Published in Image Processing On Line on 2019–09–17.
Submitted on 2019–03–21, accepted on 2019–08–01.
ISSN 2105–1232 © 2019 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
https://doi.org/10.5201/ipol.2019.255

An Implementation of the Mean Shift Algorithm

Damir Demirović

Faculty of Electrical Engineering, University of Tuzla, Tuzla, Bosnia and Herzegovina
(damir.demirovic@untz.ba)

Communicated by Luis Gómez and Pascal Monasse *Demo edited by* Pascal Monasse

Abstract

In this paper, we present an implementation and analysis of the mean shift algorithm. The mean shift is a general non-parametric mode finding/clustering procedure widely used in image processing and analysis and computer vision techniques such as image denoising, image segmentation, motion tracking, etc.

Source Code

The source code (ANSI C++), its documentation, and the online demo are accessible at the IPOL the web page of this article[1]. Compilation and usage instructions are included in the README.txt file of the archive.

Keywords: mean shift algorithm; clustering; segmentation

1 Introduction and Related Theory

The mean shift algorithm is a powerful general non-parametric mode finding procedure. It can be described as a hill-climbing algorithm on the density defined by a finite mixture or a kernel density estimate [3]. Mean shift is based on ideas proposed by Fukunaga and Hostetler [11], and can be used as a non-parametric clustering method [3], for object tracking [8], image segmentation [17] etc.

Mean shift started to attract the attention after the publication of Chang [4] describing its applications to cluster analysis, and in vision when Bradski applied it on the Camshift algorithm [1]. Comanciu [6] extended it to low-level vision problems like segmentation and adaptive smoothing. Mean shift is employed in the spatial-range domain on gray level and color images for discontinuity preserving filtering and image segmentation with filtering properties similar to the bilateral filter [18].

In [7] Comaniciu et al. present two solutions for the scale-space problem. The first is completely non-parametric and based on the adaptive estimation of the normalized density gradient. They define variable bandwidth mean shift and show superiority over the fixed bandwidth procedure.

[1]https://doi.org/10.5201/ipol.2019.255

A review of bandwidth selection for density estimation is given in [14]. In [15] Meng et al. propose a novel adaptive bandwidth strategy that combines different adaptive bandwidth strategies and bidirectional adaptive bandwidth mean shift which have the ability to escape from the local maximum density. Fashing and Tomasi [10] developed the understanding that mean shift is a bound optimization which is equivalent to Newton's method in the case of piece-wise constant kernels.

In [20] the convergence of mean shift is improved by dynamically updating the sample set during the iterations, and the procedure is called Dynamic Mean Shift (DMS). Carreira-Perpinan [2] proved that the mean shift with Gaussian kernel corresponds to the Expectation Maximization (EM) algorithm.

This paper is organized as follows. In Section 1.1 a short overview of kernel density estimation and kernel functions is presented. Section 1.2 introduces how a differentiable kernel is used to make a gradient density estimator. Insight into the working of the mean shift algorithm and its two phases, namely filtering and segmentation, is given in Section 2. Section 3 and 4 describe our implementation of the mean shift algorithm and its evaluation.

1.1 Kernel Density Estimate and Kernel Functions

Kernel density estimation is a non-parametric method where the parameter search window radius must be defined. The advantage of non-parametric methods is dealing with arbitrary coupled/joined probabilities. With an infinite number of observations, non-parametric methods can reconstruct the density of the original probabilities. An unknown density distribution with finite observations of a point in the sample space can be estimated using the kernel density estimation.

To estimate the density of a point $x \in \mathbb{R}^D$ in a D-dimensional feature space, N observations are required x_i with $1 \leq i \leq N$ and $x_i \in \mathbb{R}^D$ within a search window that is centered around point x. The probability density in x is the mean of the probability densities that are centered in the N observations $x_1, \cdots, x_N$.

Let X be a random variable and N observations x_i with $1 \leq i \leq N$ and $x_i \in \mathbb{R}^D$ given. The kernel density estimator $\hat{f}(x)$ in a point $x \in \mathbb{R}^D$, with a kernel $K(x)$ and $D \times D$ bandwidth matrix $\mathbf{H}$ is

$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K_H(x - x_n), \tag{1}$$

where

$$K_H(x) = \frac{1}{\sqrt{|\mathbf{H}|}} K\left(\frac{x}{h}\right), \tag{2}$$

where the simplification $\mathbf{H} = h^2\mathbf{I}$ has been applied. Here, only the single bandwidth parameter, the window radius, will be regarded. Equation (1) can then be written

$$\hat{f}(x) = \frac{1}{Nh^D} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right). \tag{3}$$

This equation is valid for several kernels. It follows the definition and a profile of the kernel. The following definition is from Cheng [4]. The norm $\|x\|$ of x is a non-negative number so that $\|x\|^2 = \sum_{d=1}^{D} |x_d|^2$. A mapping $K : \mathbb{R}^D \to \mathbb{R}$ is called a kernel when there is a function $k : [0, \infty] \to \mathbb{R}$, the profile of the kernel, such that

$$K(x) = c_{k,D} k(\|x\|^2), \tag{4}$$

where K is radially symmetric, and k is non-negative, non-increasing and piece-wise continuous with $\int_0^\infty k(r)dr < \infty$. The value $c_{k,D}$ is a positive normalization constant so that $K(x)$ integrates to 1.

The previous equation can be transformed into a new equation. The two parameters K and h represent which kernel and radius are used for the density estimator.

With the profile notation k, Equation (3) is transformed to

$$\hat{f}_{h,K}(x) = \frac{c_{k,D}}{Nh^D} \sum_{i=1}^{N} k\left(\left\|\frac{x - x_i}{h}\right\|^2\right). \tag{5}$$

In [13] Huang et al. show that the original Epanechnikov mean shift terminates at non-critical points due to its non-smoothness nature and propose a simple remedy to fix it.

This section will introduce three uni-variate profiles and their associated multivariate radial symmetric kernels.

From the Epanechnikov profile

$$k_E(x) = \begin{cases} 1 - x, & 0 \leq x \leq 1 \\ 0, & x > 1 \end{cases}, x \in \mathbb{R}, \tag{6}$$

follows a radial symmetric kernel

$$K_E(x) = \begin{cases} \frac{1}{2}c_D{}^{-1}(D+2)(1 - \|x\|^2) & \|x\| \leq 1 \\ 0 & \text{otherwise} \end{cases}, x \in \mathbb{R}^D, \tag{7}$$

where c_D is the volume of the D-dimensional globe. The Epanechnikov kernel is used often to minimize the Mean Integrated Squared Error (MISE).

From the normal profile

$$k_N(x) = \exp\left(-\frac{1}{2}x\right) \text{ where } x \geq 0, x \in \mathbb{R}, \tag{8}$$

follows the normal kernel

$$K_N(x) = (2\pi)^{-D/2} \exp\left(-\frac{1}{2}\|x\|^2\right), x \in \mathbb{R}. \tag{9}$$

The normal distribution, like every other kernel with infinite support, is often capped for finite support. Finite support is important for convergence. Capping the normal kernel can be accomplished by multiplying it by a uniform kernel where the inner part of the normal kernel is cut out and weighted with 1 and the outer part is set to 0. The derivative of the normal profile is again a normal profile.

From the uniform profile

$$K_U(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}, x \in \mathbb{R}^D, \tag{10}$$

follows the uniform kernel

$$K_U(x) = \begin{cases} 1 & \|x\| \leq 1 \\ 0 & \text{otherwise} \end{cases}, x \in \mathbb{R}^D. \tag{11}$$

a hyper unit ball in the origin.

Assuming that a derivative of a profile $k(x)$ exists for all $x \in [0, \infty]$, it follows a new profile $g(x)$. Now a kernel $G(x)$ can be defined

$$G(x) = c_{g,D}\, g(\|x\|^2), \tag{12}$$

where $c_{g,D}$ a normalizing constant and $K(x)$ is the shadow kernel of $G(x)$[4]. The mean shift vector of a kernel points to the same direction as the gradient of the shadow kernel.

1.2 Density Gradient Estimation

The density gradient estimator can be expressed as the gradient of the density estimator

$$\hat{\nabla} f_{h,K}(x) \equiv \nabla \hat{f}_{h,K}(x) = \frac{2c_{k,D}}{Nh^{D+2}} \sum_{i=1}^{N} (x - x_i) k' \left(\left\| \frac{x - x_i}{h} \right\|^2 \right), \tag{13}$$

where the inner part and a part of the prefactor originate from the differentiation of $k \left(\left\| \frac{x-x_i}{h} \right\| \right)$.

$$\begin{aligned} \nabla \left(k \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \right) = & \\ &= \left(\left\| \frac{x - x_i}{h} \right\| \right)' k \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) = \\ &= 2(x - x_i) \left(\frac{1}{h} \right) k' \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) = \\ &= \frac{2}{h^2} (x - x_i) k' \left(\left\| \frac{x - x_i}{h} \right\|^2 \right). \end{aligned} \tag{14}$$

Using $g(x) = -k'(x)$ and with Equation (12) a new kernel $G(x)$ with profile $g(x)$ can be defined. Transforming Equation (13) with the new profile $g(x)$ the gradient of density becomes

$$\begin{aligned} \hat{\nabla} f_{h,K} = & \\ &= \frac{2c_{k,D}}{Nh^{D+2}} \sum_{i=1}^{N} (x_i - x) g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) = \\ &= \frac{2c_{k,D}}{Nh^{D+2}} \left[\sum_{i=1}^{N} g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \right] \left[\frac{\sum_{i=1}^{N} x_i g \left(\left\| \frac{x-x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{N} g \left(\left\| \frac{x-x_i}{h} \right\|^2 \right)} - x \right], \end{aligned} \tag{15}$$

The first term in Equation (15) conforms with the density estimator $\hat{\nabla} f_{h,G}$ for kernel G except for a factor, whereas the second term is the difference between the center x of the kernel windows which conforms to the mean shift vector from Equation (13). To localize the maxima (which are the roots of the gradient) with mean shift, first it has to be shown that the mean shift vector is moving along the direction of the gradient. Inserting $\hat{\nabla} f_{h,G}$ and $m_{h,G}(x)$ into Equation (15) follows

$$\hat{\nabla} f_{h,K}(x) = \frac{2c_{k,D}}{h^2 c_{g,D}} \hat{f}_{h,G}(x) m_{h,G}(x), \tag{16}$$

transformed to $m_{h,G}(x)$ follows

$$m_{h,G}(x) = \frac{1}{2} h^2 c \frac{\hat{\nabla} f_{h,K}(x)}{\hat{f}_h G(x)}, \tag{17}$$

whereas

$$c = \frac{c_{g,D}}{c_{k,D}}. \tag{18}$$

The denominator of Equation (18) is the normalizing factor that originates from the density estimator with kernel G in X and the numerator is the gradient density estimator with kernel K. In fact, the mean shift vector is proportional to the gradient, that means it is adaptive. Kernel K is the shadow kernel of kernel G. Cheng firstly introduced the term shadow kernel.

Let

$$mi_{h,K}(x) = \frac{\sum_{i=1}^{n} x_i k\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} k\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x, \tag{19}$$

be the D-dimensional mean of the observations $x_1, \cdots, x_N$ from $\mathbb{R}^D$ weighted with kernel K and a window radius h. Then K is the shadow to the kernel G if the mean shift vector with kernel G

$$m_{h,G}(x) = mi_{h,G}(x) - x = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x, \tag{20}$$

lies in the gradient density estimator direction with kernel K

$$\hat{f}_{h,K}(x) = \frac{c_{k,D}}{Nh^d} \sum_{i=1}^{N} k\left(\left\|\frac{x-x_i}{h}\right\|^2\right). \tag{21}$$

2 Mean Shift Algorithm

Gradient-based methods of feature space analysis use gradients of the probability density function to find the maxima. Such methods are complex because, among other things, of the need for an estimate of the probability of density. The gradient-based methods first calculate the gradient and then the kernel is shifted by a specific length vector in the direction of a maximum increase of density. The magnitude is the step size which has to be chosen appropriately. The task is how to choose a suitable step size because a small step size will slow down the convergence.

The mean shift algorithm solves the main problem of gradient methods. The main idea of the mean shift is to treat the points in D-dimensional feature space as an empirical probability density function where dense regions correspond to the local maxima of the underlying distribution. Gradient ascent is performed in the feature space on the local density estimation until convergence. After the procedure, stationary points correspond to the modes of the distribution, and the same stationary points are considered members of the same cluster.

The step size of the mean shift is adaptive and depends on the gradient of the density of probability. The gradient is not calculated, instead, a more efficient mean shift vector is calculated. The mean shift vector points in the same direction as the gradient in gradient-based methods.

In contrast to the well known K-means clustering approach, mean shift does not need assumptions on the number of clusters and the shape of the distribution, but its performance relies on the selection of scale parameters. Bandwidth is the only parameter to tune, so for the one-dimensional case this is a relatively simple procedure, but in a multidimensional case, it can be difficult. Mean shift might not work well in higher dimensions.

The mean shift procedure consists of two steps:

1. Construction of probability density in some feature space,

2. The mapping of each point to the maximum (mode) of the density which is closest to it.

Each data point is shifted to the weighted average of the data set. The mean shift algorithm tries to find stationary points of an estimated Probability Density Function (PDF).

2.1 Mean Shift Filtering

Filtering with the mean shift algorithm has an advantage because discontinuities like edges are preserved. Smoothing the pixels with a weighted average of the neighbors in both space and color range systematically excludes pixels across the discontinuity.

An image is usually defined as a two-dimensional lattice of p-dimensional vectors. The space of the lattice is known as the spatial domain, while the gray level or color is represented in the range domain. The L*u*v* feature space can be regarded as the probability density function of the color [6].

The multivariate kernel can be defined as the product of two radially symmetric kernels with two bandwidth parameters for each domain

$$K_{h_s,h_r}(x) = \frac{C}{h_s^2 h_r^p} k\left(\left\|\frac{x^s}{h_s}\right\|^2\right) k\left(\left\|\frac{x^r}{h_r}\right\|^2\right), \tag{22}$$

where x^s is the spatial part, x^r is the range part of a feature vector, and $k(x)$ is the common profile used in both domains, h_s and h_r the employed kernel bandwidths, and C is the corresponding normalization constant. For color images, filtering is in 5D feature space, two for lattice coordinates and three for a color.

For each pixel of the image and the set of the neighboring pixels within the specified parameters, spatial radius and color distance are determined, h_s and h_r respectively. For this set of neighbor pixels, the new spatial center and the new color mean values are calculated. These new values will serve as the new center for the next iteration. This procedure will iterate until the spatial and color means will stop changing or the maximum number of iterations is achieved.

In practice, an Epanechnikov truncated normal kernel always provides satisfactory performance, so the user only has to set the bandwidth parameter $h = (h_s, h_r)$. The resolution of the mode detection is controlled by the size of the kernel.

Pseudo-code for mean shift filtering is given in Algorithm 1. The implementation of this algorithm is given in the C++ function *MS_Filter* within the related source code. The algorithm first converts the input image from RGB to L*u*v* color space. Here dimension D is 5, so we have spatial part $X_n^s = (i,j) \in I \times J$, and range part which is a color range $X_n^r = (R,G,B)$, so we have $x_n = (x_n^s, x_n^r) \in \mathbb{N}^5$ for $n = 1, \ldots, N$. Here N is the number of pixels in the image. Input parameters in the filtering algorithm are h_s and h_r, and initial shift and maximum number of iterations which controls the convergence of the algorithm. The filtering procedure reads the corresponding input pixel from the image and then moves the mean shift vector to the next pixel on its path to the convergence point $z_i = (x_i^s, y_{i,conv}^r)$, i.e. the pixel with spatial data x_i^s will have the range component of the point of the convergence $y_{i,conv}^r$. The new calculated pixel is the basis for the next iteration of the calculation of the mean shift vector. After calculation of z_i which is a matrix of 5-dimensional points, conversion back to RGB color space is performed.

For the purpose of testing mean shift filtering, we choose the following parameters for Mandrill image $(h_s, h_r) = (8, 16)$ and uniform kernel, as shown in Figure 1. One can see that the texture of the fur has been smoothed, but the eyes and whiskers remain relatively crisp. It can be seen that the edges are mainly left intact in a similar fashion like with a bilateral filter, which is consistent with the results presented in [18]. The difference between the mean shift and bilateral filtering is the use of local information. The kernel in the mean shift moves in the direction of the maximum increase of the joint density gradient, while in the bilateral case uses a fixed static window [8].

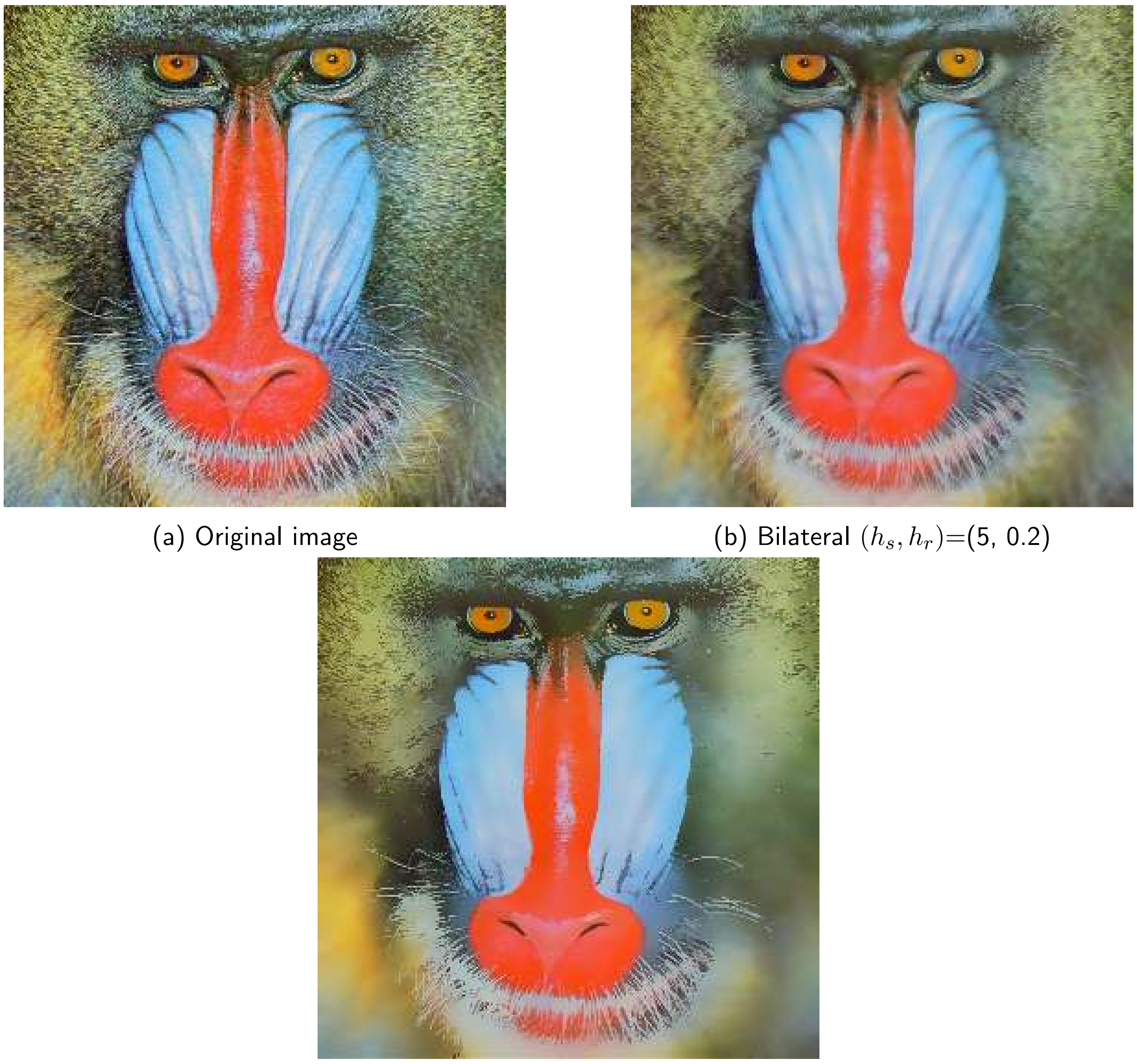
Algorithm 1: Pseudo-code for the Mean shift filtering

```
Input     : $x_n = (x_n^s, x_n^r), n = 1, \dots, N$ 5-dimensional RGB points
Parameter: $h_s$, $h_r$
Data: $c_i = (c_i^s, c_i^r), i = 1, \dots, N$ 5-dimensional L*u*v* points
Data: $z_i = (z_i^s, z_i^r), i = 1, \dots, N$ 5-dimensional filtered points
Output    : $o_n = (o_n^s, o_n^r),\ n = 1, \dots, N$ 5-dimensional RGB points
for $n = 1, \dots, N$ do
    $c_n^r = ConvertRGB2LUV(x_n^r)$
for $i = 1, \dots, N$ do
    initialize $j = 1$ and $y_{i,1} = c_i = (x_i^s, c_i^r)$
    while not converged do
        calculate $y_{i,j+1}$ according to $y_{i,j+1} = \frac{\sum_{i=1}^{n} c_i g\left(\left\|\frac{y_{i,j}-c_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{y_{i,j}-c_i}{h}\right\|^2\right)}$,
        $y_{i,j+1} \in \mathbb{R}^D$ is a new position of the kernel window.
        $n$- the number of points in the spatial kernel centered on $y_{i,j}$
    $y_{i,conv} = y_{i,j+1}$
    assign $z_i = (x_i^s, y_{i,conv}^r)$
for $n = 1, \dots, N$ do
    $o_n^r = ConvertLUV2RGB(z_n^r)$
```

2.2 Mean Shift Segmentation

Segmentation is a process that partitions an image into homogeneous regions. The segmentation algorithm is a straightforward extension of the mean shift filtering algorithm. After applying the filter, all convergence points are found, and clusters are built from them. All convergence points that are closer than h_r in the spatial domain and h_s in the range domain are grouped together, in fact the basins of attraction of the corresponding points are concatenated. In the end, all points are labeled. The basins of attraction of the modes, located within $h_r/2$ in the color space, are recursively fused until convergence. When the mean shift procedure is applied to every point in the feature space, the points of convergence aggregate in groups that can be merged. These are the detected modes, and the associated data points define their basins of attraction. An image region is defined by all the pixels associated with the same mode in the joint domain. The clusters are separated by the boundaries of the basins, and the value of all the pixels within are set to their average. The process of delineation of the clusters is a natural outcome of the mode seeking process [5]. After convergence, the basin of attraction of a mode, i.e. data points visited by all the mean shift procedures converging to that mode, automatically separate a cluster of arbitrary shape. The number of significant clusters present in the feature space is automatically determined by the number of significant modes detected [5]. The parameter M is used for the last step of the algorithm: if the number of pixels in each group is smaller than M, that pixel group is eliminated, i.e that pixel group is merged to a similar neighbor region. It is important to emphasize that the segmentation processes gray level and color images in the same way. The only difference is that in the former case the feature space has three dimensions, the gray value and the lattice coordinates. The following paragraph is the description of the implementation from paper [5].

Pseudo-code for mean shift segmentation is given in Algorithm 2 [6]. The algorithm starts with the filtering phase. After filtering the information about convergence, points z_i are saved. Here M is the input parameter, which defines the minimal region size. This algorithm is implemented in C++



(a) Original image (b) Bilateral (h_s, h_r)=(5, 0.2) (c) Mean shift $(h_s, h_r) = (8, 16)$

Figure 1: Comparison with bilateral filtering

functions *MS_Segment* which call the functions *MS_Cluster* and *TransitiveClosure*. The algorithm firstly converts the input image from RGB to L*u*v* color space. After calculation of z_i, which is a matrix of D-dimensional pixels, conversion back to RGB color space is performed. All points are labeled after their cluster assignment. As one can notice, there are two noticeable differences between Algorithms 1 and 2. The first two and last steps in both algorithms are the same, the segmentation includes the phase of labeling. The number of clusters P is controlled by the parameters h_s and h_r.

In Figure 2, a House image is segmented with the following parameters $(h_s, h_r) = (32, 8)$ with the uniform kernel. Segmentation in this case recovers the sharp edges. As it can be noticed, there is oversegmentation of shadows around the roof. The result of this segmentation can be further refined by tuning the parameters.

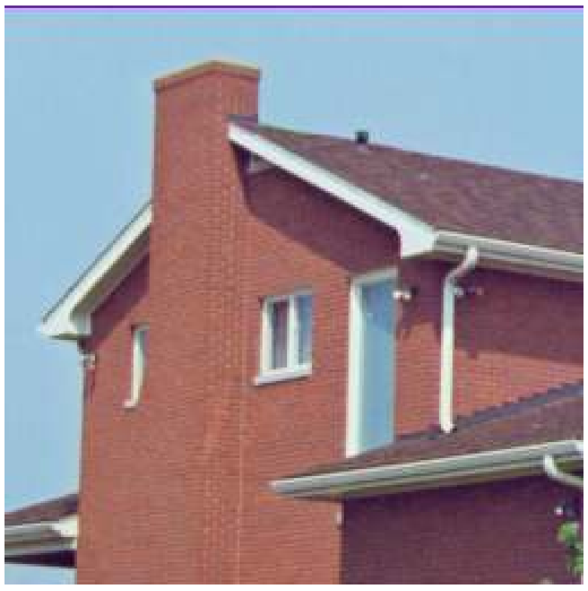
2.3 Mean Shift Strengths and Weaknesses

The mean shift algorithm is an iterative process which computes the mean shift value for the current position and then moves the point to its mean shift value. The process of computing mean

Algorithm 2: Pseudo-code for the Mean shift segmentation

```
Input      : $x_n = (x_n^s, x_n^r), n = 1, \dots, N$ 5-dimensional RGB points
Parameter: $h_s$, $h_r$, $M$
Data: $c_i = (c_i^s, c_i^r), i = 1, \dots, N$ 5-dimensional L*u*v* points
Data: $z_i = (z_i^s, z_i^r), i = 1, \dots, N$ 5-dimensional filtered points
Output     : $o_n = (o_n^s, o_n^r),\ n = 1, \dots, N$ 5-dimensional RGB points

Run the mean shift filtering (Algorithm 1) and store
all information about convergence points $z_i = (x_i^s, y_{i,conv}^r)$.
for $i = 1, \dots, N$ do
    identify clusters $\{C_p\}_{p=1,\dots,P}$ of convergence points by
    linking together all $z_i$ which are closer than $h_s$
    in the spatial domain and $h_r$ in the range domain
for $i = 1, \dots, N$ do
    assign label $L_i = \{p | z_i \in C_p\}$
eliminate spatial regions containing less than M pixels
for $i = 1, \dots, N$ do
    $o_n = ConvertLUV2RGB(z_i)$
```


(a) Original image

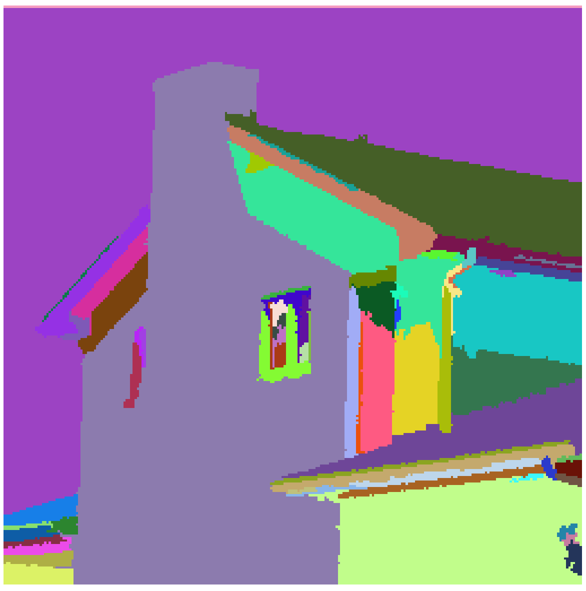

(b) Segmented $(h_s, h_r) = (32, 8)$

Figure 2: Mean shift segmentation of House image.

shift iterates until it fulfills a certain convergence condition and it is limited by the fixed kernel bandwidth. Among the strengths of the mean shift algorithms we have that it does not assume any prior shape (e.g. elliptical) on data clusters, and it can handle arbitrary feature spaces, unlike the K-Means algorithm. The algorithm is not sensitive to outliers, and convergence is guaranteed. A detailed analysis for the bandwidth selection problem is presented in [7].
Weaknesses of the Mean shift algorithm include a need to use adaptive window size because improper window size can lead modes to be merged, which results in bad clusters (segments). The window size (bandwidth selection) is not trivial to choose. The inappropriate window size can cause modes to be merged or generate additional shallow modes. A limitation of the standard mean shift procedure

is that the value of the bandwidth parameter is unspecified. For representative solutions to the bandwidth selection problem, see Comaniciu [8], Singh [16] and Wang [19].

2.4 Mean Shift Computational Complexity

The computational complexity of mean shift can be written as O(Tn^2), where T is the number of iterations and n is the number of data points. So, by doubling the side length of the quadratic image the run time increases by a factor of 4. For each pixel of the image, the Mean shift vector has to be calculated, and if one increases the number of pixels by a number n, we have to deal with n times longer run time.
The convergence speed and quality depend on the kernel type. For example, the algorithm converges quickly with a limited number of steps when we use the uniform kernel. On the other hand, the algorithm is slow when utilizing the normal kernel. The most computationally expensive component of the mean shift procedure corresponds to identifying the neighbors of a point in a space (as defined by the kernel and its bandwidth). This problem is well-known as a multidimensional range searching. This computation becomes cumbersome for high-dimensional feature spaces. Proposed solutions to this problem include embedding the mean shift procedure into a fine-to-coarse hierarchical bandwidth approach [9] and employing approximate nearest-neighbor hashing-based search [12].

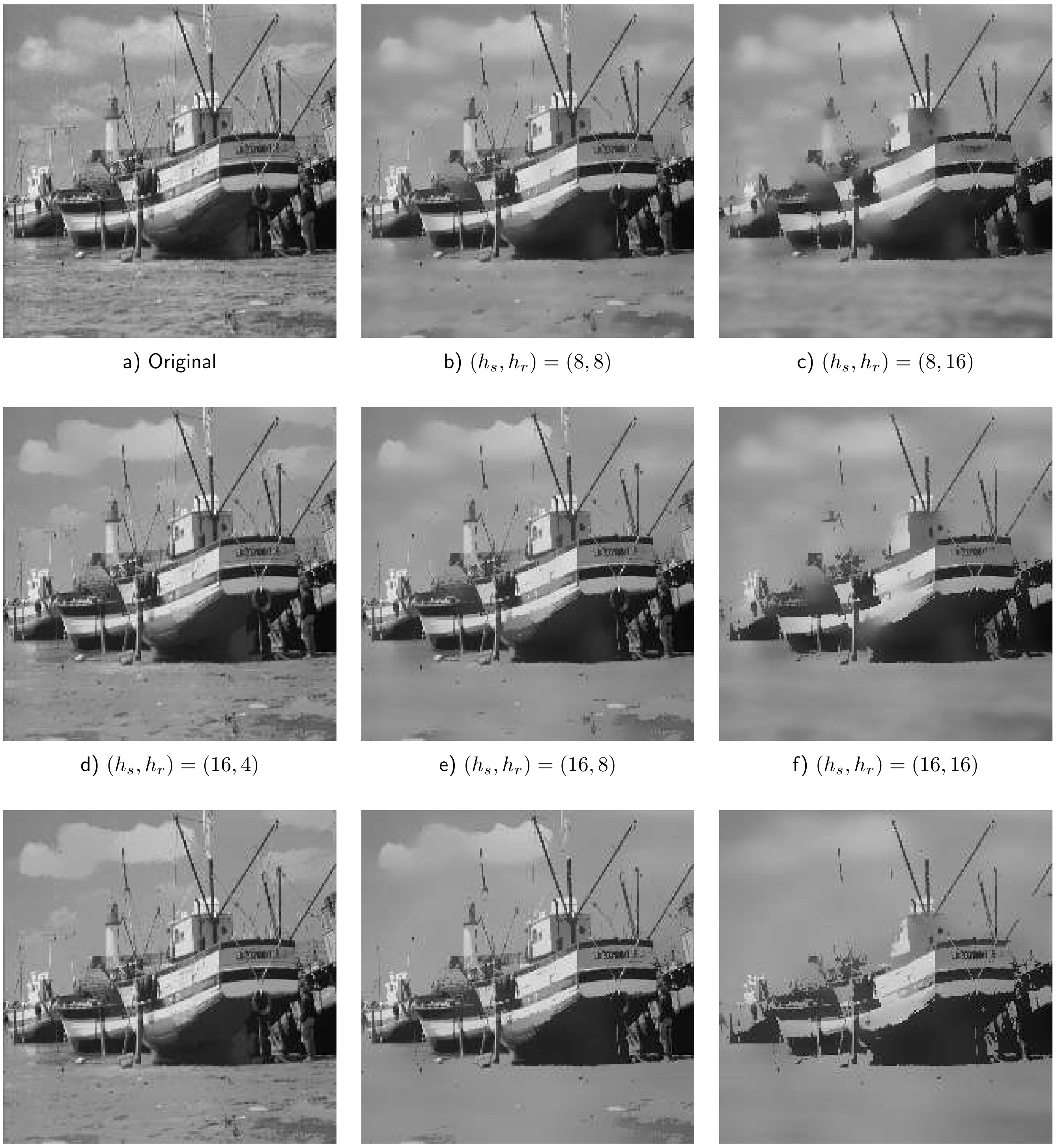
3 Implementation of the Mean Shift Algorithm

Our algorithm is implemented in the C++ programming language and based on code parts[2] from EDISON [5]. The front-end for libpng for reading and writing PNG images from Nicolas Limare has been used, and Region Adjacency List class and Transitive Closure[2], and the function for filtering is based on the implementation from ImageJ[3]. The mean shift algorithm runs in two phases, namely filtering and segmentation, and the last phase is labeling. In the start, an image is converted to L*u*v* color space. Users can decide to run the whole mean shift procedure or only the filtering phase. The input to mean shift algorithms is color radius, space radius, and the minimal region size. Our implementation uses the uniform kernel.

4 Case Study

We run several experiments on standard images Boat, Mandrill and Peppers with 512 × 512 pixels, Cameraman, House with 256×256 pixels to evaluate the filtering phase of Mean shift. All experiments were carried out with various spatial and range resolutions and the uniform kernel. Figure 3 shows the example of the filtering gray-scale Boat image. The image was filtered with eight combinations of parameters h_s and h_r, as can be seen in Figure 3. Evaluation of color filtering is done with the Peppers image as shown in Figure 4. For this experiment, parameters h_s and h_r were the same than for the gray-scale image. For both gray-scale and color images, it is noticeable the image preserving filtering, and more smoothing is for the larger range parameter. The details on Boat image are destroyed with a larger range value (more than 4), which is more noticeable on the gray-scale image. For assessing the filtering quality, structural similarity SSIM is used to evaluate the similarity between the input

[2]EDISON (Edge Detection and Image SegmentatiON), in an implementation of the Mean shift in the C++ language called EDISON and the source code is available. EDISON provides a versatile graphical interface (can also be run in the command line) for discontinuity preserving filtering, segmentation, and edge detection.
http://coewww.rutgers.edu/riul/research/code/EDISON/ implemented by Chris M. Christoudias, Bogdan Georgescu
[3]https://imagej.nih.gov/ij/plugins/download/Mean_Shift.java



a) Original b) $(h_s, h_r) = (8, 8)$ c) $(h_s, h_r) = (8, 16)$

d) $(h_s, h_r) = (16, 4)$ e) $(h_s, h_r) = (16, 8)$ f) $(h_s, h_r) = (16, 16)$

g) $(h_s, h_r) = (32, 4)$ h) $(h_s, h_r) = (32, 8)$ i) $(h_s, h_r) = (32, 16)$

Figure 3: Mean shift filtered Boat image

image and the mean shift filtered image. As shown in Table 1, SSIM between original image a) and filtered images b)-i) can be used to assess the quality of mean shift filtering. It can be seen that for both images $(h_s, h_r) = (16, 4)$ gives the best SSIM value.

In Figure 5 and Figure 6 several filtering and segmentation experiments were carried out for all test images. In this case, the parameters were used for the gray-scale image. Here, for display purposes, just one segmentation for each test image is shown, which corresponds to the best segmentation for each case. Mandrill image segmentation results are noticeably the worst. From our experiments,

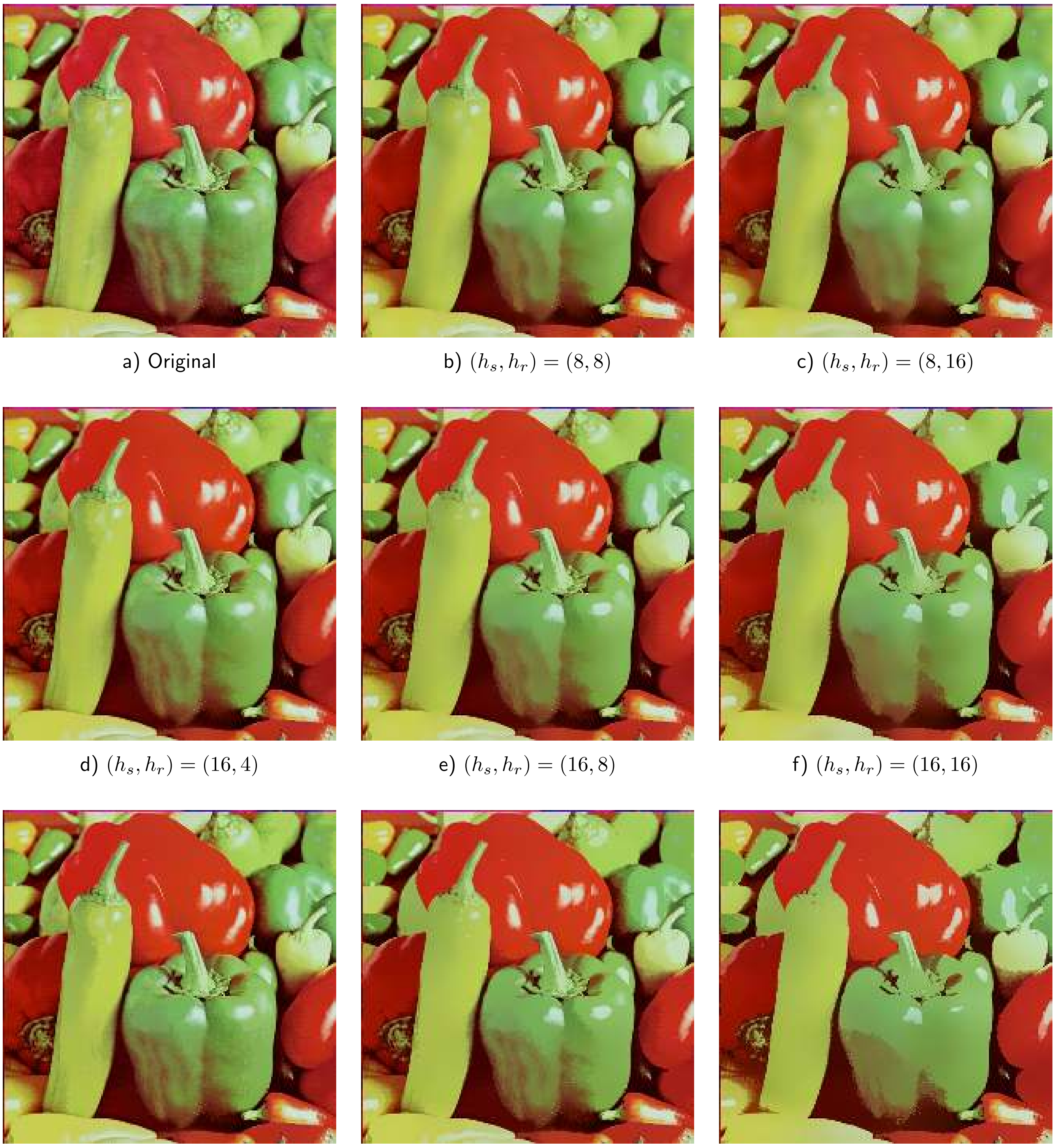
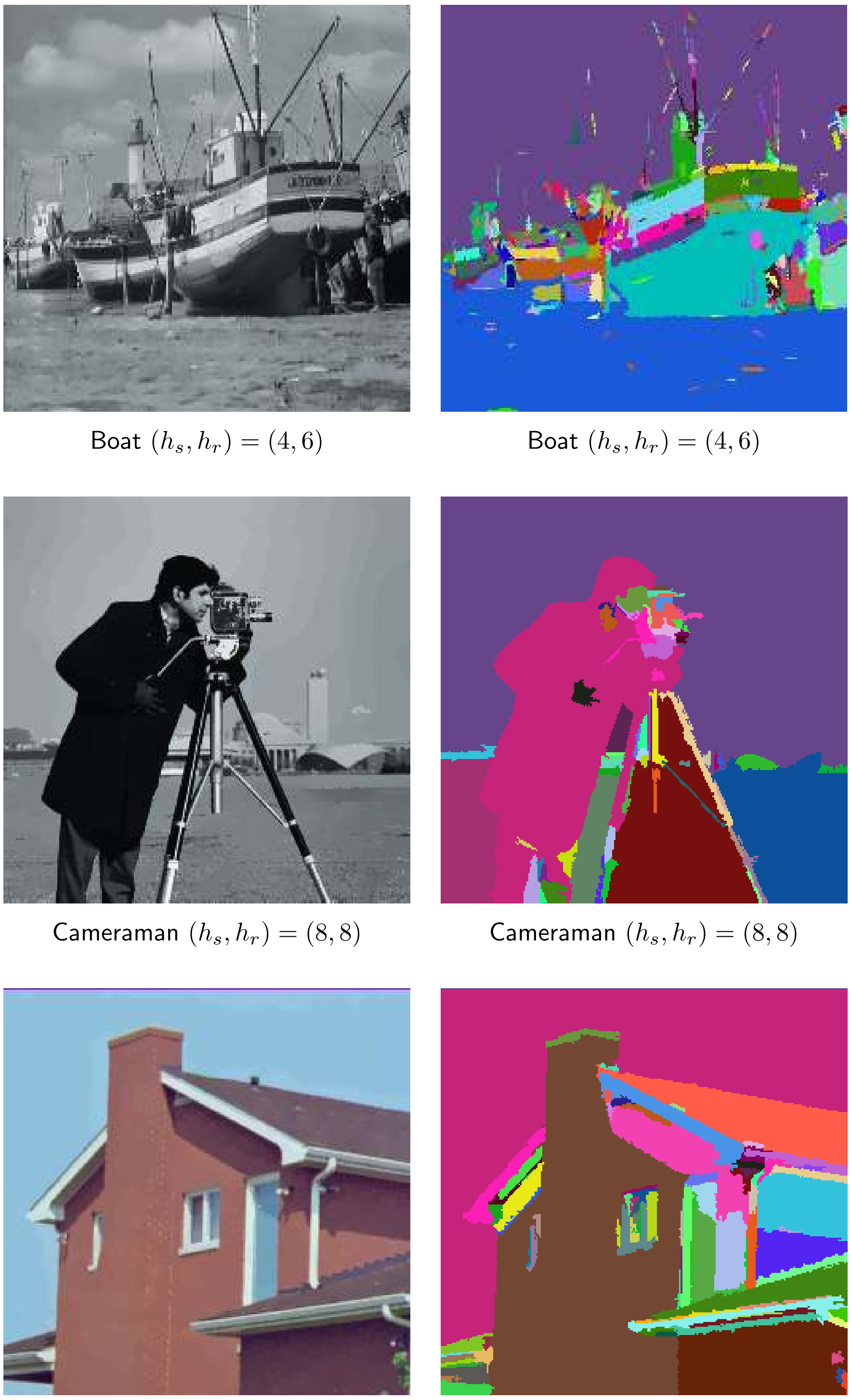

a) Original
b) $(h_s, h_r) = (8, 8)$
c) $(h_s, h_r) = (8, 16)$
d) $(h_s, h_r) = (16, 4)$
e) $(h_s, h_r) = (16, 8)$
f) $(h_s, h_r) = (16, 16)$
g) $(h_s, h_r) = (32, 4)$
h) $(h_s, h_r) = (32, 8)$
i) $(h_s, h_r) = (32, 16)$

Figure 4: Mean shift filtered Peppers image

Image	b)	c)	d)	e)	f)	g)	h)	i)
Boat	0.746	0.618	**0.871**	0.721	0.601	0.862	0.715	0.595
Peppers	0.974	0.957	**0.983**	0.907	0.947	0.982	0.966	0.935

Table 1: Comparison of SSIM for filtering experiments given in Figure 3 and Figure 4.

which are in accordance with [7], segmentation is not very sensitive to the choice of the parameters h_r and h_s.

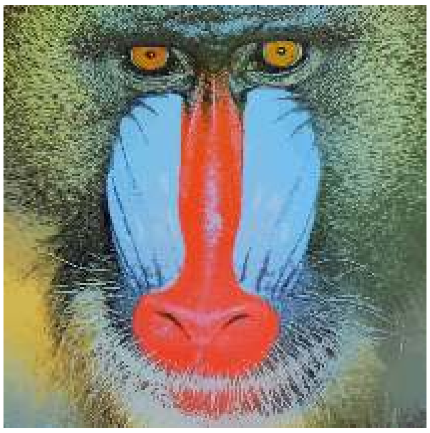

Boat $(h_s, h_r) = (4, 6)$ Boat $(h_s, h_r) = (4, 6)$

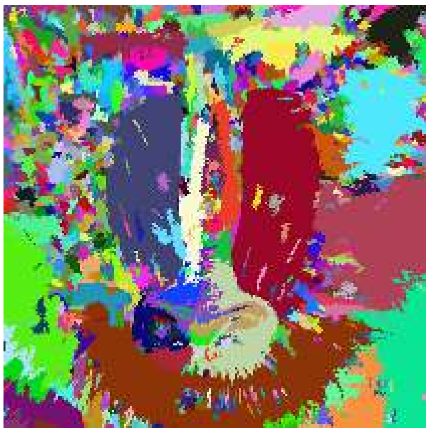
Cameraman $(h_s, h_r) = (8, 8)$ Cameraman $(h_s, h_r) = (8, 8)$

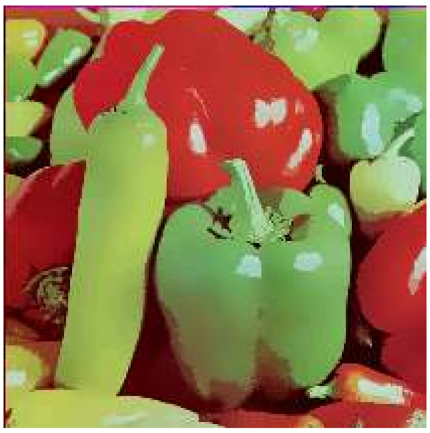
House $(h_s, h_r) = (4, 6)$ House $(h_s, h_r) = (4, 6)$

Figure 5: Mean shift filtered and segmented images: Boat, Cameraman, and House

Comparison of different sizes of the minimal region are carried out in Figure 7 and Figure 8. Here, the algorithm does not assign the same color for the corresponding regions, so different runs produce a the different coloring of the regions. The region of minimal area M for segmented regions in pixels of sizes 10 and 20 was compared. For the Boat and Cameraman, the segmentations are similar, only small regions disappear for the bigger minimal area M=20. For all experiments, the bigger value for M leads to a smaller difference in segmentation, which is small for almost all experiments. From this experiment, one can see the impact of the bandwidth on the number of clusters. Proper


Mandrill $(h_s, h_r) = (32, 8)$


Mandrill $(h_s, h_r) = (32, 8)$


Peppers $(h_s, h_r) = (16, 16)$

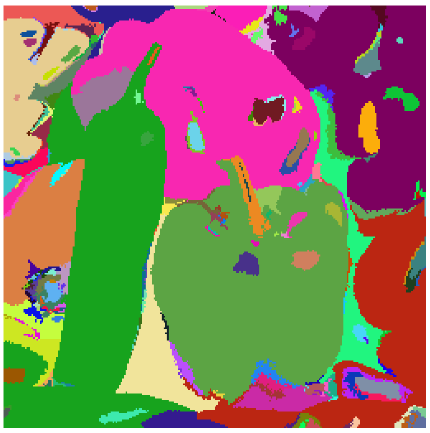
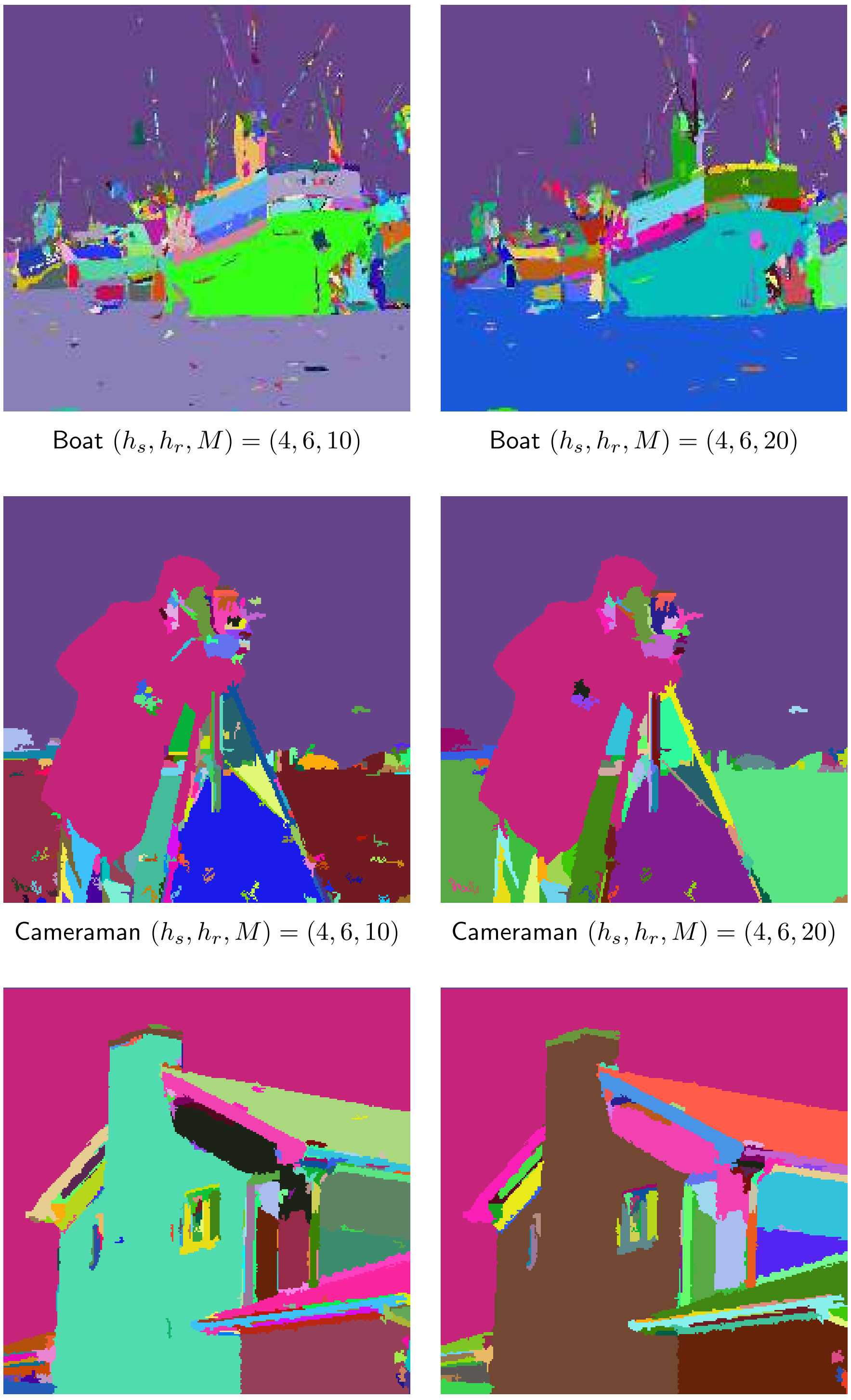

Peppers $(h_s, h_r) = (16, 16)$

Figure 6: Mean shift filtered and segmented images: Mandrill and Peppers

bandwidth is important, a large one might result in incorrect clustering and to eventually merging distinct clusters; on the other side, small bandwidth might result in too many clusters.

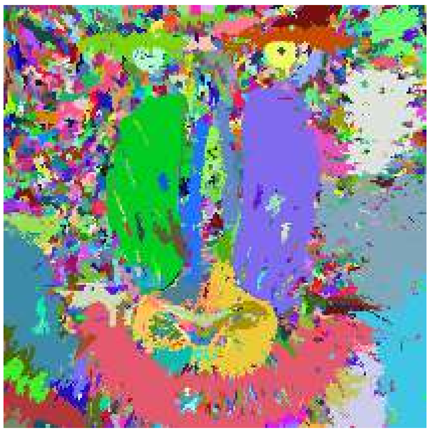


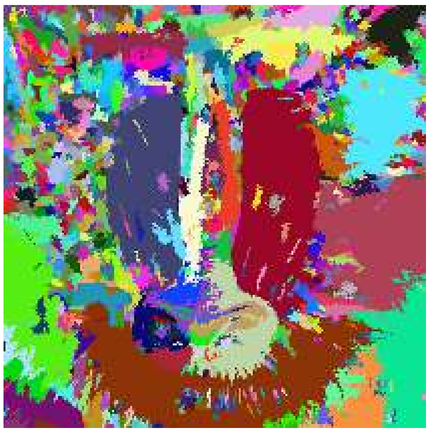
Boat $(h_s, h_r, M) = (4, 6, 10)$ Boat $(h_s, h_r, M) = (4, 6, 20)$

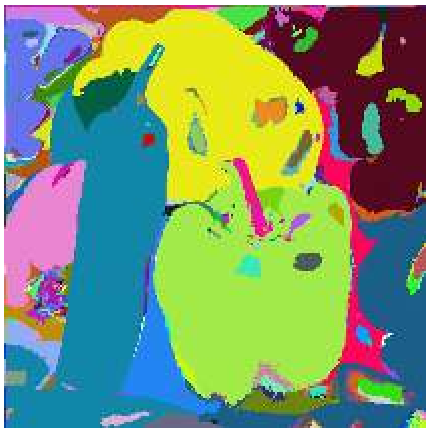
Cameraman $(h_s, h_r, M) = (4, 6, 10)$ Cameraman $(h_s, h_r, M) = (4, 6, 20)$

House $(h_s, h_r, M) = (4, 6, 10)$ House $(h_s, h_r, M) = (4, 6, 20)$

Figure 7: Comparison for different minimal regions; M=10 and M=20


Mandrill $(h_s, h_r, M) = (32, 8, 10)$


Mandrill $(h_s, h_r, M) = (32, 8, 20)$


Peppers $(h_s, h_r, M) = (16, 16, 10)$

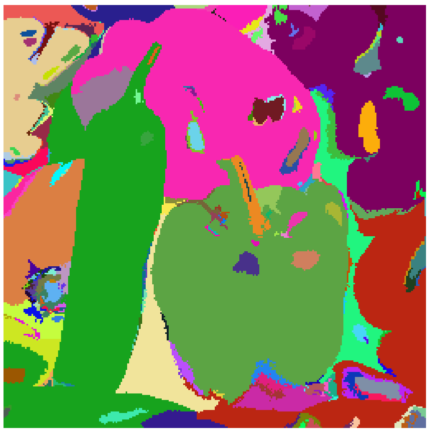
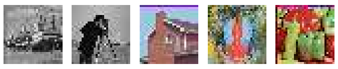

Peppers $(h_s, h_r, M) = (16, 16, 20)$

Figure 8: Comparison for different minimal regions; M=10 and M=20

5 Conclusions

In this paper, the detailed implementation of the mean shift algorithm was studied. An analysis of the algorithm and a well-commented implementation in the C++ programming language are provided. For testing purposes, a case study of the algorithm was presented for filtering and segmentation. Several standard gray-scale and color images were used for the experiments. Parameters for spatial and color radius and the minimal region size were evaluated. Future research can be done in two directions, parallelization of the proposed implementation, implementation of the variants of mean shift with quantization of the probability density function, methods of sampling etc.

Image Credits

 Standard test images[4]

References

[1] G.R. Bradski, *Computer vision face tracking for use in a perceptual user interface.* Intel Technology Journal Q2.

[2] M.A. Carreira-Perpinan, *Gaussian mean-shift is an EM algorithm*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 767–776. http://doi.org/10.1109/tpami.2007.1057.

[3] M.A. Carreira-Perpiñán, *A review of mean-shift algorithms for clustering*, arXiv preprint arXiv:1503.00687, (2015). https://arxiv.org/abs/1503.00687.

[4] Y. Cheng, *Mean shift, mode seeking, and clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17 (1995), pp. 790–799. http://doi.org/10.1109/34.400568.

[5] C.M. Christoudias, B. Georgescu, and P. Meer, *Synergism in low level vision*, in Object recognition supported by user interaction for service robots, IEEE, 2002, p. 40150. http://doi.org/10.1109/icpr.2002.1047421.

[6] D. Comaniciu and P. Meer, *Mean shift: A robust approach toward feature space analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (2002), pp. 603–619. http://doi.org/10.1109/34.1000236.

[7] D. Comaniciu, V. Ramesh, and P. Meer, *The variable bandwidth mean shift and data-driven scale selection*, in Eighth IEEE International Conference on Computer Vision (ICCV), vol. 1, IEEE, 2001, pp. 438–445. http://doi.org/10.1109/iccv.2001.937550.

[8] ——, *Kernel-based object tracking*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 564–577. http://doi.org/10.1109/tpami.2003.1195991.

[9] D. Dementhon and R.Megret, *Spatio-temporal segmentation of video by hierarchical mean shift analysis*, 2002. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.8958.

[4] http://www.imageprocessingplace.com/root_files_V3/image_databases.htm

[10] M. Fashing and C. Tomasi, *Mean shift is a bound optimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 27 (2005), pp. 471–474. http://doi.org/10.1109/tpami.2005.59.

[11] K. Fukunaga and L. Hostetler, *The estimation of the gradient of a density function, with applications in pattern recognition*, IEEE Transactions on Information Theory, 21 (1975), pp. 32–40. http://doi.org/10.1109/tit.1975.1055330.

[12] B. Georgescu, I. Shimshoni, and P. Meer, *Mean shift based clustering in high dimensions: A texture classification example*, in Ninth IEEE International Conference on Computer Vision (ICCV), IEEE Computer Society, 2003, pp. 456–. http://doi.org/10.1109/iccv.2003.1238382.

[13] K. Huang, X. Fu, and N.D. Sidiropoulos, *On Convergence of Epanechnikov Mean Shift*, arXiv preprint arXiv:1711.07441, (2017). https://arxiv.org/abs/1711.07441.

[14] M. C. Jones, J. S. Marron, and S. J. Sheather, *A brief survey of bandwidth selection for density estimation*, Journal of the American Statistical Association, 91 (1996), pp. 401–407. http://doi.org/10.2307/2291420.

[15] F. Meng, H. Liu, Y. Liang, L. Wei, and J. Pei, *A bidirectional adaptive bandwidth mean shift strategy for clustering*, in IEEE International Conference on Image Processing (ICIP), 2017, pp. 2448–2452. http://doi.org/10.1109/icip.2017.8296722.

[16] M. K. Singh and N. Ahuja, *Mean-shift segmentation with wavelet-based bandwidth selection*, in Sixth IEEE Workshop on Applications of Computer Vision (WACV), 2002, pp. 43–47. http://doi.org/10.1109/acv.2002.1182154.

[17] W. Tao, H. Jin, and Y. Zhang, *Color image segmentation based on mean shift and normalized cuts*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 37 (2007), pp. 1382–1389. http://doi.org/10.1109/tsmcb.2007.902249.

[18] C. Tomasi and R. Manduchi, *Bilateral filtering for gray and color images*, in Sixth International Conference on Computer Vision, Jan 1998, pp. 839–846. http://doi.org/10.1109/iccv.1998.710815.

[19] J. Wang, B. Thiesson, Y. Xu, and M. Cohen, *Image and video segmentation by anisotropic kernel mean shift*, in European Conference on Computer Vision (ECCV), T. Pajdla and J. Matas, eds., Springer Berlin Heidelberg, 2004, pp. 238–249. http://doi.org/10.1007/978-3-540-24671-8_19.

[20] K. Zhang, J. T Kwok, and M. Tang, *Accelerated convergence using dynamic mean shift*, in European Conference on Computer Vision (ECCV), Springer, 2006, pp. 257–268. http://doi.org/10.1007/11744047_20.