Selecting Objects With 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
{tankhlahuja}@vision.ai.uiuc.edu

Abstract

We present in this paper the design of an interactive tool for selecting objects using simple freehand sketches. The objective is to extract object boundaries precisely while requiring little skill and time from the user. The tool proposed achieves this objective by integrating user input and image computation in a two-phase algorithm. In the first phase, the input sketch is used along with a coarse global segmentation of the image to derive an initial selection and a triangulation of the region around the boundary. The triangles are used to formulate subproblems of local finer-grained segmentation and selection. Each of the subproblems is processed independently in the second phase, where a linear approximation of the local boundary as well as a local, finer-grained segmentation are computed. The approximate boundary is then used with the local segmentation to compute a final selection, represented with an alpha channel to fully capture diffused object boundaries. Experimental results show that the tool allows very simple sketches to be used to select objects with complex boundaries. Therefore, the tool has immediate applications in graphics systems for image editing, manipulation, synthesis, retrieval, and processing.

1 Introduction

Selection in digital image editing is the task of extracting an object embedded in an image, a task performed frequently in many content creation applications. A classic example of selection is in broadcast television news where blue screen matting is used to place a newscaster's image in front of background graphics. 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. Currently a number of commercial tools are available that aid in the process, but they either have severe limitations in their capabilities, or require very careful user guidance and control.

This paper is an attempt at making selection easier and usable in a wider range of applications. We present the design of a tool for selecting objects using simple, rough freehand sketches similar to those used in normal interhuman communication. Such a sketching interface is also an appropriate choice for image editing because the pen and pencil are ubiquitous tools for creating graphics, and sketch-based interaction would be very natural for use in handheld personal digital assistants and the emerging class of "tablet" computers. A typical example of the kind of sketch allowed by the selection tool we will describe is shown in figure 1(a). With the sketch shown, our selection tool was able to extract the head of the crane, as shown in figure 1(b). Despite the fairly convoluted profile of the crane's head, which consists of both sharp boundaries along the beak and complicated boundaries around the crane's crown, our selection tool needed only the simple sketch shown, which is natural and can be drawn quickly, without much skill and dexterity. We believe that this tool will enable sophisticated and high-fidelity graphical selection and manipulation operations in situations where the user may not have much time, such as during a live sports broadcast, or may not have much dextrous control, such as on a mobile handheld device, or may want to avoid the monotony and tedium of selection, without compromising the quality of the selection.

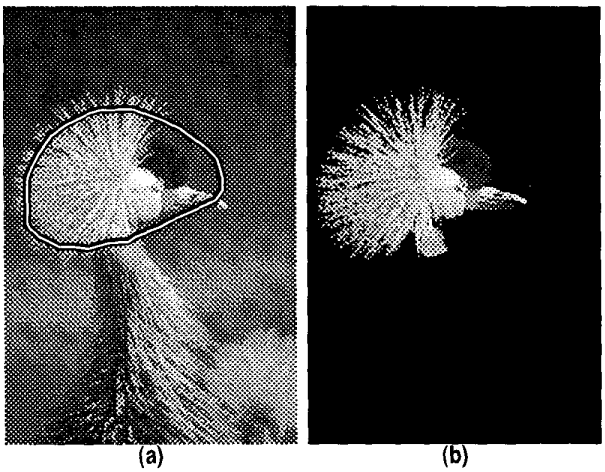

Figure 1. Selection using freehand sketches. We would like to extract the head of the gray crowned crane precisely, using minimal effort and skill. (a) The original image, with a freehand sketch as shown. (b) The selection extracted with our tool and composited against a black background. (Original image by Gerald and Buff Corsi, California Academy of Science).

2 Related Work

The original inspiration for our work came from PerSketch[18, 19], an augmented simulation of whiteboard

0-7695-1143-0/01 $10.00 © 2001 IEEE

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 a subject of current research[1, 10, 12, 21].

Interactive selection tools for still images have been attracting a lot of 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 the user-guided pointer, 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 final representation of the object segmentation is a binary-valued boundary such that every pixel may either belong completely to the selected object or be completely unselected. In most photographic images however, boundaries are not as clearly cut. 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 is to explicitly model diffused boundaries as Gaussian-blurred step discontinuities in image intensity[7]. This model has also been used in an interactive image editing system, ICE, which allows objects to be deleted seamlessly from intensity images. 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[20], 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[17]. With this 'Alpha Estimation' algorithm, the colors of pixels in the vicinity of object boundaries are mixtures of colors from the foreground object and those of 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 use a mixture model to estimate the alpha channel value at all the boundary pixels.

3 Algorithm Overview

A schematic of our interactive selection algorithm is shown in figure 3(a), and steps of the algorithm is shown in figure 3(b). The algorithm lets the user make an initial slection by drawing a sketch which the user then iteratively refines by drawing additional corrective sketches. This iterative process continues until the user is satisfied with the selection.

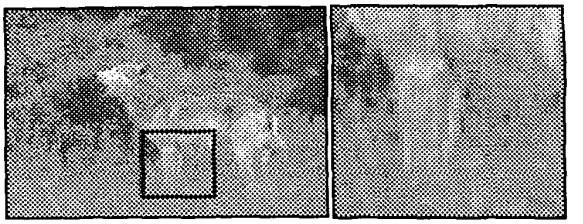

Figure 2. Examples of diffused object boundaries. (a) An image, taken from the Corel Stock Photography Collection, the boxed portion of which is shown in detail in (b), illustrating diffused edges. For high-fidelity image editing, the object boundary details need to be fully captured with an alpha channel.

The inputs to the algorithm consist of an input image and user's sketches consisting of *points*, *lines* and *regions*, which we call the *elements* of a sketch, each with predefined meanings. The intention behind the provision of these elementary strokes is that a user would use regions to give the approximate extent of the desired object, lines to roughly indicate parts of the object boundary, and points to select the entire (surrounding) object. In figure 3, the use of points, lines, and regions each produces the same final selection, and shows typical usage of the respective sketch elements. Other sketches can be constructed from these elements to convey other selection-related actions desired by the user.

Independently of the user sketches, the algorithm computes a global segmentation of the input image, which partitions the image into segments. The location of each segment is represented by its centroid, and the layout of the segments is captured by computing the Delaunay triangulation of the centroids. The segment map, the centroids and their triangulation collectively form a coarse initial description of the image structure. Segments approximate objects or parts of objects. Their centroids represent their respective spatial locations and the triangulation captures their spatial adjacency information.

Having extracted a description of image structure, user's sketch is now related to a set of image segments that serves as the initial estimate of user's intended selection. Each sketch input element adds one or more segments to the initial selection. A point adds the segment that it falls on. A line drawn by the user near the boundary between two objects first identifies the segments that are in its vicinity and then adds those that are on a specific *a priori* agreed upon side of the line (e.g. according to right hand rule). A region adds the segments that have large overlaps with it. The union of all

the segments selected by elements of the sketch forms the initial coarse selection. The user can inspect this selection and add/remove segments from the coarse selection until the coarse selection is a satisfactory representation of the desired object.

This coarse selection is based on global image analysis, and may not reflect fine local detail, e.g. exact locations of object boundaries. To increase the accuracy of the selection, the algorithm next performs local, finer segmentation in the vicinity of the initial selection. The vicinity of the initial selection is defined as the area included in the Delaunay triangles that straddle across the selection boundary. Within each triangle, a finer-grained segmentation, and a linear approximation of the initial selection's local boundary inside the triangle are computed independently. The approximate boundary is computed using Fisher's linear discriminant. Essentially, this approximate boundary represents the position and orientation of the local object boundary, derived from the coarse segmentation. This linear discriminant is then used to classify the centroids of the segments found from the local segmentation into two groups, foreground and background. In subsequent discussion, we shall use the term foreground to refer to pixels, segments, segment centroids and other entities associated with the object to be selected, and the term background to refer to entities not associated with the object.

The final selection is then computed by estimating the alpha channel for the foreground object using pixel samples from the foreground and background pixel populations. This is necessary since the selection formed by the segments has sharp boundaries, a representation that is inadequate for capturing diffused boundaries that characterize real-world images. In real images, there is a zone inside each object boundary where the images of adjacent objects diffuse during image formation. Alpha channel computation determines at each pixel near an object boundary the mix of the object and surround populations, caused by the diffusion. Alpha channel computation is carried out within each boundary triangle, and the results are pieced together to form the final selection. In the following three sections (4-6), we present the details of the algorithms used.

4 Global Segmentation

The method we use for image segmentation is binary-split vector quantization[8]. We form a 5-tuple for each pixel, (x,y,r,g,b). The first two coordinates represent the spatial location, and the last three coordinates are the RGB color component values representing the photometric features. The spatial coordinates are normalized by dividing with the height of the image. The color component values are normalized so that they range from zero to one. The vector quantization algorithm recursively partitions the data set of 5-tuples into two parts by finding the mean of the set, perturbing the mean to form two new seed vectors, and then clustering the set into two subsets using the seed vectors and the k-means clustering algorithm. We perform this recursive

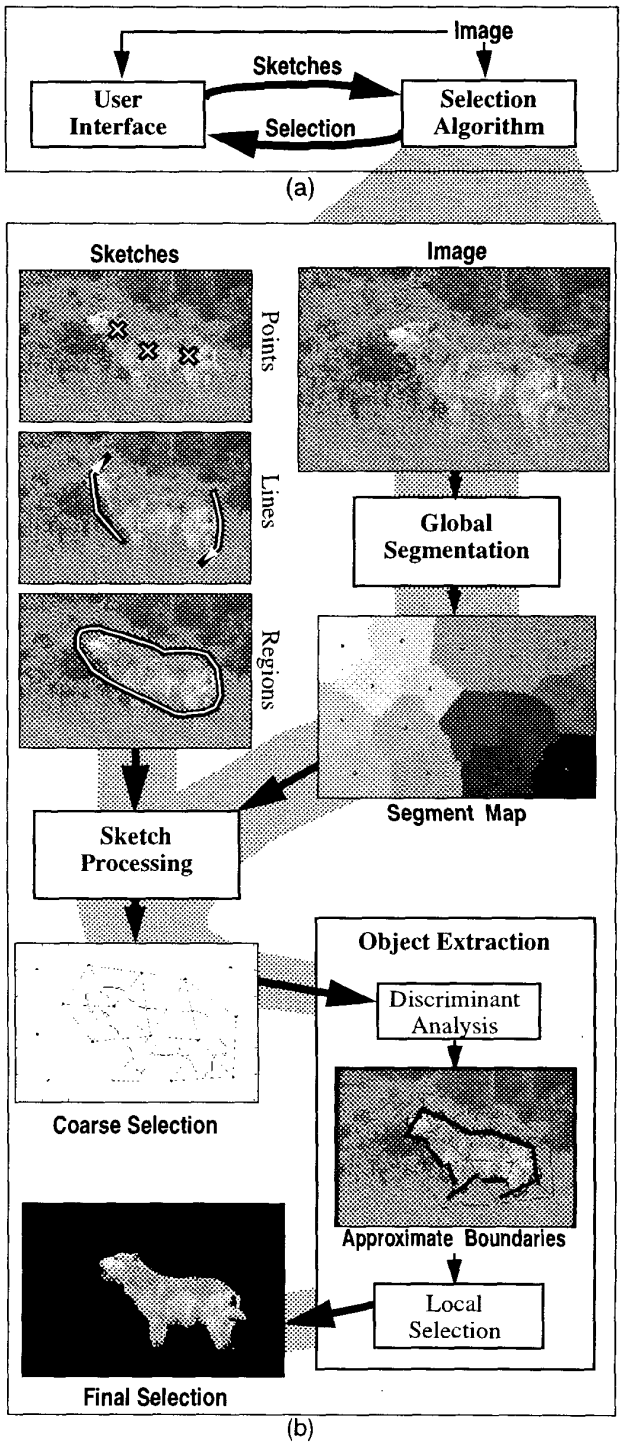

Figure 3. (a) Overview of the selection process. User supplies sketches to the selection algorithm which returns a selection that can then be iteratively modified with additional sketches. (b) Steps in selection algorithm.

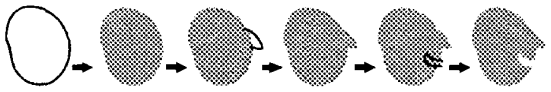

Figure 4. Developing a closed region using a set of predefined strokes. The sketch region is shaded gray, and the strokes are black.

splitting until the variances of each of photometric features within a subset are below a threshold σ_s. The lower σ_s is, the finer the segmentation. We also terminate the recursion when the number of data points in a set is below a threshold σ_t, so as to curb the splintering of the data set into insignificant pieces. An example segmentation can be seen in figure 3. Each pixel in the segment map is shown with a unique gray level corresponding to its segment label. The centroid of each segment is shown by a dot.

5 Sketch Processing

The three sketch element classes defined are as follows:

1. *Points* for specifying entire objects.
2. *Lines* for describing boundaries.
3. *Closed, include*, and *exclude* strokes for describing image/object regions.

Examples of the first two classes of strokes can be seen in figure 3. The third class of strokes, for describing a region, is shown in greater detail in figure 4. A closed stroke creates a region, while the include and exclude strokes modify the region. We assume that strokes are non-self-intersecting. Closed strokes are simply curves with endpoints close by. Include strokes are ones that begin inside an existing region, traverse outside and then end in the same region, delineating a region to be appended to the existing region. Exclude strokes are the opposite of include strokes, beginning and ending outside while carving out a piece of the existing region to be removed.

In isolation, the strokes do not have specific meaning. Their semantics become well-defined only when they are combined with information about the image over which they are drawn. In our case the image structure computed in the global segmentation stage provides this context information.

The sketch processing step takes as input the coarse global segmentation and the sketches made by the user. Each of the three main classes of sketch strokes is processed differently, since they have different meanings. However the end result of all three types of sketch processing is to classify the set of image segments from the global segmentation as being a part of the object to be selected, or otherwise. The result is a coarse initial selection of the object formed by the union of all segments classified as being a part of the desired object.

5.1 Points and Lines

Points are the simplest to process: for each point, we pick the segment label for the pixel under the point and add the corresponding segment to the foreground.

Lines carry more information than points, and we use lines to identify foreground segments as follows:

1. From the Delaunay triangulation of the segment centroids, find all triangles that touch the line.
2. For each such triangle, classify the vertices (and thus the centroids) as either foreground or background depending on the side of the line they fall on.
3. Form the union of all the sets of foreground vertices from each triangle and use it as the set of foreground centroids selected by lines. All other centroids then are the background centroids. The definition of this polarity is an arbitrary decision, though it is important for all relevant stages of the algorithm to adopt the same convention.

The specific meaning of points and lines with respect to the object selection task is now well-defined. Points simply correspond to segments, and lines correspond to segment boundaries. The benefit of using the image structure to provide a context for sketch processing is that it allows variations in the sketch while providing an unambiguous meaning of the sketch. For example, a point can fall onto any part of a segment to select the segment centroid. Lines indicating a boundary between two segments can be located anywhere between the two corresponding centroids, and the triangulation establishes an unambiguous correspondences between each line and the relevant centroids in the image.

5.2 Regions

Regions are processed differently from lines and points, and make use of the segment map. We choose the foreground regions as follows:

Let $A_1 .. A_N$ be N sets representing the segments from the global segmentation, the elements of each A_i being the pixels contained in the segment. $|A|$ thus represent the number of pixels in each segment.

If S is the sketch region, then $S \cap A_i$ is the set of pixels that are both in the sketch region and segment A_i.

A segment A_i is chosen as being a part of the foreground object if $\frac{|S \cap A_i|}{|A_i|} > \frac{1}{2}$.

The foreground object may be represented by a union of the chosen segments:

$\bigcup_i A_i$ s.t. $\frac{|S \cap A_i|}{|A_i|} > \frac{1}{2}$.

The foreground segments are thus chosen based on the extent of their overlap with the sketch region. As is with

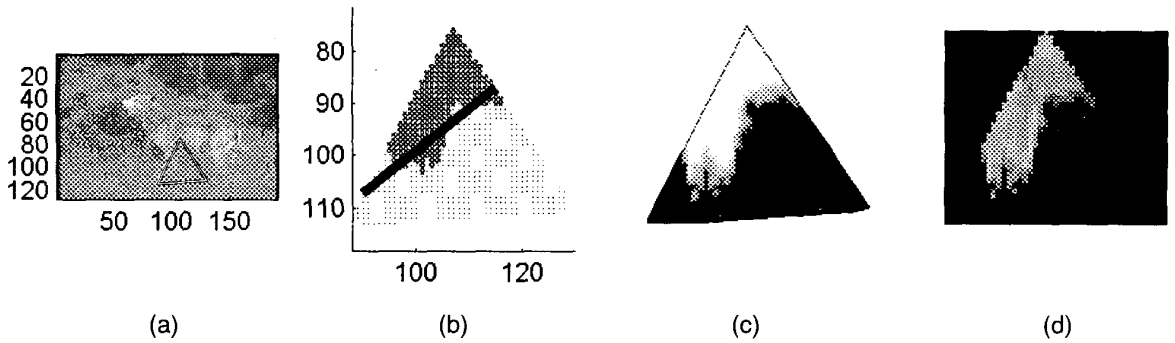

Figure 5. Local segmentation in a boundary triangle. (a) The original image, with a particular boundary triangle shown. (b) Pixels within the boundary triangle are used as two labelled populations to compute a linear discriminant. (c) The alpha channel estimated for the boundary triangle. Bright pixels indicate pixels containing the selected object. (d) Part of the foreground object contained in the boundary triangle extracted.

lines and points, the advantage of this approach is that a wide range of strokes may be used to indicate the same selection. All three types of sketches may be present in a user input, and the foreground segments can be the union of the foreground segments selected from each input stroke. The segments not selected are then classified as being in the background.

6 Object Extraction

At this point, we have labelled each segment centroid (and their corresponding pixels in the segment map) as being in the foreground or the background. We also have the Delaunay triangulation of the segment centroids. In the object extraction stage these inputs are used to compute the final selection, in the form of an alpha channel map and an estimate of the foreground pixel colors factored out from the image. We use the triangulation to partition the computational task into smaller pieces, and apply the object extraction procedure independently in each piece. We partition the problem as follows: identify all triangles from the centroid triangulation with vertices consisting of *both* foreground and background centroids. We call these the *boundary triangles* since they define the vicinity of the object boundary. The entire subsequent object extraction computation is performed within these triangles independently. The rest of the triangles are either completely contained in the foreground or in the background. The alpha channel value for pixels inside these triangles are uniformly one for pixels in foreground triangles and uniformly zero for pixels in background triangles.

We now describe the algorithm that applies to each boundary triangle (Figure 5). This algorithm can be divided into two phases, linear discriminant analysis and local segmentation:

Linear Discriminant Analysis

1. Using the coarse initial selection, label each pixel in the triangle as being foreground or background.
2. For each pixel, form a 3-tuple (x,y,c) which reflects the spatial location and foreground/background class membership of the pixel.
3. Compute Fisher's Linear Discriminant[6], which gives a direction along which to project the spatial location of each pixel so as to maximize the ratio of between-class scatter to within-class scatter.
4. If n_f, n_b are the number of pixels in the foreground and background groups respectively, and μ_f, μ_b are the corresponding centroids for these local pixel populations, let the linear discriminant pass through the point

$$w = \frac{n_b}{n_f + n_b}\mu_f + \frac{n_f}{n_f + n_b}\mu_b .$$

Local Segmentation

5. Perform a further segmentation of the local pixel population into a number of segments and compute their centroids.
6. Classify these newly computed segments by the linear discriminant into foreground/background groups.

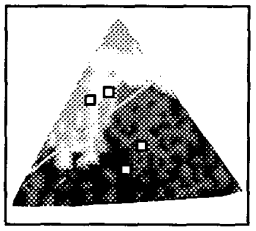

Figure 6. Foreground/background classification. In this figure, pixels within the triangle are clustered into four groups, shown in four gray levels. The spatial centroid for each cluster is also shown as a square. The linear discriminant is then used to label each centroid (and thus its corresponding cluster) as foreground or background.

7. Extract pixel samples in the neighborhood of the new centroids using their corresponding segment maps.
8. Estimate the alpha channel for each pixel in triangle using these foreground/background pixel samples.

Steps 1-4 compute an approximation of the local object boundary in the form of a linear discriminant. Steps 5-8 then perform a further local segmentation, form the new foreground/background groups and compute the alpha channel using the algorithm described in [17], with pixel samples drawn from the locally segmented foreground/background pixel populations. Steps 5-6 are illustrated in figure 6. In step 5, the local segmentation is performed with a procedure similar to the global segmentation stage, except now we recursively partition the data set until we exceed a minimum number of segments. Since each triangle contains pixels from three segments in the coarse segmentation, by partitioning the pixels in the data set into any number larger than three would result in a finer local segmentation.

The alpha channels for the boundary triangles together yield the overall segmentation. A question one may ask is whether these independently computed selections will produce a seamless result. To answer this question, recall the manner in which the selection problem is divided into subproblems of selection within triangles. Each pair of adjacent triangles share two vertices, and therefore, the color population of the segments corresponding to those centroids. These segments remain unaffected by the triangulation, thus ensuring continuity across the edges of the triangle.

7 Experiments

We implemented the selection tool as a stand-alone application written using the OpenStep AppKit and Foundation frameworks to provide the interactive sketching interface, and MATLAB for parts of computation and visualization. A user drags-and-drops an image into the application window to initiate the segmentation computation. The image is then displayed so the user may use the sketching interface to select the object(s). The selected object is then displayed in a separate panel, which is updated with each addition to the sketch made by the user. Upon completion the user may drag the selected object into a different application for further composition and manipulation. We plan to build on this stand-alone application to create an environment which allows PerSketch-like image editing with raster images.

7.1 Sketches

We tested the tool against a number of test images, as illustrated in figure 7. Very simple input sketches that one can reasonably expect a user to draw in a small amount of time consistently yielded highly detailed selections from the images shown. The only parameter that needs to be chosen by the user is the size threshold of the coarse segmentation

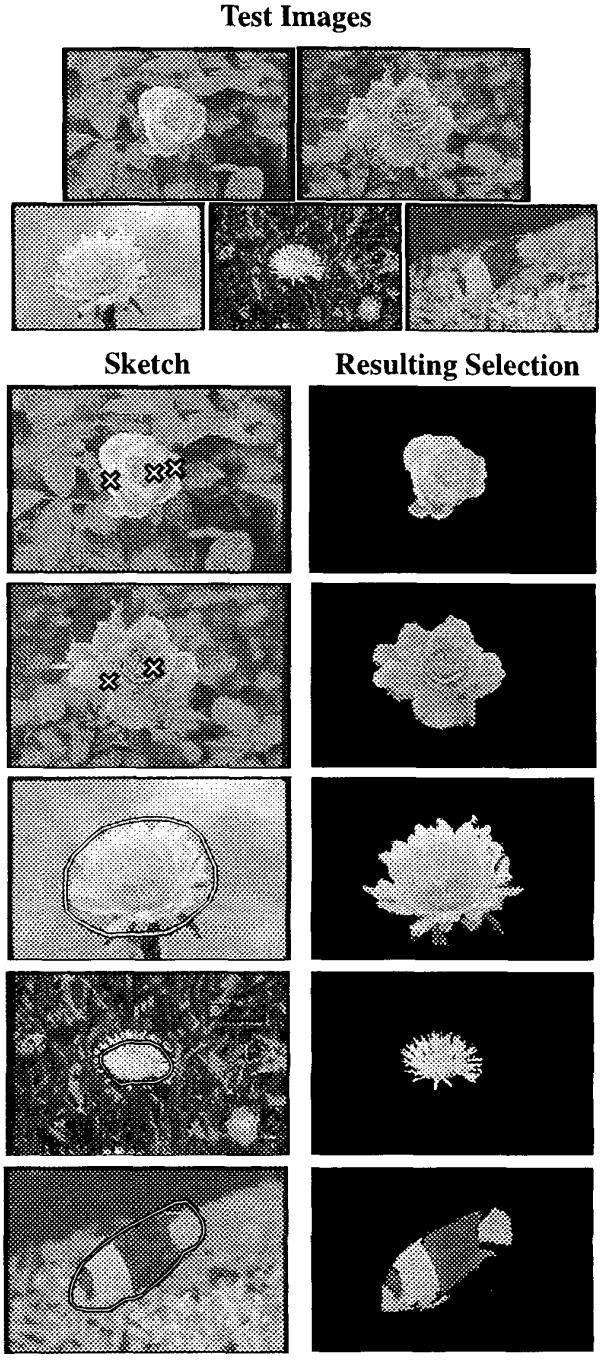

Figure 7. Experimental Results. Selection examples with regions and points are shown. Points are shown as crosses.

algorithm. Typical values for the experiments shown are one-half, one-quarter, one-eighth, and one-sixteenth of the

number of pixels in the given image. That the system is relatively parameter-free should not be surprising if one considers that the user sketch is the key input parameter, and the algorithm is just relating this rich visual information given by the user to the image content.

7.2 Comparing the Object Extraction Phase with the Alpha Estimation Tool

In the alpha channel estimation paper[17], two different methods were presented for specifying the foreground and background pixel samples. Examples of these input methods are shown in figure 8. In both methods the user ultimately classifies pixels in the image into three sets: foreground, background, and boundary. Foreground(background) pixels are assigned alpha values of one(zero), while a mixture model is used to assign intermediate values to boundary pixels based on the pixel color. The first method, boundary specification, involves a contour that is dilated by varying amounts along its length. Pixels covered by the dilated contour become the boundary pixels, while the untouched pixels on the two side become foreground and background pixels. Note that the amount of dilation also needs to be manually specified by the user. In the second method, the object specification method, the user simply 'paints' certain pixels as foreground, some others as background, and lets the untouched pixels be the boundary pixels. The kind of results derived from the two input methods are shown in figure 8.

We compared the alpha estimation algorithm with the local segmentation steps from the object extraction stage and studied the selection results. Recall that the only input required by the local segmentation steps is a linear discriminant. As can be seen in figure 8, using just one line segment to represent the linear discriminant, the object extraction phase was able to automatically form its labelled pixel sample populations and compute an alpha channel. Clearly the method we show here needed less user input than either of the two methods suggested in [17]. Couple this low-level routine we have with the high-level image analysis in the global segmentation and sketch processing stages, and it is easy to see how our selection tool is able to allow such simple user input sketches without compromising the selection quality.

7.3 Automatic Selection

While the bulk of the discussion presented in this paper assumes a human user, the input sketch regions or the initial selection may well be the result of another computational process. As an example, we experimented with using an automatic segmentation algorithm to pick out the foreground segments from the global segmentation. We perform a foreground/background segmentation over the global segmentation centroids using an eigenvector segmentation algorithm suggested in [22]. The results computed with the

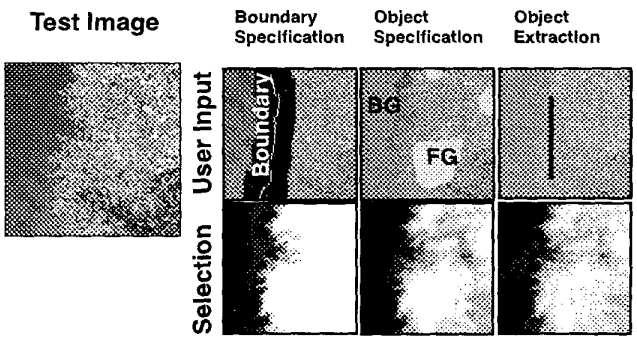

Figure 8. Comparing the alpha estimation algorithm with the object extraction stage in our tool. A test image is shown on the left (with increased contrast for better legibility). The objective is to select the leaves occupying the right half of the image. The first row shows the input given by the user, and the second row shows the computed alpha channel. The first two columns show methods proposed in[17]: boundary specification and object specification. The last column shows a linear discriminant given as input to our method. It can be seen that comparable results are achieved with markedly simpler input.

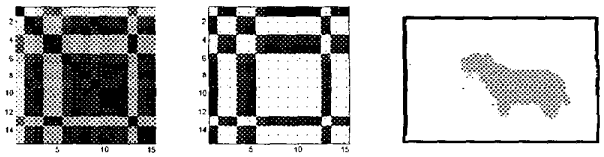

Figure 9. Automatic object selection. (a) The affinity matrix formed with the photometric feature distance among the global segmentation centroids. (b) The Q matrix computed. (c) The dominant foreground objects selected by thresholding a column of the Q matrix.

wolf image are shown in figure 9. In this case it can be seen that the algorithm correctly picked out the segments corresponding to the wolf, and would therefore subsequently result in a selection identical to the user-created selection shown previously. This suggests a method for automatic segmentation first using a simple method for coarse segmentation, and then using the eigenvector segmentation method to group the centroids. This saves computation by using simple methods to first group the large number of pixels in the image into a small number of representative centroids, and then applying global automatic segmentation methods to these centroids. Detailed boundaries between the foreground/background group of segments may then be computed with the object extraction algorithm.

8 Discussion

The algorithm we presented uses a divide-and-conquer scheme that breaks the problem of identifying object boundaries and estimating the alpha channel at object boundaries into smaller pieces based on image structure and user input, and then solving them individually. Triangulating the vicinity of the boundary and solving the alpha channel estimation problem within each triangle allows for better performance, since one can reasonably expect the amount of color variation in a small portion of the boundary to be less than that over the entire boundary. Also, the approximation of the local boundary with a linear discriminant would be less likely to work if the boundary were not split into smaller segments. Processing segments of the boundary allows the use of simpler algorithms.

9 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. We have also shown how the local segmentation stage in our tool yields results comparable to that of the alpha estimation algorithm in terms of the quality of the selection but our tool involves much simpler input from the user.

The algorithm can also be modified to perform automatic segmentation, which is one possible direction for future work: the fully automatic decomposition of an image into segments represented by alpha channels, yielding a form of 'soft' segmentation. Application of the tool to image synthesis, video analysis and object tracking are currently being investigated.

Empirical and subjective evaluation of the algorithm is underway, although it is proving to be challenging. Obvious quantities such as the time taken for the selection creation or the number of strokes required are dependent on various factors such as the speed of the computer, the skill level and habits of the user, and the complexity of the image and the desired selection, will need to be taken into account. Subjective comparisons with other selection tools will also be difficult since each tool may have a different mode of operation, and careful experiment design with a large user population and image database will be needed.

10 Acknowledgments

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

11 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] Dean Rubine. Specifying gestures by example. Computer Graphics, 25(4):329-337, July 1991.

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

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

[19] 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.

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

[21] 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.

[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.