Wavelet-based Semi-automatic Segmentation of Image Objects

Thomas Haenselmann, Claudia Schremmer, Wolfgang Effelsberg
Praktische Informatik IV, University of Mannheim, Germany
{haenselmann|schremmer|effelsberg}@informatik.uni-mannheim.de
phone: +49 621 181 {2603, 2604, 2600} fax: +49 621 181 2601
adress: L 15, 16 - 68161 Mannheim, Germany

Abstract

We present a wavelet-based algorithm to cut an object out of an image in a semi-automatic manner. The user of the algorithm selects a piece of a boundary that separates an object from the background. In contrast to other well-known segmentation algorithms, this sample boundary does not have to be a sharp edge, but might also be a smooth transition between two shades or a texture. Based on a multi-scale wavelet analysis of the sample area our algorithm seeks a path around the object by analyzing characteristic frequencies.

Keywords: segmentation, multi-scale analysis, object boundary, intelligent scissor

1 Introduction

Cutting shapes (e.g., people, cars) out of an image is a time-consuming task that occurs daily in television, news magazins, commercials or web pages. It is desirable to have a tool that detects objects and finds their boundaries automatically. In computer graphics, this is referred to as *object segmentation.* Decades of research in segmentation have taught us how difficult it is to build such a tool since objects that form a unit for a human viewer do not necessarily have a uniform color, a continuous edge, or a similar texture. For example the color and texture of trousers may differ from those of a shirt which makes it difficult to cut out an entire person rather than just the upper or lower part.
In our approach, the user defines a sample of the contour that is then tracked. The algorithm follows this contour as long as possible; it may take further user interaction if the appearance of the object changes significantly. In contrast to existing algorithms ours does not need a sharply defined edge or even any edge at all. The algorithms can follow any kind of visible transition between regions in the image that forms a continous path.

2 Related work

Despite decades of research in image analysis, reliable segmentation of real-world objects without user interaction is not yet possible. A number of attempts been made to ease this task by developing intelligent 'scissors' in image-manipulation programs.

Very early approaches operated on single pixel color only. A histogram is derived from the image that contains all possible shades of gray [JAE97]. Then the algorithms try to find a threshold in the histogram that clearly separates background intensities from object intensities. If all shades above a certain threshold are object colors and all shades below it are background colors, it can be decided by pixel which does belong to objects or to the background [ROS82]. The advantage of this class of algorithm is that objects can have an arbitrary shape or even consist of single pixels. Nonetheless this process requires very strong assumptions with regard to the color histogram.
For example, if dark shades belong to the object and light shades to the background, a light reflex on a shiny object could be mistaken for a background pixel.

A second class of well-known segmentation algorithms are based on region-growing [BUR84]. These algorithms begin with an arbitrary starting point and spread in all directions by seeking a defined color or shades of gray. The shade may differ by a certain degree of tolerance. Algorithms of this class work fine for uniform areas and if the objects differ suf-

ficiently from the background but they meet their limitations in textured objects.

A third class of algorithms trace the contour of an object along an edge that separate the object from the background [JAI89]. These algorithms work very well on objects with uninterrupted sharp edges. If the edge fades the algorithm usually extrudes into areas that don't belong to the object. Mortensen [MOR95] has adressed these kinds of problems in his *Live Wire* approach. They formulate a dynamic program which leads to globally optimal boundaries in conjunction with a cost function. In contrast to SemiSeg, the cost function is based on edges and gradients. However they also need user interaction whenever the detected boundary extrudes into the background.

Active contours have recently proven to be another promising approach. The user defines a rough outline of the object. The outline can be interpreted as a rubber band. It has inner forces which contract the boundary. On the other hand, the edges of an object act as outer forces that prevent the active contour from vanishing into a single point. While contracting, the contour seeks a position of minimum energy which hopefully converges to the outline of the object [BLA98]. This approach is not very interactive by its nature since the user does not know in advance the shape to which the contour will shrink. While it does not work well if objects are embedded into complicated environments with strong gradients, it is a useful tool e.g., in medical imaging.

3 Wavelet-Based Segmentation

3.1 Outline of the algorithm

Our algorithm tries to follow a path that separates the object from the background. Even though we will sometimes refer to an 'edge', the boundary that separates object and background can be an arbitrary transition (like the smooth borderline between a white cloud and the blue sky) and need not be a sharp edge.
Our algorithm works as follows: We try to find rectangles that lie on a user-defined boundary. We use a multi-scale analysis to compare high and low frequencies of the user-defined sample and neighboring rectangular image candidates. A measure is introduced that depicts how much these frequencies differ. At the same time, we estimate how significant a specific frequency scale is for what the user defines as a continuous structure (e.g., an edge) and then weigh it with that estimate.
For example, our algorithm might find out that the coefficients of a high frequency scale are not very characteristic for the user-defined object boundary but that all the lower frequencies characterize it very well. The algorithm will then focus on finding those lower frequencies in the neighborhood.

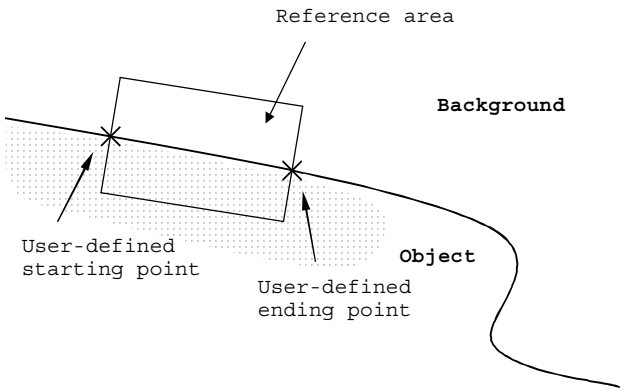

Figure 1: User-selected sample of an edge that separates the object from the background

3.2 Semi-automatic Segmentation

Step 1: The user initiates the segmentation process by selecting a starting point and an ending point. Between those two points we define a *reference rectangle* that contains a sample of the boundary. It exemplifies what the border between object and background looks like and will be used by the algorithm to find a segmentation path through the image.

Step 2: The sample rectangle is rotated until it is parallel to the axes of a 2D Euclidean coordinate system so that the line between starting point and end point is aligned with the x-axis (see Figure 2).

Unlike with other segmentation algorithms, the sample picture does not have to include a sharp edge. It could as well contain a smooth transition between two shades that occur in all kinds of fuzzy objects (e.g., clouds). In fact, the sample does not have to satisfy any constraints other than that it should be constant along the x-axis.

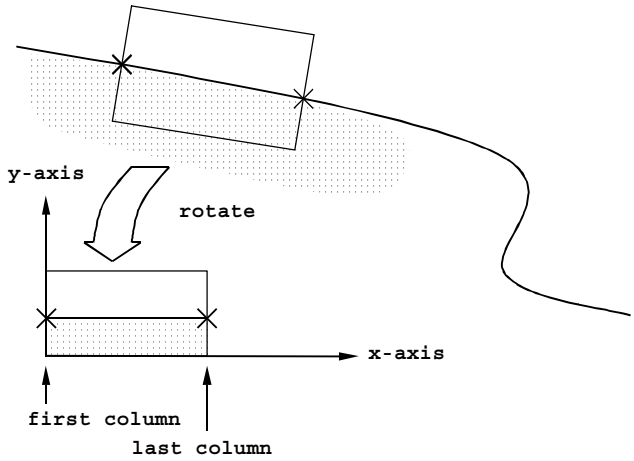

Figure 2: The reference rectangle is rotated so that it is parallel to the axes of the coordinate-system

Looking at the rotated version of the reference area in Figure 2, the only constraint is that its columns of pixels should not differ too much. Let us assume that the object is in the lower part of the reference rectangle and the background in the upper part. The same is true for every column of pixels in the rectangle. Only if there is a sufficient 'correspondence' within neighboring columns a pattern can be derived that is typical for the reference area and that might be found in other parts of the picture. The term 'correspondence' will be defined more precisely later.

Step 3: The algorithm seeks another area that starts with the ending point of the user -defined reference rectangle and has a similar appearance (see figure 3). At first only the starting-point of the new area is known. It is used as the center of a subsequent rotation of the area. The aim is to find an angle in which it is best aligned with the object.

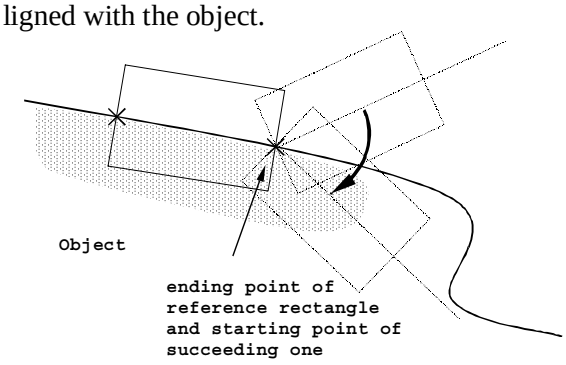

Figure 3: The succeeding rectangle of the reference rectangle is rotated until it is best aligned with the object.

For each angle, the picture within the new rectangle is rotated as described in step 2 until it is aligned with the coordinate system. In a naive approach, we could try to find the best match between both the reference rectangle and the new rectangle on a pixel basis by simply summing up the unsigned differences of the gray values.

We will now introduce a more sophisticated similarity measure that is based on frequencies rather than on gray values. Before the algorithm can find its path through the image, the axis-aligned reference area is analyzed by means of a multi-resolution wavelet analysis in the following way:

Step 4: A column of pixels is convoluted with a scaled wavelet so that the oscillations on a specific scale are derived. This is repeated for all n-pixel scales (n = 2, ..., y/2) where y is the number of pixels of the column. The procedure stops after the lowest frequency has been derived.

Hint 1: *The coefficients depict the amount of oscillation that can be found somewhere in a given signal. Obviously the fastest oscillation (resp. the highest frequency) can be found on a two-pixel scale and the lowest frequency can be found within the first and the second half of the column.*

Hint 2: *The energy of the image is stored in the coefficients of the different scales. Each coefficient describes the affinity between the image and the wavelet in a particular area of a column.*

Figure 4 shows a plot of the coefficients on different scales. This frequency analysis is carried out for each column of the reference area.

Step 4: Afterwards the average coefficients $\bar{k}_{i,s}$ are calculated by summing up the equally weighted coefficients $k_{i,s}$ of each column r.

$$\bar{k}_{i,s} = \frac{1}{R}\sum_{r=1}^{R} k_{i,s}^{r} \qquad (1)$$

R : Number of columns within th e current rectangle

where the index i denotes the number of a wavelet coefficient on scale s.

Hint: *Since the coefficients in the frequency domain map linearly to the gray values in the spatial domain, the same result would be obtained by averaging the pixels of each column and then analyzing that column.*

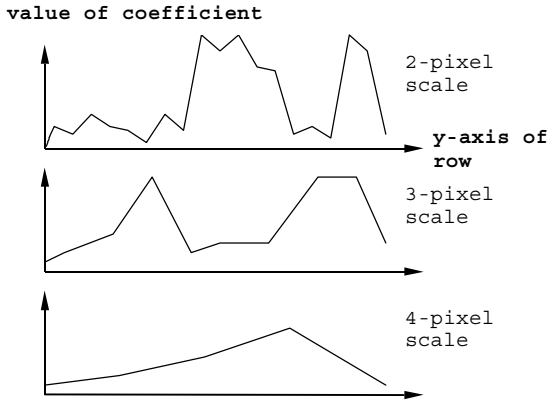

Figure 4: Plot of wavelet coefficients on

Later the algorithm will use the average coefficients $k_{i,s}$ to produce a measure of the affinity of two rectangular areas.

Step 5: It is useful to evaluate the significance of the coefficients of a specific scale for the area the user has defined. Consider an imagine with highly frequent noise. On a two-pixel scale the coefficients would be almost random, whereas the coefficients of larger scales might not be influenced that much by the noise.
For the reference area the user will use a sample that has a continuous structure however perhaps that continuity is only implicitly defined in the image. If a high variance is found on a certain scale, we can conclude that the continuity cannot be found within the frequencies (resp. the coefficients) of that scale. A low variance var_s on the coefficients of a scale s means that they might depict the continuous similarity the user is looking for. So the need of the variance is not only to reduce the influence of noise but also to weigh the frequencies that are characteristic for the structure of the reference area.

$$\mathrm{var}_s = \frac{1}{I_s}\sum_{i=1}^{I_s}\frac{1}{R}\sum_{r=1}^{R}\left(k_{i,s}^{r} - \bar{k}_{i,s}\right)^2 \qquad (2)$$

I_S : Number of coefficients in scale s (with index i)

$k_{i,s}^{r}$: coefficient number i on scale s of column $r \in \{1,\ldots,R\}$

As pointed out above the difference in the coefficients of two areas is weighted by the inverse of the variance of its scale. That causes a low influence of a scale with a high variance and a high influence of a mostly continuous scale.

$$m = \sum_{s=1}^{S}\left(\frac{1}{var_s I_s}\sum_{i=1}^{I_s}\left|\bar{k}_{i,s}^{1} - \bar{k}_{i,s}^{2}\right|\right) \qquad (3)$$

m : difference of two image areas (rectangles)

S : number of scales (with index s)

I_S : number of coefficients in scale s (with index i)

$\bar{k}_{i,s}^{p}$: average coefficient i on scale s of picture $p \in \{1,2\}$

The outer sum of (3) iterates over the scales. Usually S is not very large. For example if the reference rectangle is 12 pixels high, there is a 2-pixel scale with 12/2 = 6 coefficients, a 3-pixel scale with 12/3 = 4 coefficients, and so on. The number of coefficients is denoted by I_s for the upper bound of the inner sum. In the inner sum, differences of the average coefficient vector for two areas are calculated in turn for each scale. By dividing the inner sum by I_s, we prevent the small scales from becoming more important than larger scales with fewer coefficients. At the same time, a high variance leads to a smaller influence in the total difference m.

Our difference measure m is proportional to the similarity of two rectangular image areas. The similarity is based on frequencies, whereas a specific frequency is weighted by its estimated importance for a user-defined structure.
As shown in Figure 3, the current rectangle is rotated around the end point of the preceding one. In each step of the discrete rotation the value m can be calculated that indicates how different the images in both rectangles are. The rectangle with the smallest m is the one we choose for the continuation of our segmentation path.

4 Implementation

The implementation is written in C++ for 32-bit Windows environments. Please have a look at our segmentation homepage [HOME]. There you will also find some examples and a version of the program for Windows 9x/NT. After starting the program a 24-bit bitmap file can be opened. It appears as a color image on the right side of the screen and as grayscale image on the left side. By pressing the left mouse button, the starting-point of the reference rectangle can be defined. Releasing the button defines the ending-point. The program will try to follow the user-defined structure for a number of pixels. The path is indicated by a red line. In the other half of the window the part of the image that is enclosed by the red line is cut out of the background in realtime while the algorithm finds its segmentation path through the image. In addition to the automatic approach just described, a user-defined red line can be defined using the right mouse button. By moving the mouse while pressing the shift key on the keyboard parts of the segmentation path can be deleted manually.

5 Experimental Results

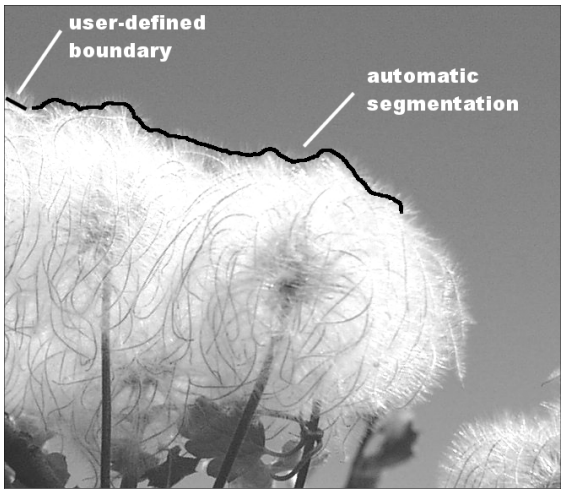

Figure 5: Segmentation of a fuzzy object

Figure 5 shows the result of the segmentation process. The small black line segment at the very left is the user-defined piece of boundary. The rest of the line was derived by the algorithm without further intervention.

We evaluated our algorithm against other semi-automatic segmentation methods. Four different images were segmented. Initially we had planed to use the segmentation methods mentioned in section 2 (related work) for comparison. Unfortunately we were unable to obtain an implementation of the live-wire method. We thus chose the following methods:


Figure 6: Screenshot from the implementation. On the left side the black segmentation path is shown. The right side shows the image with the background removed.

Image	Clicks	P1	P2	P3	P4	P5	P6	P7	Sum*
Sea1	28	4	6	3	4	6	5	6	25
Sea2	40	2	3	2	3	3	1	5	13
Sea3	83	8	9	8	9	7	9	10	43
Sea4	21	8	8	7	10	9	8	9	34
Africa1	102	8	5	6	7	7	5	8	33
Africa2	57	2	3	2	3	2	1	4	12
Africa3	216	9	7	6	8	9	7	8	39
Africa4	59	3	4	5	5	3	5	6	22
Fashion1	55	4	5	5	8	5	6	8	29
Fashion2	2	6	7	7	8	8	8	8	38
Fashion3	110	5	6	7	8	7	7	7	34
Fashion4	2	9	9	9	9	10	10	9	46
Noise1	60	6	6	5	8	5	7	8	32
Noise2	4	1	2	2	3	1	1	3	9
Noise3	150	8	7	6	8	7	7	9	37
Noise4	2	10	8	8	9	9	10	10	46

Table 1: Human perception of segmentation quality

Method 1 (edge guided line trace): The user defines a closed polygon that fully encloses the object. Since the polygon consists of concatentated lines the algorithm processes each line in turn. It must begin with the starting-point of a line of the polygon and arrive at the ending-

point of the same line. However, in between those two points the algorithm follows the strongest edge that can be found in a defined neighborhood around the line.

Method 2 (region growing): The user defines a starting point. The algorithm fills in an area around the starting point whose color lies within a defined tolerance. The filled in area is a part of the object. Usually the user has to define some more points until the whole shape of an object is filled in (resp. selected by that area). In some cases it is easier to fill in the background and take the inverse of the selected area in order to get the object.

Method 3 (fully user guided segmentation): As in method 1 the user outlines the object line-by-line until he gets a polygon that encloses the object. No further computation is done. The polygon already is the segmentation path.

Method 4: SemiSeg is used.

Figure 7: Test-images with no segmentation applied to it

All four methods were applied to the images *sea*, *africa*, *fashion* and *noise* (the results of the segmentation can be seen on our homepage [HOME] – the image names consist of the basic image as shown in figure 7 followed by the number of the segmentation method used). Earlier work in the field of semi-automatic segmentation measured the average time a user needs to carry out the segmentation [MOR95]. We believe that the time is too dependent upon the user's familiarity with a specific tool. So we counted the number of necessary user interactions for segmentation. In our evaluation the number of mouse clicks proved to be an acceptable measure of the amount of user interaction. Furthermore the user had unlimited time to become familar with each of the segmentation tools so that an optimal number of clicks could be achieved for each segmentation method.

After the four images were segmented according to each method in turn, seven trial persons, P1-P7, had to evaluate the quality of the segmentation results with marks ranging from 1-10 (1 poor, 10 very good), as depicted in table 1.
For each row the lowest and the highest marks were removed. The rest are summed up in column *Sum**. The column *clicks* shows the number of user clicks that were needed.
Afterwards, the accumulated marks *sum** were added for each method. The result can be interpreted as a measure of the quality of each method (see table 2). In the end, each of these marks is divided by the number of user interactions since greater user effort delivers better results. The resulting quotient depicts the quality of the segmentation method and is normalized with regard to the user effort.

Method	sum (marks)	# user clicks	Sum/#clicks
Method 1	119	245	0.486
Method 2	72	103	0.699
Method 3	153	559	0.274
SemiSeg	148	84	1.762

Table 2: Summary of perception quality and user effort

In some cases such as in *sea2*, *africa2* and *noise2*, the segmentation process was aborted at the point where the improvement of image quality was not justified by further user interaction. In other words, the user interaction was only continued to the point where a significant improvement in image quality could be expected. Column *sum (marks)* in table 2 shows that the fully user-guided segmentation method 3 delivers the best image quality. However, at the same time, it is the most time-consuming. The area-growing algorithm can lead to superb

results in some cases (like in fashion 2) but can totally fail in other cases like *africa2* and *noise2*.
In most cases the results produced by SemiSeg are among the best. It totally outperforms the others with regard to the total quality measure in the last column of table 2.

7 Conclusion

In this paper we have introduced a new method for tracing an arbitrary object boundary. The user defines a sample of a piece of that boundary and the algorithm tries to find the same transition properties in a continuous path. This is achieved by stringing rectangles with a similar content onto one another. A measure of the difference between two rectangular image areas is derived from weighted wavelet coefficients.

8 Outlook

The implementation currently uses the Haar wavelet. Filterbanks of other wavelets might be more appropriate.
The rotation of the pixels in the reference area introduces aliasing effects if no filtering is applied to the result. This especially affects the 2-pixel scale.

For now the implementation uses the same reference area for comparison all the time. In practice the appearance of an object can chance abruptly or continuously. In this case it will become more difficult to find the border of an object, and the path will sometimes intrude out into the background. In an adaptive version of the implementation the reference rectangle could be continuously adapted to the new border in the following way:

$$k_{i,s}^{l} = (1-\alpha)k_{i,s}^{l-1} + \alpha k_{i,s}^{new} \qquad (4)$$

where
$\alpha \in [0;1]$

$k_{i,s}^{l}$ is coefficient number i on scale l. It is dependent with the weight $(1-\alpha)$ on the coefficient (l-1) but it is also influenced by the weight α by the coefficient of the last best-matching rectangle.

Finally, the implementation only uses gray-scale images. In color images the gray-scale version corresponds to the intensity map of the image. Other dimensions of the color space like hue or saturation can also be interpreted as a gray-scale image and be analyzed in the same way as described above. The analysis of the intensity, hue and saturation maps of an image would result in three independent suggestions as to the direction in which an object boundary could be followed. Eventually this would lead to a more stable prediction for a path through the image.

References

[BLA98] A. Blake, M. Isard, **Active contours**, Springer Verlag, Berlin, Heidelberg, 1998

[BP] W. Boehme, H. Prautzsch, **Geometric Concepts for Geometric Design**, A K Peters, 1994

[BUR84] P. J. Burt, **The Pyramid as a Structure for Efficient Computation**, Springer Verlag, New Y-ork, 1984

[FVF90] J. Foley, A. vanDam, S. Feiner, J. Hughes, **Computer Graphics**, Second Edition, Addison-Wesley, 1990

[GS97] M. M. Gutzmann, R. Scholz, Manuskript - Wavelets: **Grundlagen und Anwendung in der Bildkompression**, Friedrich-Schiller-Universität Jena, Lehrstuhl für Rechnerarchitektur und –kommunikation, 1997
http://www2.informatik.uni-jena.de/pvs/gutzmann/

[HOME] www.informatik.uni-mannheim.de/informatik/pi4/projects/L3/SemiSeg/
(no whitespaces – as it is printed)

[JAE97] Bernd Jähne, **Digitale Bildverarbeitung**, 4. Auflage, Springer-Verlag Heidelberg 1997, page 482ff

[JAI98] A. K. Jain, **Fundamentals of digital image processing**, Prenti-

ce-Hall, Englewood Cliffs, NJ, 1989

[MA82] D. Marr, **Vision**, W.H. Freeman and Co., San Fransisco 1982

[MOR95] E. N. Mortensen, W. A. Barret, **Intelligent Scissors for Image Composition**, ACM Proceedings, Computer Graphics (SIGGRAPH 1995), Los Angeles, CA, August 1995, pp. 191-198

[ROS82] A. Rosenfeld and A. C. Kak. **Digital picture processing**, 2nd ed., Volume I and II. Academic Press, Orlando 1982

[RT71] A. Rosenfeld and M. Thurston, **Edge and curve detection for visual scene analysis**, IEEE Trans. Comput., C-29, 1971