In Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.

A Representation of Image Structure and Its Application to Object Selection Using Freehand Sketches

Kar-Han Tan
Department of Computer Science
and Beckman Institute

Narendra Ahuja
Department of Electrical and Computer Engineering
and Beckman Institute

University of Illinois at Urbana-Champaign
{tankh|ahuja}@vision.ai.uiuc.edu

Abstract

We present an algorithm for computing a representation of image structure, or image segmentation, and use it for selecting objects in the image with freehand sketches drawn by the user over the image. The sketches are mapped onto image segments whose union forms the intended object. The mapping operation is performed with the aid of a simplicial decomposition of the image segmentation - a triangulation formed with vertices chosen to lie along the medial axes of the segments. Each edge of a triangle lies entirely inside the two segments that contains its vertices. This decomposition captures the adjacency information about the segments as well as the shape of the segment boundaries. Any object boundary is completely contained in a set of triangles. The triangles are also used to formulate the problem of estimating gradual photometric transition across an object boundary, called alpha channel estimation, as a set of local, intratriangle alpha channel estimation problems that can then be solved more accurately, independently, and in parallel. Experimental results are included to show how the algorithm allows selection of image objects with complex boundaries using roughly drawn simple sketches.

1 Introduction

Selection in digital image editing is the task of extracting an object embedded in an image, and is performed frequently in many visual content creation applications. Artists creating magazine covers routinely extract people or products from photographs to remove unwanted background and compose the new images such that magazine titles appear to be occluded by the object naturally. Existing tools for creating selections focus primarily on the quality of the end result, and not the ease with which objects may be selected. As a result, most existing tools require skillful user guidance and typically cannot be used when the user does not have much time, such as during a live sports broadcast, or do not have much dextrous control, such as on a mobile handheld device, or simply want to avoid the monotony and tedium of selection, without compromising the quality of the selection. To enable the use of sophisticated and high-fidelity graphical selection and manipulation operations in these situations, we propose a new tool that allows a user to select objects with loosely hand-drawn sketches similar to those that humans use to communicate with one another.

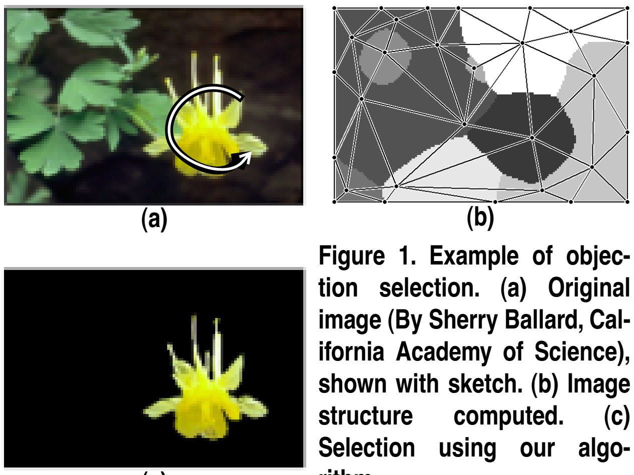

Figure 1. Example of objection selection. (a) Original image (By Sherry Ballard, California Academy of Science), shown with sketch. (b) Image structure computed. (c) Selection using our algorithm.

Figure 1 shows an example of selection with the new tool. The sketch drawn by the user is shown in Figure 1(a), and the resulting selection found is shown in Figure 1(c). Evidently, the sketch can be made quickly and without much skill, but the quality of the selection is not compromised. We believe that such an interface would be very natural for use in the emerging class of 'tablet' computers, and will easily find applications in image and video editing applications.

2 Related Work

The original inspiration for our work came from PerSketch[17, 18], an augmented simulation of whiteboard sketching which operates exclusively on line drawings. The system automatically analyzes line drawings made by the user and allows access to the underlying structure of the line drawing, enabling objects to be selected with intuitive gestures. Generalizing PerSketch from line drawings to raster images is however non-trivial because fully automatic analysis of an image and its decomposition into corresponding primitive elements remain subjects of current research[1, 10, 12, 20, 22].

Interactive selection tools for still images have been attracting significant interest from both researchers and commercial developers in recent years[2, 9, 13]. The Intelligent Scissors[13] and Image Snapping[9] techniques are tools that assist users by detecting high-contrast edges in the vicinity of user-indicated locations, thus relieving users of the need to trace over the exact boundaries. Snakes[11] may also be placed in this same category. A common feature of these methods is that the boundary is sharp so that every pixel is either inside a selected object or is completely unselected. In most photographic images, however, boundaries are not as clearly localized. This is true even in high-quality stock photographs such as those from the Corel Stock Photography collection[5]. Object boundaries are frequently not defined by step edges, for example when the object is not in focus (intentionally or otherwise) or is in motion. Sometimes objects, such as wispy strands of hair, are simply too small to be represented fully by a finite-resolution digital image. Examples of diffused object boundaries can be seen in Figure 2.

One approach to account for diffused boundaries is to model them as blurred step discontinuities in image intensity. This model has also been used in an interactive image editing system, ICE, which allows objects to be deleted seamlessly from intensity images[7]. More conventionally, this problem is addressed by the use of an alpha channel[15] to represent the selection. An alpha channel, in our context, is a real-valued map with one entry corresponding to each pixel ranging from zero(not selected) to one(fully selected). It is the representation used with great success in blue screen matting[19], and a number of commercial selection tools that produce selections in the form of alpha channels[3]. While the techniques used by these commercial tools are not published, recently an algorithm that estimates the alpha channel in the vicinity of object boundaries was proposed[16]. With this 'Alpha Estimation' algorithm, the colors of pixels in the vicinity of object boundaries are mixtures of colors from the foreground object and the background object. It has been shown to be able to extract objects with detailed boundaries, given samples of 'pure' foreground and background, and boundary pixels. The algorithm uses a mixture model to estimate the alpha channel value at all the boundary pixels.

3 Overview

We present an object selection algorithm summarized in Figure 3. The inputs to the algorithm are the image containing the object to be selected and the freehand sketches drawn by a human user over the image. The output is the selection in the form of an alpha channel, with each pixel having a real value ranging from zero (completely not selected) to one (fully selected). A previous paper[21] presented parts of this work. This paper complements the work reported there. The previously reported work is pointed out in this section, and is excluded from the rest of this paper.

Our algorithm involves solving two problems: mapping the sketches to the appropriate object in the image, and computing the alpha channel representation for the object. These two problems are solved using a representation of the image

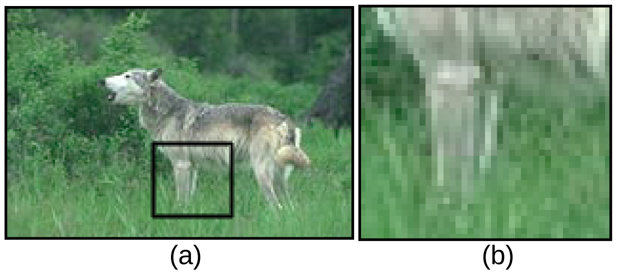

Figure 2. Examples of diffused object boundaries. Image is taken from the Corel Stock Photography Collection. (a) Original image, the boxed portion of which is shown in detail in (b), revealing examples of diffused edges. For high-fidelity image editing, the object boundary details need to be fully captured with an alpha channel.

structure that is extracted automatically from the image. This representation consists of a segmentation and a segmentation-guided decomposition of the image into triangles whose vertices and edges reflect the shapes and spatial adjacency of the segments, and is the main focus of this paper. To provide motivation and context, we now briefly describe the algorithms that make use of this representation. Details may be found in [21].

First, we consider the problem of mapping freehand sketches to objects in the image. We allow the user to draw sketches consisting of *points*, *lines*, and *loops*. Points are typically mouse clicks, and are typically used to select small objects. Lines are non-self-intersecting curves and used to indicate object boundaries. Loops are curves that indicate the spatial extent of objects. We allow the user to specify the selection with different degrees of precision, according to the complexity of the image, so that where there is less ambiguity the user sketches can be less precise. Figure 3 shows this step as *sketch processing*, and its output is a set of segments whose union forms an *initial selection*, and a set of triangles whose vertices straddle the boundary of the selection. These *boundary triangles* also completely cover the boundary of the selection, and thus decompose the spatial vicinity of the boundary. For details on sketch processing, see [21].

The initial selection and the boundary triangles are then processed for *local alpha estimation*. The objective of this stage is to compute the final selection using the alpha estimation algorithm[16], a method for factoring out the foreground objects' contribution to a pixel's value when the pixel value is a mixture of the foreground object and the background. The alpha estimation algorithm, as originally proposed, assumes that pure samples of the foreground and background are provided by the user. We automate this process by using the initial selection to get the pixel samples. The problem of estimating the alpha channel over the entire boundary is also broken into small, local subproblems using the boundary triangles. The final selection is then obtained by combining the alpha channels computed within each boundary triangle. For additional detail and discussion on

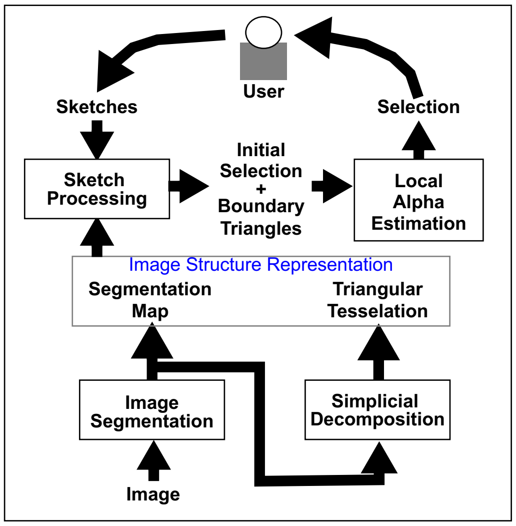

Figure 3. Schematic of the object selection algorithm.

the local alpha channel estimation algorithms, see [21].

In this paper, we present our algorithm for image segmentation (section 4), and decomposition of the segmentation that is used to derive the set of boundary triangles (section 5). Together these parts comprise the image structure representation as shown in Figure 3. Experimental results using the new algorithm are presented in section 6.

4 Image Segmentation

We use a segmentation of the image to represent groups of pixels that form objects or parts of objects. The algorithm we use is based on binary-split vector quantization in color space. For each pixel in the image, we create a three-tuple (r, g, b), with one component for each color space component. Initially the data points will all be placed in a single cluster. We split this cluster into two at its mean along the direction of largest variation, and recursively split the resulting clusters until the number of data points in each is below a given threshold. Typically we let the threshold be one half, quarter, eighth, or sixteenth of the number of pixels in the image. The cluster label for the pixels thus forms a raw segmentation map. We then apply a morphological ‘majority’ filter that replaces the label of a pixel by the value that occurs most frequently within a square window centered at the pixel. We then relabel the pixels so that each 4-connected component in the segmentation map has a unique label. The segmentation computed by this procedure is an approximate representation for the spatial extent of objects in the original image, although shape details are lost due to the morphological filters employed. However, this form of segmentation is adequate for our purposes.

5 Simplicial Decomposition

The segmentation map provides a pixel-resolution

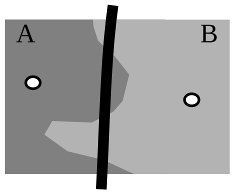

Figure 4. Using points to represent the relative positions of regions.

approximation of the objects in the image. While this form of representation of image structure is useful for many applications, it is still not sufficient for mapping the sketches to the segments. For example, it is still difficult, in general, to answer the question: “is a segment to the left side or right side of a line?” when the line crosses the segment boundary. This is, of course, the type of questions we need to answer in order to map freehand sketches onto image segments. Figure 4 illustrates the way we answer the above question. The two regions A and B have a curved shared boundary, which intersects the thick black line, so that no segment is strictly on one side of the line. However if we can represent the two segments by two points, then there will be no ambiguity unless the line passes through the points. Obviously, such a representation is only applicable in the local vicinity of the points. In order to fully represent the shape and spatial configuration of a segmentation, we also need to decompose it into small, manageable pieces such that the geometric queries necessary for the object selection problem can be answered.

In this section, we will describe how such a representation can be computed. We use a simplicial decomposition of the image into triangles. In order for the triangles, edges and vertices to reflect the structure of the segmentation map, we require the triangulation to satisfy the following constraints:

Triangulation Constraints
1. *All vertices lie on the medial axes of the segments.*
2. *For an edge between two vertices, one in region A and one in region B, the edge must lie entirely in the union of the two regions.*
3. *Each triangle must be contained in at most three regions.*

.

The first property ensures that vertices are always ‘inside’ the respective segments they are representing and are not positioned too near to boundaries. The second property ensures that the triangles formed reflect the adjacency relationship well, in the sense that an edge is always between two vertices representing adjacent segments. The third constraint ensures that segment boundaries are always enclosed by vertices representing the respective segments. Figure 5 illustrates the second property. It is worth noting that the example edge $e3$ cuts the region boundary more than once, but is still considered a valid edge because it lies entirely in the union of the two segments. Edge $e4$ is invalid because its two vertices lie in the same region, but the edge passes through two different regions.

The algorithm for triangulating the segmentation map is

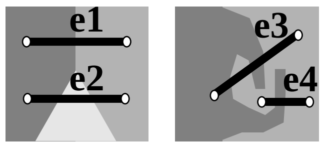

Figure 5. Examples of valid and invalid edges in the desired triangle decomposition: Edges e1 and e3 are valid, while edges e2 and e4 are invalid.

essentially a procedure for systematically covering the segment boundaries while choosing points and adding triangles that satisfy the constraints. We need to define a few terms to facilitate the description: a pair of adjacent regions meet at their shared boundaries. We call each continuous piece of the shared boundary an *opening*. The word opening is used because in a triangulation there will be at least one edge passing through each opening, and these edges can only pass through this piece of shared boundary between the two regions-an opening. *Junctions* are points in the segmentation map where three or more regions meet. In the case of a digital image, where pixels are on a rectangular grid, at most four regions can meet at a point. Examples of openings and junctions are shown in Figure 6. Openings are shown as solid black lines and junctions white dots. Openings X and Y are on the share boundaries of the same regions, but the two are distinct. Opening Z forms a closed loop. Openings that do not form loops will have two end points, each one either a junction or a point at the edge of the image. Each opening thus can have two, one, or no endpoints. The main steps of the decomposition algorithm are as follows:

Triangulation Algorithm

1. Cover each junction with a triangle.
2. Merge vertices within each segment as much as possible.
3. Cover openings with 2,1 or 0 endpoint(s).
4. Cover the remaining area in the image.

Figure 7 illustrates the steps in the algorithm. We now discuss the individual steps in more detail.

5.1 Covering Junctions

Triangle placement starts at the junctions. Where three regions meet, we pick one vertex from each of the three region skeletons corresponding to the junction such that they form a triangle with edges that are valid. At junctions where four regions meet, we place two triangles. The two cases are shown in Figure 8. We use the following goodness measure for a triangle:

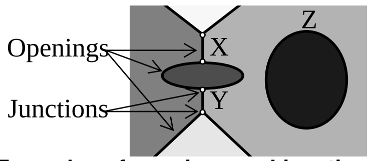

Figure 6. Examples of openings and junctions. The diagram shows a segmentation map with six segments, shown in different shades of gray.

$$\frac{\sum_{i=1\ldots3} w_i}{C+P}\theta_{min}$$

where

w_i is the *depth* of a vertex i, its distance from the nearest boundary of its containing segment

θ_{min} is the minimum of the three internal angles of the triangle

P is the perimeter of the triangle

C is a constant

Thus we favor small triangles with large internal angles, and vertices that are positioned deep inside their respective segments. Search for the triangle proceeds in the following manner: we rank all points on the respective segment medial axes by their depths, so that vertices with large depths are highly-ranked. The search is then constrained by using only the points in the 90 percentile of each of the segment medial axes involved. With this small set of points, we examine all possible triangles and pick the best. If no triangle is found due to the vertex weight constraint, we drop to a lower percentile and redo the search. Figure 9(a) illustrates a typical set of triangles satisfying the tesselation constaints found with this search procedure.

Figure 9(b) shows another set of triangles satisfying the same constraints, but using fewer distinct vertices. We obtain this latter result by a procedure that merges all possible vertices within a segment. For each vertex v, we call the two vertices on the same triangle its neighbour vertices. We say that two vertices can be connected if there is a valid edge between the two vertices. Two vertices can be merged if they each can be connected to the neighbours of the other vertex. Generalizing, a vertex can be added to a set of vertices with a neighbour set formed by the union of all the neighbours of the vertices of a set if the vertex can be connected to all vertices in the neighbour set and all vertices in the set can be connected to the two neighbours of the new vertex.

We use a greedy procedure to identify a number of maxi-

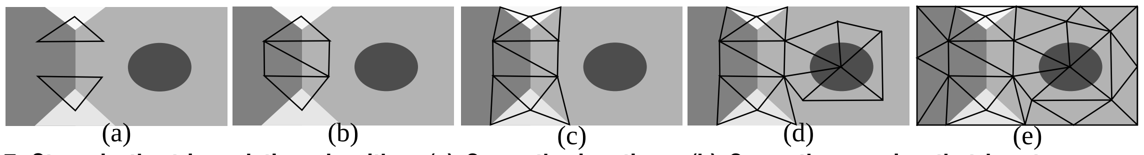

Figure 7. Steps in the triangulation algorithm. (a) Cover the junctions. (b) Cover the opening that has two covered end points. (c) Cover the openings that has one covered end point. (d) Cover the openings with no end points (in this case a loop). (e) Cover the remaining parts of the image between the hull of the triangles and the border of the image.

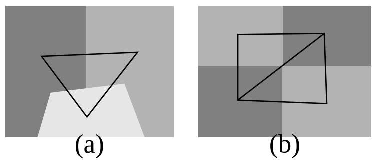

Figure 8. Placing initial triangles at junctions. (a) At a three-region junction, one triangle is placed. (b) At a four-region junction, two triangles are placed. In a digital image, these two cases are exhaustive.

mal merge sets by starting with a single vertex and then incrementally adding to the merge set vertices that can be merged with the vertices in the set. A merge set is maximal when no vertices from the same segment can be added to it. We then merge the vertices in the set and start growing the next merge set by a remaining vertex not in the merge set. We repeat this until all vertices have been included in a merge set. In the worst case, all merge sets contain only one vertex, and no merging occurs.

5.2 Openings with Two End Points Covered

Since the initial set of triangles is placed at junctions, and junctions are endpoints for openings, each initial triangle thus cover the endpoints for either three or four openings. This property is retained by the new set of triangles obtained by the vertex merging operation. Unless an opening ends at the border of the image or forms a closed loop, both its end points would be covered by these initial triangles. We identify such openings and place additional triangles between the two initial triangles at its two ends to completely cover these openings. Figure 10 shows such an opening, with its two end points covered, leaving the middle portion still uncovered. It can be seen from the diagram that the area in which the triangles need to be placed is bounded by the two skeletons, and the two triangle edges at the two ends. This suggests that we can cover the opening by traversing along the opening from one end to the other, placing new triangle vertices on the two opposite skeletons. The traversal algorithm we use is as follows:

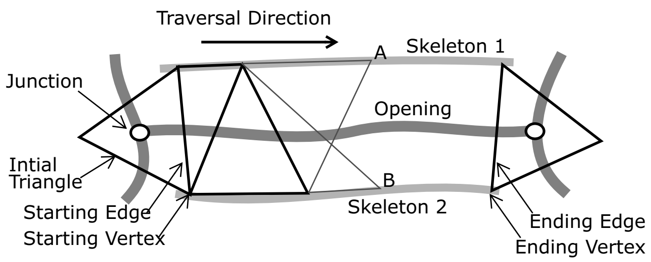

Figure 10 Traversing an opening and covering it with triangles. Triangles that have been placed are shown with thick edges, while the candidate triangles are shown with thin edges.

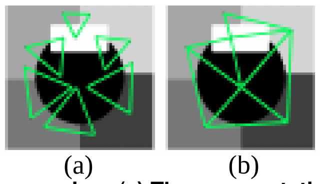

Figure 9. Vertex merging. (a) The segmentation map, shown with the initial set of triangles placed at the junctions. (b) The result of vertex merging.

Adding Triangles by Skeleton Traversal

1. Given an opening with its two end points covered by two triangles, identify the two skeletons, and the starting and ending triangle edges.
2. The starting and ending edges each have a vertex on the two skeletons. These are the starting and ending vertices for each of the skeleton. Use Dijkstra's shortest path algorithm to find a path on each skeleton from the starting vertex to the ending vertex.
3. On each of the skeleton paths, traverse the path. At each point along the traversal, while the following conditions are true:
    a. The point is connected to the two vertices of the starting vertex (and thus is able to form a candidate triangle using the starting edge and the new point).
    b. In the new candidate triangle, the angle at the new point is larger than a preset value θ_{min}. This avoids the creation of thin 'sliver' triangles.
    c. The new point has not reached the end of the path.
4. Given the two new candidate triangles, pick the one that covers more of the opening. Add the new triangle to the existing set. Add new triangles on the other side of the skeleton on which the new point was added, using the new point.
5. Let the new starting edge be the new edge between the two skeletons that is added with the new triangle.
6. Repeat steps 3-5 until the starting edge coincides with the ending edge.

5.3 Openings with One End Point Covered

If an opening starts at a junction and ends at the edge of the image, it would have only one end point covered. We can treat this end point as a junction by having a virtual, infinitely-thin segment at the edge of the image, running around the frame of the image. This is illustrated in Figure 11. Then opening end points at the edge of the image can be thought of as a junction involving the virtual segment. In order to place a triangle covering this junction, a vertex would need to be placed at the point where the opening meets the edge of the image since that is the only point on the virtual segment that can be connected to both of the skeletons for the opening. Now we can choose a triangle in the same manner as before. The portion of the opening between the new triangle and the triangle at the other end of the opening can then be covered by vertex merging and skeleton traversal.

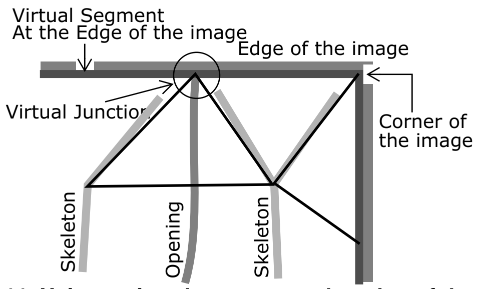

Figure 11. Using a virtual segment at the edge of the image.

5.4 Openings with No End Point Covered

In the case where both opening endpoints are at the edge of the image, we can place two triangles at the two ends as described above, and then covering the rest of the opening by vertex merging and skeleton traversal between the two new triangles.

Openings forming loops do not have endpoints, and as such do not have initial triangles. We simply choose an edge between the two respective region skeletons, and add triangles to cover the loop formed by the opening by skeleton traversal. If there are vertices already placed on the skeleton, we can use the existing vertex to place the edge. We simply use the chosen edge as the starting edge and also the ending edge.

5.5 Covering Borders

Finally, we need to place triangles in the remaining portion of the image, between the image boundary and the triangles placed thus far, to ensure that the entire image is covered with triangles. All the regions that need to be covered in this step is by now bounded by the edge of the image and the hull formed by the triangles already placed. We just need to identify these uncovered regions, and again use the skeleton traversal method to cover these regions, using the "virtual skeleton" of the virtual segment at the edge of the image and the hull of the existing triangles as the opposite skeleton. During this traversal, we use the area covered to choose between candidate triangles. As a result of using the virtual segment reasoning, there will always be a vertex at the corners of the image, since those are the only points along the virtual skeleton that can be connected to points on the two edges meeting at the corner. This is illustrated in Figure 11.

6 Experiments

Our first results illustrate the use of lines for selection, and they are shown in Figure 12. Selection with lines offfers the most freedom to the user, and more examples are shown in Figure 13(a)-(b). The same figure also shows how the same selection can be obtained using points and loops. This shows how the tool allows a reasonable degree of freedom

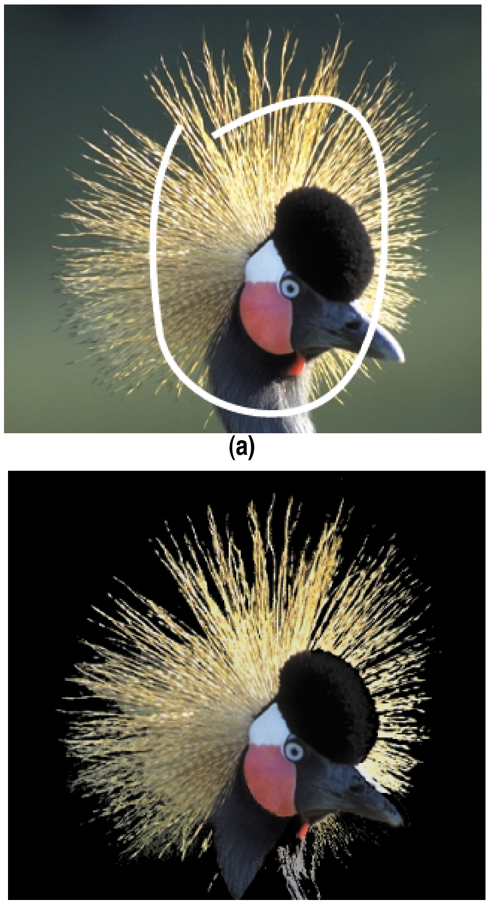

Figure 12. Examples of selection with lines. (a) Original image shown with sketch. (b) Resulting selection composited against a black background. Crane example cropped from an image by Gerald and Buff Corsi, California Academy of Science.

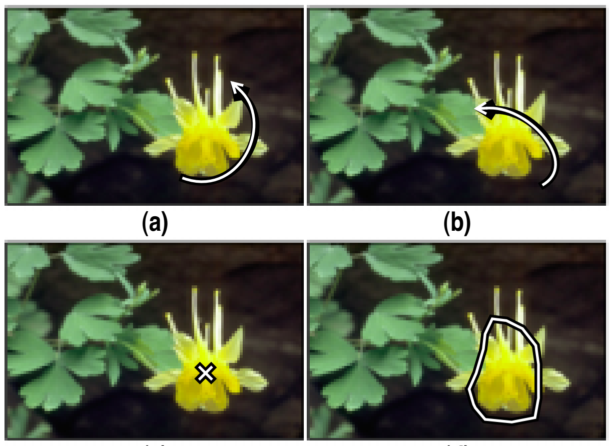

Figure 13. Different sketches that yield equivalent results. (a)-(b) Lines. (c) Point. (d) Loop.

so that users do not have to be very precise drawing the sketches.

We can also use the triangulation to further decompose the original coarse segments into smaller pieces, such that each piece corresponds to a vertex. The new decomposition is easily found from the triangulation: examine each triangle with two vertices that fall in the same segment. Identify the edge between these two vertices and find its midpoint. Draw a line from the midpoint to the third vertex in the triangle. If the edge is shared by a second triangle on the other side, do the same for the second triangle. Now cut the segment containing the two vertices along these lines. The resulting segments are each represented by a vertex in the triangulation, has the property that if two segments are adjacent, then there is a valid edge between the two representative vertices. This is a reasonable way to decompose an image into small pieces, and is useful in a number of applications. For example, one can use these segments in matching and recognition tasks, among others.

7 Conclusion and Future Work

We have presented a method by which simple freehand sketches may be used to select objects with complex boundaries. We have demonstrated with experimental results that the method is able to extract complex objects with a minimal amount of user input.

Ongoing work includes improving the computational complexity of the simplicial decomposition stage and additional constraints on the triangles. For example, it might be interesting to place the vertices such that the edges can be found automatically by computing the Delaunay triangulation of the vertices. It may also be interesting to place the vertices such that the Voronoi diagram of the vertices approximates the actual segment boundaries. While the motivation for computing the image structure representation is object selection, the representation may be used in a variety of applications, such as image retrieval, registration, and object tracking. We plan to investigate some of these possibilities.

8 Acknowledgements

We would like to thank Mark Ruzon and Carlo Tomasi for making available the source code for alpha estimation, and the anonymous reviewers for their thoughtful comments. The support of the Office of Naval Research under grant N00014-96-1-0502, and the Beckman Institute Graduate Fellowship is gratefully acknowledged.

9 References

[1] Narendra Ahuja. A transform for multiscale image segmentation by integrated edge and region detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(12):1211-1235, 1996.

[2] Alberto Del Bimbo and Pietro Pala. Visual image retrieval by elastic matching of user sketches. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(2):121-132, 1997.

[3] Sue Chastain. Knock it Out! Tools and Techniques for Removing Backgrounds. http://graphicssoft.about.com/compute/graphicssoft/library/weekly/aa000607a.htm, 2000.

[4] Christophe Chesnaud, Philippe Refregier, and Vlady Boulet. Statistical region snake-based segmentation adapted to different physical noise models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(11):1145-1157, 1999.

[5] Corel Stock Photography Collection. http://www.corel.com (The images may also be viewed at http://elib.cs.berkeley.edu/photos/about.shtml).

[6] Richard O. Duda and Peter E. Hart. Pattern classification and scene analysis. John Wiley & Sons, Inc., 1973.

[7] James H. Elder and Rick M. Goldberg. Image editing in the contour domain. In Proceedings IEEE International Conference on Computer Vision and Pattern Recognition, pages 374-381, 1998.

[8] Allen Gersho and Robert M, Gray. Vector Quantization and Signal Compression. Kluwer Academic Publishers, 1992.

[9] Michael Gleicher. Image Snapping. In Proceedings SIGGRAPH, pp. 183-190, 1995.

[10] Robert M. Haralick and Linda G. Shapiro. Survey: Image segmentation techniques. Computer Vision, Graphics, and Image Processing, vol. 29, 100-132, 1985.

[11] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active Contour Models. In Proceedings of the First International Conference on Computer Vision, pp. 259-268, 1987.

[12] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and Texture Analysis for Image Segmentation. Submitted to the International Journal of Computer Vision.

[13] Eric. N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In Proceedings SIGGRAPH, pp. 191-198,1995.

[14] Elin R. Pedersen, Kim McCall, Thomas P. Moran, and Frank G. Halasz. Tivoli: An electronic whiteboard for informal workgroup meetings. In Proceedings INTERCHI'93, pp391-398, 1993.

[15] Thomas Porter and Tom Duff. Compositing digital images. Computer Graphics, 18(3):253-259, 1984.

[16] Mark Ruzon and Carlo Tomasi. Alpha Estimation in Natural Images. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pp. 597-604, 2000.

[17] Eric Saund and Thomas P. Moran. A perceptually-supported sketch editor. In Proceedings UIST, pp. 175-184, 1994.

[18] Eric Saund and Thomas P. Moran. Perceptual Organization in an interactive sketch editing application. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pp. 597-604, 1995.

[19] Alvy Ray Smith and Jim Blinn. Blue Screen Matting. In Proceedings SIGGRAPH, 1995.

[20] Mark Tabb and Narendra Ahuja. Multiscale image segmentation by integrated edge and region detection. IEEE Transactions on Image Processing, 6(5):642-655, May 1997.

[21] Kar-Han Tan and Narendra Ahuja. Selecting objects with freehand sketches. IEEE International Conference on Computer Vision, vol. 1, pp. 337-344. July 2001.

[22] Yair Weiss. Segmentation using eigenvectors: a unifying view. Proceedings IEEE International Conference on Computer Vision, 1999.

[23] Donna J. Williams and Mubarak Shah. A fast algorithm for active contours and curvature estimation. CVGIP: Image Understanding, 55(1):14-26, 1992.