Intelligent Segmentation Tools

William A. Barrett
william_barrett@byu.edu

Eric N. Mortensen

See next page for additional authors

Intelligent Segmentation Tools

William A. Barrett L. Jack Reese Eric N. Mortensen
Brigham Young University

Abstract

Intelligent Scissors and Intelligent Paint are complementary interactive image segmentation tools that allow a user to quickly and accurately select objects of interest using simple gesture motions with a mouse. With Intelligent Scissors, when the cursor position comes in proximity to an object edge, a live-wire boundary "snaps" to, and wraps around the object of interest. The Intelligent Paint tool uses the cursor position to sample the image data interior to the object and grows the current region, in discrete, snapping increments, to include similar neighboring regions. Both techniques make use of a watershed algorithm called toboganning. With Intelligent Scissors, image segmentation is formulated as a piece-wise globally optimal graph searching problem, while Intelligent Paint approaches it as an adaptive, cost-ordered connected component labelling scheme. Using these tools, objects or regions can be selected in a few seconds, with better accuracy and reproducibility than can be obtained using manual selection tools. In particular, interobserver reproducibility using intelligent segmentation tools is far better than intraobserver reproducibility using manual segmentation methods.

1. Introduction

Fully automated, general purpose image segmentation is still an unsolved problem due to the wide range of image content and complexity. Even advanced techniques require the user to interact with the image in *some* way to specify a region or object of interest, initialize a process, or provide some guidance in the segmentation. Since *some* user input *is* required, it makes sense to maximize the information derived from that input. For this reason intelligent segmentation tools which exploit high level visual expertise but require minimal user interaction become appealing.

Intelligent Scissors and Intelligent Paint are complementary, general-purpose, segmentation tools which allow rapid and accurate object extraction from arbitrarily complex backgrounds using simple gesture motions with a mouse [1-4]. Intelligent Scissors allow the user to quickly define object boundaries using circumnavigational gestures. Intelligent Paint allows the user to efficiently select the *region* associated with an object of interest by stroking the cursor through the interior of the object. Both tools utilize toboganing to partition an image into homogeneous regions which are effectively identical to the catchment basins produced by computing the watershed of the image's gradient magnitude map. These regions become the basis for region-based (Intelligent Paint) or boundary-based (Intelligent Scissors) object selection.

Among the more popular boundary-based segmentation approaches, active contours or snakes have recently received significant attention [5-11]. Active contours require a (user-supplied) initial approximation to the boundary and then iteratively minimize the boundary's energy functional to obtain an optimal boundary. Both external forces (gradient magnitude) and internal forces (boundary curvature) are imbedded in the functional in an attempt to produce a final smooth boundary that conforms to the targeted object, resulting in a robust, globally optimal formulation of the boundary finding problem. However, even if interactive "nudging" is allowed [5,6], it is not clear what the final boundary will look like until convergence is achieved. If the desired result is not obtained, additional user interaction, editing and iterations are necessary.

Region-based segmentation algorithms have also seen and increase in robustness and adaptivity in terms of what defines a connected component, and at what level in a hierarchy connectivity is defined. [12-15]. Region competition schemes have also been introduced, combining both region and boundary-based information, including active contour models driven by underlying watershed algorithms [16-18]. However, like some boundary-based approaches, the steps are still initialize (with possibly multiple seed points), connect, and repeat if the result is not what was wanted.

Since most of these approaches require the user to interact with the image at least at some level, if only to point to the object of interest, we would like to exploit that interaction while providing immediate feedback to the user as to what object or region is being selected.

2. Intelligent Segmentation Tools

Unlike previous object selection methods where segmentation typically occurs only after user input is complete, Intelligent Scissors and Intelligent Paint integrate human input with the segmentation process to provide dynamic interaction and feedback between the user and the computation *while* the object is being segmented. The notion of "intelligence" in a segmentation tool is that, with a little help from the user, the tool seems to get the idea of what is wanted and snaps to the boundary or region defining the object of interest. Both approaches make use of toboganing - an efficient watershed algorithm.

2.1. Hierarchical Toboganing

Toboganing, introduced by Fairfield [19] and extended by Yao and Hung [20], oversegments an image into small regions by sliding in the derivative terrain. The basic idea is that, given the gradient magnitude of an image, each pixel determines a slide direction by finding the pixel in a neighborhood with the lowest gradient magnitude. Pixels that "slide" to the same local minimum are grouped together, thus segmenting the image into a collection of small regions (Fig. 1). Toboggan-based regions are similar to the catchment basins produced using a watershed algorithm [20-22] but more computationally efficient.

The derivative terrain often contains many local minima resulting in a large number of small regions containing only a few pixels each. These small regions are of little more use for region-based segmentation than are the individual pixels. This is overcome by reapplying the tobogganing algorithm to each of these small regions (rather than the original pixels), forming a segmentation hierarchy consisting of successively higher level regions which are not dependant on heuristic parameters and which are largely associated with image objects or subobjects. Figure 2 illustrates the various levels in the hierarchy created from an image of a parrot (Fig. 2.a). Notice that since we combine existing smaller regions rather than recomputing new ones using a looser grouping criteria, the boundaries of the resulting parent regions correspond exactly to existing boundaries of their children.

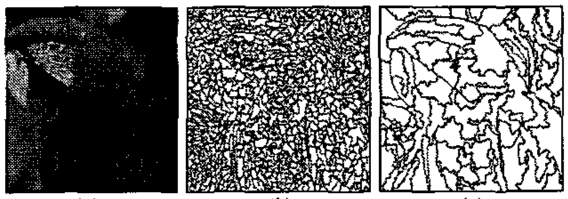


Figure 1: Region hierarchy. (a) Original parrot image. (b) Regions resulting from tobogganing at 2nd region level. (c) 4th level in hierarchy.

Hierarchical tobogganing proceeds in the same fashion as pixel-based tobogganing with the exception that we are grouping regions rather than pixels. To accommodate this, we redefine the *neighborhood* and adopt a general discontinuity measure rather than the pixel derivative used previously. For all levels above the pixel level, we define the neighborhood $N(\boldsymbol{p})$ for a region $\boldsymbol{p}$ as the set of all regions $\boldsymbol{q} \neq \boldsymbol{p}$ such that $\boldsymbol{q}$ is 4-connected to $\boldsymbol{p}$. The discontinuity measure between two regions, $\boldsymbol{p}$ and $\boldsymbol{q}$, is computed using the student's t-score, which computes the probability that two measured distributions have different means. The t-score is given by

$$C(\boldsymbol{p}, \boldsymbol{q}) = p(t|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu}} \int_t^{\infty} \left(1 + \frac{x^2}{\nu}\right)^{-\left(\frac{\nu+1}{2}\right)} \tag{1}$$

where

$$t = \frac{\left|\bar{x}_p - \bar{x}_q\right|}{\sqrt{\sigma_p^2/N_p + \sigma_q^2/N_q}} \tag{2}$$

is the student's t-score and

$$\nu = \frac{\left(\sigma_p^2/N_p + \sigma_q^2/N_q\right)^2}{\frac{\left(\sigma_p^2/N_p\right)^2}{N_p - 1} + \frac{\left(\sigma_q^2/N_q\right)^2}{N_q - 1}} \tag{3}$$

is the degrees of freedom. $\bar{x}$, σ^2, and N are the mean, variance and number of samples of each region, respectively. A solution to equation (1) is given in [23].

2.2. Intelligent Scissors

Intelligent Scissors finds piece-wise optimal boundaries in an image by imposing a weighted, planar graph on the hierarchical toboggan partition. Region junctions (Fig. 1) (shaded circles) define nodes in the graph. Segments between two nodes form (weighted) edges. Optimal paths through the graph define object boundaries. The region-based graph is many times smaller than the pixel-based version used in the original Intelligent Scissors [1], which greatly speeds up the graph search and increases interactive responsiveness.

Intelligent Scissors presents a novel approach to object segmentation. Rather than optimizing a user-initialized approximate contour, we allow the user to interactively select a boundary from a collection of optimal solutions. Thus the user knows exactly what the resulting contour will be *during* interaction. Intelligent Scissors achieves this goal by interactively computing the optimal path from a user selected "seed" point to all other points in the image. As the cursor moves, the optimal path from the pointer position to the seed point is displayed, allowing the user to select an optimal contour segment which visually corresponds to a portion of the desired object boundary. As the mouse position comes in proximity to an object edge, a live-wire [1] boundary "snaps" to and wraps around the object of interest.

Computation of optimal paths corresponding to minimum cost contour segments is accomplished using Dijkstra's [24] optimal graph search. Given a user selected "seed" node, η_s, the optimal graph search algorithm is:

Algorithm 1: Optimal Graph Search.

```
Input:
  $\eta_s$                  {Seed node.}
Data Structures:
  L                         {List of active nodes sorted by total cost (initially empty).}
  edge($\eta$)              {Edge set containing edges emanating from $\eta$.}
  done($\eta$)              {Boolean function indicating if $\eta$ has been expanded.}
  g($\eta$)                 {Total cost function from $\eta_s$ to $\eta$.}
Output:
  opt($\eta$)               {Edge from $\eta$ indicating optimal path back to $\eta_s$.}
Algorithm:
  g($\eta_s$)←0;  L←$\eta_s$;          {Initialize active list with zero cost seed node.}
  while L ≠ ∅ begin                    {While still nodes to expand:}
    $\eta$←min(L);                     {Remove minimum cost node $\eta$ from active list.}
    done($\eta$)←TRUE;                 {Mark $\eta$ as expanded (i.e., processed).}
    for each e ∈ edge($\eta$) begin
      $\eta_n$←e_dst($\eta$);                          {Get neighbor connected to edge.}
      if not done($\eta_n$) then begin
        g'←g($\eta$)+f(e);                             {Compute total cost to neighbor.}
        if $\eta_n$ ∈ L and g' < g($\eta_n$) then      {Remove higher cost }
          $\eta_n$←L;                                  { neighbor from list.}
        if $\eta_n$ ∉ L then begin                     {If neighbor not on list, }
          g($\eta_n$)←g'; opt($\eta_n$)←e;             { assign total cost, set opt. edge}
          L←$\eta_n$;                                  { and place on (or return to) }
        end                                            { active list.}
      end
    end
  end
```

While the optimal graph search algorithm presented here is similar to our original work [1], the reduced graph size allows optimal paths to be computed anywhere from two to 10 or more times faster.

2.3. Intelligent Paint

Intelligent Paint uses the toboggan region hierarchy to segment objects by interactively collecting regions in cost order. Each paint stroke interactively samples the properties of the underlying regions as the intelligent paint adaptively flows into similar neighboring regions. This strategy coordinates human-computer interaction to extract complex regions of interest from arbitrary backgrounds using simple paint strokes with a mouse.

Intelligent Paint uses an efficient cost ordered collection algorithm with on-the-fly training using dynamically updated cost look-up-tables, dynamically spawned seed or sample regions, adaptive cooling, and adaptive thawing to maintain efficiency and accuracy. This allows minimal user input to guide the algorithm where the segmentation becomes difficult. The region collection algorithm is based on the same graph expansion used by Intelligent Scissors with a few important distinctions. First, we use the graph expansion to collect regions within the desired object rather than to select boundary segments from a collection of optimal paths. Second, the weighted graph is formulated by treating regions as weighted nodes with edges connecting neighboring regions. Third, we utilize local region-based costs. Fourth, we do not want to expand the entire graph; region collection terminates either by releasing the mouse button or automatically via adaptive cooling. Fifth, the user is provided real-time visual feedback and interacts directly with the graph expansion process itself, not just it's results.

Algorithm 2: Cost-ordered Region Collection

```
Input:
  s                {Seed region.}
Output:
  L(p)             {Label for each region p (Bkgrnd, Foregrnd, Active, Collect).}
Methods:
  A                {Active list of regions sorted by cum. cost (initially empty).}
  C                {List of collected regions (initially empty).}
  l(p,q)           {Local cost measure between regions p and q.}
  N(p)             {Neighbors of region p.}
  t(p)             {Total (cumulative) cost of region p.}
  $T_d$, $T_s$           {Dynamic and static thresholds for cooling.}
Algorithm:
t(s) = 0;  A ← s;                  {Initialize active list with seed region.
while A not empty begin            {Process while regions on active list.}
  p = min(A);                      {Remove min cost region from A.}
  L(p) = Collect;                  {Mark removed region as collected.}
  C ← p;                           {Place region on collected list.}
  for each q in N(p) begin         {For each neighboring region of p.}
    if L(q) ≠ Collect and           {If neighbor not collected and}
        $l(p,q) < T_d$ and              {local cost < dynamic and}
        $l(p,q) < T_s$ begin            {static threshlds: Add to active list.
      t(q) = t(p) + l(p,q);        {Assign cumulative cost to neighbor.}
      A ← q;                       {Place q on active list.}
      L(q) = Active;               {Mark q as being on the active list.}
    end                            {Done adding region to active list.}
  end                              {Done processing neighbors of p.}
end                                {Done expanding region.}
while C not empty begin            {Process all regions on Collect list.}
  p ← C;                           {Remove region from list.}
  L(p) = Foregrnd;                 {Relabel region as Foregrnd.}
end                                {Done relabelling collected regions.}
```

Collection begins when a user clicks in the image with the mouse. First, a dynamic 3-D color look-up-table, which acts as a remapping function for region pixels, is initialized to zero. Then, the region (at whatever level in the hierarchy the tool is currently using) corresponding to the cursor position is placed on a sorted, cumulative cost list (called the active list) with a total cost of 0. Intelligent Paint then iteratively removes the lowest cost region from the active list, marks it as processed (which means that it becomes part of the segmented region), updates the remapping function by incrementing each look-up-table entry corresponding to a pixel color in the region, and then expands the region by computing the total cost to all of its neighboring regions and placing them on the active list in sorted order. The total cost to a neighboring region is simply the sum of the neighbor region's local cost and the accumulated cost of the region being expanded (which is the total cost used to place it on the active list). A region's local cost is computed by summing the remapped pixel values for the neighboring region and then scaling the resulting sum by the mode of the color look-up-table. Thus, given region, p, just pulled off the active list with total cost $t(p)$, and a neighboring region, q, the total cost for region q is given by

$$t(q) = t(p) + l(p, q) \tag{4}$$

where

$$l(p, q) = l_{max}\left(\frac{m_{mode} \cdot N_n}{\sum_{r \in P(q)} m(I(r))} - 1\right) \tag{5}$$

is the local cost of adding region q from region p, m is the 3D color remapping function (i.e., color look-up-table), m_{mode} is the mode of m, $P(q)$ is the set of pixel positions contained in region q, N_q is the number of pixels in region q, and l_{max} is a scaling factor indicating the largest integer local cost. After computing the total cost for a neighboring region, the region is placed on the active list in sorted order based on its total cost. Intelligent Paint uses an efficient cost-ordered collection algorithm and performs on-the-fly training using dynamically updated look-up tables, interactively spawned seed regions, and adaptive cooling.

3. Results and Conclusion

Figure 2 illustrates Intelligent Paint segmentation of abdominal anatomy from the visible human project in just a few seconds with only a few mouse clicks while maintaining connectivity through the interior of the object. Figure 3 illustrates the application of Intelligent Scissors to CT scans where the results are accurate and virtually instantaneous. A more extensive analysis (Figure 4) shows Intelligent Segmentation Tools to far superior to manual tracing in efficiency, accuracy and reproducibility. In fact, multiple users using these tools are *collectively* (interobserver) more consistent than a single individual (intraobserver) performing manual tracing!

4. References

[1] E. N. Mortensen and W. A. Barrett, "Intelligent Scissors for Image Composition," in *Proc. of the ACM SIGGRAPH 95: Computer Graphics and Interactive Techniques*, pp. 191-198, Aug. 1995.

[2] E. N. Mortensen and W. A. Barrett, "Toboggan-Based Intelligent Scissors with a Four Parameter Edge Model," in *CVPR*, 452-458, 1999.

[3] E. N. Mortensen, L. J. Reese, and W. A. Barrett, "Intelligent Selection Tools," in Proc. IEEE: Computer Vision and Pattern Recognition (CVPR '00), Vol. II, pp. 776-777, Hilton Head, SC, June 2000.

[4] D. Daneels, et al., "Interactive Outlining: An Improved Approach Using Active Contours," in *SPIE Proc. Storage and Retrieval for Image and Video Databases*, 1908: 226-233, Feb. 1993.

[5] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Int. Journal of Computer Vision*, **1**(4): 321-331, Jan. 1988.

[6] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, "Dynamic Programming for Detecting, Tracking, and Matching Deformable Contours," *IEEE PAMI*, **17**(3): 294-302, Mar. 1995 (Correction in *PAMI*, **18**(5): 575, May 1996)

[7] D. J. Williams and M. Shah, "A Fast Algorithm for Active Contours and Curvature Estimation," *CVGIP: Image Understanding*, **55**(1): 14-26, Jan. 1992.

[8] T. McInerney and D. Terzopoulos," Topologically Adaptable Snakes", ICCV95, 840-845, 1995.

[9] N. Paragios and R. Deriche, "Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects", *PAMI*, 22, No. 3, 266-280, 2000.

[10] L. Lorigo, O. Faugeras, W. E. L. Grimson, R. Keriven, R. Kikinis and C.Westin, "Co-dimension 2 geodesic active contours for mra segmentation," In Int'l Conf. Inf. Proc. in Med. Imaging,126--139, 1999.

[11] P.K. Saha and J.K. Udupa, "Relative Fuzzy Connectedness among Multiple Objects: Theory, Algorithms, and Applications in Image Segmentation,", CVIU, V. 82, No. 1, 1-32, 2001.

[12] G. T. Herman and B. Carvalho, "Multiseeded Segmentation Using Fuzzy Connectedness," PAMI, V. 23, 5, 460-474, 2001.

[13] J. Shi and J. Malik, "Normalized cuts and image segmentation," *CVPR'97*, 731-737.

[14] M. Tang and Songde Ma, "General Scheme of Region Competition Based on Scale Space," *PAMI*, V. 23, No. 12, 1366-1378, 2001.

[15] S. C. Zhu, T. S. Lee, and A. L. Yuille, "Region Competition: Unifying Snakes, Region Growing, and Bayes/{MDL} for Multiband Image Segmentation", *PAMI*, V. 18, No. 9, 884-900, 1996.

[16] I. Jermyn and H. Ishikawa, "Globally Optimal Regions and Boundaries as Minimum Ratio Weight Cycles," *PAMI*, V. 23, No. 10, 1075-1088, 2001.

[17] J. Park and J. Keller, "Snakes on the Watershed," *PAMI*, V. 23, No. 10, 1201-1205, 2001.

[18] J. Fairfield, "Toboggan Contrast Enhancement for Contrast Segmentation," in *IEEE Proc. 10th Int. Conf. on Pattern Recog.*, **1**: 712-716, 1990.

[19] X. Yao and Y. P. Hung, "Fast Image Segmentation by Sliding in the Derivative Terrain," in *SPIE Proc. Intelligent Robots and Computer Vision X: Algorithms and Techniques*, **1607**: 369-379, Nov. 1991.

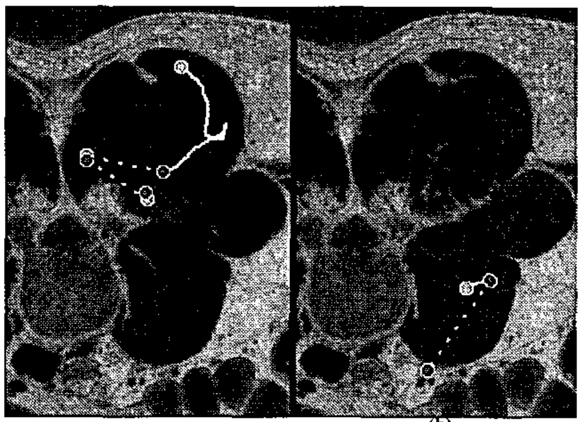
[20] S. Beucher and C. Lantuéjoul, "Use of Watersheds in Contour Detection," in *Proc. Int. Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Sept. 1979.

[21] L. Najman and M. Schmitt, "Geodesic Saliency of Watershed Contours and Hierarchical Segmentation," *IEEE PAMI*, **18**(12): 1163-1773, Dec. 1996.

[22] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE PAMI*, **13**(6): 583-598, June 1991.

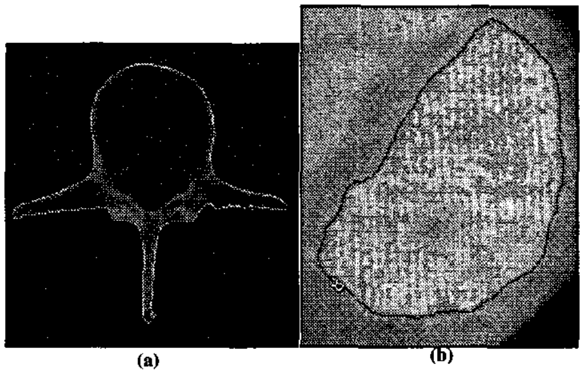
[23] W. H. Press, et al., *Numerical Recipes in C*, ch. 15: 681-688, Cambridge University Press, 2nd ed., 1992.

[24] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, **1**: 269-270, 1959.



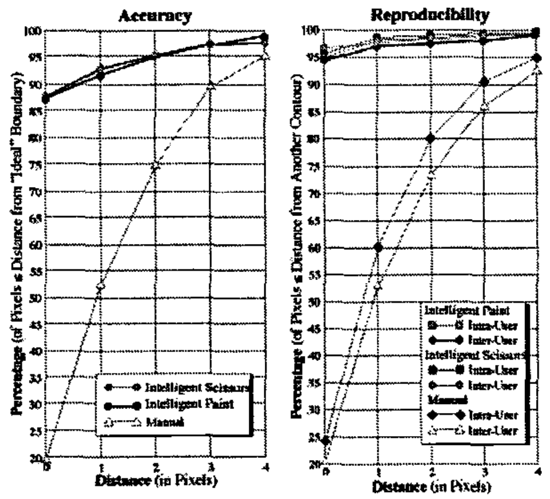
(a) (b)

Figure 2: Abdomen. Segmentation with Intelligent Paint showing the mouse movements (white lines) and mouse clicks (press → green circle, release → red circle): (a) 7 seconds with 3 mouse clicks (b) 2 seconds with 2 mouse clicks.



(a) (b)

Figure 3: (a) CT - Lumbar Spine. Segmented in .21 sec. with 1 seed point. (b) Cine CT - Left Ventricle. Segmented in .11 sec. with 1 seed point.



(a) (b)

Figure 4: Accuracy and Reproducibility for Intelligent Scissors and Intelligent Paint is far superior to manual tracing.