

Chapter 8

Conclusions and Future Directions

This dissertation has demonstrated that raising the granularity of the fundamental image primitive from low-level, device-dependent pixels to high-level, object-centered regions (as generated via tobogganing) provides a more efficient and consistent framework for identifying, analyzing, and manipulating objects within images. Within the context of Intelligent Scissors, using an object-centered image primitive as the basic processing unit—in conjunction with the other contributions arising from toboggan-based regions—significantly reduces the human effort involved in object selection (in both single images and temporal sequences) while also allowing for more accurate boundary definition and facilitating future image editing operations. Increasing the size of the basic segmentation unit not only reduces the size of the underlying graph used by the live-wire selection tool (resulting in faster, more interactively responsive optimal path computations), but also reduces the number of potential paths, thereby increasing reproducibility, and provides a reliable mechanism for computing an edge model (which provides, among other things, for subpixel boundary location) and an edge confidence measure (which forms the basis

for confidence threshold snapping and the live-wire path extension algorithm). In fact, in many cases, an object is selected with little more effort than moving the cursor close to the object's boundary. Further, this dissertation has also demonstrated that parallelizing both the user input and visual feedback provides an effective interface for simultaneously defining the boundary of a moving object in multiple frames of a temporal sequence; thus maintaining the same interactive style that has proven effective in the original Intelligent Scissors (i.e., the live-wire feedback), but within a multi-frame segmentation environment.

The ultimate goal of this work has been to develop techniques that can be used in producing a widely available tool that reduces the human burden involved with accurate object selection for general purpose applications such as image/movie editing, quantitative medical image analysis, etc. As such, using object-centered image primitives within the Intelligent Scissors framework have resulted in several contributions, both major and minor, that seem to increase the efficiency, accuracy, and reproducibility of user-guided boundary definition and that provide automatic techniques for extracting alpha mattes as well as assisting in other image editing operations. This chapter lists these contributions and expounds on several possible directions for future improvement.

8.1 Contributions

The major contributions of this work are:

1. The use of a toboggan-based graph increases the granularity of the segmentation problem above the pixel-level and greatly reduces the time required to compute the minimum spanning tree used for live-wire optimal path selection, thereby providing for much greater responsiveness in the interactive live-wire environment.
2. Tobogganed regions also provide a consistent and reliable mechanism for computing a four parameter edge model, which includes an estimate of the sub-pixel boundary position and a measure of the standard error in the fitted parameter allowing the displayed object boundary to be interactively smoothed.
3. The other three edge model parameters estimate the edge blur and the foreground and background color. This allows the blur to be adjusted on a per contour basis and provides a means for automatically computing an alpha channel while simultaneously removing the background color from transition pixels.
4. Live-wire path extension represents a new and powerful local edge following technique that reduces the number of seed points required for boundary definition to the point where, in some cases, a closed object boundary can be defined with a single seed point and minimal cursor movement.
5. Confidence threshold snapping practically eliminates the need for precise cursor positioning by snapping to the nearest high confidence edge point within a radius up to several tens of pixels away.
6. The ability to compute faster optimal paths allows live-wire selection to occur simultaneously in multiple frames displayed as a montage; thereby facilitating segmentation of temporal image sequences while maintaining, in each visible frame, the interactive live-wire object selection style that has proven effective for Intelligent Scissors.

While each of these contributions represent unique additions or extensions to previously existing techniques, all of them rely, to some degree or another, on the over-segmentation and the subsequent weighted planar graph resulting from tobogganing. Though tobogganing is, in and of itself, certainly not a contribution of this work¹, using tobogganed region

1. For some reason, tobogganing has been largely ignored by the computer vision and image processing communities in favor of the more computationally expensive, yet theoretically equivalent, watershed algorithm. Prior to its “rediscovery” by Reese [144], the only previous references to tobogganing seem to be the original work by Fairfield [49] and a follow on paper by Yao and Hung [183].

boundaries to create a weighted planar graph and toboggan slide paths to sample edge model pixels are both new ideas.

Besides the major contributions just enumerated, other (minor) contributions¹ of this work include:

- Section 2.2 provides a concise list of most (if not all) of the desirable properties that an object selection (or, for that matter, any interactive) tool should possess.
- While this dissertation has, admittedly, not focused on prior art in edge operators, this researcher has not encountered any previous technique for normalizing Gaussian-based gradient kernels (whether separable or not) based on their maximum amplitude response as is detailed in Section 3.1.1.
- Section 3.2.1 presents an efficient and topologically sound algorithm for creating a graph from the 4-connected region boundaries resulting from tobogganing.
- While various curvature measures have been used in the past (both in live-wire methods as well as active contours and other segmentation strategies), the extent of interdependence of the curvature cost given in Section 3.2.3.4 exceeds that of any previously known measure and even though its current use inhibits guaranteed optimality, it has the potential to be both optimal (see Section 8.2) and provide a means of coupling live-wire feature costs between frames.
- There appears to be no previous implementation of the efficient two-level active list sort described in Section 3.3.3 that combines the standard wrap-around bin sort at one level with the (apparently novel) linear-time insertion sort at the second level.
- While computing the gradient magnitude of vector valued images using multiple kernel scales is well established, combining the gradient direction vectors from multiple image bands and over multiple kernel scales (as described in Section 4.3.1.1) has not been previously encountered.
- Extracting a one-dimensional (in both the domain and range) edge profile from a two-dimensional, vector valued (i.e., RGB) image using tobogganing information and the domain and range projection vectors (Section 4.3) seems to be a new approach.

1. These contributions are listed here for one (or maybe both) of following reasons: first, while similar techniques or contributions may have already been published, they represent fields of study with which this researcher is only partially familiar and may, therefore, be “reinventing the wheel” or second, they represent problems that needed to be solved only as a means to an end in that they are just a piece of one of the larger contributions listed previously.

- The method for computing the estimated standard deviation of the global image noise, as detailed in Section 4.4.1, also seems to be novel.
- The scanline fill algorithm for a closed, crack-based contour provides an efficient and topologically sound alternative to previous pixel-based inside/outside contour filling methods.
- The edge confidence computation as described in Section 5.1.2 seems to be a new and effective means of estimating whether or not an edge element belongs to an object boundary and may find application in other techniques such as edge linking or perhaps as an edge based cost for Intelligent Paint.
- While filling algorithms abound, these are usually scanline (for polygons) or depth-first flood fill (for arbitrary areas) techniques that are intended for the regular (i.e., uniform) pixel grid rather than a breadth-first expansion of non-uniform (4-connected) regions (Algorithm 5.3) that partitions the space to be filled.

Although some of these contributions are more “minor” than others, they all represent problems that had to be solved (via new ideas and techniques or by adaptation and modification of previous methods) in order to achieve the goal of developing a tool that reduces the human burden involved with object selection and image editing.

In conclusion, even in its original, pixel-based form, Intelligent Scissors/live-wire has proven to be an effective object selection tool that, compared to tedious manual tracing or even other semi-automated approaches like magic wand, provides a faster, more accurate, and more reproducible method of defining object boundaries. As such, the ideas and techniques behind Intelligent Scissors have been incorporated into several image editing and medical image analysis packages (commercial and otherwise) and have consequently benefited many people in tasks ranging from the creation of web content to graphic design for advertising and illustration to segmentation of medical scans for analysis and treatment of patients. Thus, it seems that the extensions presented in this work have the potential to, hopefully, further assist many (and more) people in their endeavours to segment and/or edit images and movies.

8.2 Future Work

Tobogganed-based Intelligent Scissors and the extensions presented in this dissertation provide many advantages over the previous, pixel-based Intelligent Scissors (as well as other object selection strategies). However, there is still plenty of room for additional improvements to this work. Some of the future directions to explore and possible extensions to this work include:

1. Automatic determination of filter scale.
2. Adaptive transitions between tobogganed-based and pixel-level graphs.
3. Efficiently incorporate the curvature measure while guaranteeing optimal paths.
4. Eliminate (or greatly reduce) preprocessing.
5. Replace the single element edge model with a multi-element multi-edge model.
6. Fit the subpixel boundary points with a spline rather than a polyline.
7. Reduce the exponential complexity of the edge confidence computation.
8. Enhance live-wire path extension to facilitate object selection.
9. Improve the interface by allowing zooming, panning, and scrolling.
10. *A posteriori* addition and/or editing of seed points.
11. Automatic, efficient, and reliable multi-frame free-point correspondence.
12. Multi-frame live-wire feature cost coupling.

Each of these possible extensions are addressed in more detail in the following paragraphs.

Automatic determination of filter scale: While the gradient magnitude operator automatically selects, on a per pixel basis, the kernel with maximal response, the choice of possible kernels is still set either by default or by the user. Even though the default kernel

sizes of 5×5 ($\sigma = 0.5$) and 9×9 ($\sigma = 1$)—which are the smallest two kernels available—perform well for most of the images used in this dissertation, some images are better suited to larger kernel scales due to their high noise and/or low SNR. For example, the test image in Fig. 5.5(b) works well with a 13×13 ($\sigma = 1.5$) or 17×17 ($\sigma = 2$) kernel while the low SNR CT image of the left ventricle in Fig. 5.5(a) requires a much larger 45×45 ($\sigma = 5.5$) kernel¹. Since smaller kernels have a larger response to fine detail (such as noise), they are not well suited for these types of images. One possible extension to this work would be to automatically adjust the default kernel scales based on the global noise image—provided by the estimated noise measure given in Section 4.4.1—so that larger kernels are used in images with a lot of noise. Alternatively, the kernel scale could be automatically determined on a per pixel basis using a technique similar to that used by Elder et al. [46-48]) so that a small kernel is applied only in areas of high SNR while avoiding its use in flat, low SNR areas.

Adaptive transitions between toboggan-based and pixel-level graphs: While toboggan-based Intelligent Scissors provides several benefits over its pixel-based predecessor, the one advantage that pixel-based Intelligent Scissors has is its ability to force, if necessary, any desired pixel-level boundary. Since the toboggan-based graph requires that live-wire paths adhere to the region boundaries, there is currently no means to specify a path through a toboggan region. While the toboggan-based graph is sufficient for a majority of object selection situations, there are times when a portion of an object boundary is not adequately defined by the region-based graph (such as those identified in Section 3.1.3).

1. While this is a large kernel, it only results in a 0.11 second increase in preprocessing (1.13 seconds compared to 1.02 for the original 480×480 image) due to the fact that convolution with separable kernels is still quite fast even for large kernels and that the resulting reduction in toboggan regions reduces the time (and memory) needed for graph creation.

Further, there may be occasions when the user simply wants to draw a contour through an area where there is no object boundary. Thus, a very desirable future extension would be to incorporate the ability to resort to a pixel-level graph, either automatically or on demand, to create live-wire paths that are not constrained to the tobogganed-based graph. The decision to automatically switch to the pixel level could be based on an analysis of the tobogganed graph edges in the current live-wire segment. Prior to displaying the live-wire, graph edges in the live-wire that exhibit weak edge properties (such as a low gradient magnitude) would trigger the transition to a pixel-level recomputation of the optimal paths bounded by the regions on either side of the “offending” edge(s). Further, since the gradient magnitude has been found to be lacking in these areas, its relative weight in the local cost function could be reduced to favor the curvature measure, thereby increasing the smoothing cost.

Efficiently incorporate the curvature measure while guaranteeing optimal paths: As noted in Section 3.3.5, the curvature cost, as a multiple graph element feature, is not currently being utilized to its full potential, resulting in live-wire graph paths that may be sub-optimal. Thus, another future development could explore efficient techniques for limiting the maximum potential difference between curvature costs in order to restrict the number of paths that need to be maintained at each node during graph expansion; thereby guaranteeing optimality in all live-wire paths. One possible mechanism for bounding the maximum curvature difference is to precompute and maintain the maximum and minimum possible path curvature through each graph node.

Eliminate (or greatly reduce) preprocessing: While convolution with separable kernels greatly reduces preprocessing time, most commercial applications try to avoid even short preprocessing delays. Further, the memory needed to store the resulting graph information

limits the number of frames from a temporal sequence that can be in memory at any given time. Consequently, future work can also include methods for computing the necessary graph information on an “as needed” basis (as if there wasn’t already enough to compute during interaction) to avoid preprocessing delays and reduce high memory requirements. One possible strategy is to “tile” the image (perhaps into blocks of 256×256) and compute and maintain the graph information for each tile as it is needed.

Replace the single element edge model with a multi-element multi-edge model: The edge model described in Chapter 4 assumes that the sampled pixel values are influenced solely by the edge being modelled and thus it has difficulty in areas where multiple edges are interacting. Also, the edge model for each boundary element is computed in isolation, thus it is susceptible to noise since the parameter values computed for one edge element is not influenced (i.e., smoothed) by the edge model for neighboring edge elements. Further, the gradient direction used as the domain projection vector does not always approximate the normal vector of the object boundary (due also to interactions with other edges within the gradient kernel). Finally, the slide path used to sample pixels sometimes follows “strange paths” that results in sampling pixels that are too far away from the domain projection vector while missing pixels that are close to the object boundary. Therefore, another future extension would explore a means of computing the edge model for multiple edge elements simultaneously within a neighborhood to reduce the effects of noise as well as incorporate the cumulative effects of multiple interacting edges. In essence, this would compute the local two-dimensional image function within the neighborhood examined, thereby accounting for and correcting problems such as the loss of true color in narrow image features (as with Fig. 4.11(b)) and the apparent rounding of sharp corners due to blurring by the point spread function. Obviously, this would increase the number of

parameters solved for (one of which could be the direction vector itself), but would greatly augment the edge model's capacity for determining the "true" object edges and colors.

Fit the subpixel boundary points with a spline rather than a polyline: Representing and drawing the live-wire and subsequent object boundaries as a polyline is visually sufficient as long as the error tolerance remains low or as long as the smoothed (i.e., higher tolerance) polyline is representing a straight portion of the object boundary. However, when the object boundary is curved, it is difficult to justify an increase in the error tolerance in order to "smooth out" a "rough" section only to replace it with an unsatisfying polyline. Therefore, another future extension is to fit the subpixel edge points with a second or third degree spline curve (or perhaps some combination of linear, quadratic, and cubic curve sections) in order to better accommodate nonlinear object boundaries.

Reduce the exponential complexity of the edge confidence computation: A major portion of the computation involved in cursor snapping and live-wire path extension lies in the exponential complexity of the extended branch cost given by Eq. 5.13 (which is used in computing an edge's confidence). In fact, the computation of edge confidences for snapping and the computation of the edge model for each live-wire path element are the two major causes of interactive "sluggishness" in the multi-frame live-wire environment. However, the extended branch cost (which basically finds the lowest cost path from among all fixed-length paths in a subtree) is recursive in nature. Further, edge confidences are typically computed for all edges in a neighborhood or a path at the same time (i.e., during the same snapping or path extension operation). Thus, portions of the subtree explored by each extended branch cost are often computed multiple times during a cursor snapping or path extension operation since the subtree computed for one edge overlaps with the subtree computed for an adjacent edge. Hence, future work could explore methods for reduc-

ing the computational complexity involved in computing edge confidences by taking advantage of the extended branch cost's overlapping subtree nature, thereby eliminating redundant computation.

Enhance live-wire path extension to facilitate object selection: Live-wire path extension requires, by definition, a live-wire path to extend and therefore also requires at least one seed point to be able to generate a live-wire. Thus, even though live-wire path extension is able to define the majority of the object boundaries in Figure 5.9, the user is still required to define a seed point and a short live-wire segment. To reduce the human effort even further, future work could explore techniques for automatically displaying high confidence closed boundaries as the cursor moves near and snaps to object boundaries. For example, recall that confidence threshold snapping generated high confidence subtrees that are bounded by the snap neighborhood. If the branches of subtree containing the snap point are extended further such that they form loops, the highest confidence loop or the loop that, based on some analysis of boundary properties, is deemed to be the most likely object boundary could be displayed immediately, thus allowing the user to simply indicate a completed boundary if the displayed loop corresponds to the desired object boundary. Even if the displayed loop does not quite agree with what is wanted, the interface could allow the user to select a portion of the displayed loop and then either indicate an alternative branch or use traditional live-wire to complete the boundary.

Improve the interface by allowing zooming, panning, and scrolling: Both the single image and the multi-frame montage interface currently allow only an *entire* image or frame to be displayed at a scale factor of one (i.e., no zooming or reduction). Any image that is larger than the screen dimensions (horizontally and/or vertically) is not accepted by the current system. Also, the maximum number of frames that can be displayed simulta-

neously in the montage is limited to the number of *full* frames that will fit side-by-side on the screen both horizontally and vertically¹. Since there is nothing inherent in Intelligent Scissors (either pixel- or toboggan-based) that requires it to display a full image, there is no reason the interface could not be improved to show a portion of the image (i.e., a viewport) that pans and scrolls to access the whole image. Further, since the subpixel live-wire path and the object boundaries used in this dissertation are no longer constrained to the pixel grid, an image or frame could be zoomed (via pixel replication) or reduced (via pixel averaging) to allow various levels of user control and visualization detail—more control and detail in difficult regions of the object boundary and less in well-defined areas.

A posteriori addition and/or editing of seed points: There is currently no mechanism for adding new seed points or editing existing seed points for an already defined object boundary². While this ability would be beneficial for both single image and multi-frame Intelligent Scissors, it has much more potential utility in the multi-frame environment where occasions for temporarily ignoring an incorrect boundary segment are more frequent. For example, if, in the current system, the live-wire segment in all but one frame is able to extend well past the point where the single, “holdout” frame fails to follow the object boundary, then all the live-wires must be shortened to accommodate the “weak link”, often resulting in more seed points than are necessary. However, given the ability to return to a frame and “fix up” any problem areas, the user could simply overlook the problem in the individual frame with the intent of returning later to correct the error by adding

1. However, limiting the maximum number of visible frames may be desirable since it limits the number of frames that require immediate interactive feedback.

2. This ability was incorporated into pixel-based Intelligent Scissors, but it has not yet been included into the toboggan-based version.

an extra seed point, thus allowing the live-wire to “realize its potential” in the other frames.

Automatic, efficient, and reliable multi-frame free-point correspondence: Intelligent Scissors for movies would certainly exhibit a higher degree of intelligence if it could quickly and accurately determine inter-frame free-point correspondence automatically rather than relying on the current manually controlled interpolation strategy. Automatic free-point matching between frames would greatly reduce the time and effort required to define the boundary of a moving object. One idea for inter-frame free-point correspondence is to use the high confidence subtree rooted at the free-point in the active frame to find the minimum cost match between a path in this subtree with a high confidence path in the appropriate neighborhood of an adjacent frame. The minimum cost match would include a curvature difference measure and would allow for bounded changes in translation (obviously), scale, rotation, and non-rigid transformations by using an approach similar to DP warping. Further, once a correspondence has been established between two frames, a motion model for a Kalman filter can begin to be built up in order to reduce the size of the neighborhood to search.

Multi-frame live-wire feature cost coupling: Currently, each frame computes its minimum spanning tree independently without any attempt to match optimal path features between frames. This inter-frame anarchy resulting from each frame “following its own path” is the primary reason that the live-wire fails to follow the desired object boundary in one frame while adhering to it in other frames. Thus, along with exploring techniques for matching free-points between frames, future work will also look at coupling the live-wire feature costs from frame to frame. As with inter-frame free-point correlation, the live-wire

path in the active frame would provide a template for computing differential feature costs used during graph expansion in indirect frames.

As can be seen, though the extensions to Intelligent Scissors presented in this dissertation enhance and extend an already effective and useful object segmentation tool, there does not appear to be any immediate end in sight for the possibilities to improve this tool even further. As such, Intelligent Scissors promises to remain on the “cutting edge” of image segmentation technology.