Color based image segmentation as edge preserving filtering and grouping

Abstract—In this paper we contend that color based image segmentation can be performed in two stages: an edge preserving filtering stage followed by pixel grouping. Furthermore, for the first step, we introduce a framework under which many current filtering approaches (and some novel ones) can be classified. We present experiments where a new method, called Color Mean Shift, outperforms all the other methods in producing more uniform regions and preserving the edge boundaries. Then, applying standard clustering methods for the grouping step, we present extended experimental results for the Berkeley segmentation dataset of the combined filtering+grouping segmentation methods. We show that this new approach to image segmentation produces better segmentations compared to current state-of-the-art methods based on grouping only.

Index Terms—Image filtering, image segmentation, mean shift, bilateral filtering.

1 Introduction

We consider the problem of image segmentation, based only on the intensity values of an image. Color based segmentation is a fundamental and well studied problem in computer vision and many algorithms exist in the literature. Although many different methods have been suggested, all of them perform some kind of grouping of the pixels in the image based on some pairwise similarity criteria.

This paper does not argue against this common view of segmentation. On the contrary we believe that the grouping of pixels is a necessary step of the segmentation process. The main contention of the paper is that an edge preserving filtering step[1] should proceed the grouping step. As a result, we perceive segmentation as a two-step process; a *smoothing step* followed by a *grouping step*. Intuitively, the smoothing step attempts to bring closer intensities of neighboring pixels that belong to the same segment, while preserving (or even enhancing) the intensity difference across segment boundaries. The grouping step, on the other hand, makes the final decision whether two neighboring pixels belong to the same segment or not. We also argue that both steps are equally important, even though current methods only concentrate on one step of the process. As expected, their combination affects the final result.

1. In the rest of the paper we use the terms filtering and smoothing interchangably. In our view both terms indicate the process of smoothing the image while preserving the strong edges.

In the first part of the paper, we study a number of smoothing techniques; the original mean shift [1] and its modified version[2], [3][2], bilateral filtering [4],[5], local mode filtering [6] and anisotropic diffusion [7]. We present all the above techniques as variations of a general optimization problem. Using such a formulation the similarities and differences between them are made clear. This framework also provides a natural way to classify them using two criteria. Using the classification criteria we propose three novel methods. Two of them (color mean shift and spatial mean shift) are variations of the mean shift filtering and the third one is an extension of bilateral filtering. Filtering experiments show that color mean shift actually outperforms the other filtering methods in smoothing the images, while preserving the edges.

In the second part of the paper, we present segmentation methods by combining the previously described filtering methods with four grouping methods. We perform a number of experiments using the Berkeley [8] and Weizmann Institute [9] datasets and answer the following critical questions; Is the filtering step importart for the segmentation and does filtering in different color spaces and using different kernel functions matter?

1.1 Related Work

Our work on filtering methods is motivated by the mean shift algorithm so first we present related work on mean shift. Following the success of Comaniciu and Meer's version of mean shift [3] the same basic algorithm for non parametric clustering has been used for object tracking [10], 3D reconstruction [11], image filtering [3], texture classification [12] and video segmentation [13] 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 [14] for Normal ker-

2. In the recent papers, the original "mean shift" approach is called "blurring mean shift". We use a different name for the mean shift variant used in computer vision, namely "mode finding". So in the rest of this chapter the term **Mode Finding** refers to **Comaniciu and Meer's version of mean shift** and is abbrievated as **CMMS**.

nels and the Locality Sensitive Hashing based mean shift [12].

Cheng [2] was the first to recognize the equivalence of mean shift to a step-varying gradient ascent optimization problem, and much later Fashing and Tomashi [15] showed that it is equivalent to Newton's method with piecewise constant kernels, and is a quadratic bound maximization for all other kernels. Yuan and Li [16] prove that mean shift is a half quadratic optimization for density mode detection when the profiles of the kernel functions are convex. Finally, Carreira-Perpinan [17] proves that it is equivalent to an EM algorithm when the kernel is the Normal function.

Concerning the grouping methods, we use three algorithms: the grouping method used by Comaniciu and Meer [3] (in $3D$ and $5D$) and the method of Felzenszwalb and Huttenlocher[19]. We have chosen the first two because they are directly related to the filtering methods. The fourth method while not directly related to filtering is still a fast, local method that is considered the state of the art in color based segmentation. A number of other grouping (i.e., segmentation) algorithms exist in the literature; energy minimization [20], spectral clustering [21], [22], algebraic multigrid [23] based methods to name a few. The reason why we did not include them in our comparison was because they were either a) slow and/or b) hard to parameterize and/or c) difficult to implement. Nevertheless, we believe that the grouping methods we used were sufficient to prove our points.

1.2 Paper organization

This paper is organized as follows. After a short section describing the notation and some necessary mathematical prerequisites we proceed to describe the framework for the filtering algorithms. Then, we present the criteria used to classify the methods as well as the individual methods themselves. We conclude the first part of the paper with a number of experiments applying the methods to different images. The second part, begins with the introduction of the grouping methods we used and the measures to quantify the segmentation quality. Then we proceed with the experiments. We conclude this paper with the final conclusions and future work.

2 Notational Preliminaries

We represent the color image as a mapping $\mathbf{S}$ from the 2D space of the pixel coordinates to the 3D space of the intensity values (for color images). $\mathbf{x}_i$ is a 2D vector representing the spatial coordinates of pixel $i (i = 1 \dots N)$ and $\mathbf{S}(\mathbf{x}_i)$ is a vector that represents the three color channels. To simplify the notation we denote the intensities for a pixel $\mathbf{x}_i$ with a subscript, so $\mathbf{S}(\mathbf{x}_i) = \mathbf{S}_i$. We also denote the set of all pixels as $\mathtt{X}$ and the whole image $S(\mathtt{X})$. The cardinality of $\mathtt{X}$ is N.

In the following sections we use *bold letters* to represent *vectors* and the notation $[\mathbf{x}_i, \mathbf{S}_i]^T$ to indicate a concatenation of vectors. When we want to indicate the evolution of a vector over time we use superscripts, e.g. $[\mathbf{x}_i^0, \mathbf{S}_i^0]$ indicates the initial values of pixel $\mathbf{x}_i$ having intensity $\mathbf{S}_i$.

2.1 Kernel Functions

Definition(Kernel Function): Let $\mathbb{X}$ be a d-dimensional Euclidean space and $\mathbf{x} \in \mathbb{X}$. We denote with x_i the i^{th} component of $\mathbf{x}$. The L_2 norm of $\mathbf{x}$ is a non-negative number $||\mathbf{x}||$ such that $||\mathbf{x}||^2 = \sum_{i=1}^{d} x_i^2$. A function $K : X \rightarrow R$ is a kernel if and only if there exists another function $k : [0 \cdots + \infty] \rightarrow R$ such that

$$K(\mathbf{x}) = k(||\mathbf{x}||^2) \tag{1}$$

and

1) k is non negative
2) k is non increasing i.e.,

$$k(a) \geq k(b),\ if\ a < b \tag{2}$$

3) k is piecewise continuous and

$$\int_0^{+\infty} k(a)da < +\infty \tag{3}$$

Function $k(x)$ is called the *profile* of the kernel $K(\mathbf{x})$.

Often the kernel function is normalized i.e.,

$$\int_X K(\mathbf{x})d\mathbf{x} = 1. \tag{4}$$

Even though kernel functions are mostly used for kernel density estimation, we use them in order to define optimization problems that we subsequently solve using standard gradient descent methods. Thus, we are not only interested in the kernel function $K(\mathbf{x})$ but also on its partial derivatives $\frac{\partial K(\mathbf{x})}{\partial \mathbf{x}}$. Next we define two kernel functions that we use; the Epanechnikov and the Gaussian kernel.

2.1.1 Epanechnikov kernel

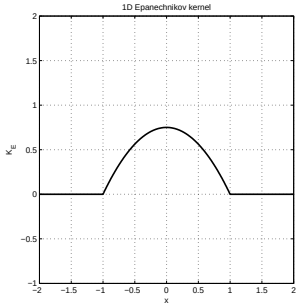
The Epanechnikov kernel [24] has the analytic form

$$K_E(\mathbf{x}) = \begin{cases} c_E(1 - \mathbf{x}^T\mathbf{x}) & \mathbf{x}^T\mathbf{x} \leq 1 \\ 0 & otherwise \end{cases} \tag{5}$$

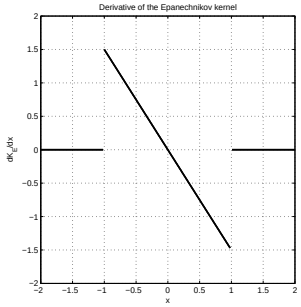
where $c_E = \frac{d+2}{2\pi^{d/2}}\Gamma(\frac{d+2}{2})$ is the normalization constant. Fig. 1(a) presents this kernel in the $1-D$ case. The partial derivative of $K_E(\mathbf{x})$ with respect to element x_i of vector $\mathbf{x}$ is

$$\frac{\partial K_E(\mathbf{x})}{\partial x_i} = \begin{cases} -2 \cdot c_E \cdot x_i & -1 < x_i < 1 \\ 0 & |x_i| > 1 \end{cases} \tag{6}$$

and is depicted in Fig. 1(b).

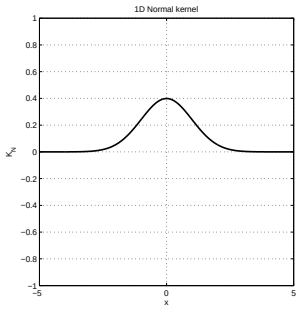


(a) 1-D Epanechnikov Kernel

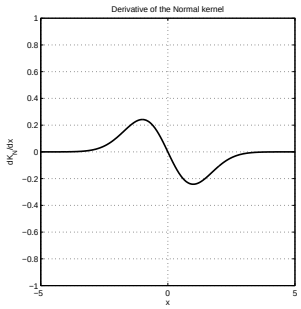


(b) Derivative of 1-D Epanechnikov Kernel

Figure 1: $1-D$ Epanechnikov kernel.



(a) 1-D normal kernel



(b) Derivative of 1-D normal kernel

Figure 2: $1-D$ Normal kernel.

2.1.2 Multivariate Normal (Gaussian) kernel

The multivariate Normal kernel with variance 1 has the analytic form

$$K_N(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}). \quad (7)$$

In Fig. 2(a) a $1-D$ Normal kernel is displayed.

The partial derivative of $K_E(\mathbf{x})$ with respect to element x_i of vector $\mathbf{x}$ is

$$\frac{\partial K_N(\mathbf{x})}{\partial x_i} = -x_i \cdot (2\pi)^{-\frac{d}{2}} exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}) = -x_i \cdot K_N(\mathbf{x}) \quad (8)$$

and is depicted in Fig. 2(b).

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

3 Edge Preserving Filtering

3.1 A taxonomy of filtering methods

In Fig. 3 we present our scheme for classifying the various edge preserving filtering methods. The figure can be split into two parts. On the left part optimization based methods are shown, while on the right part filtering methods. The only difference between filtering and optimization methods is that the former methods perform a single iteration of the corresponding optimization problem. The three new methods are spatial mean shift, color mean shift and joined bilateral filtering.

3.2 Classification criteria

Careful examination of the previous defined optimization problems reveal that there are only two differences in their objective functions; the presence of $[\mathbf{x}_i, \mathbf{S}_i]$ or $[\mathbf{S}_i]$ as the optimization argument; and the comparison against the points in the original image $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ or the points on the previous iteration $[\mathbf{x}_j, \mathbf{S}_j]$. Finally two of the methods (bilateral filtering and joined bilateral filtering) are an one-iteration methods, while all the other methods perform multiple iterations till convergence. Next we explain in details these differences.

3.2.1 $\arg\min_{[\mathbf{x}_i, \mathbf{S}_i]}$ *vs* $\arg\min_{\mathbf{S}_i}$

In the first case the optimization problem is defined over the joint spatial and range domain $(5-D)$, i.e. both the position of the pixels as well as their intensities change in each iteration. In the second case, where the optimization is over the range domain $(3-D)$, only the intensities of the pixels change while their position remain the same. This is not to be confused with the use of $[\mathbf{x}_i, \mathbf{S}_i]$ in the objective function. While the position of the pixel is always considered in the computation of the objective function, that position might change or not (depending on the method).

At this point we should also make clear that the optimization is defined for the whole image, that is the values of all the pixels change. For the sake of simplicity we do not make this explicit when we write down the optimization equation.

3.2.2 $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ *vs* $[\mathbf{x}_j, \mathbf{S}_j]$

With a subscript we denote the value of the pixels at a specific iteration, so $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ is the value of pixel $\mathbf{x}_j$ at the very beginning, i.e. in the original image. The lack of a superscript denotes the current value of pixels, i.e. the value of the pixel at a previous iteration. Two pairs of algorithms (mean shift/mode finding and local mode filtering/anisotropic diffusion) only differ in whether we compare the current value of a pixel against the original image or the image obtained in the previous iteration. As we will demonstrate in the experiments, the results vary significantly because of that (also see [25] for a theoretical analysis and justification).

Furthermore, there are two valid hybrid combinations that have not been proposed before.

- $[\mathbf{x}_j^0, \mathbf{S}_j]$: In this case the comparison is performed against the original position of the pixels and the previously computed range image.
- $[\mathbf{x}_j, \mathbf{S}_j^0]$: In this case the position of the pixels in the previous iteration is used along with their original intensity values.

Apparently the previous cases only make a difference when the optimization is defined over the joint spatial/range domain. Otherwise the position of the pixels never changes, thus $[x_j] \equiv [x_j^0]$.

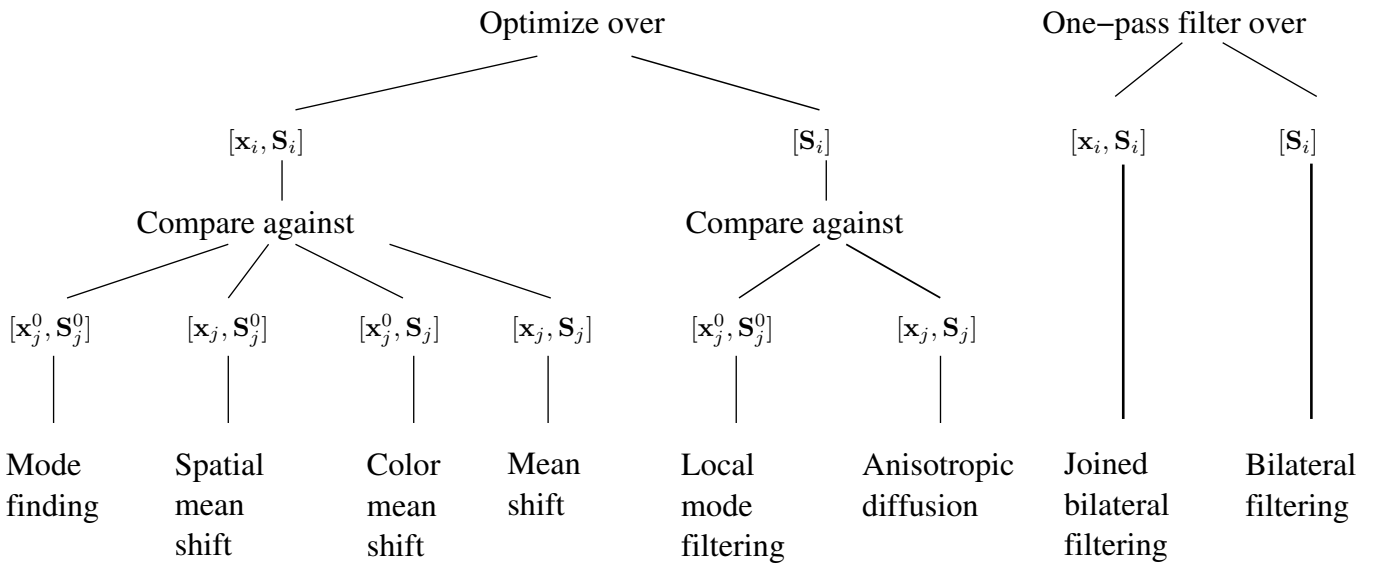


Figure 3: Classification of various filtering methods.

3.3 Filtering methods

In the following subsections we define a number of image filtering techniques as optimization problems. In previous formulations these methods were defined as the result of applying an algorithm to an image. Using our formulation we aim to achieve two goals; to simplify the methods (since we only need a single equation to describe it) and to describe all the methods in a uniform way. Note that some methods (i.e. mean shift and mode finding) are defined for any kernel function, while others (i.e., bilateral filtering, local mode filtering and anisotropic diffusion) are only defined with respect to the Normal kernel $K_N(\mathbf{x})$.

3.3.1 Mean Shift (MS)

The original mean shift formulation [1] (applied to a color image) treats the image as a set of $5-D$ points (i.e., 2 dimensions for the spatial coordinates and 3 dimensions for the color values). 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

$$\arg\min_{[\mathbf{x}_i,\mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i,\mathbf{S}_i]-[\mathbf{x}_j,\mathbf{S}_j]), \tag{9}$$

where $\sum_{i,j}$ defines the summation over all pairs of pixels in the image. Note that this problem has a global maximum 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.

3.3.2 Mode Finding or Comaniciu/Meer Mean Shift (CMMS)

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

$$\arg\min_{[\mathbf{x}_i,\mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i,\mathbf{S}_i]-[\mathbf{x}_j^0,\mathbf{S}_j^0]) \tag{10}$$

There is a subtle difference between mode finding and mean shift, that significantly affects the performance. In the former formulation each current 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 $[\mathbf{x}_j, \mathbf{S}_j]$. In a recent paper [25] S. Rao et al. study those two variations from an information theoretic perspective and conclude that mean shift is not stable and hence should not be used for clustering.

Fig. **4** presents the results of both methods in a smoothly varying intensity image. Notice that the gradient of the kernel function is zero everywhere but in the boundaries. Thus, mode finding filtering only changes the intensity on the boundaries (that change is not very visible in Fig. 4). Mean shift, on the other hand, produces artificial segments of uniform intensity. Intuitively, each iteration of the process results in more clustered data which in turn results in better clustering results for the next iteration. On the downside, a fast mean shift implementation is challenging due to the fact

that the feature points and the comparison points do not lie on a regular spatial grid anymore. Thus in a naive implementation one would have to compare the current feature $[\mathbf{x}_i, \mathbf{S}_i]$ against all the remaining feature points.

3.3.3 Spatial Mean-Shift (SMS)

One of our proposed methods that lies between mean shift and mode finding, spatial mean shift performs mean shift in the spatial dimensions and mode finding in the color dimensions. SMS can be viewed as the gradient descent solution of the optimization problem

$$\arg \min_{[\mathbf{x}_i, \mathbf{S}_i]} - \sum_{i,j} K([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j^0]). \tag{11}$$

Spatial mean shift suffers from the same computational problems as mean shift, so it is mentioned here for the sake of completeness. We exclude the results of both mean shift and spatial mean shift in our filtering and segmentation experiments.

3.3.4 Color Mean-Shift (CMS)

Color mean shift is our proposed method that alleviates the computational problem of mean shift by using the original spatial location of the points for comparison, while it uses the updated intensity values of the previous iteration for improved clustering ability. In a sense, mean shift is performed on the color dimensions and mode finding 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

$$\arg \min_{[\mathbf{x}_i, \mathbf{S}_i]} - \sum_{i,j} K([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j]). \tag{12}$$

3.3.5 Local Mode Filtering (LMF)

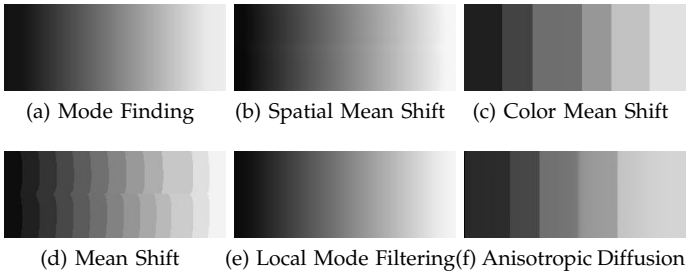
Local mode filtering [6] was introduced as a method to find the local mode in the range domain of each pixel of the image. A generalization of the spatial Gaussian filtering to a spatial and range Gaussian filter is used to iterate to the local mode (on the $3-D$ color domain). On each iteration the intensity of each pixel is replaced by a weighted average of its neighbors. From an optimization point of view the problem can be expressed as

$$\arg \min_{\mathbf{S}_i} - \sum_{i,j} K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0]). \tag{13}$$

3.3.6 Bilateral Filtering (BF)

In bilateral filtering [4],[5] the intensity of each pixel is replaced by a weighted average of its neighbors. The weight assigned to each neighbor decreases with both the distance in the image plane (spatial domain) and the distance on the intensity axes (range domain). Formally the intensity at each pixel $\mathbf{S}_i$ takes the value

$$\mathbf{S}_i = \frac{\sum_j \mathbf{S}_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}. \tag{14}$$



(a) Mode Finding (b) Spatial Mean Shift (c) Color Mean Shift

(d) Mean Shift (e) Local Mode Filtering (f) Anisotropic Diffusion

Figure 4: All the described algorithms applied on a 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.

Bilateral filtering can be considered *as the first iteration of local mode filtering with a specific step size* (Sec. 3.4).

3.3.7 Joined Bilateral filtering

In this variation of the bilateral filtering both the intensity and position of each pixel is replaced by a weighted average of its neighbors. Formally, the new coordinates and color of each pixel are

$$[\mathbf{x}_i, \mathbf{S}_i] = \frac{\sum_j [\mathbf{x}_i, \mathbf{S}_i] K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}. \tag{15}$$

Analogous to bilateral filtering this method can be considered as the first iteration of mode finding with a specific step size.

3.3.8 Anisotropic Diffusion (AD)

Anisotropic diffusion is a non-linear process introduced by Perona and Malik [7] for edge preserving smoothing. In the original formulation a diffusion process with a monotonically decreasing diffusion function of the image gradient magnitude is used to smooth the image while preserving strong edges. Since then other functions have been proposed and the equivalence of this technique to robust statistics has been established [26]. In [6] the connection with local mode filtering was also made. Here we provide an alternative view of the diffusion process as an optimization problem

$$\arg \min_{\mathbf{S}_i} - \sum_{i,j} K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j]). \tag{16}$$

The difference between this method and local mode filtering is analogous to the difference between the original mean shift and mode finding. Namely in local mode filtering the current point is compared against the original image pixels $[\mathbf{x}_j^0, \mathbf{S}_j^0]$, while in anisotropic diffusion the comparison is against the intensity value of the pixels in the previous iteration $[\mathbf{x}_j, \mathbf{S}_j]$.

Color Mean Shift (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 $[\mathbf{x}_i, \mathbf{S}_i]$

$$[\mathbf{x}_i, \mathbf{S}_i] \leftarrow \frac{\sum_j [\mathbf{x}_j, \mathbf{S}_j] g(||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j]||^2)}{\sum_j g(||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j]||^2)}$$

Mode Finding (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

$$[\mathbf{x}_i, \mathbf{S}_i] \leftarrow \frac{\sum_j [\mathbf{x}_j, \mathbf{S}_j] g(||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0]||^2)}{\sum_j g(||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0]||^2)}$$

Figure 5: The algorithms that we use in the experiments. Note that $g(x) = [x \leq 1]$ (indicator function in Iverson notation) for the Epanechnikov kernel and $g(x) = \exp(-x/2)$ for the Normal kernel. Local mode filtering is performed in a similar way as mode finding and mean shift, anisotropic diffusion are performed in a similar way as color mean shift.

3.4 Optimization steps sizes

From the above optimization problems *mean shift, spatial mean shift, color mean shift and anisotropic diffusion are joint optimization problems* i.e., the whole image needs to be optimized simultaneously. In mode finding and local mode filtering, on the other hand, each pixel can be optimized independently from the rest of the image. Next we present two claims concerning the step size of these optimization problems.

Claim 1: Local mode filtering (and mode finding with a Gaussian kernel) can be considered as gradient descend methods for solving the corresponding optimization problem (Eqs. 13 and 10 respectively) with a step size at iteration t of

$$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])}. \tag{17}$$

Claim 2: Mode finding with an Epanechnikov kernel can be considered as a gradient descend method for solving the corresponding optimization problem (Eq. 10) with a step size at iteration t of

$$\gamma_i^t = -\frac{1}{2c_E \sum_{j,||[\mathbf{x}_i^t, \mathbf{S}_i^t] - [\mathbf{x}_j^0, \mathbf{S}_j^0]||<1} 1} \tag{18}$$

As a consequence the result after one iteration of the gradient descent is

$$[\mathbf{x}_i^{t+1}, \mathbf{S}_i^{t+1}] = \frac{\sum_{j,||[\mathbf{x}_i^t, \mathbf{S}_i^t] - [\mathbf{x}_j^0, \mathbf{S}_j^0]||<1} [\mathbf{x}_j^0, \mathbf{S}_j^0]}{\sum_{j,||[\mathbf{x}_i^t, \mathbf{S}_i^t] - [\mathbf{x}_j^0, \mathbf{S}_j^0]||<1} 1}. \tag{19}$$

In the Appendix we provide the proof of the first claim along with a table (Table 4) that summarizes the optimization step sizes for each method along with the results after one iteration. Note that in the case of mean shift and anisotropic diffusion we are using the *block gradient descent* method and optimize one pixel vector at a time[3].

4 Filtering Experiments

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_c) resolution. 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 optimization equations for simplicity reasons). Then, we perform the optimization; one pixel at a time in the case of mode finding (Fig. 5, top right), or one iteration of the whole feature set at a time in the mean shift and color mean shift cases (Fig. 5, top left). Fig. 6 displays the original images that we use for all the experiments in the rest of the section.

4.1 Epanechnikov vs Normal Kernel

First we present some filtering results when using different kernels; namely the Epanechnikov and Normal kernel (Figs. 7,8). Each column of the figures depicts the filtering result with a different algorithm; CMMS, LMF, CMS and AD stand for mode filtering, local mode filtering, color mean shift and anisotropic diffusion respectively. In all cases the Normal kernel produces smoother results, while preserving edge discontinuities. As a matter of fact the color resolution h_r is the one that defines the gradient 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 on the background, while a value of $h_r = 10$ retains most the texture (in RGB color space with a Normal kernel).

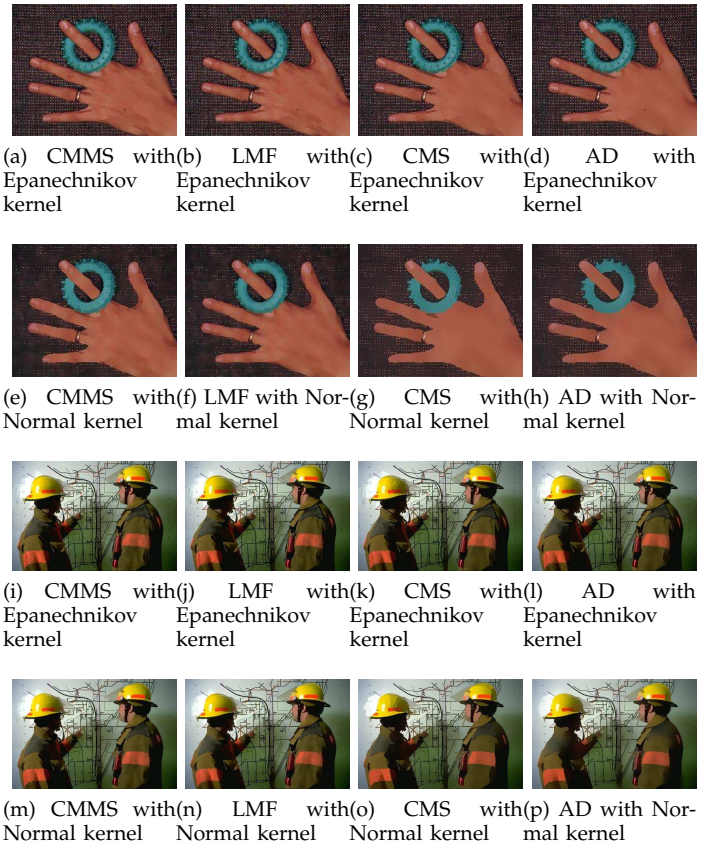
In all the images mode finding and local mode filtering produced very similar results. Furthermore color mean

3. We use the symbols $\mathbf{x}_j$, $\mathbf{S}_j$ to denote the current value of pixel p_j. These might be the values of pixel p_j at iteration t or $t+1$ depending on whether p_j is processed after or before p_i.

(a) Hand (b) Workers

(c) Woman (d) Houses

Figure 6: The original images we use for the filtering experiments. The first image is taken from Comaniciu and Meer's mean shift segmentation paper, while the remaining are training images of the Berkeley segmentation database collection. Their sizes are 303×243 and 481×321 pixels respectively.



(a) CMMS with Epanechnikov kernel (b) LMF with Epanechnikov kernel (c) CMS with Epanechnikov kernel (d) AD with Epanechnikov kernel

(e) CMMS with Normal kernel (f) LMF with Normal kernel (g) CMS with Normal kernel (h) AD with Normal kernel

(i) CMMS with Epanechnikov kernel (j) LMF with Epanechnikov kernel (k) CMS with Epanechnikov kernel (l) AD with Epanechnikov kernel

(m) CMMS with Normal kernel (n) LMF with Normal kernel (o) CMS with Normal kernel (p) AD with Normal kernel

Figure 7: Epanechnikov vs Normal kernel experiment. We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 19$. All the images are processed in RGB color space.

shift and anisotropic diffusion gave similar results. Color mean shift seems to produce more crisp edges while anisotropic diffusion smooths some of the edges. Overall, color mean shift and anisotropic diffusion produce more uniform regions (e.g. suppresses the skin color variation on the "hand" image) and more crisp boundaries between segments compared to mode finding and local mode filtering. The latter is particularly important for the segmentation step. We further investigate this phenomenon in subsection 4.3.

For the remaining filtering experiments we use a Normal kernel.

4.2 RGB vs Luv Color Space

In Figs. 9, 10 we present the results when filtering in the RGB and Luv color space. In general, filtering in Luv color space 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 the colors as perceived by a human observer. This is not true in RGB, where very similar colors might be located far away and the opposite. Furthermore the range of values for each component (L, u, v) is different (for example in our implementation $L \in [0 \dots 100]$, $u \in [-100 \dots 180]$, $v \in [-135 \dots 110]$.), while each of the Red, Green and Blue components have values from 0 to 255.

In these experiments, mode finding and local mode filtering seem to produce almost identical images, while color mean shift preserves the boundaries better than anisotropic diffusion. Both latter methods smooth the image considerably more than the former ones.

4.3 Color uniformity of regions after filtering

Next we compare the ability of the filtering algorithms to suppress texture and produce uniform regions. One issue is how to measure the color uniformity of regions. Here, we use the zero order (i.e., color histograms) and first order (i.e., gradient histograms) statistics to measure the intensity variation in an filtered image. Then, we compute the entropy of the two histograms. The entropy definition[4] measures how "random" an image is. Thus, an image created by sampling each pixel's color value from a uniform random distribution is expected to have a large entropy value, while a single uniform color image has an entropy of 0. In general lower entropy values indicate more uniform colored images, i.e. images with less number of segments of more uniform color.

In Table 1 we display the entropy measures for each method with the different kernels and color spaces (and

4. If X is a discrete random variable with possible values $\{x_1, \dots, x_n\}$ then the entropy is defined as $H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$, where b is the base of the logarithm (in our case we use $b = 2$).

Table 1: Entropy measures for the color and gradient histograms for the four images after performing the filtering with different methods and different kernels in the two color spaces. The first number is the entropy for the color and the second for the gradient histogram. The lower the values the smaller the variation.

Hand Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	**6.14**, 12.97	**6.14**, 12.97	**6.14**, 12.97	**6.14**, 12.97
Epanechnikov, Luv	7.02, 12.91	7.02, 12.91	7.42, 12.82	7.50, 12.83
Normal, RGB	7.15, 12.68	7.32, 12.59	8.91, 11.89	9.32, 11.94
Normal, Luv	10.47, 10.85	11.20, 11.02	9.84, **8.87**	10.93, 9.16

Workers Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	13.95, 9.59	14.64, 9.59	12.34, 9.21	13.31, 9.35
Epanechnikov, Luv	13.72, 8.78	14.70, 8.75	12.51, 8.16	13.59, 8.21
Normal, RGB	12.46, 8.47	14.16, 8.48	**10.82**, 7.85	12.61, 8.14
Normal, Luv	12.74, 7.05	14.31, 7.16	11.80, **6.17**	13.16, 6.28

Woman Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	14.25, 8.49	14.58, 8.43	13.12, 8.43	13.79, 8.39
Epanechnikov, Luv	13.67, 7.30	14.37, 7.15	12.37, 6.13	13.24, 6.07
Normal, RGB	13.26, 7.72	14.16, 7.41	**11.58**, 7.51	12.81, 7.35
Normal, Luv	13.08, 5.18	13.92, 5.11	12.07, **4.23**	12.86, 4.30

Houses Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	14.27, 9.12	14.59, 9.07	13.07, 9.04	13.70, 8.98
Epanechnikov, Luv	13.39, 7.75	14.17, 7.60	11.71, 6.29	12.78, 6.46
Normal, RGB	13.05, 8.53	14.10, 8.22	**10.94**, 8.08	12.53, 8.12
Normal, Luv	12.72, 5.57	13.62, 5.67	11.48, **4.36**	12.56, 4.71

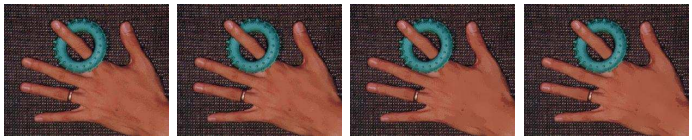
(a) CMMS with Epanechnikov kernel (b) LMF with Epanechnikov kernel (c) CMS with Epanechnikov kernel (d) AD with Epanechnikov kernel

(e) CMMS with Normal kernel (f) LMF with Normal kernel (g) CMS with Normal kernel (h) AD with Normal kernel

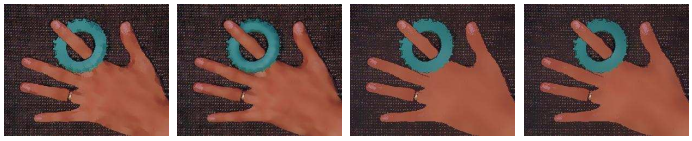
(i) CMMS with Epanechnikov kernel (j) LMF with Epanechnikov kernel (k) CMS with Epanechnikov kernel (l) AD with Epanechnikov kernel

(m) CMMS with Normal kernel (n) LMF with Normal kernel (o) CMS with Normal kernel (p) AD with Normal kernel

Figure 8: Epanechnikov vs Normal kernel experiment. We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 19$. All the images are processed in RGB color space.



(a) CMMS on RGB color space (b) LMF on RGB color space (c) CMS on RGB color space (d) AD on RGB color space

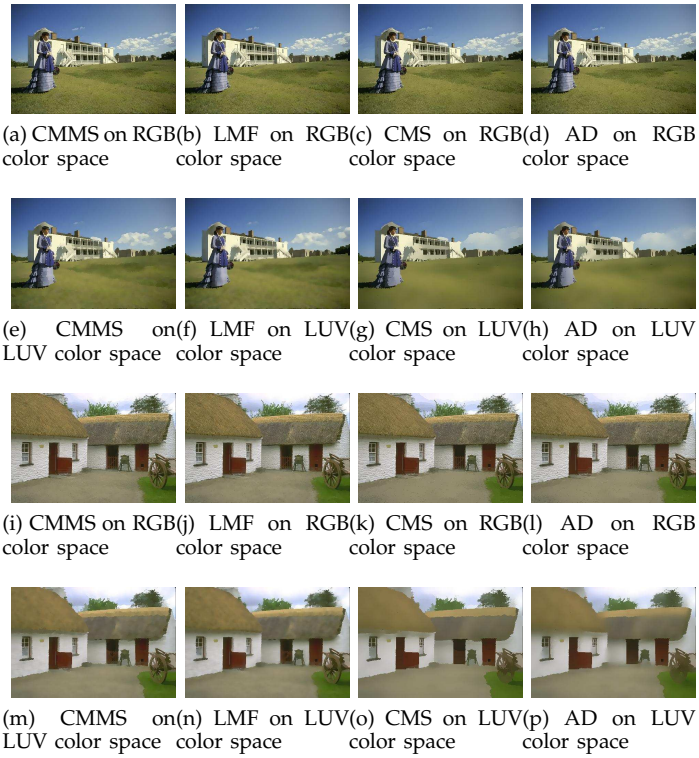


(e) CMMS on LUV color space (f) LMF on LUV color space (g) CMS on LUV color space (h) AD on LUV color space

(i) CMMS on RGB color space (j) LMF on RGB color space (k) CMS on RGB color space (l) AD on RGB color space

(m) CMMS on LUV color space (n) LMF on LUV color space (o) CMS on LUV color space (p) AD on LUV color space

Figure 9: RGB vs Luv color space experiments (1/2). We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 5$. All the images are processed with a Normal kernel.



(a) CMMS on RGB color space (b) LMF on RGB color space (c) CMS on RGB color space (d) AD on RGB color space

(e) CMMS on LUV color space (f) LMF on LUV color space (g) CMS on LUV color space (h) AD on LUV color space

(i) CMMS on RGB color space (j) LMF on RGB color space (k) CMS on RGB color space (l) AD on RGB color space

(m) CMMS on LUV color space (n) LMF on LUV color space (o) CMS on LUV color space (p) AD on LUV color space

Figure 10: RGB vs Luv color space experiments (2/2). We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 5$. All the images are processed with a Normal kernel.

constant spatial and color resolutions $h_s = 5$, $h_r = 5$). From the results of Table 1 we observe that Color Mean Shift with Normal kernel gives the smallest entropy values for all the images, hence producing the most uniform regions. Anisotropic diffusion follows, while Mode finding and local mode filtering produce very similar results. A natural question to ask is whether the above results are due to *over smoothing*. From the sample filtering results presented above this does not seem to be the case. The only way to verify that though is to perform the segmentation and then compare the results against human segmented images. In Sec. 8 we present these experiments. As we discuss there the segmentation results for color mean shift are better than the ones for the other filtering methods, thus we can safely conclude that *color mean shift produces more uniform regions without over smoothing the original image*.

4.4 Filtering speed comparison

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. We perform all the experiments on a desktop computer with an Intel Core2 Quad CPU @$3GHz$[5].

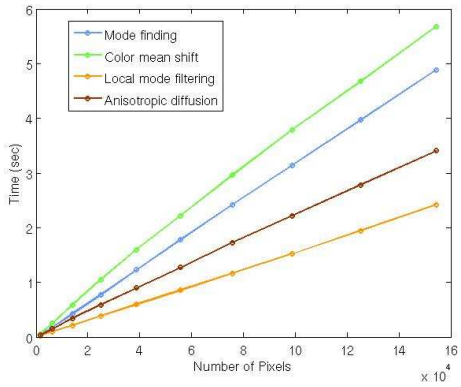


Figure 11: The filtering speed as a function of the image size (i.e., number of pixels) for all four methods. We use the "workers" image (whose original size is 321×481 pixels) and perform the filtering on the RGB color space with an Epanechnikov kernel with spatial and color resolutions $h_s = 5, h_r = 15$ respectively. 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.

4.4.1 *Image size*

The number of pixels directly affect the filtering speed. In theory, the complexity of the algorithm increases linearly with the number of pixels, since each pixel represents a feature vector that needs to be processed. The theoretical prediction is verified in practice as Fig. 11 shows.

4.4.2 *Spatial resolution (h_s)*

Theoretically, all the filtering methods (but Mean Shift and Spatial Mean Shift) depend quadratically on the spatial bandwidth. In practice, other parameters, explained below, make the dependence less than quadratic. Fig. 12 displays the filtering speed with respect to the spatial resolution for the methods, when all the other parameters are the same.

4.4.3 *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

5. Due to Matlab's limitation only one core is used in the experiments.

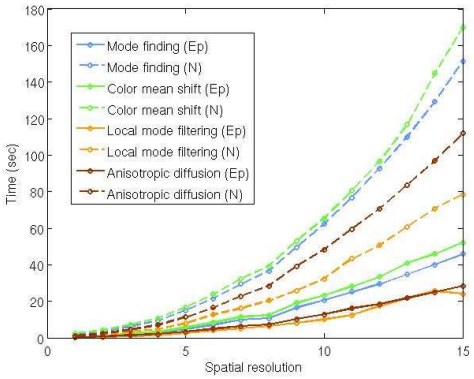


Figure 12: The filtering speed as a function of the spatial resolution (h_s) for all four methods. We use the "workers" image (321×481 pixels) and perform the filtering on the RGB color space with an Epanechnikov kernel (continuous line) or Normal kernel (dotted line). We also limit the number of iterations to 20 and stop the optimization for pixels that move less than 0.001 between two iterations. We perform the filtering 5 times for each value of h_s and only plot the median value.

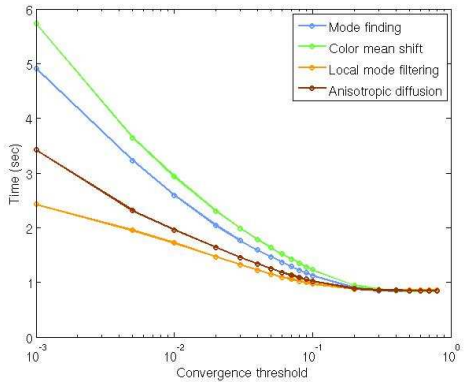


Figure 13: The filtering speed as a function of the convergence threshold for all four methods. We use the "workers" image (321×481 pixels) and perform the filtering on the RGB color space with an Epanechnikov kernel with spatial and color resolution $h_s = 5, h_r = 15$ respectively. We also limit the number of iterations to 50. We perform the filtering 5 times for each value of the convergence threshold and only plot the median value. Notice that the X-axis is on logarithmic scale.

and thus filtering with an Epanechnikov kernel is faster compared to filtering with a Normal kernel as is shown in Fig. 12. Other researchers (e.g. **[27]**) have proposed the use of lookup tables to approximately compute the exponents much faster.

At this point we should note that the overall speed of the segmentation process is also affected by the quality of the result of the filtering process. We experimentally found, that using a normal kernel produced better results and as a consequence sped up the grouping step. Overall the use of a Normal kernel still resulted in slower segmentation times, but the time difference was not as large as Fig. 12 shows.

4.4.4 Convergence threshold

As described above, 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. At this point we would like to emphasize that for the mode finding and the local mode filtering methods the shift of each pixel is a monotonically decreasing function of the iteration number, while for color mean shift and anisotropic diffusion it is not. Fig. 13 displays the filtering speed with respect to the convergence threshold. As expected the higher the threshold the faster the filtering. Especially for thresholds less than 0.1 the filtering time decreases almost exponentially. According to this graph and all the previous ones, local mode filtering is the fastest filtering operation followed by anisotropic diffusion, and then mode finding, while color mean shift is slightly slower. This is expected due to the extra number of calculations needed to estimate the $5D$ feature vector instead of the $3D$ feature vector in the other methods.

4.4.5 Filtering speed optimization

In the tests above, we use our own implementation of all the filtering methods, that is a straightforward translation of Table 4 to Matlab and C code, to perform the speed experiments. A number of methods can be used to perform the filtering faster.

In the core of all the filtering algorithms the pairwise distance between feature points needs to be computed for all pairs of points. As suggested in [3] employing data structures and algorithms for multidimensional range searching can speed up the filtering. This technique can be used in all the filtering methods and is expected to significantly improve the speed of slow methods such as mean shift and spatial mean shift.

In mode finding the trajectory of most feature points lay along the path of other feature points. Christoudias et al. in [18] report a speed up of about five times relative to the original algorithm when they "merge" the feature points together. This trick can directly be used in local mode filtering. A variation of the same concept could also be used to speed up the filtering in all the other methods.

The introduction of the multicore CPUs and, especially, GPUs has provided new way to improve the execution speed of algorithms through a parallel implementation. From Table 4 and Fig. 5 it is clear that the filtering of each feature point can be performed in parallel. We expect that a careful implementation of any of the four algorithms (i.e. mode finding, color mean

Table 2: Synopsis of the filtering results

- Normal kernel gives smoother images compared to Epanechnikov kernel
- Luv color space produces smoother filtering results compared to RGB color space.
- Mode finding and local mode finding produce similar filtering results. Mode finding performs slightly better filtering.
- Color mean shift and anisotropic diffusion produce similar filtering results. Color mean shift preserves the edges better than anisotropic diffusion.
- $3-D$ filtering (i.e. local mode filtering) is almost equivalent to $5-D$ filtering (i.e. mode finding) when the original image is used for the comparison. When the image obtained in the previous iteration is used then $5-D$ filtering (i.e. color mean shift) preserves edges better than $3-D$ filtering (i.e. anisotropic diffusion).
- Whether we use the original image for comparison or not affects the filtering more than whether we perform it in $3-D$ or $5-D$.
- Local mode filtering is the fastest; mode finding and local mode filtering are a little bit slower; color mean shift is even slower. All the methods are fast enough to perform the filtering in real time for a reasonably large image when implemented in GPUs.

shift, local mode filtering and anisotropic diffusion) on a modern GPU will run in real time for VGA or larger images.

5 Filtering Conclusions

So far, we presented a unifying framework under which we can express different filtering algorithms. Using the new understanding of filtering, we developed three new edge preserving filtering methods, that we named Color Mean Shift, Spatial Mean Shift and Joined Bilateral Filtering. The first one exhibits similar clustering characteristics with the original Mean Shift method while being almost as computationally efficient as the Mode Finding method, so it was included in our filtering comparison. We performed a comparison of four different methods (Mode Finding, Color Mean Shift, Local Mode Filtering and Anisotropic diffusion) on a number of images with different configurations for the color space and the kernel function. Overall we noticed that Color Mean Shift outperforms (i.e. creates more uniform segments with better boundary separation) than the other methods with the drawback of being slightly slower. Table 2 synopsizes the results of the experimental comparison for performing edge preserving filtering.

6 Grouping methods

A variety of grouping methods exist in the literature for image segmentation. As a matter of fact almost all the color based image segmentation methods are grouping methods. Next, we describe the three methods that we have chosen to use in the segmentation experiments. The first two methods are based on a simple connected components algorithm with a global threshold, while the last method is an extension of that algorithm. All methods are simple, namely they don't require the use of complicated tuning parameters and they are used widely for image segmentation. Another advantage is that they are fast so they can be used for (almost) real time segmentation.

6.1 Greedy Connected Components grouping (CC3D and CC5D)

This is the same strategy that Comaniciu and Meer implicitly use in their image segmentation algorithm [3]. The method is a good starting point for our comparison; its simplicity allows us to compare the smoothing algorithms for the task of segmentation without worrying that the result has been "changed" by the grouping algorithm. Thus, the quality of the segmentation is directly related to the quality of the filtering.

In a nutshell, the algorithm groups neighboring pixels together if and only if their Euclidean distance is within a user defined threshold. Note that there is a $3-D$ and a $5-D$ variant of this algorithm since pixel x_i is represented by either a $3-D$ vector ($\mathbf{S}_i$) or a $5-D$ vector ($[\mathbf{x}_i, \mathbf{S}_i]$) (Fig. 14). In our implementation we use an union-find data structure to perform the merging so the complexity of the algorithm is almost linear on the number of pixels. A similar implementation was used in the EDISON system [18].

The biggest problem with this simple grouping method is the "segment diffusion" problem, when two quite different segments are merged together because there is a single weak (blurry) edge between them (e.g. the clouds and the sky are merged into a single segment in the first images of the top row of Fig. 16). In order to reduce the impact of this problem we reduce the grouping threshold (t in Fig. 14, top row) to 0.5.

6.2 Grouping with an Adaptive Threshold (GAT)

Felzenszwalb and Huttenlocher in [19] present a variation of the connected component algorithm where an adaptive threshold for merging segments is used. Each segment C_i keeps track of the maximum distance between two pixels belonging to it[6](denoted $Int(C_i)$) and two segments $C_i,\ C_j$ are merged only if the minimum distance between the pixels belonging to their common boundary is smaller than the internal distance $Int(C_i), Int(C_j)$. The method is described in Fig. 14. This algorithm is also linear on the number of pixels.

7 Segmentation as Filtering plus Grouping

The notion of segmentation consisting of a filtering followed by a grouping step is not new, but it is underemphasized in the literature. Most image segmentation (i.e. grouping) algorithms operate on the original image, while the filtering algorithms are usually applied to the problems of edge preserving smoothing or noise removal. Comaniciu and Meer [3] talk about "segmentation consisting of a filtering and a fusion step", but

6. Only the edges belonging to the minimum spanning tree of the segment are considered

Connected Components $3D$ (CC3D)

Input:
set of pixels $\mathbf{x}_i$ with intensities $\mathbf{S}_i$
a grouping threshold t
Output:
a set of labels (label l_i for $\mathbf{x}_i$)
Algorithm:
for all pixels $\mathbf{x}_i$
assign label l_i
repeat until convergence
for all pixels $\mathbf{x}_i$
for all pixels $\mathbf{x}_j$
if $||\mathbf{S}_i - \mathbf{S}_j|| < t$ and $l_i \neq l_j$
merge the labels of $\mathbf{x}_i$ and $\mathbf{x}_j$ ($l_i \equiv l_j$)

Connected Components $5D$ (CC5D)

Input:
set of pixels $\mathbf{x}_i$ with intensities $\mathbf{S}_i$
a grouping threshold t
Output:
a set of labels (label l_i for $\mathbf{x}_i$)
Algorithm:
for all pixels $\mathbf{x}_i$
assign label l_i
repeat until convergence
for all pixels $\mathbf{x}_i$
for all pixels $\mathbf{x}_j$
if $||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j]|| < t$ and $l_i \neq l_j$
merge the labels of $\mathbf{x}_i$ and $\mathbf{x}_j$ ($l_i \equiv l_j$)

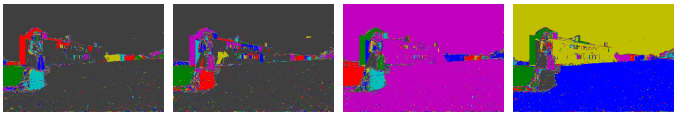
Grouping with an Adaptive Threshold (GAT)

Input:
An image as a graph $G = (V, E)$ with n vertices and m edges
Output:
A segmentation of V into components $S = (C_1, ...C_r)$
Algorithm:
sort E into $\pi = (o_1, \ldots, o_m)$ by non decreasing edge weight
in the initial segmentation S^0 each vertex v_i is its own segment
for $q = 1, \ldots, m$ construct S^q given S^{q-1} as follows
let v_i, v_j be the vertices connected by the q^{th} edge $o_q = (v_i, v_j)$
let pixels v_i, v_j belong to components C_i, C_j with
$|C_i|, |C_j|$ number of elements respectively
let $Int(C_i), Int(C_j)$ be the maximum edge weights of the minimum spanning tree of components C_i, C_j
let e_q be the weight of edge o_q
if v_i, v_j belong to different components C_i, C_j and $e_q < \min\{Int(C_i) + \frac{k}{|C_i|}, Int(C_j) + \frac{k}{|C_j|}\}$
merge C_i, C_j
return $S = S^m$

Figure 14: The grouping algorithms that we use in the segmentation experiments.

they focus on the filtering step and they use the simple connected component algorithm of Fig. 14 top left, to obtain the final segments. Subsequent work from the same group [18] focuses on how to bring edge information into the filtering and grouping step, but they still use a similar connected components algorithm. Close to our philosophy is the work of Unnikrisnan et al. [28] where they combine the filtering algorithm of [18] with the grouping algorithm of [19]. Their focus, thought, is to introduce a new measure called Normalized Probabilistic Rand to compare the quality of segmentation.

One of the main points of this paper is that both steps are important to obtain good segmentation results. In Figs. 15, 16, for example, we present the segmentation results we obtained using different combinations of filtering and grouping methods. First, we use the same grouping method, namely CC3D, along with the four different grouping algorithms. It is clear that depending on the filtering method the sky is merged with the grass or not. On the second figure the filtering method is kept constant (color mean shift) while the grouping method changes. Here the results significantly depend on the method, with the adaptive threshold method producing the most intuitive segments. In the next section we experimentally study the problem of color based segmentation



(a) CMMS+CC3D (b) CMS+CC3D (c) LMF+CC3D (d) AD+CC3D

Figure 15: We present the segmentation results when we use the same grouping method (CC3D) coupled with different filtering methods. The filtering is performed on the RGB color space with an Epanechnikov kernel with spatial and color resolution $h_s = 5, h_r = 4$ respectively.

(a) CMS+CC3D (b) CMS+CC5D (c) CMS+GAT

Figure 16: We present the segmentation results when we use the same filtering method (Color mean shift) followed by a different grouping method. The filtering is performed on the RGB color space with an Epanechnikov kernel with spatial and color resolution $h_s = 5, h_r = 4$ respectively.

by comparing different combinations of filtering and grouping algorithms. More specifically we couple each of the four filtering algorithms that we studied above with the three grouping algorithms that we introduced in the previous section to obtain a new segmentation method.

8 Segmentation Comparison

There is little effort to classify image segmentation algorithms and compare their characteristics due to two main factors. The multiplicity of methods each having a number of parameters make the comparison extremely tedious. Moreover, the "right" segmentation is hard to define, since there are many levels of detail in an image and therefore multiple different meaningful segmentations. S. Paris [29] for example, creates a hierarchical structure of segmentations where starting from a large number of segments, regions are merged together to create more coarse segmentations. Furthermore, in complex scenes the evaluation of a given segmentation mostly relies on subjective criteria. Borra and Shankar [30], for example, go as far as suggesting that the proper segmentation is task and domain specific. The difficulty of formally defining the quality of a segmentation explains the lack of segmentation databases for natural images.

The most complete attempt at comparing segmentation algorithms is presented on the Berkeley database and segmentation website [8]. A large set of images along with human created segmentations are made available for segmentation evaluation. This is the testbed we use in this paper for the evaluation of the different segmentation methods[7]. More specifically we use the 200 training images along with the 1087 human created segmentations. Next, we first describe the different measures that we use for the comparison, and then we present the segmentation results.

8.1 Comparison measures

A number of measures have been proposed in the literature in order to compare two different segmentations of the same image. In general the segmentation measures can be classified in two categories; region based and boundary based. The first group includes measures, such as the Global Consistency Error [8], the Variation of Information [31],[32] and the Probabilistic Rand index [33], that consider the overlap of the segments in the two segmentations, while the second consists of measures that count the overlap or the distance of the boundaries, such as the Boundary Displacement Error [34]. We compared the segmentations using all the above measures, but we report results on the Probabilistic Rand index and the Boundary Displacement Error only. This is due not only on the lack of space, but mainly because the other measures were either not discriminative (Variation of Information) or misleading (Global Consistency Error).

Boundary Displacement Error (BDE) This quantity measures the average displacement error of the boundary pixels between two segmented images. Particularly, it defines the error of one boundary pixel in one segmentation as the distance between the pixel and the closest pixel in the other segmentation. BDE is not symmetric, thus we use it to measure the average distance of the human segmentation to the computer generated one. Intuitively, the lower the BDE value the more similar the two segmentations are. A BDE measure of 0 indicates that all the boundaries of the human segmentation are covered by the boundaries of the computer one, but not vice versa.

Probabilistic Rand Index (PR) This measure counts the fraction of pairs of pixels whose labellings are consistent between the computed segmentation and the ground truth, averaging across multiple ground truth segmentations to account for scale variation in human perception. PR is a measure of similarity and as such a value of 0 indicates no similarity, while a value of 1 indicates the highest similarity.

8.2 Methodology

To produce the following segmentation figures we only vary the value of the color resolution h_r of the filtering methods. More specifically, we let h_r to obtain values from 0.6 to 20 on 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. We also use a spatial resolution of $h_s = 5$, resulting on a 11×11 smoothing window around each pixel. Furthermore, we utilize constant parameters for the grouping methods. More specifically the grouping threshold (parameter t of Fig. 5) is set to 1 and 0.5 for the CC5D and CC3D grouping algorithms respectively. We use the excellent C++ code provided by Felzenszwalb and Huttenlocher [19] with two different sets of parameters to implement the grouping with the adaptive threshold (GAT). On the static setting we used $\sigma = 0.5$ and the $k = 500$ as suggested in their paper. On the dynamic (varying k setting) we change the value of k depending on the

7. In Appendix **??** we also present segmentation results using the Weizmann Institute dataset [9].

value of h_r. In the following experiments we use a linear relation between h_r and k[8], namely

$$k = 45.83 * h_r + 142.5. \quad (20)$$

We computed the comparison measures for each image of the database and further aggregated the results for the whole database using the median value[9]. 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. The motivation behind this choice is the following; a major goal of a segmentation algorithm is to create as large segments as possible without merging areas belonging to different objects. Thus the measures described above in conjunction with the segment size only, can indicate whether a segmentation is good and useful. For the computation of the Boundary Displacement Error and the Probabilistic Rand Index we use the code provided by J. Wright and A. Yang [35].

8.3 Filtering+Grouping vs Grouping

In the first set of experiments we compare the segmentation methods with and without filtering. We start with the simple segmentation method of connected components (CC5D) in Fig. 17. Note that filtering the image before performing the final grouping improves the segmentation results in both measures. In Fig. 18 we present similar results when the GAT grouping is used. As mentioned in 8.2 there are two variations of the GAT; one with a constant parameter $k = 500$ and one where k changes according to Eq. 20. We observe that in both cases the results when we performed the filtering and the grouping were better than when we performed the grouping on the original images only. Especially in the case of GAT with varying k there was a significant improvement on both measures.

8.4 Epanechnikov vs Gaussian kernel and RGB vs Luv color space

On our previous work [36] we presented a comparison between the two kernels; Epanechnikov and Gaussian, and the two color spaces; RGB and Luv. As it is shown in Fig. 19 significantly better results were obtained with the Luv/Normal kernel combination.

In Fig. 20 we extend the results for the case where the GAT algorithm is used for grouping. Confirming our previous observation the best combination is also Luv color space and Normal kernel function. Furthermore, for average segment sizes greater than 250 pixels the clear winner for the filtering algorithm is CMMS. This is contrary to the results for CC5D where CMS outperformed CMMS in all cases. Thus, it is evident that to obtain the best segmentation results *one needs to consider the combination of filtering and grouping algorithms*.

8. Out of the infinite number of combinations for the pair (k, h_r) we match the average segment size obtained with $CMS + CC5D$ with the one obtained by GAT only to compute the coefficients. Thus, we calculated the coefficients of the linear system by solving the system of (k, h_r) for values (170, 0.6) and (1050, 19.8).

9. 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.

9 Conclusions

In this paper we presented our position that the problem of color based segmentation should be subdivided into a filtering and a grouping component. We used the Berkeley segmentation dataset to validate our position. Furthermore, we created a number of new segmentation algorithms by combining existing and new filtering and grouping methods and we evaluated all the methods extensively. Table 3 synopsizes the results of the experimental comparison for performing edge preserving filtering and color based segmentation.

There are two main results that we want to emphasize here. In all the experiments, processing the image with an edge preserving filter before using a grouping method produced significantly better results. Thus it is beneficial to consider *the segmentation process to be a combination of a filtering and a grouping step*.

Second, depending on the grouping method that is used, a different filtering process produces best results. For grouping with a hard threshold (i.e. CC3D and CC5D methods) Color Mean Shift filtering worked best. When grouping with an adaptive threshold (i.e. GAT method) Mode Finding proved to be the best method. As a conclusion, *when considering the problem of color based segmentation, one should study the combination of the filtering and the grouping method to obtain the best results.* Studying only one component in isolation is not sufficient.

Our overall comparison showed that for the Berkeley dataset the best method to use is a combination of Mode Finding with Grouping with Adaptive Threshold (with variable k). Furthermore the results are better when the filtering is performed in Luv color space with a Normal kernel.

There are many interesting directions for future research. In this paper we focused on the filtering more than the grouping step. It would be interesting to perform the comparison using a more wide range of grouping methods, namely global energy minimization methods (e.g. graph cut), eigenvector based methods (e.g. normalized cuts) and soft assignment methods based on algebraic multigrid.

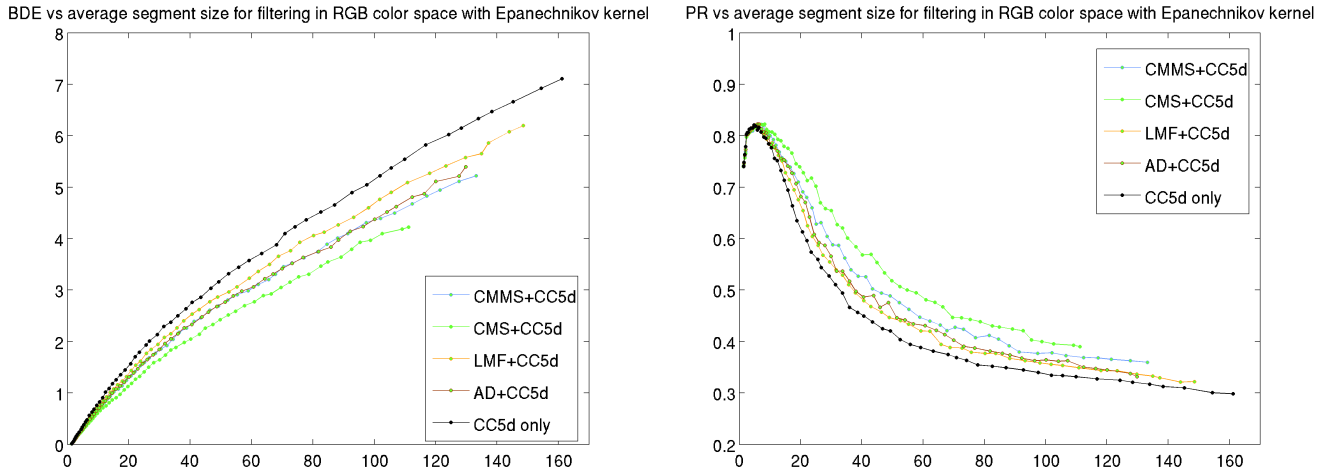


Figure 17: Comparison of CC5D grouping with and without filtering the images. The filtering in the plots was performed on the RGB color space with an Epanechnikov kernel. Similar results were obtained on the Luv color space and with the Gaussian kernel. Both BDE and PR measures show that filtering improves the quality of segmentation.

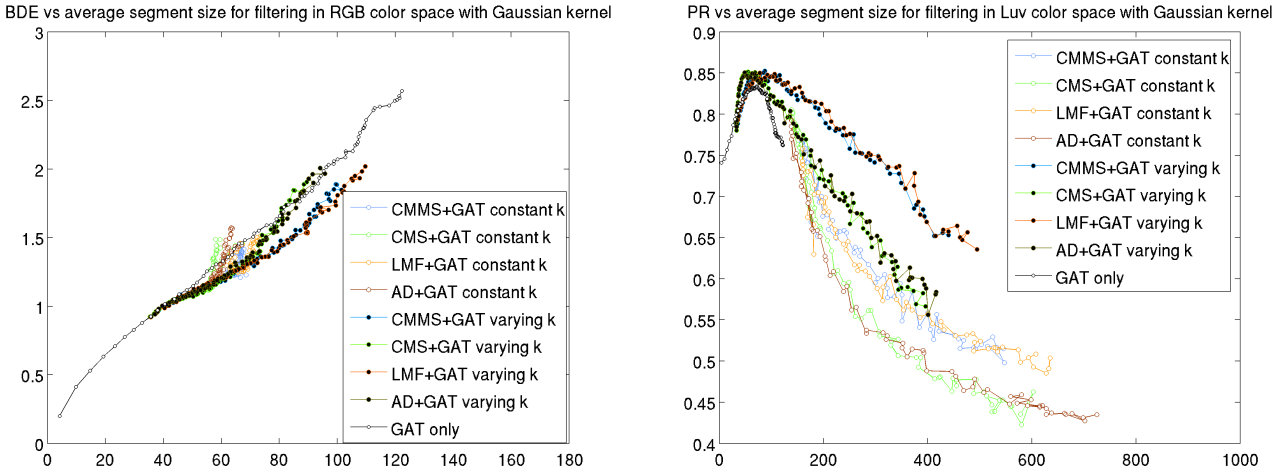


Figure 18: Comparison of GAT grouping with and without filtering the images. The filtering in the plots was performed on the Luv color space with a Gaussian kernel. Similar results were obtained on the RGB color space and with the Epanechnikov kernel. Both BDE and PR measures show that filtering improves the quality of segmentation.

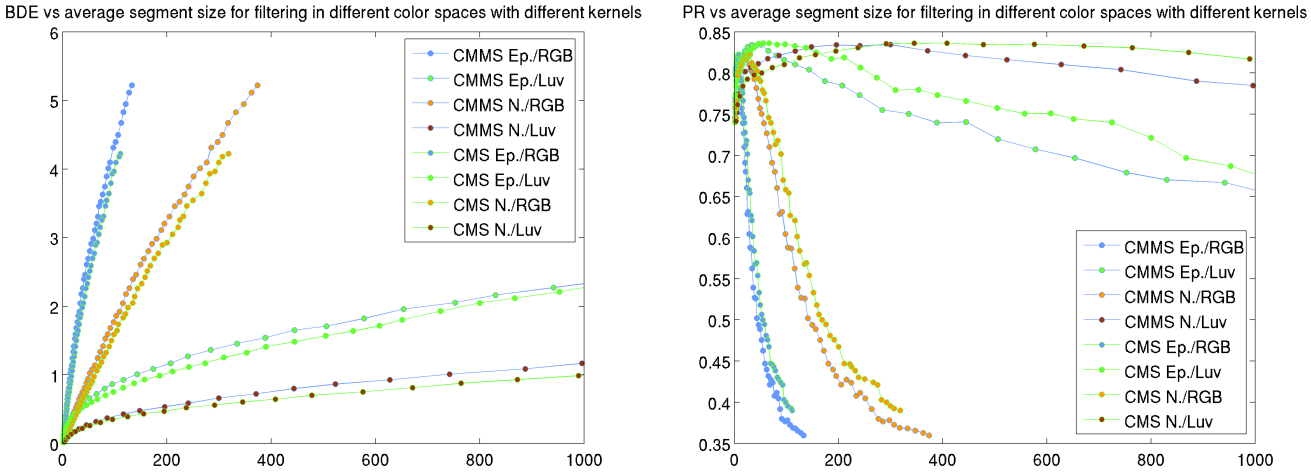


Figure 19: Comparison of segmentation methods when performing the filtering on different color spaces using different kernels. Two color spaces (RGB and Luv) and two kernel functions (Epanechnikov and Normal) were compared. We used the two best filtering methods, namely CMMS and CMS and the CC5D grouping method.

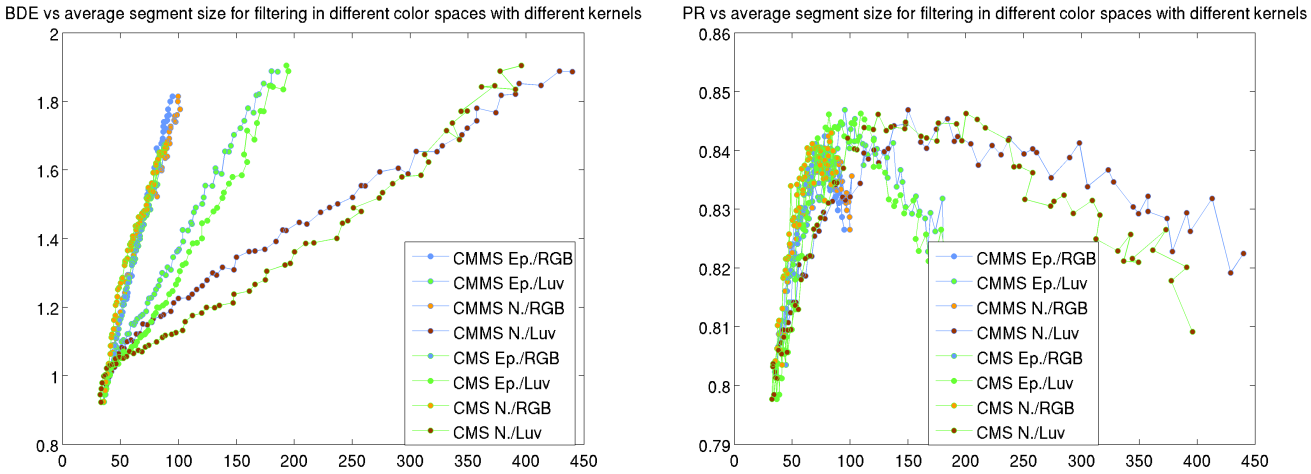


Figure 20: Comparison of segmentation methods when performing the filtering on different color spaces using different kernels. Two color spaces (RGB and Luv) and two kernel functions (Epanechnikov and Normal) were compared. We used the two best filtering methods, namely CMMS and CMS and the GAT grouping method.

APPENDIX

Claim 3: Local mode filtering (and mode finding with a Gaussian kernel) can be considered as gradient descend methods for solving the corresponding optimization problem (Eqs. 13 and 10 respectively) with a step size at iteration t of

$$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])}. \quad (21)$$

Proof: A proof for local mode filtering follows. Each pixel p_i is optimized separately. So if we replace the step size γ_i in the general gradient descent algorithm we get

$$\mathbf{S}_i^{t+1} = \mathbf{S}_i^t - \gamma_i^t \nabla \sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0]) \quad (22)$$

$$\mathbf{S}_i^{t+1} = \mathbf{S}_i^t - \gamma_i^t \sum_j \nabla K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0]) \quad (23)$$

$$\mathbf{S}_i^{t+1} = \mathbf{S}_i^t - \gamma_i^t \sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])[\mathbf{S}_j^0 - \mathbf{S}_i^t] \quad (24)$$

$$\begin{aligned} \mathbf{S}_i^{t+1} &= \mathbf{S}_i^t + (\gamma_i^t \textstyle\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j^0]))\mathbf{S}_i^t \\ &\quad -\gamma_i^t \textstyle\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])\mathbf{S}_i^t \end{aligned} \quad (25)$$

$$\mathbf{S}_i^{t+1} = \frac{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])\mathbf{S}_j^0}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])} \quad (26)$$

that is exactly the intensity values for pixel $\mathbf{x}_i$ at the next iteration $t+1$. □

To prove the claim for mode finding with a Gaussian kernel one only needs to replace the occurrence of $\mathbf{S}_i^t$, $\mathbf{S}_i^{t+1}$, $\mathbf{S}_j^0$ with $[\mathbf{x}_i^t, \mathbf{S}_i^t]$, $[\mathbf{x}_i^{t+1}, \mathbf{S}_i^{t+1}]$, $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ respectively, because the optimization is performed on the $5-D$ domain.

Table 3: Synopsis of the filtering results

- Segmentations obtained by grouping methods alone have much lower quality than the ones obtained using a combination of a filtering and a grouping method.
- All segmentation methods are very sensitive to image variations. The methods based on Grouping with an Adaptive Threshold (GAT) are the least sensitive to inter image variation. They also exhibit the least sensitivity to the segmentation parameters (h_r, k) when segmenting the same image.
- Segmentation methods based on GAT grouping are not monotonic.
- Segmentation methods based on GAT grouping outperform , on average, all the other segmentation methods.
- Segmentation methods based on GAT grouping are the most stable to color resolution changes i.e., exhibit less variation of the average segment size.
- Segmentation methods based on CC3D and CC5D grouping have very similar performance, with the CC3D ones producing slightly better segmentation results.
- The graphs of the Probabilistic Rand Index (PR) and Boundary Displacement Error (BDE) measures are the most discriminative.
- Color Mean Shift (CMS) based segmentation methods outperform all the other filtering methods when they are combined with CC3D or CC5D grouping methods.
- When using GAT grouping with varying parameter k Mode Finding (CMMS) produces the best results.
- Filtering in Luv produces much larger segments than filtering in RGB for a given color resolution h_r. Filtering with a Normal kernel results in larger segments compared to using a Epanechnikov kernel.
- The selection of the kernel function seems to be very important for the segmentation results. More specifically, we obtained the best segmentation results when the filtering was performed with a Normal kernel in the Luv color space. The second best configuration is a Normal kernel with an RGB color space, while the results obtained with an Epanechnikov kernel in either RGB or Luv color spaces are much worse.

REFERENCES

[1] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function with applications in pattern recognition," *IEEE Trans. Information Theory*, vol. 21, pp. 32–40, 1975.

[2] Y. Cheng, "Mean shift, mode seeking, and clustering," *PAMI*, vol. 17, pp. 790–799, 1995.

[3] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. on PAMI*, pp. 603–619, 2002.

[4] S. Smith and J. Brady, "Susan a new approach to low level image processing," *IJCV*, vol. 23, pp. 45–78, 1997.

[5] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *ICCV*, pp. 839–846, 1998.

[6] J. van de Weijer and R. van den Boomgaard, "Local mode filtering," *CVPR*, vol. 2, pp. 428–432, 2001.

[7] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *PAMI*, vol. 12, no. 7, pp. 629–639, 1990.

[8] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," *ICCV*, vol. 2, pp. 416–423, 2001.

Method	Step Size	Single iteration result
Mode Finding with K_E	$\gamma_i^t = -\frac{1}{2c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0]\|\|<1}[\mathbf{x}_j^0,\mathbf{S}_j^0]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0]\|\|<1} 1}$
Mode Finding with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])[\mathbf{x}_j^0,\mathbf{S}_j^0]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$
Mean Shift with K_E	$\gamma_i^t = -\frac{1}{4c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j]\|\|<1}[\mathbf{x}_j,\mathbf{S}_j]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j]\|\|<1} 1}$
Mean Shift with K_N	$\gamma_i^t = -\frac{1}{2\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])[\mathbf{x}_j,\mathbf{S}_j]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$
Spatial Mean Shift with K_E	$\gamma_i^t = -\frac{1}{2c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0]\|\|<1}[\mathbf{x}_j,\mathbf{S}_j^0]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0]\|\|<1} 1}$
Spatial Mean Shift with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0])[\mathbf{x}_j,\mathbf{S}_j^0]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0])}$
Color Mean Shift with K_E	$\gamma_i^t = -\frac{1}{2c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j]\|\|<1}[\mathbf{x}_j^0,\mathbf{S}_j]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j]\|\|<1} 1}$
Color Mean Shift with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j])[\mathbf{x}_j^0,\mathbf{S}_j]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j])}$
Local Mode Filtering with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$	$\mathbf{S}_i^{t+1} = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])\mathbf{S}_j^0}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$
Anisotropic Diffusion with K_N	$\gamma_i^t = -\frac{1}{2\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$	$\mathbf{S}_i^{t+1} = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])\mathbf{S}_j}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$

Table 4: Step sizes and iteration results for the different filtering methods with different kernels.

[9] S. Alpert, M. Galun, R. Basri, and A. Brandt, "Image segmentation by probabilistic bottom-up aggregation and cue integration," *CVPR*, pp. 1–8, 2007.

[10] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *PAMI*, vol. 25, pp. 564– 577, 2003.

[11] Y. Wei and L. Quan, "Region-based progressive stereo matching," *CVPR*, pp. 106–113, 2004.

[12] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example," *ICCV*, pp. 456–463, 2003.

[13] D. DeMenthon and R. Megret, "Spatio-temporal segmentation of video by hierarchical mean shift analysis," tech. rep., 2002.

[14] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, "Improved fast gauss transform and efficient kernel density estimation," *ICCV*, pp. 464–471, 2003.

[15] M. Fashing and C. Tomasi, "Mean shift is a bound optimization," *PAMI*, vol. 27, pp. 471–474, 2005.

[16] X. Yuan and S. Z. Li, "Half quadratic analysis for mean shift: with extension to a sequential data mode-seeking method," *Computer Vision, IEEE International Conference on*, vol. 0, pp. 1–8, 2007.

[17] M. Carreira-Perpinan, "Gaussian mean-shift is an em algorithm," *IEEE Trans. PAMI*, vol. 29, no. 5, pp. 767–776, 2007.

[18] C. Christoudias, B. Georgescu, and P. Meer, "Synergism in low-level vision," *ICPR*, vol. 4, pp. 150–155, August 2002.

[19] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, no. 2, pp. 167–181, 2004.

[20] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. on PAMI*, vol. 23, no. 11, pp. 1222–1239, 2001.

[21] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE PAMI*, vol. 22, no. 8, pp. 888–905, 2000.

[22] Y. Weiss, "Segmentation using eigenvectors: a unifying view.," *ICCV*, pp. 975–982, 1999.

[23] E. Sharon, A. Brandt, and R. Basri, "Fast multiscale image segmentation," *CVPR*, pp. 70–77, 2000.

[24] V. Epanechnikov, "Nonparametric estimation of a multivariate probability density," *Theory Prob. Appl. (USSR)*, vol. 14, pp. 153–158, 1969.

[25] S. Rao, A. Martins, and J. Principe, "Mean shift: An information theoretic perspective," *Pattern Recognition Letters*, vol. 30, no. 3, pp. 222 – 230, 2009.

[26] M. Black, G. Sapiro, D. Marimont, and D. Heeger, "Robust anisotropic diffusion," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 421–432, 1998.

[27] D. Defour, F. D. Dinechin, and J. Muller, "A new scheme for table-based evaluation of functions," Tech. Rep. ISSN 0249-6399, Institut National de Recherche en Informatique et en Automatique, 2002.

[28] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," *PAMI*, vol. 29, no. 6, pp. 929–944, 2007.

[29] S. Paris and F. Durand, "A topological approach to hierarchical segmentation using mean shift.," *CVPR*, 2007.

[30] S. Borra and S. Sarkar, "A framework for performance characterization of intermediate level grouping modules," *PAMI*, vol. 19, no. 11, pp. 1306–1312, 1997.

[31] M. Meila, "Comparing clusterings by the variation of information," *Conference Learning Theory*, 2003.

[32] M. Meila, "Comparing clusterings: an axiomatic view," *ICML*, pp. 577 – 584, 2005.

[33] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "A measure for objective evaluation of image segmentation algorithms," *Workshop on Empirical Evaluation Methods in Computer Vision, CVPR*, 2005.

[34] J. Freixenet, X. Munoz, D. Raba, J. Marti, and X. Cuff, "Yet another survey on image segmentation: Region and boundary information integration," *ECCV*, pp. 408–422, 2002.

[35] A. Y. Yang, J. Wright, Y. Ma, and S. Sastry, "Unsupervised seg-

mentation of natural images via lossy data compression," *Comput. Vis. Image Underst.*, vol. 110, no. 2, pp. 212–225, 2008.

[36] K. Bitsakos, C. Fermüller, and Y. Aloimonos, "An experimental study of color-based segmentation algorithms based on the mean-shift concept," *ECCV*, vol. 2, pp. 506–519, 2010.