Image Segmentation Via Iterative Geodesic Averaging

Asmaa Hosni, Michael Bleyer and Margrit Gelautz
Institute for Software Technology and Interactive Systems, Vienna University of Technology
Favoritenstr. 9-11/188/2, A-1040 Vienna, Austria
[asmaa, bleyer, gelautz]@ims.tuwien.ac.at

Abstract

We present a simple and fast method for performing unsupervised segmentation. Our method works by centering a square window on each pixel of the input image. Each pixel is then assigned to a new color which is computed by averaging the pixel colors inside the window. The idea is that if this averaging operation is repeated a few times then we should obtain an image in which pixels of the same color surface are assigned to the same (or at least to very similar) color values. Consequently, the desired color segments are formed by groups of spatially neighboring pixels that share the same color in the convolved image.

Obviously, our method would deliver poor performance if the averaging operation is applied in a naive manner, as pixel colors of different segments would be mixed. To overcome this problem, we propose to compute a geodesic weight mask that regulates a pixel's influence in the averaging operation. A pixel's weight in the window is determined by computing the geodesic distance to the center pixel. In other words, we enforce that a pixel obtains high influence only if there exists a path to the center pixel along which the color does not change significantly (connectivity). The proposed method is evaluated on some widely used test images. Our method seems to produce accurate segmentation results and to capture object outlines correctly. We show by quantitative evaluation that our segmentation algorithm outperforms two competing segmentation methods.

1. Introduction

Unsupervised image segmentation is one of the most important tasks in low-level vision. It often represents an essential step for image analysis and image understanding. In this paper, we concentrate on color segmentation. Therefore, our goal is to divide an image into regions of homogeneous colors.

The contribution of this paper lies in a simple and effective method for performing the color segmentation task. The building block of our algorithm is formed by the geodesic distance transform. We center a window on each pixel of the input image. Geodesic distances are then computed inside the window to determine a weight mask. The derived masks serve to compute a weighted color average for each pixel of the image. The idea is to iterate this averaging procedure. After some iterations it can be observed that pixels of a homogenous region show the same color value. Therefore, we build our segments by grouping spatially neighboring pixels of the same color in the iteratively averaged image.

Although in a quite different form, the geodesic distance transform has already been applied in image segmentation. In [7], geodesic distances are employed for integrating spatial proximity information into a graph-based segmentation algorithm. Other papers [1, 6] use the geodesic distance for performing foreground/background segmentation. Here, the geodesic distance is computed between user-defined foreground and background regions and not in individual windows as in our work.

In the context of previous work, our method is also related to approaches that use the geodesic distance transform in other computer vision research areas. For example, geodesic windows have been applied in local image denoising approaches [2, 8, 9]. These local approaches assume that a small neighborhood around a pixel contains sufficient information to remove noise from the pixel's original color. The original color is therefore reconstructed by using a function that operates on the colors of all pixels inside the window.

We also consider our segmentation algorithm related to the work of [11]. Here, a bilateral filter is iteratively applied to generate cartoon-like images. Apart from cartoonization being a different application, we believe that our geodesic filter is better-suited for generating regions of piecewise constant colors, since we implement the concept of *connectivity*.

2. Algorithm

Figure 1 presents an overview of our algorithm. The input for our algorithm is formed by a color image (Figure

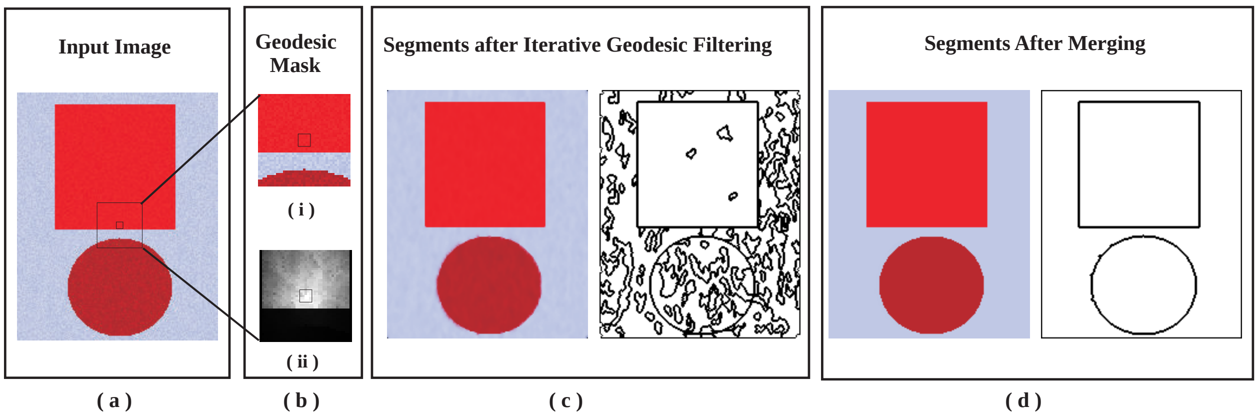


Figure 1. Outline of our algorithm. (a) Input image. (b) Geodesic mask computation: (i) Zoomed-in view. (ii) Computed geodesic mask. (c) Segments after iterative averaging. (d) Segments after segment merging.

1(a)). We center a window on each pixel of the input image. The colors of pixels inside the window are averaged. Here, it is important to assign different influence to the pixels of the window, since we want to avoid mixing colors of different color surfaces. Ideally, pixels that lie on the same color surface as the center pixel obtain high influence, while the influence of pixels outside the color surface is low. We implement this concept by computing the geodesic distance from the center pixel to all other pixels of the window. The effect is that a pixel can only have high influence if there is a path to the center pixel along which the color remains nearly constant. In other words, the pixel needs to be connected to its center pixel.

Figure 1(b) shows a geodesic mask that we have computed for the marked pixel of Figure 1(a). It is important to notice that high influence is only assigned to pixels of the rectangular structure on which also the center pixel resides. This is because only these points are connected to the center pixel. In the image, there is also a circular structure that belongs to a different object. Although pixels of the circular structure show similar color in comparison to the center pixel, they are not connected. Consequently, their influence is low. We can therefore avoid mixing the colors of these different objects in the subsequent averaging procedure. This is an advantage over using other filtering methods such as the bilateral filtering [10] where the concept of connectivity is not implemented.

Once the geodesic masks are determined, we compute a new color for each pixel of the image. A pixel's new color is thereby obtained by computing the weighted average of color values inside the window using the weights of the corresponding geodesic mask. This averaging procedure is repeated a few times. After some iterations the averaged image consists of regions of piecewise constant colors (Figure 1(c)). This result can already be interpreted as the final segmentation output. Alternatively, we provide a simple segment merging procedure to reduce the number of segments (Figure 1(d)). We go into detail on the individual steps in the following sections.

2.1. Geodesic Masks

The basic idea in the proposed segmentation method is to compute the support weights within a square window via the use of geodesic distances. This is explained as follows.

The geodesic distance $D(p,c)$ between a pixel p of the support window and the window's center c is defined as the shortest path that connects p with c in the color volume:

$$D(p,c) = \min_{P \in \mathcal{P}_{p,c}} d(P). \tag{1}$$

Here, $\mathcal{P}_{p,c}$ denotes the set of all paths between p and c. A path P is defined as a sequence of spatially neighbouring points in 8-connectivity $\{p_1, p_2, \cdots, p_n\}$. The costs $d()$ of a path are computed by

$$d(P) = \sum_{i=2}^{i=n} d_C(p_i, p_{i-1}) \tag{2}$$

with $d_c()$ being a function that determines the color difference. This function is implemented by

$$d_C(p,q) = \sqrt{\sum_{i=1}^{i=3} (f_i(p) - f_i(q))^2} \tag{3}$$

with $f_i(p)$ denoting the value of the i^{th} color channel at pixel p. In our implementation, we represent color using the RGB system.

Intuitively spoken, the geodesic distance from pixel p to the window center c is low, if there exists a path between these points along which the color varies only slightly. We refer to this property as connectivity. Low geodesic distances should therefore convert into high support weights.

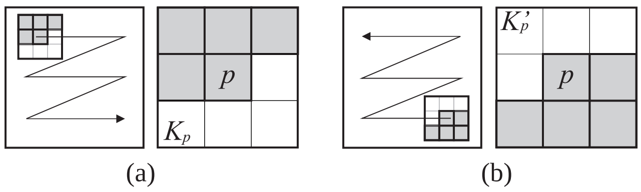


Figure 2. Approximation of the geodesic distances inside a window. (a) Forward pass. (b) Backward pass.

The function $w()$ implements this conversion by

$$w(p,c) = \exp\left(-\frac{D(p,c)}{\gamma}\right) \tag{4}$$

with γ being a user-defined parameter that defines the strength of the resulting segmentation. Small values of γ thereby lead to a large number of segments, whereas large values lead to a smaller number of segments.

We apply the method of [3] for efficiently approximating the geodesic distances of each pixel within the window to the center pixel (equation (1)). This method is reviewed as follows.

Each pixel p inside the window is assigned to costs $C(p)$. Initially, the costs of the center pixel are set to 0, while the costs of all other pixels are set to a large constant value. In the forward pass of the algorithm, we traverse the support window in a row major order. The costs of a pixel p are thereby updated by

$$C(p) := \min_{q \in K_p} C(q) + d_C(p,q) \tag{5}$$

with the kernel K_p being a set of pixels consisting of p itself as well as its left, left upper, upper and right upper neighbours (Figure 2(a)). The cost update is thereby performed immediately so that the new costs already affect the cost computation of the next pixel. Once the forward pass is completed, we invoke the backward pass. This pass traverses the window in reverse direction (see Figure 2(b)). It thereby updates the costs using equation (5) in conjunction with the kernel K_p' of Figure 2(b). Forward and backward passes are iterated. (In our experiments, we found three iterations to be sufficient for giving reasonable results.) The final costs $C(p)$ represent our estimate of the geodesic distance of p to the center pixel.

Figure 3 shows some examples of computed geodesic masks at some specified pixels in the Peppers and House images.

2.2. Iterative Color Averaging

After computing the geodesic weight masks we compute a new color for each pixel. This is accomplished by calculating the weighted average for each color channel:

$$f_i'(c) = \frac{\sum_{p \in \mathcal{W}_c} w(p,c) \cdot f_i(p)}{\sum_{p \in \mathcal{W}_c} w(p,c)}. \tag{6}$$

Here, $\mathcal{W}_c$ represents the set of all pixels that form the window centered on pixel c.

Once we have determined the filtered image I', we can repeat this procedure. We therefore apply geodesic filtering on I' to obtain I'' and so on. In our experiments, we have used three iterations.

The result of the iterative filtering is an image of piecewise constant colors. We can interpret groups of neighboring pixels that share the same color as our final segments. Optionally, we can further try to reduce the number of segments in a segment merging step that is described in the following section.

2.3. Segment Merging

Our segment merging procedure is straightforward. Let us assume that we have two segments S and T. For merging S and T, they need to fulfill two criteria:

1. S and T are spatial neighbors, i.e. they have a common border in 4-connectivity.

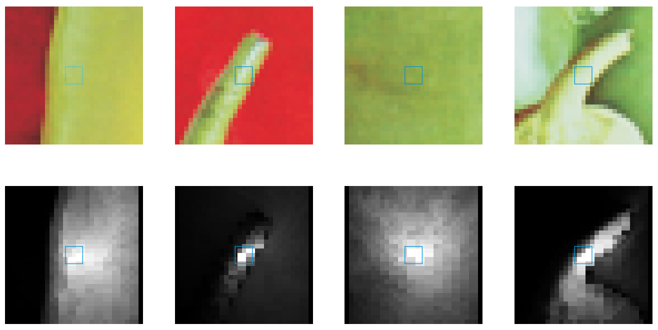
2. The difference in colors between S and T is smaller than a pre-specified threshold thr.

We record all pairs of segments that fulfill our criteria. In the segment update step, we enforce that two segments that form a pair are part of the same new segment. For example, we have the pairs $< S, T >$ and $< S, U >$ with U being another segment. Since we enforce that pairs of segments must lie in the same new segment, S, T and U must all lie in the same new segment after the segment update step. Once the new segments are known, we update each segment's color by computing the mean color over all pixels inside the segment. We then repeat our procedure until there are no segments left to merge.

Our algorithm also offers the possibility to define a threshold on the minimum region size $minR$. Segments whose number of pixels is smaller than $minR$ will be merged with the spatially neighboring segment of the most similar color.

3. Experimental Results

To evaluate the performance of our algorithm, we run our method using constant parameter settings. In our experiments, we set the parameters as follows: γ = 25 (segmentation strength), thr = 25 (color difference threshold in the segment merging step) and $minR$ = 20 (minimum region size). The size of the window used in the color averaging procedure is set to 9×9 pixels. We have estimated these parameters empirically in order to generate good-quality results. To be fair, we have also used the same parameter tuning procedure for the two competing segmentation algorithms that we use for performance comparison. These algorithms are described in the following.


(a)

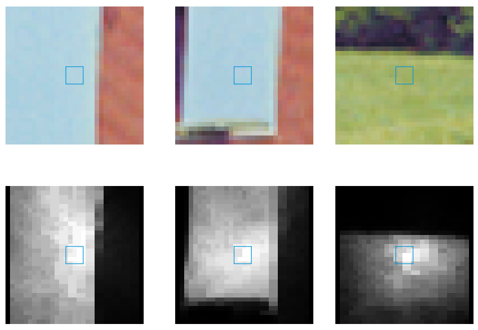

(b)

Figure 3. Examples of geodesic masks. (a) Masks computed for the ”Peppers” image. (b) Masks computed for the ”House” image.

As a first competitor, we apply the mean shift algorithm [5]. We have chosen this algorithm due to its popularity in the vision community. As a second competitor, we employ a method that is identical to our algorithm except that we use a different way for computing the weight masks used in the color averaging procedure. Instead of computing the weights in equation (4) via the geodesic distance transform, we apply a bilateral filter [10].

Our test set is composed of eight real-world images that are widely adopted in the image segmentation literature. These images are: House, Peppers, Lena, Baboon, Flowers, Sailboat, Pens and Yacht. The images have different color segmentation complexities. Figure 4 shows the results of our experiment. As can be seen from Figure 4(d), the eight test images are well segmented by our segmentation algorithm. Object edges are well preserved in all test images.

To obtain quantitative results, we use three evaluation metrics proposed in the literature. These metrics are: (1) the number of segmented regions for the same setting of the parameter $minR$, (2) Goodness and (3) computation time.

We compute the Goodness function F as defined in [4] by

$$F(I) = \sqrt{R} * \sum_{i=1}^{R} \left(\frac{e_i^2}{1 + \log(A_i)} + \left(\frac{R(A_i)}{A_i} \right)^2 \right). \quad (7)$$

Here, I is the image to be segmented and R is the number of regions in the segmented image. A_i denotes the area of the i^{th} region (number of pixels). The function $R(A_i)$ returns the number of segments having identical area as A_i. Finally, e_i is the difference in color between the i^{th} segment in the segmented and original images. For example, e can be derived by computing the absolute color difference between images of Figure 4(a) and the color images of Figure 4(b). In general, a good segmentation result will lead to small values of F.

In the context of computation time, our implementation can be divided into three components: (1) geodesic weights computation, (2) the pixel averaging operation and (3) the segment merging operation. We have implemented components (1) and (2) on the Graphics Processing Unit (GPU), whereas component (3) is implemented on the CPU. Note that a major advantage of our simple segmentation strategy is that its parallelization is very straightforward. In our implementation, each processor computes the geodesic weight mask and the averaging procedure for a single pixel. As a consequence, the computational performance of our algorithm is relatively high.

Table 1 shows the quantitative results for the images of Figure 4. It can be depicted that the performance of our algorithm seems to be superior in comparison to the mean shift algorithm and to the method using the bilateral filter. Our algorithm shows better Goodness, a smaller number of segments and a faster computation time. Table 1 also shows that for highly textured images with high color complexities (e.g. Baboon, Flowers, Sailboat and Pens), the performance of mean shift and the bilateral filter algorithm are relatively poor, since they tend to over-segment these images. In contrast to this, our algorithm performs relatively well on these images.

In the context of faster computation times, it is not surprising that our algorithm outperforms the mean shift method, since the mean shift algorithm does not use the graphics hardware. It would be interesting to test whether a comparable computational behavior can be achieved by implementing the mean shift algorithm on the GPU in future work. In order to avoid the labor-intensive task of porting the bilateral method to the GPU, we have also used a CPU version of this segmentation algorithm.

4. Conclusions

We have presented a method for color segmentation, which is based on the geodesic distance transform. Our algorithm is simple and can easily be parallelized, which enables a fast GPU-based implementation. Another advantage is that our method seems to correctly preserve object boundaries in the segmented image. In the results, we have shown that our algorithm can compete with the popular mean shift

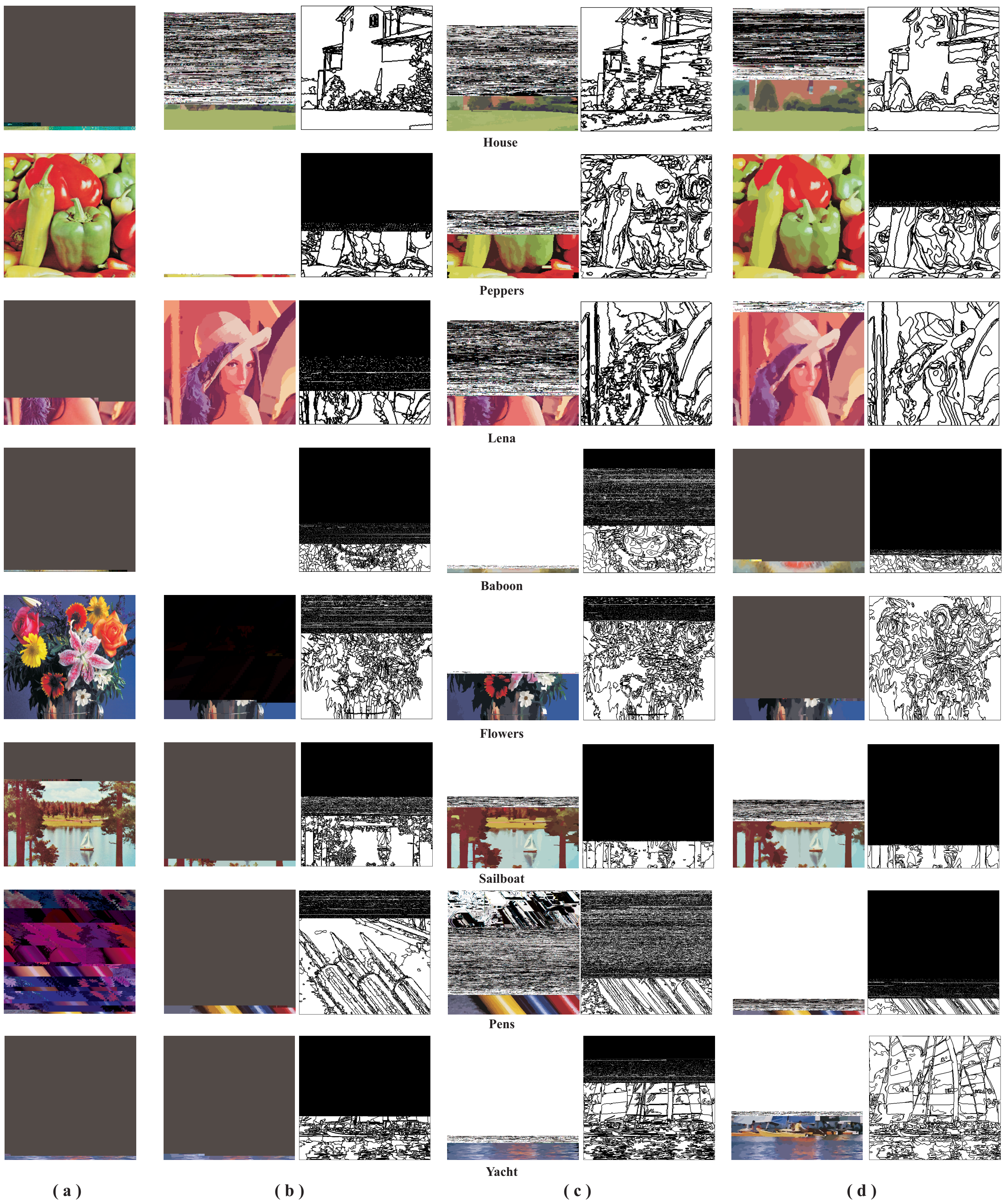


Figure 4. Segmentation results. We use two representations for displaying the segmentation results. In the left images of (b), (c) and (d), pixels of the same segment are given the same color value. In the right images, we plot the segment borders. (a) Input images. (b) Results of the mean shift algorithm [5]. (c) Results of the method using bilateral segmentation. (d) Results computed by the proposed algorithm.

Image	Size	Number of Regions			Goodness			Time(sec.)		
		Mean Shift	Bilateral Filter	**Our Algorithm**	Mean Shift	Bilateral Filter	**Our Algorithm**	Mean Shift	Bilateral Filter	**Our Algorithm**
House	256 x 256	483	215	**150**	2.93	3.51	**2.31**	1.1	2.703	**0.250**
Peppers	512 x 512	654	311	**279**	6.40	5.43	**3.95**	0.75	2.704	**0.249**
Lena	512 x 512	582	233	**166**	4.79	3.97	**2.43**	0.76	2.720	**0.249**
Baboon	512 x 512	4666	1084	**895**	411.2	21.31	**15.11**	4.57	11.532	**0.812**
Flowers	512 x 512	2319	933	**679**	60.78	12.62	**8.24**	2.25	7.485	**0.594**
Sailboat	512 x 512	4125	747	**623**	284.9	15.61	**7.75**	5.73	10.625	**0.827**
Pens	512 x 480	2034	848	**570**	63.91	8.28	**4.69**	4.28	9.953	**0.765**
Yacht	512 x 480	1861	1002	**844**	11.32	11.13	**7.83**	3.57	9.970	**0.766**