An Experimental Study of Color-Based Segmentation Algorithms Based on the Mean-Shift Concept

K. Bitsakos, C. Fermüller, and Y. Aloimonos

Center for Automation Research,
University of Maryland, College Park, USA
kbits@cs.umd.edu, {fer,yiannis}@cfar.umd.edu

Abstract. We point out a difference between the original mean-shift formulation of Fukunaga and Hostetler and the common variant in the computer vision community, namely whether the pairwise comparison is performed with the original or with the filtered image of the previous iteration. This leads to a new hybrid algorithm, called Color Mean Shift, that roughly speaking, treats color as Fukunaga's algorithm and spatial coordinates as Comaniciu's algorithm. We perform experiments to evaluate how different kernel functions and color spaces affect the final filtering and segmentation results, and the computational speed, using the Berkeley and Weizmann segmentation databases. We conclude that the new method gives better results than existing mean shift ones on four standard comparison measures ($\backsim$ 15%, 22% improvement on RAND and BDE measures respectively for color images), with slightly higher running times ($\backsim$ 10%). Overall, the new method produces segmentations comparable in quality to the ones obtained with current state of the art segmentation algorithms.

Keywords: image segmentation, image filtering, mean-shift.

1 Introduction

Mean shift is an unsupervised clustering technique that over the last decade gained popularity and is now widely used in computer vision for color based segmentation. Though conceptually simple, an extensive amount of mathematical formalism has been used to precisely describe the method. As a result, some of the important characteristics of the method were "hidden underneath the surface". This paper simplifies the formulation and brings forth its important features by describing mean shift as an optimization problem. This leads to two contributions; a) we propose a new variation, denoted Color Mean Shift, that combines Fukunaga's mean shift superior cluster ability with most of the computational advantages of Comaniciu's variant, and b) we experimentally compare different variations of the algorithm both in terms of the computational speed and the segmentation quality. Color Mean Shift is found to outperform the current methods in terms of the quality of segmentation, while it is slightly ($\backsim$ 10%)

slower . More specifically, it produced $\backsim$ 15%, 22% better results on the Berkeley dataset with the RAND and the BDE measure respectively.

1.1 Related Work

Despite its existence for more than three decades [1], mean-shift only recently gained popularity in the computer vision community. Cheng [2] first modified the method and used it for non-parametric clustering and then, Comaniciu and Meer [3] used it for image filtering and segmentation. Since then, mean-shift has been used in computer vision for object tracking [4], 3D reconstruction [5], texture classification [6] and video segmentation [7] among other problems. The relatively high computational cost of a naive implementation of the method combined with the need for fast image processing led researchers to propose fast approximate variations of it. Most notably, two solutions for finding pairs of points within a radius have been proposed; the Improved Fast Gauss Transform based mean shift [8] for Normal kernels and the Locality Sensitive Hashing based mean shift [6].

Cheng [2] was the first to recognize the equivalence of mean shift to a step-varying gradient ascent optimization problem, and later Fashing and Tomashi [9] showed that it is equivalent to Newton's method with piecewise constant kernels, and is a quadratic bound maximization for all other kernels. Still the dominant way to describe it is by using density estimation terms [3], namely using kernels and their shadow and profile functions.

1.2 Contributions

In this paper, we describe mean shift as an optimization problem. The simplicity of the formulation not only leads to a better understanding of the method, but also brings forth the difference between the original method and its variation that is used in computer vision[1]. In the same section (Sec. 2), we propose our own variant of mean shift, denoted *Color Mean Shift* (CMS), that lies between the two methods. The next two sections contain an experimental comparison between the methods. First, in Sec. 3, we present the filtering results for different kernel functions and color spaces. Then, we study the filtering speed of the algorithms with respect to a number of optimization parameters. In Sec. 4 we show results on two different segmentation datasets (the Berkeley [10] and Weizmann Institute [11] databases) containing 300 images and 1387 human segmentations (in total) using 4 standard comparison measures. In these experiments the new method (i.e, color mean shift) exhibits an improvement of $>$ 15% compared to the existing method on color images. A similar improvement was also achieved for the grayscale images of Weizmann dataset. Summary and future work (Sec. 5) conclude the paper.

[1] In the recent papers, the original "mean shift" approach is called "blurring mean shift". In the rest of the paper we use the abbreviations **FHMS** and **CMMS** for Fugunaga and Hostetler's and Comaniciu and Meer's method of mean shift, respectively.

2 Image Filtering Using the Mean Shift Algorithm

2.1 Notation

We consider the image on the 5D space with spatial and color dimensions. More specifically, $\mathbf{x}_i$ is a 2D vector representing the spatial coordinates and $\mathbf{s}_i$ is a vector that represents the three color channels of pixel i ($i = 1 \ldots N$).

In the following paragraphs we use bold letters to represent vectors and the notation $[\mathbf{x}_i, \mathbf{s}_i]$ to indicate a concatenation of vectors. To indicate the evolution of a vector over time we use superscripts, eg. $[\mathbf{x}_i^0, \mathbf{s}_i^0]$ indicates pixel $\mathbf{x}_i$ having the initial intensity values $\mathbf{s}_i^0$.

2.2 Kernel Functions

In our experiments we use two different kernel functions; the Epanechnikov and the Normal (Gaussian) kernel. The Epanechnikov kernel has the analytic form

$$K_E(\mathbf{x}) = \begin{cases} c_E(1 - \mathbf{x}^T\mathbf{x}) & \mathbf{x}^T\mathbf{x} \leq 1 \\ 0 & otherwise \end{cases}, \tag{1}$$

where c_E is the normalization constant.

The multivariate Normal kernel with variance 1 has the analytic form

$$K_N(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} \exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}). \tag{2}$$

The Normal kernel is symmetrically truncated to obtain a kernel with finite support.

2.3 Fukunaga and Hostetler's Mean Shift (FHMS)

The original mean shift formulation [1] (applied to a color image) treats the image as a set of $5 - D$ points. Each point is iteratively moved proportionally to the weighted average of its neighboring points. At the end, clusters of points are formed. We define mean shift to be the gradient descent solution of the optimization problem

$$\arg\min_{[\mathbf{x}_i, \mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i, \mathbf{s}_i] - [\mathbf{x}_j, \mathbf{s}_j]), \tag{3}$$

where $\sum_{i,j}$ defines the summation over all pairs of pixels in the image. This problem has a global minimum when all the pixels "collapse" into a single point. We seek a local minimum instead. That's why we initialize the features $[\mathbf{x}_i, \mathbf{s}_i]$ with the original position and color of the pixels of the image and perform gradient descent iterations till we reach the local minimum. The instabilities caused by this behavior are studied in a recent work of Rao et al. [12].

2.4 Comaniciu and Meer's Mean Shift (CMMS)

The modified mean shift formulation proposed by Comaniciu and Meer [3] (CMMS) can also be expressed as a gradient descent solution of the optimization problem

$$\arg\min_{[\mathbf{x}_i,\mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i,\mathbf{s}_i]-[\mathbf{x}_j^0,\mathbf{s}_j^0]). \tag{4}$$

There is a subtle difference between CMMS and FHMS, that significantly affects the behavior. In the former formulation each feature point is compared against the original set of $5-D$ points $[\mathbf{x}_j^0,\mathbf{s}_j^0]$, while in the latter case the point is compared against the set of points from the previous iteration $[\mathbf{x}_j,\mathbf{s}_j]$.

Fig. 1 presents the results of both methods in a smoothly varying intensity image. Notice that the gradient of the kernel function, everywhere but in the boundaries, is zero and so CMMS filtering only changes the intensity on the boundaries (that change is not very visible). FHMS, on the other hand, produces artificial segments of uniform intensity. Intuitively, each iteration of the process results in more clustered data which in turn leads to better clustering results in the next iteration. On the downside, a fast FHMS implementation is challenging (if not impossible) due to the fact that the feature points and the comparison points do not lie on a regular spatial grid anymore. Thus one would have to compare the current feature $[\mathbf{x}_i,\mathbf{s}_i]$ against all the remaining feature points.

2.5 Color Mean Shift (CMS)

Our method *alleviates the computational problem* of FHMS by using the original spatial location of the points for comparison, while using the updated intensity values of the previous iteration for *improved clustering* ability. In a sense, we perform FHMS on the color dimensions and CMMS on the spatial dimensions (that is the reason for naming the method "color mean shift"). As above, CMS can be expressed as the gradient descent solution of the optimization problem

$$\arg\min_{[\mathbf{x}_i,\mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i,\mathbf{s}_i]-[\mathbf{x}_j^0,\mathbf{s}_j]). \tag{5}$$

We have included the results of color mean shift filtering in the smoothly varying image of Fig. 1. It is clear that individual clusters of uniform intensities are formed (as in the case of the original mean shift). Note that in this example there is not a single right solution for the segmentation problem and one can argue that a single segment is the best solution. We present this example only to exhibit one "weakness" of the CMMS algorithm, that is addressed in both our solution and the original mean shift algorithm. In Fig. 2 we present both CMS and CMMS algorithms.

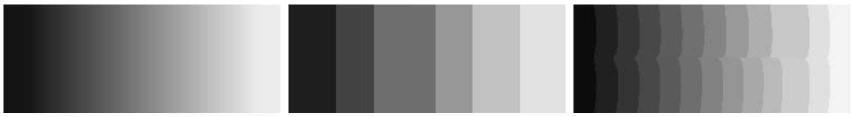

(a) CMMS (b) CMS (c) FHMS

Fig. 1. All the described algorithms applied on a 256 × 100 pixels smoothly varying image. All the filtering algorithms were executed with spatial resolution $h_s = 21$ and range resolution $h_r = 10$ and used a Normal kernel.

CMS

```
Input:
  set of pixels $\mathbf{x}_i^0$ with intensities $\mathbf{s}_i^0$
  a function $g$
Output:
  feature vector $[\mathbf{x}_i, \mathbf{s}_i]$
Algorithm:
  initialize feature points $[\mathbf{x}_i, \mathbf{s}_i] \leftarrow [\mathbf{x}_i^0, \mathbf{s}_i^0]$
  repeat until convergence
      for all features $[\mathbf{x}_i, \mathbf{s}_i]$
        $[\mathbf{x}_i, \mathbf{s}_i] \leftarrow \frac{\sum_j [\mathbf{x}_j, \mathbf{s}_j] g(||[\mathbf{x}_i, \mathbf{s}_i] - [\mathbf{x}_j^0, \mathbf{s}_j]||^2)}{\sum_j g(||[\mathbf{x}_i, \mathbf{s}_i] - [\mathbf{x}_j^0, \mathbf{s}_j]||^2)}$
```

CMMS

```
Input:
  set of pixels $\mathbf{x}_i^0$ with intensities $\mathbf{s}_i^0$
  a function $g$
Output:
  feature vector $[\mathbf{x}_i, \mathbf{s}_i]$
Algorithm:
  initialize feature points $[\mathbf{x}_i, \mathbf{s}_i] \leftarrow [\mathbf{x}_i^0, \mathbf{s}_i^0]$
  for all features $[\mathbf{x}_i, \mathbf{s}_i]$
      repeat until convergence
        $[\mathbf{x}_i, \mathbf{s}_i] \leftarrow \frac{\sum_j [\mathbf{x}_j, \mathbf{s}_j] g(||[\mathbf{x}_i, \mathbf{s}_i] - [\mathbf{x}_j^0, \mathbf{s}_j^0]||^2)}{\sum_j g(||[\mathbf{x}_i, \mathbf{s}_i] - [\mathbf{x}_j^0, \mathbf{s}_j^0]||^2)}$
```

Connected Components Grouping

```
Input:
  set of pixels $\mathbf{x}_i$ with intensities $\mathbf{s}_i$
  grouping threshold $t$
Output:
  label $l_i$ for pixel $\mathbf{x}_i$
Algorithm:
  repeat until convergence
    for all pixels $\mathbf{x}_i$
      for all $\mathbf{x}_j$ adjacent to $\mathbf{x}_i$
        if $||s_i - s_j||^2 < t$ and $x_i, x_j$ have different labels:
          merge the labels of $x_i$ and $x_j$ $(l_i \equiv l_j)$
```

Fig. 2. In all algorithms $g(x) = [x \leq 1]$ (indicator function in Iverson notation) for the Epanechnikov and $g(x) = \exp(-x/2)$ for the Normal kernel

3 Filtering Comparison

Following the example of Comaniciu and Meer [3], we normalize the spatial and color coordinates of each pixel vector by dividing by the spatial (h_s) and color (h_r) resolutions. Thus, the original feature vector $[\mathbf{x}_i, \mathbf{s}_i]$ is transformed to $[\frac{\mathbf{x}_i}{h_s}, \frac{\mathbf{s}_i}{h_r}]$ (not included in the equations for simplicity). The spatial resolution

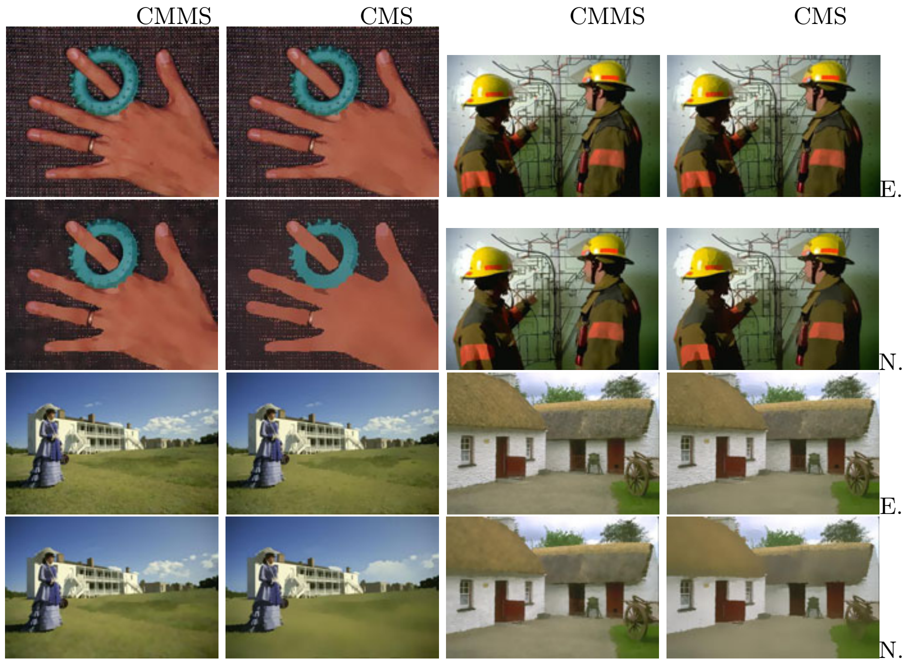


Fig. 3. Epanechnikov vs Normal kernel. We use $h_s = 5$ and $h_r = 19$. All images are processed in RGB color space. E., N. stand for Epanechnikov kernel and Normal kernel respectively. The Normal kernel produces smoother regions. Also, CMS produces more uniform regions even in heavily textured areas, eg. the grass and the roof.

h_s affects the size of the neighborhood around each pixel that the algorithm considers and in all the experiments is constant ($h_s = 5$ corresponding to a 11×11 window). Then, we perform the optimization; one pixel at a time in the case of CMMS (Fig. 2, top right), or one iteration of the whole feature set at a time for FHMS and CMS (Fig. 2, top left). FHMS has a complexity that is quadratic on the number of pixels of the whole image. Thus, its running time for a reasonably size image (eg. 640×480 pixels) is several minutes, making it prohibitively slow for any computer vision application. For that reason, we omit the results of this algorithm in the experiments.

3.1 Filtering Using an Epanechnikov or a Normal Kernel

First we present the effect of using different kernels: Epanechnikov and Normal (Fig. 3). Each column of the figure depicts the filtering result with a different algorithm (CMMS or CMS) and each row for a different kernel function (N., E. stand for Normal and Epanechnikov kernels respectively). In all cases the Normal kernel produces smoother results, while still preserving edge discontinuities. As a matter of fact, the color resolution h_r is the parameter that defines the gradient

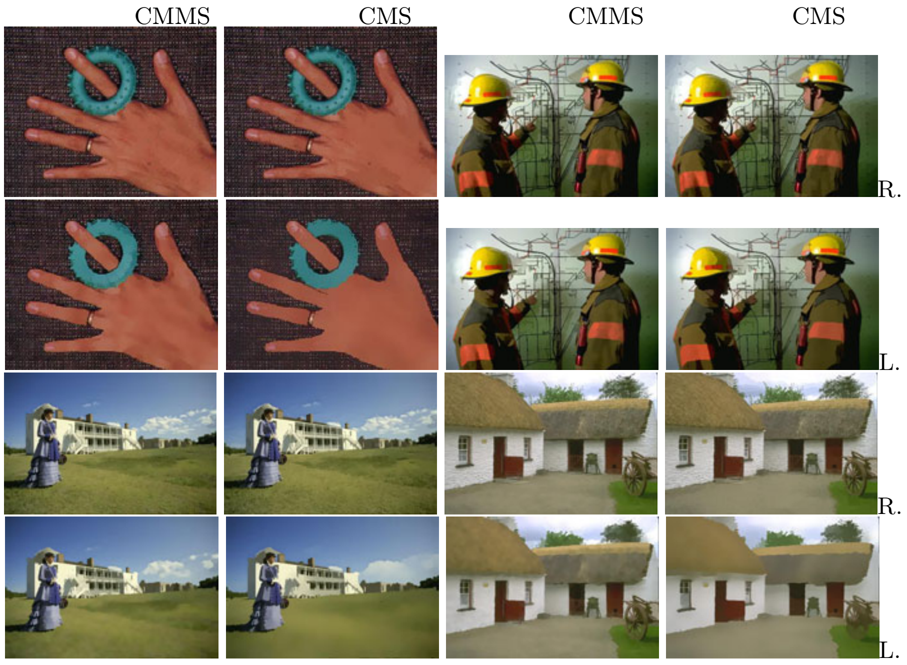


Fig. 4. Filtering in RGB vs Luv color space. We use $h_s = 5$ and $h_r = 5$. All images are processed with a Normal kernel. R, L stand for RGB and Luv respectively. Filtering in Luv makes smoother images. Moreover, CMS produces more uniform regions.

magnitude above which there is an edge (to be preserved). So for the "hand" image, a color range of $h_r = 19$ results in smoothing most of the texture of the background, while a value of $h_r = 10$ retains most of it (in RGB with a Normal kernel).

Overall CMS seems to produce more crisp boundaries between segments while creating more uniform regions within a segment (eg. it suppresses the skin color variation on the "hand" image). The former is particularly important for the segmentation step as we will see in Sec. 4.

3.2 RGB vs. Luv Color Space

In Fig. 4 we present the results when filtering on the RGB or Luv color space. In general, filtering in Luv produces smoother images.This is due to two facts; the Euclidean distance between two Luv values is perceptually meaningful, i.e., it is proportional to the distance of colors as perceived by a human observer, and the range of values for each component (L, u, v) is different (for example in our implementation $L \in [0 \ldots 100]$, $u \in [-100 \ldots 180]$, $v \in [-135 \ldots 110]$.), while each of the red, green and blue components have values from 0 to 255.

Overall, CMS smoothes the image more than CMMS, while preserving the boundaries better.

3.3 Filtering Speed Comparison

With the increasing demand for processing large volumes of data computational speed has become an important characteristic of any algorithm, that along with accuracy determines its usefulness. That is the reason why a number of approaches to speed up mean shift filtering have been proposed [6,8]. In this section we try to compare the speed of the two methods.

An objective comparison of the filtering speed of the different methods is not a simple task. Besides the implementation details that greatly affect the speed, there is also a number of algorithmic parameters that can significantly speedup or slow down the convergence of the optimization procedure. We start our comparison by evaluating the role of these parameters and then we discuss whether general speed up techniques that have been proposed in the literature can be applied to the different methods or not. For fairness sake, we use our own implementation of all the filtering methods that consists of Matlab files for the image handling and the general input/output interface, while the optimization code is written in C[2]. We perform all the experiments on a desktop computer with an Intel Core2 Quad CPU @3GHz[3].

Image Size. In theory the complexity of both CMS and CMMS increases linearly with the number of pixels (if the kernel is bounded), since each pixel represents a feature vector that needs to be processed[4]. The theoretical prediction is verified in practice as Fig. 5a shows.

Spatial Resolution h_s. Theoretically, both filtering methods depend quadratically on the spatial bandwidth. In practice, other parameters, explained below, make the dependence less than quadratic. Fig. 5b displays the filtering speed with respect to the spatial resolution for the methods, when all the other parameters are the same.

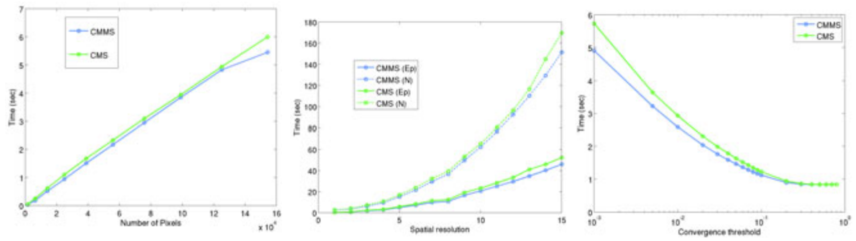
Epanechnikov vs. Normal kernel. For each pair of pixels, computation of the weight using the Epanechnikov kernel only requires a comparison, while the calculation of an exponential number is necessary for the case of the Normal kernel. As a result the former operation is much cheaper than the latter and thus filtering with an Epanechnikov kernel is faster compared to filtering with a Normal kernel as is shown in Fig. 5b.

The overall speed of the segmentation process is also affected by the quality of the result of the filtering process. We experimentally found, that a Normal kernel produced better results and as a consequence sped up the grouping step.

[2] All the code is available and can be downloaded from the author's website http://www.cs.umd.edu/~kbits/code.htm

[3] Due to Matlab's limitation only one core is used in the experiments.

[4] FHMS's complexity, on the other hand, is not linear with respect to the image size since whole areas can collapse into single points.



(a) Speed vs Image Size ($h_s = 5$) **(b)** Speed vs h_s **(c)** Speed vs Convergence threshold

Fig. 5. We use the "workers" image (size 321 × 481 pixels) and perform the filtering on the RGB color space with $h_r = 15$. A solid line denotes the use of Epanechnikov kernel while the dotted line (middle figure) the use of Normal kernel. We also limit the number of iterations to 20 and the convergence threshold is 0.001. We perform the filtering 5 times for each image size and only plot the median value.

The use of a Normal kernel still resulted in slower segmentation times, but the time difference was not as large as Fig. 5b shows.

Convergence Threshold. On each iteration of the optimization procedure each pixel vector is compared against its neighbors and shifted. If this shift is less than a predefined value (denoted convergence threshold) then we ignore that pixel in subsequent iterations of the optimization procedure. Intuitively the convergence threshold denotes how close to the "true" solution the optimization should reach before termination. Note that in CMMS the shift of each pixel is a monotonically decreasing function of the iteration number, while for CMS it is not. Fig. 5c displays the filtering speed with respect to the convergence threshold. The higher the threshold the faster the filtering. Especially for thresholds less than 0.1 the filtering time decreases almost exponentially.

Overall, from Fig. 5, CMS is $\backsim$ 10% slower than CMMS. A number of techniques can be used to perform the filtering faster. In the core of all filtering algorithms the pairwise distance between feature points needs to be computed. As suggested in [3] employing data structures and algorithms for multidimensional range searching can significantly improve the running time of all methods. In CMMS the trajectory of most feature points lay along the path of other feature points. Christoudias et al. [13] report a speed up of about five times when they "merge" the feature points together. This trick can directly be used in CMMS. A variation of the same concept could also be used to speed up CMS. The introduction of multicore CPUs and, especially, GPUs has provided a new way to improve the execution speed of algorithms through a parallel implementation. Both filtering algorithms are parallel in nature, so a careful implementation on a modern GPU is expected to run in real time for VGA or even larger sized images.

4 Segmentation Comparison

In a number of applications, like image denoising or deblurring, filtering is the final step. In most other applications filtering is an intermediate step followed by image segmentation. We are interested in the latter case. Thus, following the example of [3], we use the connected component grouping algorithm described in Fig. 2 to perform color-based segmentation. The simplicity of the grouping step allows for an objective evaluation of the filtering methods for the task of image segmentation. This algorithm has a single parameter, namely the grouping threshold t. In all our experiments $t = 0.5 * h_r$[5].

We use the Berkeley database of human segmentations [10] to evaluate the performance of the two methods. This is the biggest, publicly available database containing 200 color, training images and 1087 human created segmentations. We also present the results from the Weizmann Institute segmentation database [11], that consists of 100 grayscale images and 300 segmentations into foreground and background. Before presenting the results we need to describe the different measures that are used in the evaluation.

We use all the standard measures for the evaluation of the two algorithms, namely the Global Consistency Error (GCE) [10], the Variation of Information (VI) [14], the Probabilistic Rand index (PR) [15] and the average Boundary Displacement Error (BDE) [16][6]. From the previous measures for GCE, VI and BDE the lower the value the better the quality of the segmentation, while PR is a measure of similarity and as such a value of 0 indicates no similarity with the human created database, while a value of 1 indicates the highest similarity.

We create the following graphs by varying the color resolution h_r of the filtering methods. More specifically, we let h_r obtain values from 0.6 to 20 in increments of 0.3. We keep the remaining filtering parameters constant i.e., the maximum number of iterations for convergence is set to 20 and the convergence threshold to 0.1. For comparison we use the algorithm by Felzenswalb and Huttenlocher [18], denoted as GAT (Grouping with an Adaptive Threshold) on the figures. Again we vary the grouping threshold k ($k = [10 \dots 1500]$ in increments of 20).

We compute the comparison measures for each image of the database and further aggregate the results for the whole database using the median value[7]. These values are plotted on the Y-axis of each figure. On the X-axis we plot the average segment size, instead of the color resolution h_r. Thus all the plots below show the implicit curve of one comparison measure with respect to the average segment size.

[5] This is the same value for t that the EDISON system [13] uses. In practice, the threshold does not affect the resulting segmentation much, as long as it is larger than the convergence threshold of the optimization problem. In our experiments $t = 0.5 \gg 0.1 =$ convergence threshold.

[6] We use the code provided by J. Wright and A. Yang [17] to compute them.

[7] Since the comparison measures vary significantly for different images we choose the median value as opposed to the mean value because it is more robust to outliers.

4.1 Segmentation Results

First we present the collective segmentation results from the Berkeley database. We compare the two mean shift versions (CMMS and CMS) in two different color spaces (RGB and Luv) and using two different kernel functions (Epanechnikov and Normal kernel) for a total of $2 \times 2 \times 2 = 8$ combinations. That is why we display 8 curves on each graph of Fig. 6 plus a red curve for GAT.

Before analyzing the results any further we want to emphasize two facts. The results of the Global Consistency Error measure are misleading. As Martin et al. [10] mention, this measure only produces meaningful results when the number of segments in the computer segmentation is similar to the one in the human segmentation. In all other cases, i.e., when the number of computer generated segments is too high or too low GCE goes to zero. Indeed, as we observe in Fig. 6, all the curves for the GCE measure start from close to 0 (for very small average segment size) and asymptotically go to 0 (for very large average segment sizes). In between the two extremes, GCE values are larger, but since we display

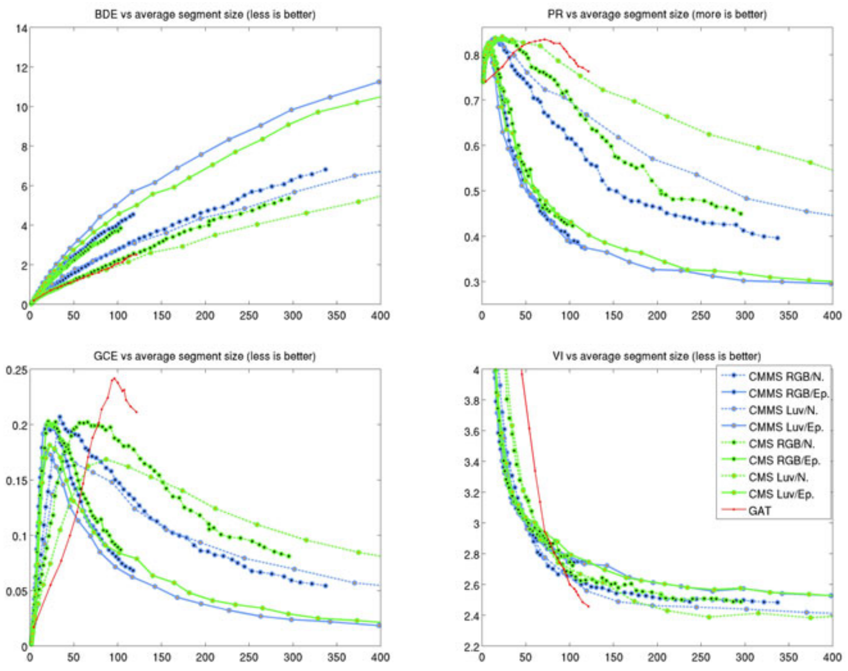


Fig. 6. Segmentation results for the Berkeley database. The solid and dash-dot lines represent the use of the Epanechnikov (Ep.) and Normal (N.) kernel, and the black and orange circle the use of the RGB and Luv color space, respectively. Note that the new method (CMS) is in *green*, while the existing method (CMMS) is in *blue*. From the top graphs it is clear that the green plots are better than the corresponding blue ones.

the average value for all the images it is impossible to determine the range of average segment sizes where GCE values are meaningful. The second fact is that the values of the Variation of Information measure for all the curves are really close together, making VI the least discriminative measure. On the other hand, both the Probabilistic Rand index and the average Boundary Displacement Error are discriminative enough to compare the different segmentation algorithms in this setting.

The segmentation results verify our earlier observations about the effect of the different kernels (Sec. 3.1) and color spaces (Sec. 3.2) on the amount of smoothing performed (for a given color resolution h_r). Filtering on the RGB color space results in less smoothing of the images and as a consequence in more image segments (and smaller average segment sizes). This is denoted by the close placement of the circles on the RGB plots compared to their Luv counterparts. The same observation, i.e., smaller average segment sizes, is valid for the Epanechnikov kernel function compared to the Normal kernel.

In the mean shift literature there are references that the Normal function produces better results than the Epanechnikov kernel [3], but so far a thorough analysis was not performed. According to the plots of Fig. 6 this prediction is absolutely right. The use of a Normal kernel produced better results in both measures (PR and BDE) and for both filtering methods (CMMS and CMS). Furthermore, the coupling of the Normal kernel with the Luv color space produced far superior results than all the other combinations.

Finally, the newly introduced variant of mean shift, i.e., Color Mean Shift, outperformed CMMS in all combinations of kernel functions and color spaces. Overall, CMS filtering on Luv color space with a Normal kernel produced the best results compared to all other methods. Compared to CMMS filtering on Luv color space with a Normal kernel (i.e., the next best algorithm) the new method produced on average $\backsim$ 17% better on the PR index and $\backsim$ 22% better on the BDE measure. Furthermore, this algorithm in most cases outperformed the current state of the art segmentation algorithm [18].

On Fig. 7 we present the segmentation results for the Weizmann dataset consisting of 100 images and 300 manual segmentations into foreground and background. Before analysing them we want to mention that this dataset is different from the previous one in the following aspects. All the images are grayscale and not color. Furthermore, the texture variation is significantly less than the one in the Berkeley database. The purpose of the dataset is to provide a testbed for segmentation into objects and as such only the single dominant object per image is marked as forground and the rest is background[8]. As a result many boundary edges are not reported in the manual segmentation. Both algorithms performed very well, with CMS performing better than CMMS on the BDE measure. In this database CMS performed slightly worse than GAT.

[8] The PR measure is misguiding in this dataset because of the existance of only two segments. Thus, a uniform segmentation of the whole image produces a result of ~ 0.97, i.e., very close to the maximum 1.

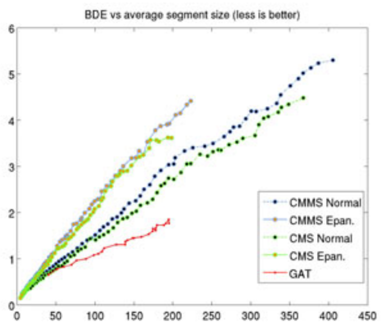


Fig. 7. Segmentation results for the Weizmann Institute database. The solid and dash-dot lines represent the use of the Epanechnikov (Epan.) and Normal kernel, respectively.

5 Conclusions

This paper presents the current variations of the mean shift algorithm from an optimization viewpoint and emphasizes the difference between Fukunaga's and Comaniciu's versions of the method, namely whether the pairwise comparison for moving each point is performed with the original image or with the filtered image of the previous iteration. A new variation of the mean shift algorithm, denoted Color Mean Shift, that lies between the existing two is also proposed. Extended experiments are presented both for the edge-preserving filtering and the segmentation tasks. In filtering, we mostly focus on the effect of different parameters on the speed of the filtering process. For segmentation, we use the Berkeley and the Weizmann Institute datasets to evaluate the performance of the algorithms using different kernel functions and color spaces. We conclude that Color Mean Shift performed on Luv color space using a Normal kernel function outperforms all other mean shift based algorithms for color images and is marginally better than current of the art segmentation algorithms. In the future we want to investigate how the methods perform when they are coupled with more sophisticated grouping techniques, such as [18].

Acknowledgements

The support of the EU under the Poeticon project (Cognitive Systems) is gratefully acknowledged.

References

1. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function with applications in pattern recognition. IEEE Trans. Information Theory 21, 32–40 (1975)
2. Cheng, Y.: Mean shift, mode seeking, and clustering. PAMI 17, 790–799 (1995)

3. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. on PAMI, 603–619 (2002)
4. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. PAMI 25, 564–577 (2003)
5. Wei, Y., Quan, L.: Region-based progressive stereo matching. In: CVPR, pp. 106–113 (2004)
6. Georgescu, B., Shimshoni, I., Meer, P.: Mean shift based clustering in high dimensions: A texture classification example. In: ICCV, pp. 456–463 (2003)
7. DeMenthon, D., Megret, R.: Spatio-temporal segmentation of video by hierarchical mean shift analysis. Technical report (2002)
8. Yang, C., Duraiswami, R., Gumerov, N., Davis, L.: Improved fast gauss transform and efficient kernel density estimation. In: ICCV, pp. 464–471 (2003)
9. Fashing, M., Tomasi, C.: Mean shift is a bound optimization. PAMI 27, 471–474 (2005)
10. 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: ICCV, vol. 2, pp. 416–423 (2001)
11. Alpert, S., Galun, M., Basri, R., Brandt, A.: Image segmentation by probabilistic bottom-up aggregation and cue integration. In: CVPR, pp. 1–8 (2007)
12. Rao, S., Martins, A., Principe, J.: Mean shift: An information theoretic perspective. Pattern Recognition Letters 30, 222–230 (2009)
13. Christoudias, C., Georgescu, B., Meer, P.: Synergism in low-level vision. ICPR 4, 150–155 (2002)
14. Meila, M.: Comparing clusterings: an axiomatic view. In: ICML, pp. 577–584 (2005)
15. Unnikrishnan, R., Pantofaru, C., Hebert, M.: A measure for objective evaluation of image segmentation algorithms. In: Workshop on Empirical Evaluation Methods in Computer Vision, CVPR (2005)
16. Freixenet, J., Munoz, X., Raba, D., Marti, J., Cuff, X.: Yet another survey on image segmentation: Region and boundary information integration. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 408–422. Springer, Heidelberg (2002)
17. Yang, A.Y., Wright, J., Ma, Y., Sastry, S.: Unsupervised segmentation of natural images via lossy data compression. Comput. Vis. Image Underst. 110, 212–225 (2008)
18. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. IJCV 59, 167–181 (2004)