (October 21-24 2004), AAAI Fall Symposium, pp. 145–151. 2

[SD04c] SIMHON S., DUDEK G.: Pen stroke extraction and refinement using learned models. In *Proc. of Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM '04)* (2004). 2

[SIJ*07] SCHMIDT R., ISENBERG T., JEPP P., SINGH K., WYVILL B.: Sketching, scaffolding, and inking: A visual history for interactive 3d modeling. In *Proc. of the Int'l Symposium on Non-photorealistic Animation and Rendering (NPAR'07)* (2007), pp. 23–32. 5

[SS01] SHAPIRO L., STOCKMAN G.: *Computer Vision*. Prentice Hall, 2001. 3, 4, 5

[WWL07] WOBBROCK J. O., WILSON A. D., LI Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. of ACM symposium on User interface software and technology (UIST '07)* (2007), pp. 159–168. 2

[YSvdP05] YANG C., SHARON D., VAN DE PANNE M.: Sketch-based modeling of parameterized objects. In *Proc. of Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM '05)* (2005). 7

© The Eurographics Association 2010.