Layered Mean Shift Methods

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@vsb.cz

Abstract. Segmentation is one of the most discussed problems in image processing. Many various methods for image segmentation exist. The mean-shift method is one of them and it was widely developed in recent years and it is still being developed. In this paper, we propose a new method called Layered Mean Shift that uses multiple mean-shift segmentations with different bandwidths stacked for elimination of the over-segmentation problem and finding the most appropriate segment boundaries. This method effectively reduces the need for the use of large kernels in the mean-shift method. Therefore, it also significantly reduces the computational complexity.

Keywords: layer, segmentation, image, mean shift, over-segmentation.

1 Introduction

Segmentation can be solved by many various algorithms. They differ in speed and accuracy. Both goals are often contradictory, very fast methods are often not very accurate and vice versa. Mean shift (MS) is one of the most popular methods in recent years, although it was firstly presented in 1975 [1]. Nowadays, this method is known as Blurring MS (BMS) and it was deeply discussed in 2006 [2]. The mean-shift methods belong to the more precise methods giving very nice filtration results. Many of them give nice segmentation results too. The problem of MS is in a high computational complexity, although many faster variants were presented in few recent years. The high computational complexity is most obvious in general mean-shift method usually denoted as MS. It was presented in 1995 [3] and deeply studied in [4], [5], and [6].

Mean shift is an iterative method that seeks for a position with the locally highest density of data points. During computation, a kernel density estimate is computed for every data point. Because we are segmenting images, the pixels in images are used as these data points in our case. Each pixel is shifted according to the density estimate and computation is carried out until the convergence when the shift is very small or zero. Two datasets are used in general MS. We distinguish between an original and a shifted data. In the first iteration, both are the same. In the following iterations, we compute mean shift for the already shifted pixel, but the neighboring pixels in the kernel placed on the computed

A. Kuijper et al. (Eds.): SSVM 2013, LNCS 7893, pp. 465–476, 2013.

pixel are taken from the original dataset that is never changed. BMS uses a slightly different approach because only one dataset is used. After each iteration, the output from the previous iteration is used as an input dataset for the next one. Therefore, the dataset is slightly blurred after each iteration (the computed pixel is not taken from the original dataset like in MS but it is taken from the slightly blurred dataset) and convergence is faster. All the pixels that converged to the same position, create one segment in the processed image.

The speed of all mean-shift methods is highly dependent on the size of kernel (the number of pixels that are needed to compute the kernel density estimate) and the number of iterations needed to achieve the convergence. It was proved that MS has a higher number of iterations per pixel than BMS [2]. In 2009, Evolving MS (EMS) was presented in [7] and [8]. It promises even lower number of iterations per pixel but each iteration requires a lot of overheads. Minimizing of the kernel sizes is mostly utilized in the hierarchical approaches [9], [10] and [11], where a small kernel is used in the first stage of these algorithms and then larger kernels are used in the following stages where the input is the computed segmentation from the previous stage. In this paper, we present a new Layered Mean Shift method family that is based on minimization of kernel sizes in order to achieve a faster segmentation. Our method also improves the detection of significant boundaries of objects and minimizes the over-segmentation problem.

In the next section, the basics of Mean Shift are going to be described. Section 3 is devoted to our new method called Layered Mean Shift. We use layered versions for several mean-shift methods, but for explanation, LxMS abbreviation for an unspecified layered mean-shift method will be used generally.

2 Mean Shift

Let $X = \{x_i\}_{i=1}^{n} \subset R^d$ be a dataset of n points in the d-dimensional space. The *kernel density estimator* is given by the equation

$$p(x) = \frac{1}{n\sigma^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{\sigma}\right), \tag{1}$$

where σ is a bandwidth parameter limiting the size of kernel function $K(x)$. In some literature, denomination the bandwidth parameter as h is also used. We can distinguish between two types of bandwidths in images. The spatial bandwidth σ_s is the first one and limits the neighbourhood of the processed pixel in x and y axis. The range bandwidth is the second one and it is denoted by σ_r. It indicates the maximal luminance difference of the sample that can fall inside the kernel. We can have more bandwidths in colour images, for example, three bandwidths for each colour channel. In our work, we are focused on the greyscale images and only one σ_r is needed. The fraction before the sum in Eq. (1) is a normalization constant. The processed pixel is denoted as x and all pixels in the neighbourhood (kernel) are labeled as x_i.

Many types of kernel functions exist. The *Gaussian* is the most popular and often gives the nicest results, but it has also few drawbacks. It is not trun-

cated kernel and covers the whole dataset. The bandwidth is not limiting the size of kernel but only the contribution of the samples. For computation of one mean-shift vector, we need to compute the kernel function with all image pixels. Therefore, the Gaussian kernel is very slow and inherently not appropriate for using it in our method that limits the size of kernel. Of course, there is a possibility of truncation of the Gaussian kernel in a defined distance.

In our approach, we use another very famous kernel, the *Epanechnikov* kernel. It is truncated and the σ parameters limits the contribution of the pixels in the distance of σ. All pixels that are out of the hypersphere given by these parameters, are not involved in computation and the algorithm can be much faster, especially with smaller values of σ. The Epanechnikov kernel is given by the equation

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

All inferences of kernel estimates and their relationship to mean-shift methods are deeply described in [4] and [6]. Therefore, we do not deal with them deeply in this paper. On the other hand, the mean-shift vector should be mentioned at least. Its equation is given by

$$m_{\sigma,k}(x) = \frac{\sum_{i-1}^{n} x_i k\left(\left\|\frac{x-x_i}{\sigma}\right\|^2\right)}{\sum_{i-1}^{n} k\left(\left\|\frac{x-x_i}{\sigma}\right\|^2\right)} - x\,, \tag{3}$$

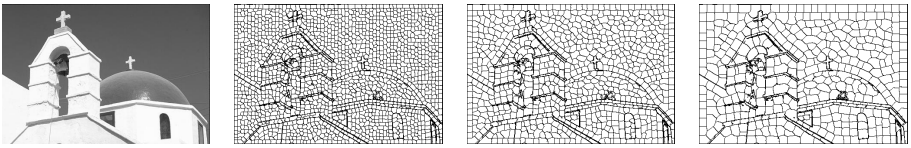
where the function $k(x)$ is a derivative of the kernel $K(x)$. The first term on the right-hand side is a new position of the processed pixel x (estimate of the position with the highest density of data points), the second term is the former position. The difference $m_{\sigma,k}(x)$ between them is called the *mean-shift vector*. In this case, we present the equation for Blurring MS that is faster than general MS. It uses the modified dataset in each iteration and its results are more regular. General experiments with our LxMS method will be carried out with the LBMS version of it.

3 Layered Mean Shift Methods

We present a new *Layered Mean Shift* (LxMS) that is aimed to the reduction of computational time as well as to reduction of the over-segmentation problem. It is well known that the size of segments is highly dependent on the value of σ parameter. The larger the σ parameter is, the larger are the segments. If we expect large segments, we are forced to use larger σ_s. This leads to slower computation because of $O({\sigma_s}^2)$ complexity for evaluation of one mean-shift vector for one pixel. Our LxMS method solves this speed and over-segmentation problem. LxMS does not suffer from over-segmentation even if it is used with small bandwidths.

The main idea of LxMS is in execution of multiple mean-shift segmentations with different kernel sizes that are not very large in any of executed segmentations. General MS, Blurring MS, and Evolving MS can be used as a basic method that will be executed repeatedly. Layered mean-shift method using BMS segmentation as its base can be called LBMS (Layered Blurring Mean Shift), the same applies to MS in HMS method and EMS in HEMS method. As we already said, presented results that explain the layered approach use BMS as its base in all cases (LBMS method).

We use m segmentations, each with a different spatial bandwidth. Then these segmentations are overlaid. Important edges in the image are highlighted in all segmentations, whereas over-segmentation artifacts are positioned in various locations in each segmentation. If we stack these segmentations, these artifacts are almost invisible (they are placed only once in some area) and only important edges remain (each segmentation produces the same border in the same place). For example, we can execute three different mean-shift segmentations, the first one with $\sigma_s = 3$, second one with $\sigma_s = 4$, and third one with $\sigma_s = 5$. We use the same σ_r for all segmentations but it is not necessary.



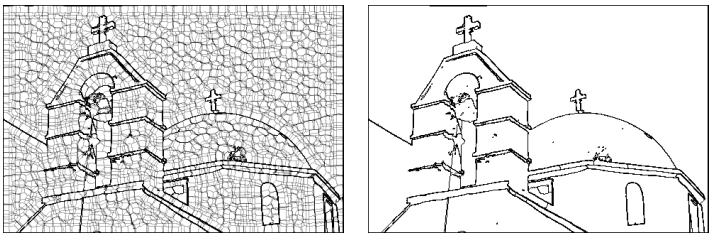
(a) original image (b) $\sigma_s = 5$ (c) $\sigma_s = 7$ (d) $\sigma_s = 9.8$

Fig. 1. Phases of the LBMS method. The original image is in the first column. The images with different bandwidths are shown in next three figures.

Three different segmentations are visible in Fig. 1. Each was computed with a different spatial bandwidth and, therefore, created a different segmentation. In all three cases, the boundaries of church are clearly evident. The number of stacked segmentations is not limited, of course.

Fig. 2(a) shows that even very small searching windows (kernels) with $\sigma_s < 10$ in the 481×321 pixel image completely reduces the problem of over-segmentation. There is no need to use spatial bandwidth with the size of hundreds of pixels. The better speed is achieved, because we carry out a small number of fast segmentations instead of one very slow segmentation.

It is obvious that only the image of stacked segmentations (Fig. 2(a)) is useless and it has to be processed to create one useful result. In Fig. 2(b), the result of merging the segments is visible. Many approaches are possible. If we want only edges, we can use simple edge following algorithms. If we need a real segmentation, another approach should be used. In LxMS, we use our own *segment merging* algorithm. We pick all pairs of pixels from the image and sum how many times they were in the same segment from m executed segmentations. The threshold t lower than the number of stacked segmentations is set. If the sum



(a) Stacked image from three segmentations (b) Result after basic processing

Fig. 2. Stacked segmentation and result of merging the segments

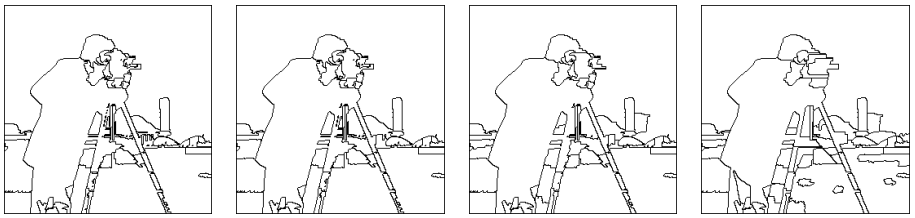
is higher or equal to the threshold, both pixels are inserted into one segment. For example, we have three segmentations and we can set the threshold value to 2. If two random pixels were in the same segment at least in two of three segmentations, they belong to one segment. It can be clearly seen in Fig. 3. After carrying out this proces, we should remove very small segments; see the small spots in 2(b).

(a) $\sigma = 5$ (b) $\sigma = 7$ (c) $\sigma = 9.8$

Fig. 3. Merging of segments. For example, if two random pixels are twice in one segment from three possible segmentations (Fig. 3(a) and Fig. 3(c)), they are given the same segment label. The number of segmentations and necessary number of the same assignments to segments is adjustable. It does not need to be 3 and 2 like in this example.

Intuitively, we should try to find the pairs between all pixels in the whole image. It leads to high complexity of $O(n^2)$ that is the same as complexity of MS and BMS algorithm. We observed that approx. $20 - 40\%$ of computational time is spent on this merging the segments with such a naive approach. It is obvious that there is almost no possibility to obtain two pixels in one segmentation if their spatial distance is larger than the spatial bandwidth (if they are not covered by the kernel, they will be hardly assigned to one attractor). Therefore, we do not need to check all pixels with all pixels in the whole image, but only with their close neighbourhood given by σ parameter. Our experiments showed that the results were the same with such an acceleration and complexity dropped to $O({\sigma_s}^2)$.

If we limit the maximal distance for merging more, we prevent the merging of the points that are divided by an another segment spatially between them.



(a) $n = \sigma$ (b) $n = 3$ (c) $n = 2$ (d) $n = 1$

Fig. 4. Segmentation merging with limiting the neigbourhood for searching pairs of pixels belonging to the same segment. Parameter n is the radius of this neigbourhood.

It should not be acceptable in statistical usage of MS method but sometimes it can be useful in digital images. This method is sort of a trade off. Some details are suppressed (segments have more regular shapes) but also does not connect corresponding parts of segments which can divide one object to more pieces and also can make additional artifacts. Therefore, we recommend to use $n = \sigma$. In Fig. 4, reduction of small segments with the size smaller than 50 pixels was used. These small segments emerge in the places where boundaries of true segments differ very slightly in different segmentations. These pixels are not assigned to any segment and create their own small segment. Such small segments are assigned to a neighboring segment.

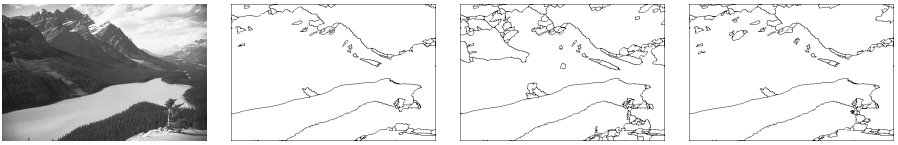
Algorithm 1. Layered Mean Shift

Input: Dataset X, spatial bandwidth σ_s, range (luminance) bandwidth σ_r, bandwidth multiplier l, number of segmentations m, threshold t.
Output: A clustered dataset X_s

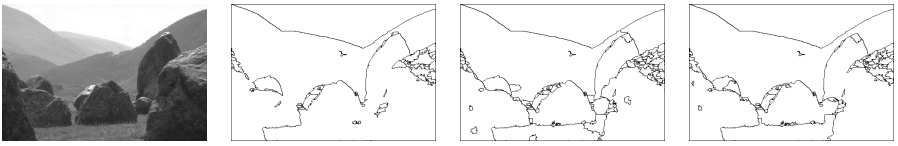
```
Set index i = 0
repeat
    Evaluate MS (or BMS) segmentation X_i with bandwidths σ_s and σ_r, where i is a index of segmentation
    Multiplicate σ_s by l and increase index i by 1
until index i = m
for all pixels x_j do
    for all pixels x_k in circle neighbourhood of pixel x_j with radius of σ_s do
        Sum the number of segmentations X_i where pixels x_j and x_k belong to the same segmentation
        If the sum is equal or higher than a threshold t, both pixels x_j and x_k are assigned to the same segment
    end for
end for
Eliminate segments with size smaller than a preset threshold (fraction of image area).
```

4 Experiments

In this section, the experiments with our algorithm are provided. We tested LBMS algorithm in comparison with the original BMS and we studied the segmentation quality as well as the speed of algorithm. Also, the hierarchical version of BMS (called HBMS) was tried. We use removal of small segments with the size smaller than a preset threshold (for example 1/5000 of image area). The bandwidth range and the number of stacked segmentations are mentioned for each test. We use images from Berkeley Image Dataset [12].



(a) original image (b) 4; 1.4; 3-2/24 (c) 4; 1.25; 4-3/24 (d) 4; 1.4; 4-3/24



(e) original image (f) 5; 1.4; 3-2/24 (g) 5; 1.25; 4-3/24 (h) 5; 1.4; 4-3/24

Fig. 5. Comparison of LBMS segmentations. The range bandwidth σ_r is the number after the slash. "The first number stands for the value of σ_s in the first segmentation, the second number is the multiplier. The notation $3-2$ says that 3 segmentations were processed and pixels that were 2 or more times in the same segmentation have been merged.

In Fig. 5, we see that different parameters lead to slightly different segmentations but we can not fully determine general influence of parameters to the segmentation quality. Of course, the larger number of processed segmentations causes a higher computational time. The higher values of multiplier increase computational time too because of higher values of σ_s parameters in the following segmentations. A small difference of spatial bandwidths between the stacked segmentations causes the increase of the number of segments. The same effect can be seen if we increase the number of stacked segmentations (of course, it can be lowered by lowering the threshold). We can say that the higher number of stacked segmentations often produces the resultant segmentation with a higher quality of details (see the incomplete stones in Fig. 5(f) and better result in Fig. 5(h)) but also with slightly more visible over-segmentation.

We can see several examples in Fig. 6 and processing times in Table 2. It is obvious that LBMS does not suffer from over-segmentation even if the kernel sizes were below $\sigma_s = 10$. Both BMS and HBMS used $\sigma_s = 20$, but there is a visible over-segmenation in both results. If we want to reduce this effect, we have to

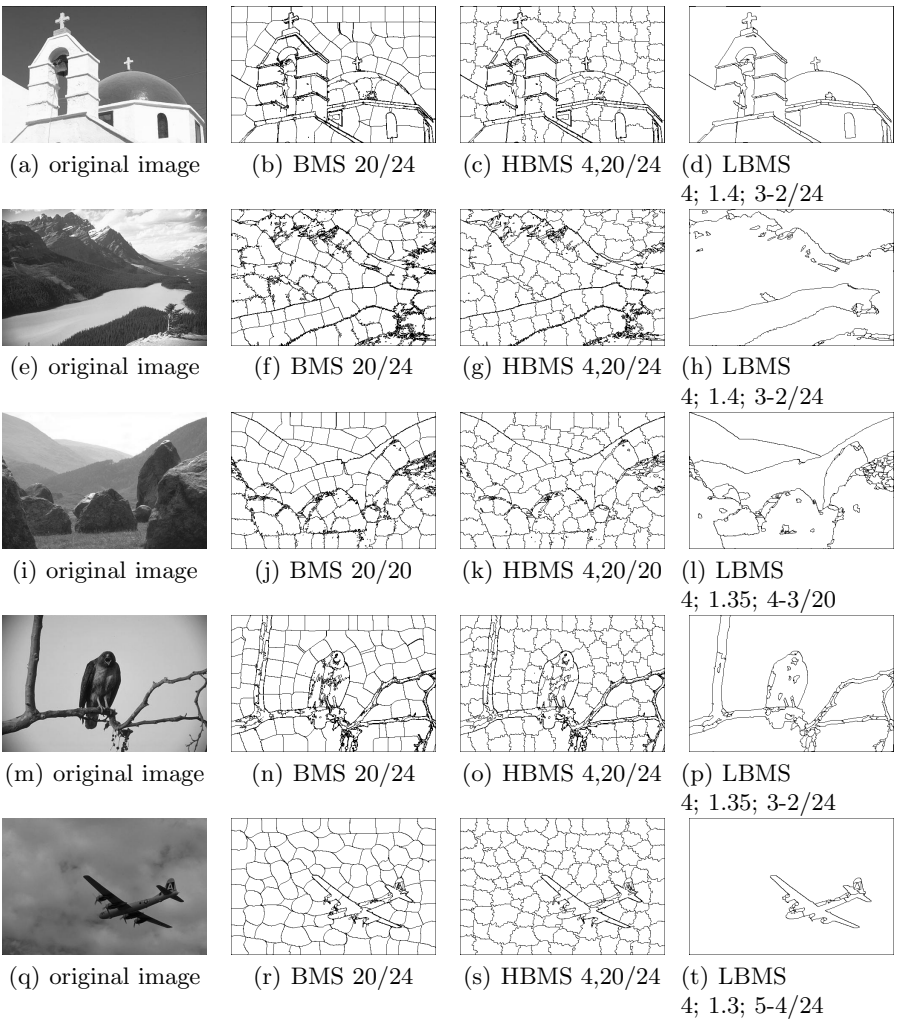


Fig. 6. Range bandwidth σ_r is the number after the slash. In BMS, the number before the slash is the spatial bandwidth σ_s. In HBMS, the numbers 4,20 mean that the first stage used $\sigma_s = 4$ and the second one used $\sigma_s = 20$. The first number in the LBMS notation is the value σ_s in the first segmentation and the second number is the bandwidth multiplier. The notation $3 - 2$ says that 3 segmentations were processed and pixels that were 2 or more times in the same segmentation have been grouped.

Table 1. Comparison of the numbers of segments and speed depending on parameters

image	bandwidth σ_s	multiplier	segmentations	threshold	time	segments
mountains	4	1.4	3	2	56.2 s	85
mountains	4	1.25	4	3	70.6 s	178
mountains	4	1.4	4	3	108.5 s	119
stones	5	1.4	3	2	87.2 s	90
stones	5	1.25	4	3	117.6 s	150
stones	5	1.4	4	3	171.6 s	121

Table 2. Comparison of the numbers of segments (seg) and the computational time (t[s]) depending on the algorithm

	BMS		HBMS		LBMS	
image	t[s]	seg	t[s]	seg	t[s]	seg
church	158.1 s	274	9.6 s	253	58.7 s	114
mountains	158.1 s	238	10.1 s	262	56.2 s	85
stones	155.9 s	198	9.7 s	185	106.3 s	149
bird	150.6 s	260	10.4 s	265	59.8 s	103
airplane	185.0 s	145	9.4 s	146	168.6 s	27

enlarge the kernel size. That would lead to enormous increase of computational time (quadratically). The segmentation is subjectively visually very nice and the computational times are much better than in BMS. On the other hand, the hierarchical approaches are still much faster but they suffer from over-segmentation. In many images, only 3 stacked segmentations are sufficient but a higher number of segmentations could be useful in images with more noticeable textures. Enlarging the number of executed segmentations would cause the increase of computational time. The more simple the images are, the smaller number of segmentations and smaller kernel size has to be used. Small bandwidths are often sufficient because they also produce different segmentation boundaries in flat areas and the same boundaries on the true edges of detected objects.

In Fig. 7, you can see five images segmented by various mean-shift methods. We used MS, BMS, EMS, their hierarchical versions HMS, HBMS, HEMS and their layered versions LMS, LBMS and LEMS. The first image is a synthetic image with smooth background gradient and smooth shapes. MS had very big problem to segment it because of zero gradient of underlying structure. The data point can not move and image is segmented only around the edges, where the non-zero gradient of density exists. The second image is the noisier version of the first image. Therefore, it can be segmented by MS. The following three images are the real-life images from the Berkeley Image Database [12]. One stage algorithms (MS, BMS and EMS) used the spatial bandwidth $\sigma_s = 25$. All of the hierarchical approaches were used in their 3-stage versions using bandwidths of $\sigma_{s_1} = 3.5$, $\sigma_{s_2} = 12$ and $\sigma_{s_3} = 50$. The layered version used two possible configurations. The first one is $3-2$, where 3 stages were processed and the pixels were merged into

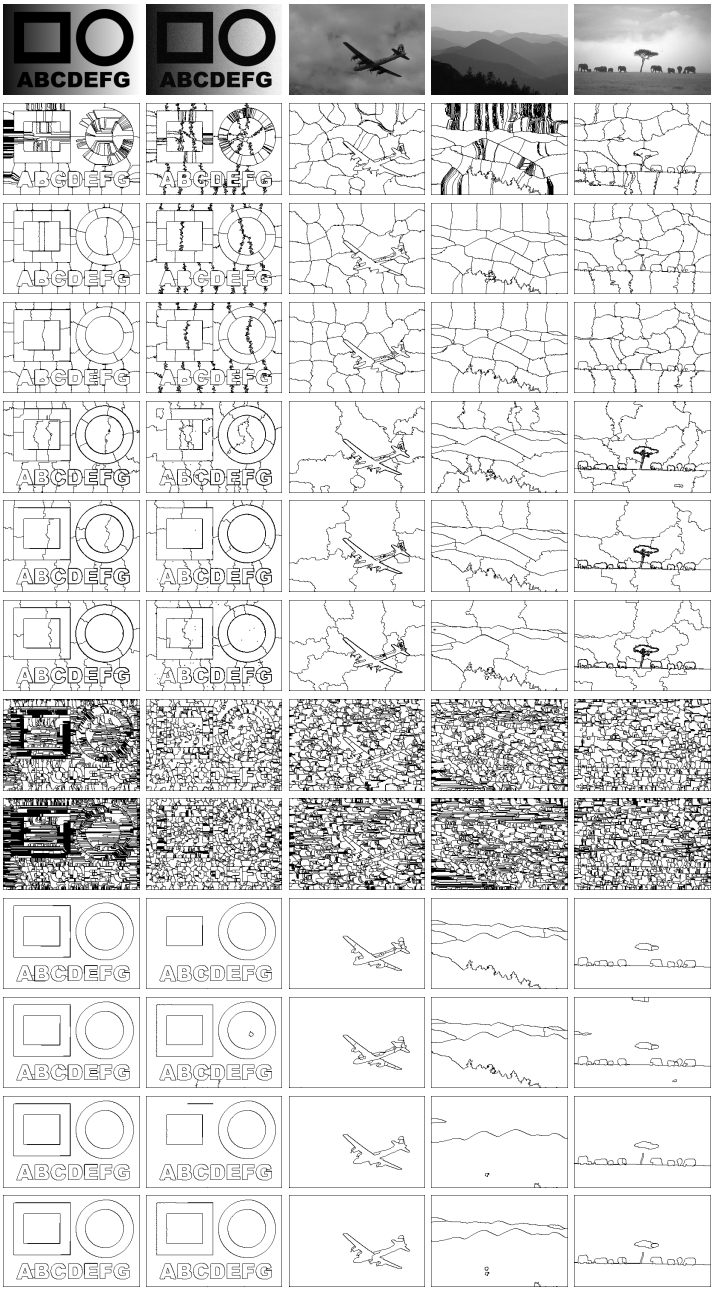


Fig. 7. Rows 1: the original image; 2/3/4: MS/BMS/EMS (spatial bandiwdth $\sigma_s = 25$); 5/6/7: HMS/HBMS/HEMS ($\sigma_s = 3.5/12/50$); 8/10/12: LMS/LBMS/LEMS (3-2 stages, $\sigma_s = 4$, multiplier of the bandwidth $mul = 1.3$); 9/11/13: LMS/LBMS/LEMS (4-3 stages, $\sigma_s = 4$, $mul = 1.3$); the notation is similar to Fig. 6.

a larger segment if they both were at least twice in the same segment. The last configuration was analogically $4-3$. It is obvious that LMS is unusable as the original MS gives an unstable result that can not be easily merged. LBMS and LEMS usually give a stable result around the most visible edges and, therefore, the layered approach is very beneficial here. In our additional tests, it has been shown that LMS needs at least $\sigma_{s_1} = 9$ for satisfactory result.

Although the largest spatial bandwidth was 6.7 in $3-2$ configuration or 8.7 in $4-3$ configuration, it definitely outperforms all other algorithms in the area of the over-segmentation problem. We can enlarge the spatial bandwidth in the classical and hierarchical algorithms to decrease the over-segmentation but it will lead to much longer computational time and potentially inaccurate results (the large σ_s will cover the large image or even the whole image and the spatial term will be unimportant - all the pixels with the same brightness in the image will be grouped even if they are separated by another segments). Such a situation does not happen in layered algorithms because of small spatial merging bandwidths.

Table 3. Comparison of the speed (t[s]) depending on algorithm

	synth. image1	synth. image 2	airplane	mountains	savana
MS	2185.35 s	2014.93 s	2112.62 s	3348.3 s	2999.28 s
BMS	82.29 s	94.34 s	107.2 s	80.58 s	103.35 s
EMS	1061.62 s	1129.69 s	629.86 s	543.26 s	753.63 s
HMS3	67.9 s	20.18 s	23.71 s	24.81 s	24.29 s
HBMS3	7.56 s	7.63 s	6.63 s	6.96 s	6.84 s
HEMS3	27.14 s	33.94 s	16.63 s	14.41 s	18.86 s
LMS3/2	318.67 s	123.08 s	282.11 s	345.74 s	222.02 s
LBMS3/2	21.21 s	19.43 s	24.08 s	23.22 s	21.62 s
LEMS3/2	79.41 s	75.21 s	70.12 s	72.05 s	73.28 s
LMS4/3	603.13 s	280.99 s	538.54 s	695.87 s	442.88 s
LBMS4/3	36.75 s	36.22 s	40.52 s	37.88 s	34.78 s
LEMS4/3	385.45 s	318.73 s	125.6 s	152.81 s	127.33 s

Table 3 shows the speed of all the algorithms. The hierarchical approaches are the fastest but they still suffer from the over-segmentation problem. The layered versions are 2.5 to 4-times slower (with the exception of LEMS4-3 in the synthetic images) with no over-segmentation problem. There rises a question whether this trade off is acceptable or not.

5 Conclusion

The layered mean-shift methods showed that they are relatively fast methods that primarily reduce the over-segmentation problem even with very small kernel sizes. They are very well suited for images with not very noticeable textures. In other cases, the number of stacked segmentations should be enlarged to achieve

a proper segmentation. Mean-shift, blurring mean-shift and evolving mean-shift approaches can be embedded into the LxMS method but it has been shown that general MS is not a very good choice. In LMS case, it needs much larger initial spatial bandwidth. The next goal is to improve the grouping of pixels in the stacked segmentations to achieve smaller a sensitivity to stronger textures.

Acknowledgements. This work was partially supported by the grant SP 2013 / 185 of VŠB-TU of Ostrava, FEECS.

References

1. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory 21, 32–40 (1975)
2. 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)
3. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 17, 790–799 (1995)
4. Comaniciu, D., Meer, P.: Mean shift analysis and applications. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1197–1203 (1999)
5. Comaniciu, D., Ramesh, V., Meer, P.: The variable bandwidth mean shift and data-driven scale selection. In: IEEE International Conference on Computer Vision, vol. 1, p. 438 (2001)
6. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 603–619 (2002)
7. Zhao, Q., Yang, Z., Tao, H., Liu, W.: Evolving mean shift with adaptive bandwidth: A fast and noise robust approach. In: Zha, H., Taniguchi, R.-i., Maybank, S. (eds.) ACCV 2009, Part I. LNCS, vol. 5994, pp. 258–268. Springer, Heidelberg (2010)
8. Yang, Z., Zhao, Q., Liu, W.: Neural signal classification using a simplified feature set with nonparametric clustering. Neurocomput. 73, 412–422 (2009)
9. 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)
10. DeMenthon, D., Megret, R.: Spatio-Temporal Segmentation of Video by Hierarchical Mean Shift Analysis. Technical Report LAMP-TR-090,CAR-TR-978,CS-TR-4388,UMIACS-TR-2002-68, University of Maryland, College Park (2002)
11. Šurkala, M., Mozdřeň, K., Fusek, R., Sojka, E.: Hierarchical blurring mean-shift. In: Blanc-Talon, J., Kleihorst, R., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2011. LNCS, vol. 6915, pp. 228–238. Springer, Heidelberg (2011)
12. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision., vol. 2, pp. 416–423 (2001)