Hierarchical Blurring Mean-Shift

Milan Šurkala, Karel Mozdřeň, Radovan Fusek, and Eduard Sojka

Technical University of Ostrava,
Faculty of Electrical Engieneering and Informatics,
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{milan.surkala.st,karel.mozdren,radovan.fusek.st,eduard.sojka}@vsb.cz

Abstract. In recent years, various Mean-Shift methods were used for filtration and segmentation of images and other datasets. These methods achieve good segmentation results, but the computational speed is sometimes very low, especially for big images and some specific settings. In this paper, we propose an improved segmentation method that we call Hierarchical Blurring Mean-Shift. The method achieve significant reduction of computation time and minimal influence on segmentation quality. A comparison of our method with traditional Blurring Mean-Shift and Hierarchical Mean-Shift with respect to the quality of segmentation and computational time is demonstrated. Furthermore, we study the influence of parameter settings in various hierarchy depths on computational time and number of segments. Finally, the results promising reliable and fast image segmentation are presented.

Keywords: mean-shift, segmentation, filtration, hierarchy, blurring.

1 Introduction

Mean-Shift (MS) is a clustering algorithm that appeared in 1975 [6] and is used for data filtration and segmentation up to now. Mean-Shift seeks for the position with the highest density of data points. These groups of points are called clusters (segments). Mean-shift is not only used as a statistical tool, but also as a method for image segmentation, object tracking [5], etc.

Recently, a lot of new Mean-Shift variations have been developed. These methods use various approaches to improve the original idea. In 1995 MS was revised by Cheng [2] and in 1999 and 2002 by Comaniciu [4,3]. One of the most interesting variations of MS is Blurring Mean-Shift (BMS) that appeared in 2006 [1]. It has been proved that BMS has a lower number of iterations per data point and, therefore, a higher speed. As for stopping conditions, it is more difficult to apply the Gaussian kernel because it is not truncated. However, using the truncated kernels as Epanechnikov makes it possible to ignore data that are too far away from its center. Even with such advantages, BMS is still slow and not appropriate for general use.

In 2009, P. Vaturri and W.-K. Wong presented Hierarchical Mean-Shift [8] based on an idea of effective application of multiple Mean-Shift filtrations with variable kernel sizes. Computational time of the Mean-Shift method is strongly

J. Blanc-Talon et al. (Eds.): ACIVS 2011, LNCS 6915, pp. 228–238, 2011.

dependent on the number of input data points and on the size of the Mean-Shift window (kernel size). The first step is to use MS with a small kernel window to get a lower number of small segments in a short time. These segments are considered as data points with the weight that is equal to the number of data points they represent. In the next hierarchical step (stage) the number of data points is reduced due to preprocessing (previous stage). This allows us to use a greater kernel without the unwanted increase of computational time. The number of hierarchical steps is not limited. In each step, a new level of hierarchy with greater segments is created. The paper is organised as follows. In Section 2, we introduce the reader to the original Mean-Shift method. Section 3 is devoted to the Blurred Mean-Shift version. A new approach (algorithm) is presented in Section 4. The experiments are presented in Section 5.

2 Mean-Shift

Consider $X = \{x_n\}_{n=1}^{N} \subset R^d$ as a dataset of N points in the d-dimensional space. Although the characteristics of these points are not limited, we focus on the images and, therefore, these points are the set of pixels in processed image. We define the *kernel density estimator* with a kernel $K(x)$ as

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K\left(\left\| \frac{x - x_n}{\sigma} \right\|^2 \right), \tag{1}$$

where x is a processed data point (pixel), x_n is a data point in neighbourhood of point x and σ is a bandwidth limiting the size of neighbourhood (radius). We can distinguish between two types of the radii in images. The first one, denoted by σ_s, is a radius in the spatial domain (for example, a circle with the radius of 20 pixels). The spatial radius is mostly identical for x and y-axis, so we need only one σ_s. The second radius, denoted by σ_r, is a radius in the range domain (colour or intensity). If the grey-scale images are processed, only one σ_r is used, whereas, in the colour images, three different quantities σ_r may be required, each for one colour channel. In most cases, the same radius is chosen for all three channels, so σ_r is simplified to one value. We can choose from many types of kernels $K(x)$. The simplest is the *uniform flat kernel* that takes the value of 1 (full contribution of pixels) for the whole area of searching window (area of kernel). This kernel is defined as follows

$$K(x) = \begin{cases} 1, & if \ \|x\| \leq 1 \\ 0, & if \ \|x\| > 1 \end{cases}. \tag{2}$$

The second most common kernel is the *Epanechnikov* kernel. It minimizes the asymptotic mean integrated square error, therefore, it is an optimal kernel. The Epanechnikov kernel is defined by the following equation

$$K(x) = \begin{cases} 1, & if \left\|1 - x^2\right\| \leq 1 \\ 0, & if \ \|x\| > 1 \end{cases}. \tag{3}$$

The *Gaussian* kernel is very popular, nevertheless, it has some disadvantages. σ_s does not limit the size of searching window but it modifies only the shape of Gaussian curve. The computation is carried out for all data points (pixels in the image). Therefore, the use of the Gaussian kernel is very computationally demanding and the only way to achieve a better speed is to truncate the kernel. Fortunately, because of small contribution of distant pixels, the truncation has only a small influence on the quality of segmentation. The Gaussian kernel is represented by the equation

$$K(x) = e^{-\frac{1}{2}x^2}. \tag{4}$$

The density gradient estimator, denoted by $k(x)$, is defined as follows

$$k(x) = -K'(x). \tag{5}$$

Therefore, we can rewrite the Epanechnikov kernel for mean-shift computation as

$$K(x) = \begin{cases} 1 - x, & if\ 0 < x < 1 \\ 0, & otherwise \end{cases}. \tag{6}$$

The mean-shift vector represents the change of position for each data point (pixel in the image). Data used for computation remain unchanged in all iterations of mean-shift. The goal is to find an attractor. Attractor is a point to which all "similar" data points from a searching window converge. At such a point, the density of input data points takes its local maximum. All pixels that converge to one attractor form a cluster (segment). The mean-shift vector is represented by the equation

$$m_{\sigma,k}(x) = \frac{\sum_{i-1}^{N} x_i k\left(\left\|\frac{x-x_i}{\sigma}\right\|^2\right)}{\sum_{i-1}^{N} k\left(\left\|\frac{x-x_i}{\sigma}\right\|^2\right)} - x. \tag{7}$$

The first term on the left side is a new position of x computed by all adequate data in the searching window; the second term is the former position of x.

3 Blurring Mean-Shift

Blurring Mean-Shift (BMS) is an another method with a slightly changed equation leading to a significantly different method of computation. MS does not change the original dataset (it moves the data points, however, the computation is done with original data). As a result, it evolves slowly to the local density maxima. BMS changes the dataset in each iteration with computed values. As source data for each iteration, the data modified in the previous stage are used. It is proven that BMS has faster convergence in comparison to MS. The main advantage of BMS is that it reduces the number of iterations, despite that the number of input points remains the same. The equation of BMS can be written as

$$m_{\sigma,k}(x) = \frac{\sum_{i-1}^{N} x_i k\left(\left\|\frac{x_m-x_i}{\sigma}\right\|^2\right)}{\sum_{i-1}^{N} k\left(\left\|\frac{x_m-x_i}{\sigma}\right\|^2\right)} - x. \tag{8}$$

BMS has several more advantages. After each iteration, for example, there is a visible progress in filtration. If the pixels that should form one cluster are close enough, without any pixel that should not be in the cluster, all these pixels will be grouped in the next iteration. Original MS does not have this feature and it iterates slowly till all the data points reach the common convergence point. This often leads to long computational times. BMS has a tendency to form clusters whose size is similar to the size of searching window, therefore, the number of clusters can be approximately predicted. Nevertheless, the shape of these clusters follows the gradients in the image very well. Many different variants of mean-shift were proposed, such as Evolving Mean-Shift [9], Medoid-Shift [7] and variations of MS and BMS. In next section we are going to describe our MS variation.

4 Hierarchical Blurring Mean-Shift

We propose a new method called Hierarchical Blurring Mean-Shift (HBMS) that is based on one of modifications of Mean-Shift, called Hierarchical Mean-Shift (HMS). It uses the original MS method, which is applied hierarchically with various sizes of kernel. First of all, it uses a very small searching window (σ_s), which leads to creation of a large number of little segments representing just few data points. Because of the small searching window size, mean-shift is processed relatively quickly. Consider this as an input to the next stage of mean-shift running with a bigger searching window. Consequently, the segments become new data points.

From the first stage, we obtained a filtered dataset which has lower number of data points than the dataset originally contained. In the new dataset, each new data point has a weight that is equal to the number of original data points. Repeatedly running mean-shift on the dataset with larger searching window results in significantly reduced computational time. The main idea is to reduce the number of segments and make them bigger in each iteration. MS and BMS are slow because they use a large searching window from the start. Hierarchical mean-shift processes data with a smaller searching window in the first stage, which results in faster computation.

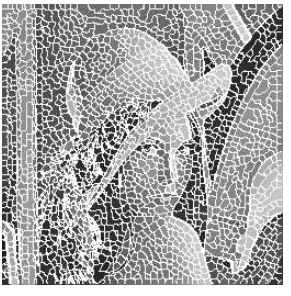
Although, HMS is faster than MS and BMS, it is still too slow for large input images. The main problem is that HMS uses the original MS as a basic algorithm. MS is not well suited for hierarchical processing since it is slower and produces a large number of segments. Therefore, each next iteration of HMS has still a large dataset. Our objective is to use more advanced mean-shift technique, which is more suitable for processing in hierarchical way.

Our method uses BMS for each step. This is much more effective and significantly faster, due to a lower number of iterations per pixel. Another advantage is that BMS forms more uniform clusters. They are also bigger and less numerous. Knowing that, we can say that BMS decreases the number of iterations per data point and, moreover, it greatly reduces the number of segments for each next stages of HBMS, so we can expect a high reduction of computational time. We will show this in Section 5. The following algorithm shows an example of implementation in a pseudo code.

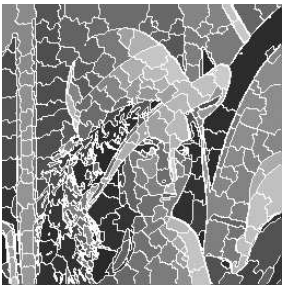
(a) The first stage filtration

(b) The second stage filtration

(c) The third stage filtration



(d) The first stage segmentation



(e) The second stage segmentation

(f) The third stage segmentation

Fig. 1. Example of filtration and segmentation evolving after each stage of HBMS

Algorithm 1. Hierarchical Blurring Mean Shift

$\sigma \leftarrow \{\sigma_1, ..., \sigma_j\}$, *where* $j = number\ of\ stages$
for $i \in \{1, ..., j\}$ **do**
  repeat
    for $m_i \in \{1, ..., N_i\}$ **do**
      $\forall n : p(n|x_{m_i}) \leftarrow \frac{k\left(\left\|\frac{x_{m_i} - x_n}{\sigma_i}\right\|^2\right)}{\sum_{n'=1}^{N_i} k\left(\left\|\frac{x_{m_i} - x_{n'}}{\sigma_i}\right\|^2\right)}$
      $y_{m_i} \leftarrow \sum_{n'=1}^{N} p(n|x_{m_i}) x_n$
    end for
    $\forall m_i : x_{m_i} \leftarrow y_{m_i}$
  until stop
end for

The problem is to find an accurate resizing of the searching window for each iteration. If we enlarge window too much, and if the number of segments from the previous stage is still high, the processing in the next stages will take too long. The greater reduction of data points is, the greater enlargement of searching window could be done. The size of initial window is discussed in Section 5.

5 Experiments

This section presents the experiments with HBMS algorithm. The first example is focused on the comparison between the computational speed of HBMS and HMS. We examine the idea that the HBMS approach is much faster than the hierarchical approach to MS (HMS) because BMS is faster than MS. We also study the effect of acceleration on image filtration quality. We also compare how the increasing amount of noise influences quality of image filtering.

As a source image, the famous Lena gray-scale photo in 256×256 pixels was used and for speed tests, original 512×512 version and resized versions were used too. To prove segmentation quality of our method we also used additional images. The algorithms were tested on a computer with Intel Core i5-M520, dual-core processor, with 2.4GHz core frequency, 4 GB of DDR3-1333 memory and a 64-bit operating system. All tests were run in the single core configuration. The first test examined the speed of HMS and our HBMS implementation using two and three-stage hierarchy. In the last run, the value of σ_s was always set to 40. The two-stage configuration started with $\sigma_{s1} = 5$, the three-stage configuration used $\sigma_{s1} = 3$ and $\sigma_{s2} = 9$. In all stages, σ_r was set to 24.

Table 1 has two columns for each tested resolution. The first column represents the values of the mean square error (MSE) between the final and the original image. MSE shows the intensity difference between the original and the filtered image. The smaller the error is, the better is the correspondence between the original and the filtered image. A good segmentation has small granularity (number of segments) and concurrently small intra-segment intensity variance (MSE error). The second column represents the time taken to complete the computation. MSE is computed as follows

$$MSE = \frac{1}{MN} \sum_{i=0}^{M} \sum_{j=0}^{N} (G_1(i,j) - G_2(i,j))^2, \tag{9}$$

where G_1 and G_2 are images to be compared, indexes i and j stand for pixel coordinates and M and N represents width and height of image.

The results were surprising. We achieved smaller MSE error especially for higher resolutions if BMS was used as a base for hierarchical segmentation.

Table 1. Comparison of speed and MSE depending on resolution

	64×64		128×128		256×256		512×512		1024×1024		1536×1536	
	MSE	t[s]	MSE	t[s]	MSE	t[s]	MSE	t[s]	MSE	t[s]	MSE	t[s]
MS	145	15	149	100	307	1516	252	12207	-	-	-	-
HMS_2	200	0,7	206	4,6	228	50	219	425	172	4310	-	-
HMS_3	317	0,45	263	1,7	196	10,5	162	60	127	313	-	-
BMS	209	4,5	212	35,5	161	138	152	1085	-	-	-	-
$HBMS_2$	232	0,4	197	1,4	203	6,2	162	33	111	156	91	358
$HBMS_3$	284	0,3	216	0,9	171	3,6	151	20	111	76	95	176

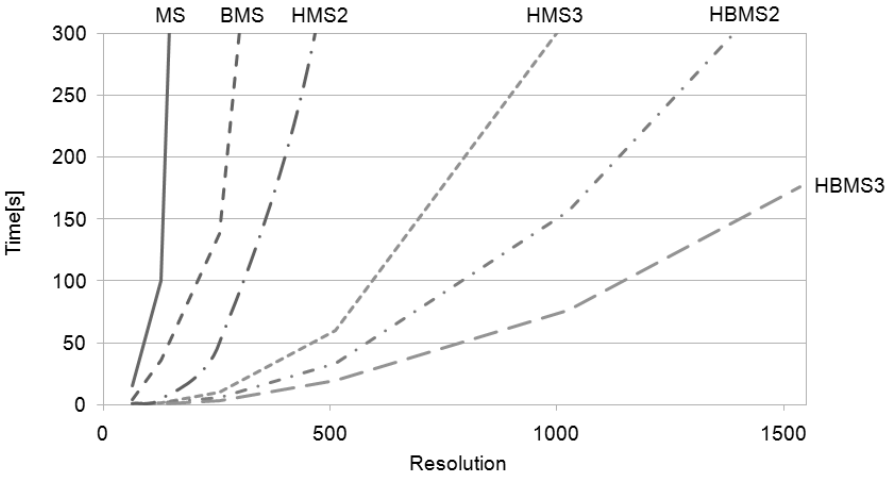
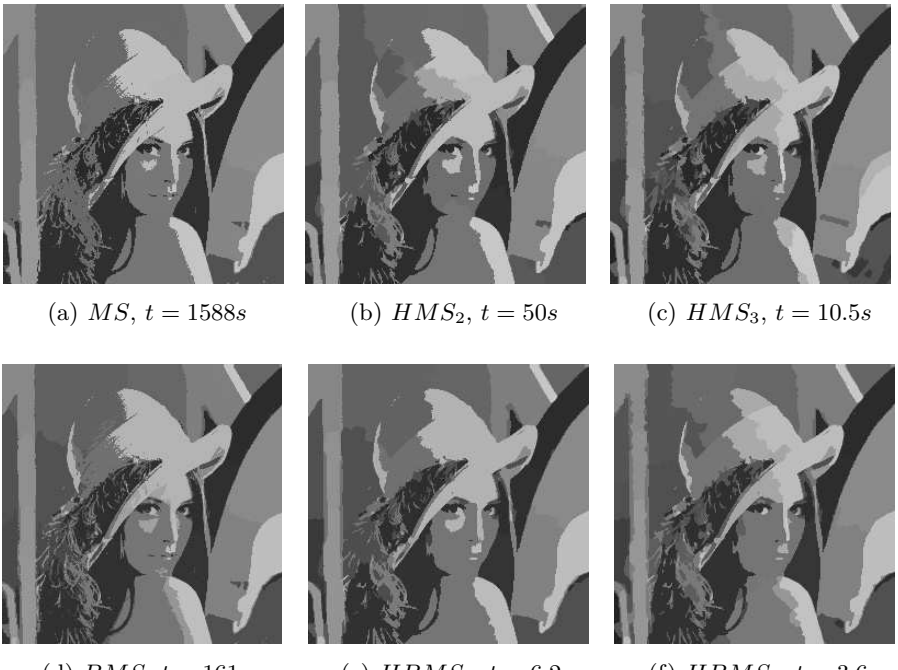


Fig. 2. Computational time with respect to image resolution



(a) MS, $t = 1588s$ (b) HMS_2, $t = 50s$ (c) HMS_3, $t = 10.5s$

(d) BMS, $t = 161s$ (e) $HBMS_2$, $t = 6.2s$ (f) $HBMS_3$, $t = 3.6s$

Fig. 3. Segmentation output from each method we compared

The authors of BMS [1] proved that this method has cubic convergence while the original MS has only linear convergence, which means that the use of BMS should lead to faster processing. Table 1 shows that the computation time increases approximately linearly with the size of the image. This means that an image with a 4-times bigger area is computed in an approximately 4-times longer time. From the table, it follows that the computation time increases approximately 8-times (or even more) if HMS is used. Therefore, the difference in performance is relatively small in the case of small images but it is significantly higher for larger images. For example, a 1 megapixel image is processed in 3 minutes with HBMS, whereas HMS needs more than one hour (2-stage configuration). We use our own implementation of HMS and HBMS.

It might have seemed that the hierarchical approach is helping more to original HMS than HBMS, however, the difference is not that significant. HMS benefits from a higher number of stages while HBMS is quite fast even with a lower number of stages. The speed-up is big enough even with two stages in HBMS, and it is not necessary to use three or more stages. The more stages used, the better filtration ability and computational speed we get. The drawback is that the granularity increases and the segmentation does not seem to be that good.

Using more stages causes creating larger segments with a higher intensity difference of the distant attractors. This increases the probability that the segment will not be included in the computation because the attractor does not fit in the searching window even though a large part of the segment is inside this window. It creates small segments on the borders of large segments, which would not happen if a larger searching window in the spacial domain was used. We can say that all of these methods achieved a good segmentation quality, but the main difference of using the HBMS is in significantly higher speed.

The next test examines the influence of noise on segmentation quality. Two-stage HMS and HBMS is used on the 256×256 image. The noise modifies each pixel in the intensity channel. The noise intensities are ranging from 5 to 70 for each pixel intensity value. We study the filtration quality (mean square error in a comparison with the original noise-free image) and the number of segments.

In Table 2, we can see that both HMS and HBMS have an increasing mean square error with the increasing amount of noise applied to the image. It means that the images with a high level of noise cannot be filtered with a low spatial radius. In HBMS, the number of segments does not depend much on the noise level, although the segments are scattered. The noise is still easily visible in the

Table 2. Influence of noise on filtration error and number of segments

	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70
HMS MSE	237	213	221	301	255	243	312	398	458	555	617	745	993	1163	1299
HMS segm	60	73	71	54	64	60	69	90	77	76	112	113	166	183	152
HBMS MSE	187	187	197	201	213	220	244	287	349	497	613	746	942	1135	1306
HBMS segm	57	64	51	50	66	51	52	48	67	60	69	61	86	67	71

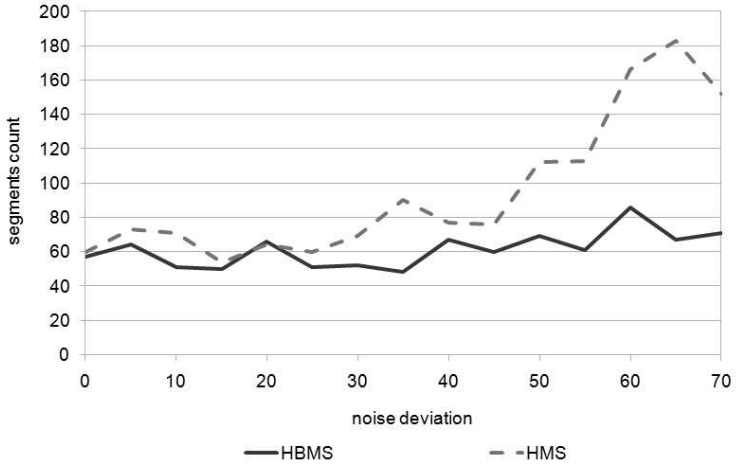


Fig. 4. Influence of noise on number of segments

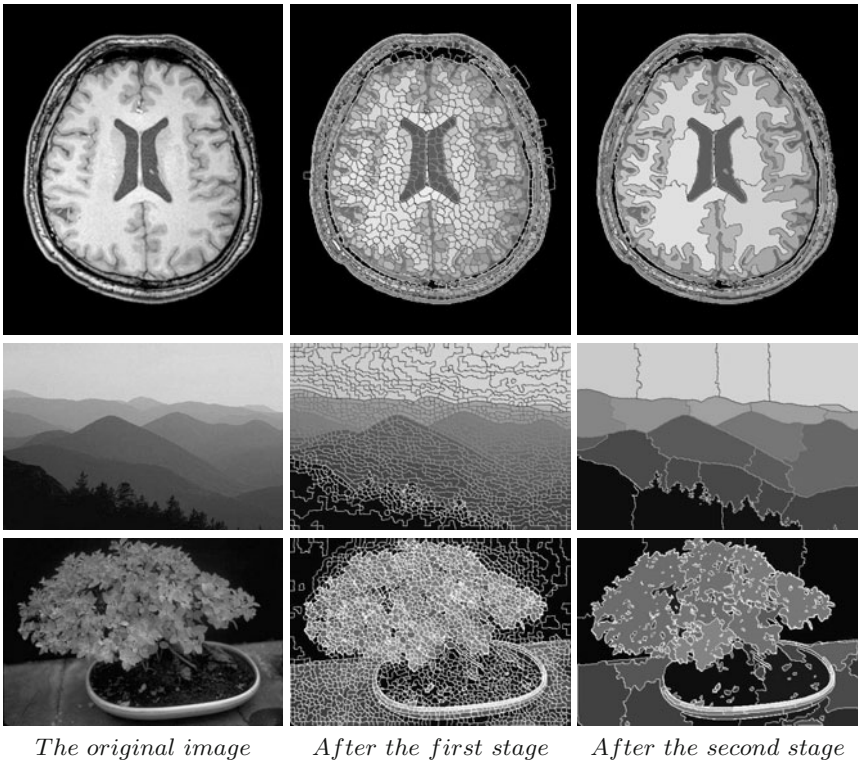


Fig. 5. Segmentation output after each stage for the brain image ($\sigma_{s1} = 7$, $\sigma_{s2} = 70$) and the images from the Berkeley image database ($\sigma_{s1} = 4$, $\sigma_{s2} = 50$)

processed image but it is obvious that the pixels darkened by noise, are grouped together. The same applies on lightened pixels. In HMS, a stronger noise does lead to an increased number of segments.

For the illustration of HBMS method capabilities, we provide three additional images. The first image is a medical scan of brain with the size of 550×650 pixels. The bandwidth σ_{s1} was set to 7, and σ_{s2} was set to 70 pixels. The range bandwidth σ_r remains unchanged with the value of 24 in pixel intensity. After that, we have chosen two images from the Berkeley image database with the size of 481×321 pixels. The mountains image is the most simple one. The tree image is more difficult one because it has a very large number of small leaves and so it could be difficult to merge them into one segment. Both images from the Berkeley database were segmented with values set to $\sigma_{s1} = 4$, $\sigma_{s2} = 50$.

The original brain image consists of 357,500 pixels. The dataset was reduced to 3,594 segments after the first stage and 72 segments after the second one. The computation required 53.8 seconds for the first stage and 26.1 second for the second one.

In the mountains image, HBMS reduced the number of pixels from 154,401 to 2,749 in the first stage and 29 segments in the final stage. The computation lasted 9.1 seconds (first stage) and 6.5 seconds (second stage). The computation took 16.3 seconds. For comparison, the original BMS with the same 50-pixel searching window lasted 894 seconds and the final segmentation had 62 segments.

For the tree image, the number of pixels was reduced from 154,401 to 3,710 segments after the first stage and 65 segments after the second stage. For this case, the execution time was 17.6 seconds.

6 Conclusion

In this paper, we have proposed a new segmentation method based on hierarchical Mean-Shift. Blurring Mean-Shift was used instead of the original MS that has many disadvantages. The proposed method decreases computational time due to the use of BMS on which HBMS is based on, and decreases the number of segments. This is why each succeeding stage is faster. The computational time can reach tens of minutes or hours with large images using the original HMS while our HBMS method needs only a few minutes and quality is still comparable to the original HMS, BMS and MS methods.

References

1. Carreira-Perpiñán, M.: Fast nonparametric clustering with Gaussian blurring mean-shift. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 153–160. ACM, New York (2006)
2. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 17, 790–799 (1995)
3. Comaniciu, D., Meer, P.: Mean shift analysis and applications. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, p. 1197 (1999)

4. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5), 603–619 (2002)
5. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Anal. Mach. Intell. 25, 564–575 (2003)
6. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory 21(1), 32–40 (1975)
7. Sheikh, Y.A., Khan, E.A., Kanade, T.: Mode-seeking by medoidshifts. In: IEEE International Conference on Computer Vision, pp. 1–8 (2007)
8. Vatturi, P., Wong, W.K.: Category detection using hierarchical mean shift. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 847–856. ACM, New York (2009)
9. Zhao, Q., Yang, Z., Tao, H., Liu, W.: Evolving mean shift with adaptive bandwidth: A fast and noise robust approach. In: ACCV (1) 2009, pp. 258–268 (2009)