ACCV2002: The 5th Asian Conference on Computer Vision, 23 – 25 January 2002, Melbourne, Australia

Trail-Dependent Intelligent Scissors Based on Multi-Scale Image Segmentation

Yi-Ping Hung and Yu-Pao Tsai
Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
{hung, yptsai}@iis.sinica.edu.tw

Abstract

Image segmentation is a very important topic in computer vision. However, due to the large variation of image content, fully automatic image segmentation for general applications is still an open problem. Therefore, our goal is to develop an interactive image segmentation tool that can accurately extract the desired object boundaries with minimal human efforts. In this paper, we propose a new trail-dependent intelligent scissors, which let the user interactively extract desired object boundaries based on multi-scale image segmentation. By utilizing the information contained in the trail of the cursor's motion, which somewhat implies the intention of the human operator, our intelligent scissors can allow the user to extract a desired object boundary with less mouse-clicking, and hence is more user-friendly. This is the major advantage of our new intelligent scissors. Another advantage is that our intelligent scissors permits the user to trace the object boundary with less tension by utilizing the coarse-to-fine region boundaries provided by multi-scale image segmentation. Our experiments have demonstrated that the new interactive segmentation tool requires less human efforts than the previously available tools.

Keywords: image segmentation, intelligent scissors, trail-dependent, multi-scale, watershed, toboggan algorithm.

1. Introduction

Image segmentation is a very important topic in computer vision. However, due to the large variation of image content, fully automatic image segmentation for general applications is still an open problem. Therefore, our goal is to develop an interactive image segmentation tool that can accurately extract the desired object boundary with minimal human efforts.

Typical examples of interactive image segmentation tools are magic wand, active contour [1] and intelligent scissors [2]. Magic wand is a segmentation tool that can be found in Photoshop. When a user selects a sample pixel with the magic wand, a connected region will be formed, which consists of all the pixels that fall within an adjustable tolerance of the sample pixel. Active contour is another common tool for image segmentation. It starts with a given initial contour around the desired object, and seeks a better contour encompassing the desired object by minimizing an energy function that combines internal forces, such as gradient magnitude, with external forces, such as boundary curvature. Intelligent scissors interactively shows the optimal path, which is supposed to coincide with the desired object boundary, from a given seed point to the current cursor position. Among the above three tools, intelligent scissors is probably the most intuitive one to use since the user can trace the desired boundary by interactively modify the cursor point, while it is harder to predict the extracted boundary produced by the magic wand or active contour algorithm. A brief review on intelligent scissors will be given in Section 2.

Although the intelligent scissors is relatively intuitive to use, there is still much room for improvements. A problem with the previous intelligent scissors is that the boundaries extracted are trail-independent. That is, the extracted boundary depends only on the image positions of the seed point and the current cursor point, and is not dependent on how the human operator moves the cursor from the seed point to the current cursor point. In this paper, we propose a new trail-dependent intelligent scissors, which let the user interactively extract desired object boundaries based on multi-scale watershed image segmentation.

The remainder of this paper is organized as follows. Section 2 gives a brief review of the conventional intelligent scissors, and introduces some basic concepts and terminology that will be used in Section 3. Details of the proposed trail-dependent intelligent scissors are presented in Section 3. In Section 3.1, we focus on the multi-scale region-based technique. In Section 3.2, we focus on the trail-dependent technique. Section 4 shows some experimental results and demonstrates the advantages of our method. Finally, Section 5 gives a conclusion.

2. Review of Intelligent Scissors

As mentioned in the last section, intelligent scissors [2] is an interactive image segmentation tool that allows the human operator to extract desired object boundaries by selecting a sequence of optimal paths corresponding to object boundaries. First, the input image is considered as a weighted graph. All pixels of the input image are nodes of the weighted graph, with weighted edges connecting each pixel with its eight adjacent neighbors. The local cost on a weighted edge *E(p,q)* that connect the vertex *p* and *q* is calculated by weighting sum of the image feature functions. After the weighted graph is constructed, the human operator then selects a seed point. The optimal paths from the seed point to each pixel are determined by applying Dijkstra's shortest path searching algorithm. Next, when the human operator moves the cursor to an image position lying on the object boundary he desires, the piece-wise optimal path between the seed point and the current cursor point will be displayed accordingly. This segment of optimal path is called a "live-wire".

In 1999, Mortensen and Barrett proposed a region-based intelligent scissors to speedup the pixel-based intelligent scissors. First, they partitioned the input image into a collection of regions using the toboggan algorithm. Notice that, after applying the toboggan algorithm, each edge of the weighted graph corresponds to a segment of region boundary that is composed of sequence of pixel "cracks" (the "crack" between two neighboring pixels). Let *L(p)* and *N(p)* denote the label of pixel *p* and the 4-connected neighborhood of pixel *p*, respectively. The pixel crack between *p* and *q* is defined as the ordered pair *(p,q)* such that $q \in N(p)$ and $L(p) \neq L(q)$, and the crack direction vector, $d_{p,q} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}(p-q)$, is the vector pointing clockwise relative to *p*. Consider a region with label *l*, and suppose that its region boundary is composed of a sequence of pixel cracks, denoted by $B(l) = ((p_1,q_1),(p_2,q_2),...,(p_i,q_i),(p_{i+1},q_{i+1}),...,(p_n,q_n))$, where $L(p_i)=l$ and $L(q_i)\neq l$. Then, there is a node on the region boundary whenever $L(q_i)\neq L(q_{i+1})$, and the node position can be computed by the following equation:

$$\eta_i = \frac{1}{2}\left(p_i + q_i + d_{p_i,q_i} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \quad (1)$$

Further, an edge is represented as a quadruple:

$$E(\eta_i, \eta_j, l_l, l_r) \quad (2)$$

which records its two connected nodes and the labels of the two adjacent regions separated by this edge *E*. The details for computing the edge weights can be found in [3]. Once the weighted graph is constructed, the remaining algorithm is the same as the pixel-based approach. However, when compared with the pixel-based approach, the number of graph nodes created by the region-based approach is much less, and hence the computational cost can be greatly reduced.

3. New Intelligent Scissors

The intelligent scissors described in the previous section is a good interactive tool for image segmentation. Our goal is to develop new techniques that can further reduce human efforts in manual operation. The major contribution of this paper is to achieve the above goal by incorporating multi-scale image segmentation and trail-dependent scheme into the intelligent scissors. The former is described in Section 3.1 and the latter in Section 3.2.

3.1 Multi-Scale Intelligent Scissors

In Section 2, we have reviewed a region-based intelligent scissors, which constructs a weighted graph based on the image partition produced by the toboggan algorithm. The toboggan algorithm is in fact an algorithm for implementing watershed segmentation, and the immersion simulation is the other approach [6].

In watershed segmentation, one can imagine the watershed regions to be the catchment basins, and the region boundary to be the ridges around the basin. The desired object boundaries almost always occur at the boundaries of those watershed regions. Notice that a higher ridge implies stronger evidence that an object boundary exists. Based on the relative height of the ridge, some researchers [4][5] have proposed methods for constructing watershed regions with a hierarchical representation. To construct a hierarchical representation of segmentation, each boundary segment is assigned a value, called "dynamics", when constructing watershed regions [4]. The dynamics of a ridge is defined to be the minimal height from bottom of basin to the ridge. Given a dynamic threshold, T_{dyn}, connecting regions sharing a ridge having a dynamics smaller than the threshold will be merged. Therefore, by adjusting the dynamic threshold, T_{dyn}, one can determine the desired level of watershed segmentation in the hierarchical representation. If a segment of region boundary exists at a coarser level of the hierarchical representation, it implies that this segment of region boundary is more salient or more important in image segmentation. Figure 1(b) shows the dynamics of the region boundaries, where the darker lines represent the region boundaries having larger dynamics.

Usually, the desired object boundaries follow the segments of region boundary that have higher dynamics. When tracing the object boundary at a coarser level of the hierarchical segmentation, the human operator can move the cursor with less stress since it is less likely to be interfered by weaker or unimportant edge signals.

Furthermore, the lengths of the boundary segments at the coarser level tend to be longer than those at the finer level.

Here, we describe a technique for integrating multi-scale image segmentation into the intelligent scissors. When constructing the weighted graph based on multi-scale watershed segmentation, we modify the edge representation described in Equation (2) as follows in order to include the dynamics of the graph edge:

$$E(\eta_i,\eta_j,l_l,l_r,d) \tag{3}$$

where d is the dynamics of the edge E. Before performing the shortest path search, we first determine whether an edge should be considered or not with a given dynamic threshold T_{dyn}. The edge will only be considered if its dynamics is larger than T_{dyn}. Notice that if the threshold T_{dyn} is set to be zero, the obtained optimal path will be exactly the same as that obtained by the conventional (finest level) region-based intelligent scissors. Figure 2 illustrates the different results obtained by applying the intelligent scissors at different levels of watershed image segmentation. In this experiment, we arbitrarily select a seed node, marked as "⊕", and then move the cursor to an image position, marked as "+" in the upperthree the images. The lower three images show the watershed regions constructed with different dynamics thresholds, T_{dyn}, and the darkest line indicates the optimal path found between the seed point and the cursor point. Figure 2(a) shows the result obtained at the finest level, which is exactly the same as that obtained by the conventional region-based intelligent scissors, and Figures 2(b) and (c) show the result obtained at the middle and coarser levels, respectively. It can be seen that the manual operation at the coarser level can extract the desired object boundary without placing the cursor right at the desired region boundaries, which allows the human operator to manipulate the cursor with less stress.

3.2 Trail-Dependent Intelligent Scissors

When extracting object contours from a single image using an intelligent scissors, the user usually has to select a series of optimal path segments by carefully clicking some nodes along the object boundary. The information contained in the trail of the cursor's motion, which somewhat implies the intention of the human operator, is not used in the conventional intelligent scissors. In this section, we introduce a new trail-dependent intelligent scissors, which utilizes the information contained in the cursor's trail, and hence can allow the user to extract the desired object boundary with less mouse-clicking, and hence is more user-friendly.

First, we apply the same procedure as the conventional region-based intelligent scissors to create a weighted graph. Next, we record the cursor's trail while the human operator is moving the cursor to select the optimal path. In our implementation, we record the cursor's trail in an ROI (region of interests) map. The ROI map is of the same size as the input image. While the human operator is dragging the cursor in the input image, we mask the pixel in the ROI map corresponding to the cursor position to be a part of ROI. To allow the human operator to move the cursor more freely, we let the cursor have an effective area. We use a mask window centered at the current cursor point to record the cursor's trail in the ROI map. That is, we assume that all the pixels falling within the mask window are the region in which the human operator is interested. We called this mask window the cursor window, $W(p)$, corresponding to the cursor position p. To start with, the human operator first selects the first seed node. Once the human operator begins to move the cursor, the ROI map will be updated. Then, we perform the shortest path search algorithm as the conventional region-based approach, except that all shortest paths must lie within the ROI. This constrained search will be referred to as *the ROI shortest path search* from now on.

This simple method works well in finding the desired optimal path from the seed node to the cursor node. However, it will fail when the ROI is connected to be a closed path along desired object contour. The desired path is the closed contour, but unfortunately the optimal (shortest) path from the seed node to the cursor node is not. To solve this problem, we must determine whether the cursor's trail is a closed path or not. Hence, we need to know where is the head of the cursor's trail and when the cursor's trail begins to form a closed path. The former problem can be solved by selecting the first seed node to be the head of cursor's trail. But the latter problem is more troublesome. It is possible that the cursor may leave from the first seed node and then begin to approach it again. But it is also possible that the human operator would like to select the optimal path around the first seed node.

To distinguish between these two situations, we assume that the cursor's trail will form a closed path only after the cursor has moved out of the bounding area of the first seed node and entry again. Centered at the first seed node, we define two areas, the core area and the bounding area, to help determining the cursor state. The cursor state is either the initial state, the starting sate, the developing state, the alerting, or the ending state. The bounding area is used to determine whether the cursor is moving out or is reentering. The core area is used to determine whether the process should be terminated or not. Figure 3(a) shows the relationship between the core area and the bounding area. The state diagram shown in Figure 3(b) illustrates the transition of the cursor state. The description of each event is listed in Table 1. At the

very beginning, we initialize the cursor state to be the initial state. In the initial state, the only process is waiting for the human operator to select the first seed node, and then the cursor state is switched into the starting state. After entering the starting state, the state is checked whenever the human operator is moving the cursor. Further, the ROI map will be updated and the ROI shortest path search is performed to compute the optimal paths, until the cursor state is switched into the ending state.

Table 1: List of events.

Event Number	Description of the event
1	The human operator selects the first seed node
2	Cursor is moved inside the bounding area
3	Cursor is moved out of the bounding area
4	Cursor is moving outside of the bounding area
5	Cursor is moved into the bounding area
6	Cursor is moved out of the bounding area again
7	Cursor is moving inside the bounding area
8	Cursor is moved into the core area

Figure 4 compares the image segmentation results obtained by using the trail-dependent and trail-independent intelligent scissors. As shown in Figure 4(a), we first select node A to be the initial seed node, and then move the cursor to node B, C, D, E and F, successively. The object boundaries extracted by the trail-independent intelligent scissor at different stages are shown in Figures 4(a), 4(b), 4(c), 4(d), 4(e) and 4(h). On the other hand, the object boundaries extracted by our trail-dependent intelligent scissors are shown in Figures 4(a), 4(b), 4(c), 4(d), 4(g) and 4(j). When the cursor is moved from A to B, to C and to D, the results obtained by both methods are the same. However, if the cursor continues to move to E and then to F, the trail-dependent method can still extract the desired object boundary, while the trail-independent one will favor an undesired shorter path. Notice that Figures 4(f) and 4(i) show the cursor's trail and ROI map corresponding to Figures 4(g) and 4(j). Here, the cursor position is indicated by darker dots, which falls within the shaded region representing the ROI.

4. Experimental Results

Some experimental results are shown in this section in order to illustrate the advantages of the new intelligent scissors proposed in this paper. First, consider Figure 5. Figure 5(a) shows the source image, and Figure 5(b) shows the watershed regions constructed with a relatively large dynamics threshold. To begin with, we arbitrarily click a seed node residing on the desired object contour, and then move the cursor to extract the desire object boundary. When the cursor is moving, the ROI map will be updated and then the shortest path search in ROI will be performed to compute the optimal path for each node within the ROI. After the cursor is moved out of the bounding area (defined in Section 3.2), if it is moved into the bounding area again, our algorithm will select a new seed node automatically. Figure 5(c) shows two seed nodes, one is the initial seed node selected manually and the other is the one selected automatically by our algorithm. Finally, the extracted object boundary is the white line shown in Figure 5(c), and the ROI map and trail information is shown in 5(d). With our trail-dependent intelligent scissors, object extraction for simple images, such as the one shown in Figure 5, usually requires only one initial button clicking, followed by simple casual tracing.

The next example is more complex than the first one. Figure 6(a) shows the source image used in the second example. Before tracing the object boundary, the user can first select a dynamics threshold (if he does not like the default one) so that most of the desired boundaries can appear at this level of watershed segmentation. For example, the watershed regions obtained with a relatively high threshold is shown in Figure 6(b).

First, we select an initial seed node and then move the cursor along the desired object boundary. When the cursor is moving around, the extracted boundary corresponding to the moving cursor will be displayed on-line to provide interactivity. Unfortunately, the extracted boundary may not continue to grow as one might expect. This is because we have selected a relatively high dynamics threshold, and thus the desired boundary segment did not appear in the result of watershed segmentation, as shown in Figure 6(c). One solution is to lower down the dynamic threshold to allow the weaker boundary appeared in the watershed segmentation results. There is an example in Figure 6(d), and the intermediate result is shown in Figure 6(e). However, more caution has to be taken and more careful clicking has to be performed by the user. Hence, after extracting the desired weak boundary, we can increase the dynamics threshold to lessen the stress caused by the requirement of accurate (or high-resolution) cursor movement and mouse clicking.

Based on the multi-scale scheme, the human operator can decrease the dynamics threshold to select the detailed edge of a desired object boundary. The final segmentation result is shown in Figure 6(f).

5. Conclusion

In this paper, we have presented a new interactive image segmentation tool, which combines two techniques, the trail-dependent scheme and the multi-scale image

segmentation, with the region-based intelligent scissors. This new method for interactive image segmentation has two major advantages over the conventional intelligent scissors. The first one is due to the utilization of the cursor trail, which contains the information related to the intention of the human operator. The use of the trail information makes our trail-dependent intelligent scissors require less mouse-clicking, and hence is more user-friendly. The second advantage is due to the use of multi-scale watershed segmentation, which prevents the user from being interfered by weak and irrelevant details and allows the user to trace the object boundary with less tension. Another power of using multi-scale watershed segmentation is that it allows the user to easily adjust the coarseness of segmentation and adaptive to different image content. Our experiments have demonstrated that this new interactive segmentation tool is highly flexible for any situation, and in general requires less human efforts than the previously available tools.

6. References

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

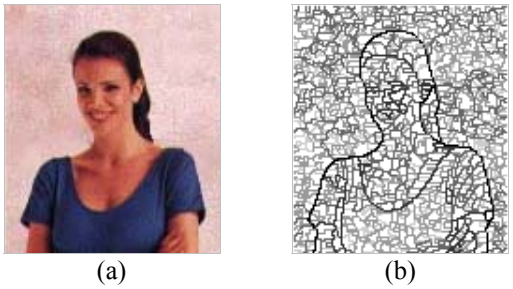
[2] E. N. Mortensen and W. A. Barrett, "Intelligent Scissors for Image Composition," in *Computer Graphics (SIGGRAPH `95)*, pp. 191-198, 1995.

[3] E. N. Mortensen and W. A. Barrett, "Toboggan-Based Intelligent Scissors with a Four Parameter Edge Model," in *Proc. IEEE: Computer Vision and Pattern Recognition (CVPR'99)*, Vol. II, pp. 452-458, June 1999.

[4] L. Najman and M. Schmitt, "Geodesic Saliency of Watershed Contours and Hierarchical Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **18**(12), pp. 1163-1173, 1996.

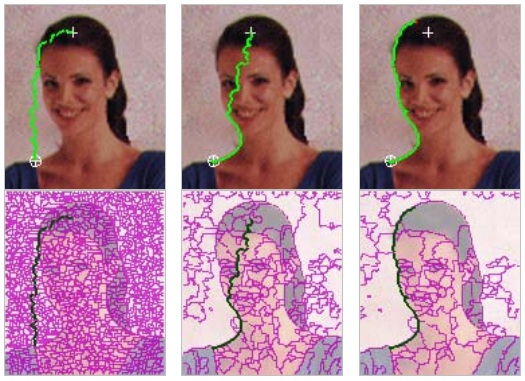
[5] L. Vincent, "Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **2**(2), pp.176-201, 1993.

[6] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **13**(6), pp. 583–598, 1991.



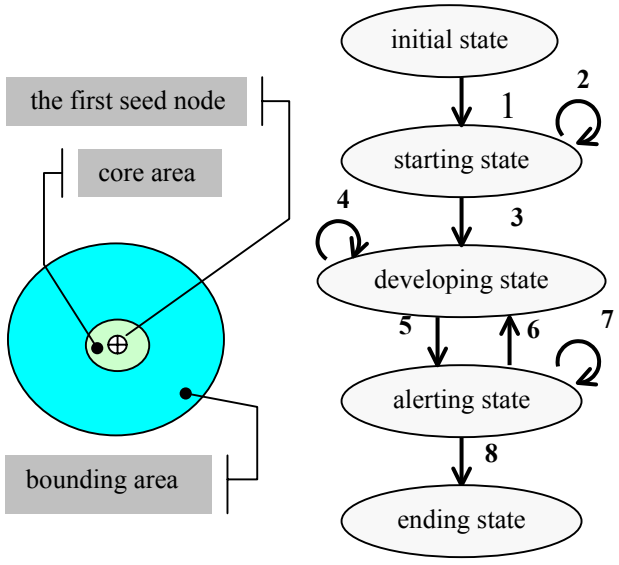
(a) (b)

Figure 1. An example of the hierarchical representation of watershed. (a) Source image. (b) Watershed regions. The region boundary segments marked by darker lines have higher dynamics.



(a) (b) (c)

Figure 2. An example of multi-scale intelligent scissors with (a) smallest (b) middle and (c) largest T_{dyn}. The images on the upper row shows the extracted boundary from the seed point, $\oplus$, to the cursor node that is nearest to the current cursor point, +. The images on the lower row show the watershed regions and the optimal paths at different levels.



(a) (b)

Figure 3. The state diagram for determining the cursor state. (a) shows the relationship between the core area and the bounding area. (b) shows the state diagram. The elliptical circles are the states. The state transitions are shown by using arrows with a number indicating the associated event. Descriptions of the corresponding event are summarized in Table 1.

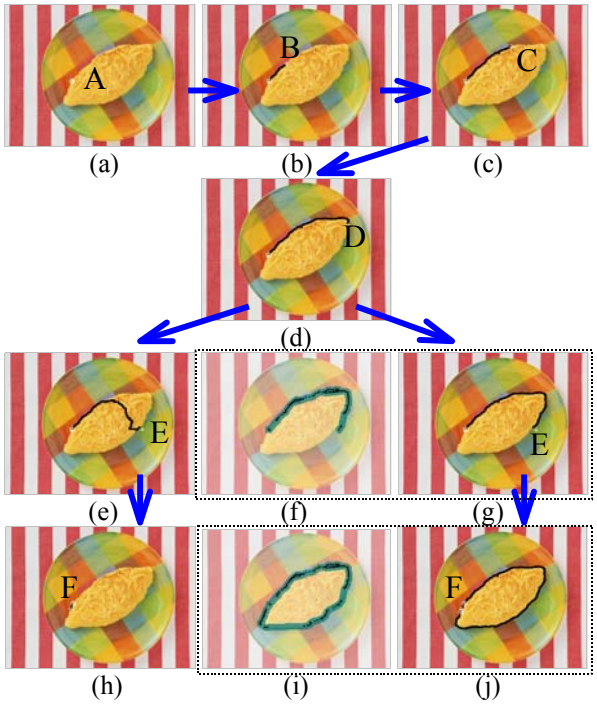


Figure 4. Comparison between trail-independent and trail-dependent intelligent scissors. (a),(b),(c),(d),(e), and (h) show the results of image segmentation obtained by using the trail-independent intelligent scissors. (a),(b),(c),(d),(g),(j) show the results obtained by using the trail-dependent intelligent scissors. The darker strokes in (f) and (i) are the cursor's trails of (g) and (j), respectively. The black dots are the cursor positions.

Figure 5. An example of image segmentation using the trail-dependent and multi-scale scheme. (a) shows the source image, and (b) shows the watershed regions obtained with a high dynamics threshold. (c) shows the segmentation result and (d) shows its cursor's trail.

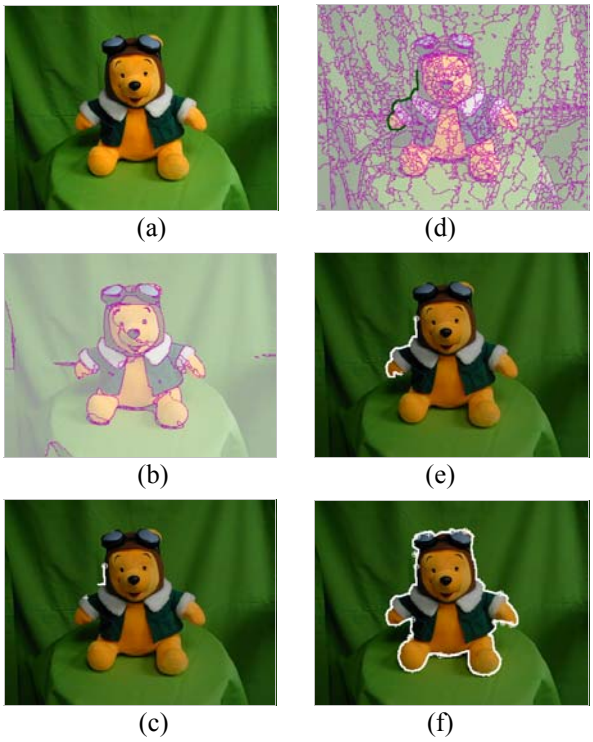


Figure 6. An example of image segmentation using the trail-dependent intelligent scissors based on multi-scale watershed image segmentation. (a) Source image. (b) Watershed regions with a relatively high dynamics threshold (c) The intermediate result. There exists no watershed ridge (with the chosen dynamics threshold) on the desired object boundary. Hence, the desired object boundary can not be extracted. (d) Watershed regions with a relatively low dynamics threshold. (e) The intermediate result based on the watershed ridge in (d). (f) The final segmentation result.