Image Editing in the Contour Domain

James H. Elder  
Department of Psychology  
Centre for Vision Research  
Human Performance Laboratory, CRESTech  
York University, Toronto, Canada M3J 1P3

Rick M. Goldberg  
Department of Computer Science

Abstract

Image editing systems are essentially pixel-based. In this paper we propose a novel method for image editing in which the primitive working unit is not a pixel but an edge. The feasibility of this proposal is suggested by recent work showing that a high-resolution image can be accurately represented by its edge map if a suitable edge model and scale selection method are employed [1]. In particular, an efficient algorithm has been reported to invert such an edge representation to yield a high-fidelity reconstruction of the original image [2]. We have combined these algorithms together with an efficient method for contour grouping and an intuitive user interface to allow users to perform image editing operations directly in the contour domain. Experimental results suggest that this novel combination of vision algorithms may lead to substantial improvements in the efficiency of certain classes of image editing operations.

1 Introduction

The ultimate commercial goal of most computer vision research is to devise autonomous algorithms which replace humans in performing visual tasks. Given the difficulty of attaining this goal, it is worthwhile to consider, as an alternative or as an intermediate step, algorithms which do not replace a human operator, but which can assist the operator, allowing visual tasks to be performed more quickly, more accurately and/or more reliably.

Image editing is one commercial application for which algorithms of this type may prove useful. An image editing system allows an image to be modified according to the specifications of a human operator. By definition, it is not possible to entirely replace the human with autonomous algorithms. However, computer vision techniques may make certain types of image editing operations more efficient. In this paper, we introduce an interconnected set of computer vision algorithms which together form the foundation of a novel kind of image editing system. This system, although still in the prototype stage and restricted to gray-level images, has the potential to be more efficient and accurate for certain operations than existing systems.

Image editing systems such as Adobe Photoshop\textsuperscript{TM} offer a broad range of tools for modifying images. An important subset of these involves the addition, subtraction or deformation of image structure on a spatially-selective basis. For example, given the photograph in Fig. 2, one may wish to remove specular reflections from the skin of the subjects, or to delete distracting features from the background. In commercial image editing systems, it can be tricky to exactly demarcate the boundary of these regions to be deleted, moved or deformed, and significant effort may be required to correct small errors and to feather boundaries to reduce artifact. Moreover, one has to make guesses about the colours to use to fill in the areas from which features are deleted. The hypothesis motivating this paper is that these operations may be greatly facilitated if the user can effect image changes by directly manipulating the bounding contours of the target image structures. In other words, it may be more efficient for the user to edit an image in the contour domain.

The main difficulty with this proposal is the problem of how to translate image editing operations expressed in the contour domain into actions on the image. A solution to this problem derives from a recent result [2] which shows that an accurate edge representation capturing edge location, brightness, contrast and blur information can be used to obtain a high-fidelity reconstruction of the original image. In other words, this edge representation is, to a good approximation, invertible. This means that any image editing operation performed in the contour domain can be translated into an associated action in the image domain. This result has enabled the development of a novel type of image editing we call Interactive Contour Editing (ICE).

ICE is based upon three component algorithms:
1. **Edge detection.** Edges are reliably detected and represented using an adaptive scale space technique known as Local Scale Control [3, 1].

2. **Image reconstruction.** In order to translate editing operations in the contour domain into actions on the image, we employ a recently reported method for reconstructing an image from its edge representation [2].

3. **Contour grouping.** To perform image editing operations in the contour domain, local edges must be grouped into extended contours. We exploit an efficient algorithm originally designed to compute closed contours [4].

In Section 2 of this paper we briefly describe each of these core algorithms. In Section 3 we describe how these algorithms have been integrated and combined with a prototype graphical user interface which allows simple ICE operations to be performed. In Section 4 we present experimental results, and in Sections 5 and 6 we discuss future work and our conclusions from this study.

2 **Algorithms**

2.1 **Edge Detection and Representation**

We model local edges as Gaussian-blurred step discontinuities in image intensity (Fig. 1). The model consists of 5 parameters [3, 1]:

- Location (to the nearest pixel)
- Orientation
- Blur scale $\sigma_b$
- Asymptotic intensity $I_d$ on dark side of edge
- Asymptotic intensity $I_t$ on bright side of edge

Detection of edges and estimation of model parameters is based upon measurement of the gradient of the intensity function using steerable first derivative of Gaussian filters [5, 6], and on estimation of the locations of zero-crossings of steerable second and third derivative of Gaussian filters, steered in the gradient direction. While the zero-crossing of the second derivative localizes the edge, the separation of the zero-crossings of the third derivative are used to estimate the blur scale of the edge. This estimate of blur scale, together with measurements of the image intensity at the third derivative zero-crossings, can be used to estimate the asymptotic intensities on either side of the edge (Fig. 1).

A major obstacle to reliable edge detection is the scale problem: how to choose the scale of local estimation filters in order to prevent false positives and distortion due to noise, while minimizing distortion caused by neighbouring image structure. Our method for edge detection solves this problem with an adaptive scale space technique called Local Scale Control [3, 1].

This technique selects, at each point in the image, the minimum reliable scale for local estimation. At this scale, hypotheses concerning the sign of response of a linear filter at each point can be tested with statistical reliability. This means in turn that zero-crossings can be reliably detected and localized.

An example of the edge map produced by this algorithm is shown in Fig. 2 (top, centre). Detection of edges and estimation of model parameters using local scale control takes roughly two minutes on a 200 MHz Pentium Pro.

2.2 **Inverting the Edge Representation**

Our method for reconstructing the image intensity from brightness estimates at edges is based upon a single major assumption: that at non-edge points the intensity function approximately satisfies the Laplace equation, i.e., $\nabla^2 f(x, y) = 0$. It is understood that the Laplacian of the intensity function is unlikely to be exactly zero over the image, however the restriction to non-edge points does mean that areas of greatest variation are excluded. In practice we find that perceptually, this approximation is a good one.

Given this assumption, an estimate of the original
intensity function can be obtained by solving the Laplace equation, with boundary conditions given by the asymptotic brightness estimates at the edge points, and with reflection boundary conditions at the frame of the image. The solution can be computed, albeit slowly, by solving the heat equation using iterative Gaussian convolution (diffusion) over the image. An intermediate solution to the heat equation for an example image is shown in Fig. 2 (top right). In practice, we use a multigrid method [7] to solve the Laplace equation quickly (roughly 10 seconds on a 200 MHz Pentium Pro).

The final solution to the Laplace equation for our example image is shown in Fig. 2 (second from bottom). It is clear from this example that representing the edge locations and brightness values alone is not enough: this yields an artificial reconstruction in which shading, shadows and highlights appear unnaturally sharp. This problem can be solved [2] by using the edge parameters and the Laplace equation model to reconstruct both intensity maps and blur maps in parallel. These two maps are then recombined through a space-varying blur of the reconstructed brightness function in which, for each point on the brightness map, the space constant of the blur kernel is drawn from the corresponding point on the reconstructed blur map (Fig. 3).

2.3 Efficient Contour Grouping

Image editing tasks would normally involve one or more extended contours, not single edge elements in isolation. For this reason, contour-based image editing depends upon an efficient method for specifying a group of edges to which an action is to be applied.

One possible method for contour grouping is a simple “lasso” function which allows a contiguous region of line segments to be selected. Unfortunately, this method fails when the user wishes to edit only a single contour, and not the other nearby contours in the image. For such cases, a more sophisticated method for grouping edges into extended contours is required.

An efficient method for grouping edges into closed contours has recently been reported [4]. The algorithm consists of three main stages:

1. Line segment approximation.
2. Computation of posterior line grouping probabilities.
3. Shortest path computation of maximum likelihood line segment cycles.

The main purpose of stage 1 is to reduce the noise in estimates of contour geometry caused by lateral displacements of edge elements, thus providing a stable
basis for one-dimensional grouping. Line segment models are constrained to lie within $\sqrt{2}$ pixels of edge elements. In addition to position, length and orientation, the mean brightness parameters over each line segment are computed. The algorithm is greedy in nature, and yields line segments that typically range from one to hundreds of pixels long. Each edge element is labelled with the line segment that approximates it, so that a computed sequence of line segments can be re-mapped to a set of edge pixels in the original edge representation.

Stage 2 employs models for prior distributions and likelihood functions on the geometric and photometric relationships between line segments to compute the posterior probability of each line segment pair forming contiguous parts of a common curve. In this way, each line segment is assigned a short, ordered list of line segments with which it is most likely to group.

Stage 3 employs a powerful independence assumption which allows the computation of maximum likelihood paths over sequences of line segments to be re-expressed as a shortest path computation over a sparsely-connected graph. In its original implementation the algorithm computed the maximum likelihood closed curve for each line segment in the image. For ICE, the algorithm has been adapted to compute the maximum likelihood contour connecting any two line segments specified by the user.

While Stage 1 and Stage 2 are executed in advance, before image editing, Stage 3 must be executed online, each time an editing operation is performed. Thus the efficiency of the shortest-path computation is key to the feasibility of this approach.

3 Interactive Contour Editing

To test the feasibility of applying these three core algorithms to image editing, we have developed a prototype ICE system using the Khoros development environment on a Linux platform.

A complete contour editor would allow users to rotate, translate, add, delete and deform contours, as well as to modify the intensity and blur variations along the contours. At present we are testing the feasibility of the approach with only two relatively simple operations:

Delete: This operation allows users to delete selected contours from the image representation. The user can then press a button to compute an estimate of the image that would have generated the edge map with these selected edges missing. This function is useful for removing glare, shadows and distracting background structure.

Crop: This operation allows users to crop the object enclosed by a selected contour. Cropping is done in the contour domain. The “closure” of the contour is computed using polygonal completion, and all edges exterior to this closed contour are deleted. In addition, the exterior brightness value for each edge point on selected bounding contour is set to a default background value (0 in our experiments). Again, the image corresponding to the cropped contours may be regenerated with a click of a button. This operation is useful for quickly isolating a foreground object of interest, and will form the basis for future object-based cut and paste operations.

Prior to editing an image in ICE, the edge representation, line segment approximation and posterior
The user initiates grouping by clicking near a contour in either the image window or the contour window. This click initiates a nearest neighbor search in the area of the mouse click to find the nearest edge point. The coordinates of this edge point are used to index the line label map to obtain the index of the line segment with which this edge point is approximated. The selected line segment is highlighted in colour on both edge and image displays. When the user clicks near a second edge point, the same process is followed to obtain a terminating line segment index.

These two line segment indices then form input to a routine which implements the third stage of the contour grouping algorithm (Section 2.3). This algorithm uses the pre-computed posterior probabilities of pairwise line segment grouping in a shortest-path algorithm to determine the most probable sequence of line segments connecting the two selected line segments. This computation rarely takes more than five seconds to complete.

Fig. 6(left) shows an example of this interactive grouping procedure. Selected line segments are indicated by bow tie markers. The most probable connecting path is shown highlighted in green.¹

Since the grouping algorithm is imperfect, selecting two line segments which are too distantly connected may lead to a nonsense path (Fig. 6(centre)). In such cases, the user may undo the path and, with ICE in Append Path mode, select a sequence of more closely-spaced points along the contour which the algorithm can more easily connect (Fig. 6(right)).

### 3.3 Two-Dimensional Grouping

Certain image editing operations may require actions on two-dimensional textures of edge elements. For this purpose, ICE supports a Box Select mode. A box is defined on the image by clicking and dragging.

¹Conclusions of contours by the frame of the image are taken into account in the computation of posterior line grouping probabilities.
3.4 Delete Operation

Clicking the Delete button on the ICE form initiates a sequence of actions in which the selected edge points and associated line segments are deleted from all data structures. While these edge points are no longer visible in the edge map, the image remains unchanged.

3.5 Crop Operation

Once a contour is selected, clicking the Crop button on the ICE form initiates the following sequence of actions:

1. The closure of the selected contour is rendered to an internal (not displayed) bitmap. This involves rendering both the selected line segments and the polygonal completions between these line segments where gaps exist.

2. The interior of this closed contour is labelled using a standard 4-connected recursive “colouring” algorithm (e.g., [8]). The background remains unlabelled.

3. All edges in the edge map at unlabelled pixel locations are deleted.

4. The brightness values on the exterior side of all edge points on the selected contour are set to a default background value (0 in our experiments)

Thus the Crop function deletes all contour structure exterior to the closure of the selected contour, and updates the selected object to have a black background.

3.6 Refresh Operation

The user can translate any actions made on the edge map into actions on the image by clicking the Refresh button on the ICE form. This initiates the reconstruction algorithm which inverts the edge representation in the modified state to produce an estimate of what an image which generated the present edge map would look like. This refresh operation typically takes 10-20 seconds to complete.

4 Results

4.1 Deletion Results

Fig. 7 shows results from two simple ICE deletions. In the first case (Fig. 7(top)), we have used primarily 1-D grouping to remove highlights and shadows from the faces of the subjects, producing a softer, more complementary look. We have also used primarily 2-D grouping to delete all extraneous background detail, thus focusing attention on the subjects. In the second example (Fig. 7(bottom)) we have used both 1-D and 2-D grouping to remove selected windows from the building. This kind of operation could be useful for running architectural “What if?” scenarios.

Effecting these image editing operations is easy and efficient with ICE because the contours to be changed may be specified directly. Note that when contours are removed, the reconstruction algorithm will automatically interpolate brightness and blur functions smoothly. No decision need be made about what grey-level values to use to fill in holes, and no post-hoc feathering and blurring need be done, as is required with conventional pixel-based systems.

4.2 Cropping Results

Fig. 8 shows how the ICE crop function can be used to rapidly isolate a foreground object in an image. There are several advantages of contour-based cropping over pixel-based cropping:

1. The object boundary is rapidly selected using a small number of only approximately localized mouse clicks (indicated by “bow tie” symbols in Fig. 7).
2. The exact boundary of the object is located accurately through the edge map.

3. The reconstruction preserves the geometry, intensity and blur properties of the bounding contour against the new background. With pixel-based cropping, these features would have to be adjusted using post-hoc feathering and blurring techniques.

5 Future Work

This paper serves to demonstrate the feasibility of interactive contour editing, in which image editing operations are performed directly in the contour domain. In the next phase of this project, we intend to implement additional operations which allow the user to translate, rotate, deform and import contours. In addition, the ability to edit intensity and blur functions along selected contour paths will be added.

In the current version of ICE, when contours are deleted the intensity function is automatically filled in smoothly. In cases where the deleted object occludes other objects or textures, deletion produces unexplained gaps in the background structures. To extend the power of contour-based image editing, more advanced filling-in algorithms must be explored.

There are many opportunities for improvement in the core algorithms on which ICE is based. Inaccuracies in the Gaussian model for edge blurring lead to inaccuracies in image reconstruction. Difficulties in detecting edges which are very close together (e.g., in hair texture) lead to reconstructed images with a
Figure 8: Results of interactive contour-based cropping. Left: Original image with selected contours. Mouse-click locations are indicated by “bow tie” symbol. Middle: Edge maps automatically cropped based on contours selected. Right: Images rendered from cropped edge maps, with default black background.

6 Conclusion

In this paper we have proposed a novel vision-based framework for image editing applications. This framework is based upon three recent computer vision algorithms for multi-scale edge detection, image reconstruction and contour grouping. The goal of the proposed system is to avoid the difficulties in exactly demarcating structures in an image by allowing users to effect editing actions directly in the contour domain. The efficient grouping algorithm allows target contours to be rapidly selected, and the reconstruction algorithm allows actions on the edge map to be quickly translated into the image domain. Although the actual image editing operations currently supported are limited, interactive contour editing has been demonstrated as a feasible basis for image editing with certain advantages over standard systems.

Acknowledgments

This work was supported by grants from NSERC.

References


