Hierarchical Methods of Image Segmentation

VŠB - Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Summary of Philosophiæ Doctor Thesis

Author: Ing. Milan Šurkala
Supervisor: doc. Dr. Ing. Eduard Sojka

September, 2016
Abstract

Segmentation and filtration of various datasets is still very alive computation problem. Many approaches for such computations exist. They can provide different outputs, speed and many other properties. I am going to deal with the mean-shift method that can be used for filtering and segmenting various datasets. In my case, I am going to filter digital images consisting of pixels. The mean-shift methods achieve good filtration results but most of them suffer from a lower speed. Mean shift can also be used for object tracking where much faster versions of this method exist and are widely used in practice. In this thesis, I am going to talk about the image segmentation problem using various mean-shift approaches and I will present some speed and quality improvements. For this goal, I will use mainly various types of hierarchical approaches. Filtration abilities and the stability of the algorithms will be studied too. Speed issues and problem of a proper setting of parameters will be studied.
Contents

1 Introduction 2

2 State of the Art 5
  2.1 Blurring Mean Shift 5
  2.2 General Mean Shift 7
  2.3 Theoretical studies of Mean Shift 7
  2.4 Further improvements of MS approaches 8
  2.5 Hierarchical approaches applied to mean-shift method 12
  2.6 Various mean-shift-like methods 13

3 My Contributions to the Area 14
  3.1 Hierarchical Mean-Shift Methods 14
  3.2 Setting the appropriate bandwidths in the hierarchical mean-shift methods 17
  3.3 Optimization of Evolving Mean Shift 19
  3.4 Layered Mean-Shift Methods 21
  3.5 Hierarchical Layered Mean-Shift Method 22
  3.6 Mean-Shift Method with Flatness Constraints 25
  3.7 Hierarchical Fast Blurring Mean-Shift for Head Position and Center Estimation in Depth Images 28

4 Conclusion 34

5 Author’s Bibliography 43
  5.1 Publications Related to the Thesis 43
    5.1.1 Conference Proceedings 43
    5.1.2 PhD Workshops 44
  5.2 Publications Unrelated to the Thesis 44
    5.2.1 Conference Proceedings 44
Chapter 1

Introduction

Clustering is a technique that is often used in data mining, statistical data analysis, image segmentation and filtration. These tasks try to find similar data in datasets and merge them into groups that are called clusters. In digital images, one should group pixels with similar properties and create segments that represent an object or some important areas in processed images. Many clustering algorithms exist and are still developed, they improve many of their properties such as speed, stability or quality of output. There are methods that deal not only with the classical 2D images but also with the 3D depth data.

All of these methods have to deal with the problem of a correct assignment of the data points to corresponding clusters. Moreover, there is one very serious problem in evaluation of these results. We have to say which result is the best and every human has a different opinion for such a problem. Some would prefer better details and more detected objects, some would prefer a lower number of segments and only the most significant objects. In [42], the segmentation results carried out by the humans are presented and it is obvious that even humans segment each image differently. They differ in the shape of segments, but the biggest difference is in the number of segments. Therefore, it is very difficult, or even impossible, to judge which segmentation method gives the best segmentation result because there is no objective evaluation for that.

The authors of new methods are trying to improve the speed of algorithms, the quality of segmentations and others. In this thesis, I especially, but not exclusively, focus on a significant reduction of computational time of the mean-shift segmentation method. Moreover, I try to preserve or even improve very good segmentation abilities of the algorithms I study the trade off between the speed gain and a change of segmentation results.

In this thesis, I focus mainly on the Mean Shift (MS) method [22]. The mean-shift method is usually used for filtering the datasets, some of its modifications are also suitable for segmenting data. In digital images, filtering can eliminate or at least reduce noise, segmenting is useful for finding and tracking the objects. Because of many tasks where MS can be used exist, there are also many various types of MS that are optimized for their goals.

Original MS was presented in 1975 [22], but the idea was not widely accepted and this method was forgotten for almost 20 years. The method has a problem with very slow speed even with today’s hardware and the first version was not suitable for practice. Since 1995, this method is studied and improved by many researchers till now. Depending on a specific type of implementation and version of MS, filtration and
CHAPTER 1. INTRODUCTION

Figure 1.1: Process of the MS method when general data are used.

segmentation quality can differ very much. Some MS variations are more useful for filtration, some of them are better in creating the precise segment boundaries. Even higher differences can be achieved between the computational times of these methods.

Mean Shift is an iterative method that seeks for the position with the highest density of data points. Because I am focused on digital images, the data points I am working with are represented by image pixels. Like in all the mean-shift methods, I have to set searching kernel bandwidths. These parameters describe the allowed similarity (difference) between a pair of data points that should be potentially in one segment. This applies with truncated kernels only. The number of dimensions and bandwidths is not limited in MS but there are usually two types of bandwidths in digital images. The first one is a spatial bandwidth that limits the size of the searching kernel in $x$ and $y$ axis and they are usually the same. The second one is a range (luminance) bandwidth that deals with luminance (colour) differences. The values of both bandwidths influence the size of the segments and the rate of filtering the data.

If the colour images are used, the range bandwidth is usually the same for all three colour channels in RGB, YCbCr or different colour space. For each image pixel, the pixels in its spatial neighbourhood (given by the spatial bandwidth) are examined and their weighted distance is computed according to a kernel function used. This distance is given by the spatial and range distance (difference in $x$, $y$ and luminance axes). If the examined pixel does not belong to the kernel (the sphere with a radius given by the spatial and the range bandwidths), the computation does not continue with truncated kernels because the contribution of such a pixel would be zero. Otherwise, the contribution of the examined pixel is computed. When non-truncated kernels are used, all pixels have at least small contribution to the computation. The closer the pixel is and the more similar luminance the pixel has, the contribution of the pixel is larger because the distance is smaller. If non-truncated kernels are used, the computation suffer from a very slow computation speed. The mean is computed
in the position where the number of the most similar pixels is the highest and the pixel is shifted to this position. The process is visible on the Fig. 1. The black pixel examines its surroundings using the mean-shift kernel window and moves to the position with a higher density of data points. The same process of searching for the higher density is repeated until convergence and the position of the highest density is reached.

The resulting segment is formed by all the pixels that converged to the same position that is called an attractor. The data points that are distinctly different in their position, luminance or both, tend to converge to another attractor and create another segment in a different location. The result of segmentation is a set of segments (attractors and their corresponding points). In a comparison with the \(k\)-means algorithm [41], Mean Shift is generally considered as a non-parametric method. To be precise, we need to set only bandwidths and type of kernel function. If variable-bandwidth version of the method is used, one does not have to specify the bandwidths as they are going to be specified by the algorithm.

In recent years, the development of the mean-shift method is more focused on the problem of object tracking. My research is focused on a reduction of the computational time with the smallest influence to the segmentation quality if possible or it should even improve the results. Whereas object tracking using Mean Shift can be easily done in real-time, the segmentation of digital images can still take at least several seconds per a small image and the first versions of the MS method may take even hours to finish properly. My new methods use small bandwidths in order to improve the speed and the hierarchical and the layered approaches are optimized to lower or to even eliminate the oversegmentation problem.

In the next chapter, the state-of-the-art is presented. I focus especially on the segmentation versions of mean-shift methods but some other segmentation methods and tracking mean-shift versions will be mentioned too.

I provide a description of my new methods in Section 3. My methods include the hierarchical versions, the layered versions, the combined hierarchical layered variants, I also present some modifications to existing MS methods that can be applicable also for my hierarchical and layered MS and a special MS method using the minimising of energy functional. A fast mean-shift method for a quick head position and center detection in depth data is presented too. This method reduces the dataset size in order to speed up the algorithm for a real-time usage with the tradeoff in a lower precision. I also carried out many experiments showing the results of my work. Proper setting the parameters in order to achieve stable and fast computation with visually good segmentation results is discussed too. All author’s publications are presented in the last Chapter 5.
Chapter 2

State of the Art

This chapter is dealing with various mean-shift algorithms that emerged in the 40-year history of the algorithm.

2.1 Blurring Mean Shift

The original mean-shift method was firstly presented in 1975, but almost no one tried to work with it for the following 20 years. Very few papers dealing with the mean-shift method emerged until the year 1995 when Cheng presented generalized mean-shift method. Since then, both methods started to be developed dramatically. In the meantime, many papers and books were dealing with statistical properties and density estimates that are also used in the mean-shift methods. Density estimation for statistics and various data analysis was deeply discussed by Silverman in 1986.

The blurring version of the mean-shift method got its name in 2006, when Carreira revisited it and brought some new ideas. The idea of Blurring Mean Shift (denoted as BMS) is relatively simple. The place with the highest density of data points is computed for each pixel in the processed image. This place is given by the similarity to the computed pixel by so called bandwidths in a kernel function. If the pixel is covered by the kernel function (it is not very far away spatially and it is not very different in colour), it is included in the computation of the mean and it influences the size and the direction of the final mean-shift vector. The mean-shift vector is the difference between the former position of the data point and the newly computed one. Only one shift per pixel is processed at this stage and these shift vectors are computed for each pixel in the image. All pixels are shifted according to their computed mean-shift vectors and new iterations for the modified dataset follow using this updated dataset. In the next iteration, all shift vectors for all (already shifted) pixels are computed, then all the shifts are processed and so on. The output from the previous iteration is considered as the input for the next one and, therefore, this is called the blurring process. Hence this method got the name Blurring Mean Shift.

Many various types of kernel functions can be used. They can influence the segmentation result, the speed and other algorithm properties. We can distinguish between two types of kernel functions. The first one is the group of truncated kernels, where all pixels that are further away than is the hypersphere given by the bandwidths are not included in the computation (the function gives the zero output if the distance is higher than 1). On the other hand, the non-truncated kernels include all the points...
in the computation and bandwidths only change the shape of the kernel function (the weight of the pixels in the computation).

The most simple kernel function is the uniform kernel that gives the value of 1 in all positions that are covered by the kernel. Other types of kernels usually give smaller output for the points in larger distances (larger values of parameter of the kernel function).

The kernel density estimator with the bandwidth \( \sigma \) (in many papers also denoted by \( h \)) is defined by

\[
p(x) = \frac{1}{N} \sum_{n=1}^{N} K \left( x - x_n \right) .
\] (2.1)

The term \( K(x) \) is the kernel function. It is nonnegative and it is used as a weighting function. When the parameter of the function is high (the data point is far away and/or it has a different luminance), the output of the function is low or even zero. The index \( H \) stands for a symmetric positive definite \( d \times d \) bandwidth matrix. The notation with fractions can be used too, see (2.5) - both notations are equivalent and widely used in practice. The kernel nonnegative function \( K(x) \) can be written as follows

\[
K(x) = k \left( ||x||^2 \right) .
\] (2.2)

\( K \) is the kernel, if there exists its profile \( k \) that is nonnegative, non-increasing \( (k(a) \geq k(b) \text{ if } a < b) \), piecewise continuous, and \( \int_{0}^{\infty} k(r)dr < \infty \). The kernel can be defined by the kernel function itself or its profile that is defined only if \( x > 0 \). The Epanechnikov kernel profile is presented in the equation

\[
k(x) = \begin{cases} 
1 - x, & \text{if } 0 \geq x \geq 1 \\
0, & \text{if } x > 1
\end{cases} .
\] (2.3)

We can rewrite it to the form of the kernel itself as follows

\[
K(x) = \begin{cases} 
\frac{1}{2} x^{-1} (d + 2) (1 - ||x||^2), & \text{if } ||x|| \leq 1 \\
0, & \text{otherwise}
\end{cases} .
\] (2.4)

In a practical application, the computation of the mean-shift vector is the most important step. This vector can be expressed by the equation

\[
m_{\sigma,k}(x) = \frac{\sum_{i=1}^{N} x_i k \left( \frac{||x - x_i||}{\sigma} \right)}{\sum_{i=1}^{N} k \left( \frac{||x - x_i||}{\sigma} \right)} - x.
\] (2.5)

The term \( x \) is the processed data point, \( x_i \) are the neighbouring points and \( k \) is the kernel function. After each iteration, the data points are changed according to the computed values. The blurring mean-shift process is ended when the shifts are zero or very small (this limit - stopping criterion - has to be set prior the computation).

Non-truncated kernels can introduce some unwanted behaviour. In the first stage, final segments emerge and the segmentation result is complete (all neighbouring pixels converged to their appropriate attractors). In the second stage, all these attractors
tend to move to each other and are slowly generating one big final segment. The process has to be stopped after the first stage. In [9], Cheng dealt with the convergence in the situation when BMS does not collapse to already mentioned one point. Unfortunately, the stopping condition was not provided as well as in the original paper. The stopping criterion after the first stage of algorithm was provided in [5]. The key is in a comparison of the entropies of histograms because simple threshold cannot be used. Some properties of the segments are changing in the second stage but the entropy of the histogram is still constant. Therefore, the histogram entropy value is a suitable stopping criterion for GBMS.

2.2 General Mean Shift

General Mean Shift (simply denoted by MS) was presented in the paper by Cheng [9]. This method handles a dataset differently. The input dataset remains the same for the whole computation and only computed positions of pixels are changed. Therefore, MS uses two datasets. The first one is the original dataset (the image that should be segmented) that is still the same for the whole computation and serves as the neighbourhood of the computed data point. The second dataset is formed of these computed data points and it is changing during the computation. Therefore, the computed pixels stored in the second dataset are floating above or under the luminance surface given by the first dataset. The drawback of this method is that in completely flat areas, data points stuck in their place and do not form any cluster. Another drawback is in much slower computational speed.

Note that the only difference in the general MS equation in comparison with the BMS equation (2.5) is the lower index 0 of the \( x \) term in the bracket. It denotes the neighbourhood of the processed pixel.

2.3 Theoretical studies of Mean Shift

Both mean-shift approaches were studied theoretically too. The algorithms were very well described by Fashing and Tomasi in [20]. In this paper, the relation between MS and the Newton’s method was added and the proof that MS is a bound optimization was mentioned too. Carreira-Perpiñán [6] found that there is a connection between Gaussian Mean Shift and the EM (expectation-maximisation) algorithm. In the paper [45], it was shown that Gaussian Blurring Mean Shift minimizes the Renyi’s quadratic entropy whereas Gaussian Mean Shift minimizes the Renyi’s cross entropy.

According to [55], the classical stopping criterion based on the entropy level is not the best for digital images. It does not consider a spatial information between images and, therefore, there is a possibility to obtain very small entropy even with two quite different images that have similar pattern in different positions. The authors presented stopping criterion with strong equivalence that segments images better.

Mean Shift was also studied by D.Comaniciu and P. Meer in 1997 [12]. They defined three parameters for class definitions and three classes of the mean-shift outputs (undersegmentation, oversegmentation and quantization).

In [7], some properties of the mean-shift method for the Gaussian kernel were rediscovered, although the term mean shift was not mentioned. The author searched for an algorithm that finds modes of Gaussian distributions and the convergence for
covariance matrices was also proved.

D. Comaniciu and P. Meer also presented two very important papers in 1999 and 2002. In the first one, they proved the convergence of the MS method for discrete data. The bandwidths \( \sigma_r \) in the range domain (luminance, colour) and \( \sigma_s \) in the spatial domain (x-axis and y-axis) were presented too. The authors also provide an acceleration technique. A supportive matrix structure of the spatial domain was used for faster indexing and searching the pixels. Similar lattice structure is one of the improvements used in my hierarchical methods.

Mean Shift was further studied by D. Comaniciu and P. Meer in 2002. They dealt with the convergence of the mean-shift method as a smooth trajectory property. It was proven that the cosine of the angle between two consecutive mean-shift vectors is always positive if the Gaussian kernel is used. Besides the relation to the kernel regression and to the location of M-Estimators, the paper dealt with the highly important problem of the bandwidth selection. The size of the bandwidths influences the filtration and segmentation results, the quality and the speed of the computation. Four different approaches for the setting of the bandwidth parameter were presented. The first approach has a statistical motivation, it should minimize the Asymptotic Mean Integrated Squared Error (AMISE). A stability of a decomposition is used in the second approach. The largest operation range over which the decomposition does not change (has the same number of segments) is taken as the bandwidth. Therefore, the third approach compares an inter- and intra-cluster variability. In the last one, the bandwidth is selected manually by a user.

The convergence of various mean-shift methods was deeply studied in the Y.A. Ghassabeh’s dissertation. New versions of the Subspace Constrained Mean Shift method were presented here and the author presented new potential applications for mean-shift methods like finding straight lines in digital images. The relation between Mean Shift and the Nadaraya-Watson kernel regression was also presented.

2.4 Further improvements of MS approaches

One of the most used practical implementations of the mean-shift algorithm is EDISON and many authors are trying to improve their algorithms in comparison with this. One of problems of MS to solve is how to set the appropriate bandwidth. The variable bandwidth was studied in and Variable Bandwidth Mean Shift was presented. In this paper, two adaptive estimators were introduced. The first one is called the balloon density estimator and the second one was marked as the sample point density estimator. Both of the estimators should adjust the bandwidth for each data point according to the local data density; therefore, each data point can use a different bandwidth. The estimators respect the data point distribution more appropriately.

The proof of proportionality of the estimators was given in .

Comaniciu revisited bandwidth selection problem in 2003 again. The bandwidth at a partition level is evaluated first. It is computed for each mode and the mean and the covariance are both associated with the corresponding point \( x_i \). Next, the bandwidth at a data level is computed. Normal distributions are computed according to the estimates. Then the most stable pair is selected for each data point \( x_i \). It was proven that this method correctly sets the appropriate bandwidth for the majority of data points and, therefore, a better quality of segmentation can be achieved.

The best stability measure across the segmentation results was used in Two-Set
Mean Shift Clustering [50] where the authors studied the number of produced segments and picked up the bandwidth in the middle between two changes of the number of segments. In order to make the mean-shift computations fast, the Locality Sensitive Hashing (LSH) data structures were used [28], [32]. These structures reduce the curse of dimensionality.

Variable bandwidth was also studied in [47], where the authors improved classical Adaptive Mean Shift that uses the \( k \)-nearest neighbours to select the bandwidth. The new Weighted Adaptive Mean Shift uses weighted bandwidths that lower the noise sensitivity and also make the algorithm less prone to curse of dimensionality. The comparison between various bandwidth selectors was also given in [8].

Microsoft Research team [64] used not only a variable bandwidth but also a variable shape of the kernel. This Anisotropic Kernel Mean Shift separated kernels for each of the domains. They were adapting the spatial and the range bandwidth separately and allowed the asymmetrical shape of the kernel according to the local structure of the data. This approach is also less sensitive to the choice of the initial bandwidth. The bandwidth choice problem was also studied in [67].

In the paper [31], the authors presented the improved mean-shift method for segmenting natural images. This algorithm presented a variable color (luminance) bandwidth. Other improvements consisted of a direct density searching, optimization of merging the segments and the authors also used the process of elimination of texture patches.

In 2003, the Improved Fast Gauss Transformation (IFGT) was presented and the authors applied it to the kernel density estimation and tested it with the Mean Shift [70]. Two improvements to the standard FGT were used. The farthest-point algorithm used for adaptively subdividing a high dimensional space is the first one and the multivariate Taylor expansion that reduced a computational speed of the fast Gauss transform is the second one. MS with IGFT acceleration runs in a linear time with satisfactory results.

Better computational times can be achieved thanks to usage of the neighbourhood consistency. This topic was studied in the paper [75] and later used and improved in [29]. New Fast Mean Shift (FMS) uses this consistency for the computation of the kernel density estimate. It uses only few representatives of the dataset and, therefore, it significantly reduces the input dataset. FMS expects that close data points will probably have very similar KDE function, and very similar mean-shift vector. The basic segmentation of the dataset is the result of the first phase of the algorithm when centers according to the distance between data points are computed. The standard mean-shift procedure is done in the next phase. The MS computation is carried out with the centers instead of all data points. Similar approach was described in [29], where BMS was used.

In 2009, Evolving Mean Shift (EMS) [77, 73, 72] was revealed. It works as an energy minimization algorithm. EMS searches for the pixel with the highest energy (largest mean-shift vector) whose movement will reduce the energy of the dataset most. Then the point is shifted according to the selected mean-shift vector. The mean-shift vectors in an affected neighbourhood have to be recomputed. It has a lower number of iterations per point, but high overheads are a significant problem. In each iteration, one have to find the maximum energy and recomputation of the affected mean-shift vectors has to be done too. Therefore, a naive implementation of EMS can be very slow in practice.
EMS provides nice filtration abilities and can utilize a variable bandwidth. Therefore, it belongs to the mean-shift methods with an adaptive bandwidth and a blurred dataset. The EMS convergence was also provided.

In the paper "Evolving Mean Shift: a Mathematical Tool for Data Analysis", the authors presented some acceleration techniques for the EMS algorithm. The authors used the pilot density estimator to update the bandwidth. The acceleration technique that was used decreased the size of the dataset by a uniform sampling. It has skipped approx. 98% points. It is highly probable that only the points in the high density areas were selected. EMS vectors are computed only for the remaining data points. This method is not deterministic.

Different Fast Mean Shift was presented in 2009 [21] but it is only one of many methods denoted as Fast Mean Shift. The input dataset is also reduced in order to speed up the computation of the density estimate. The hierarchical approach is not used for a reduction like in the superpixel algorithm [30], the authors pick up only few random samples from the neighbourhood. The acceleration is almost linearly proportional to the sampling used. One have to expect more visible line scattering.

The third ”Fast Mean Shift” [31] discretised spatial image and range coordinates by introducing a regular grid. It precomputes the values for all the data points. The grid values were used as well as the trilinear interpolation. First, an intensity range is computed according to the luminance of the computed pixel. Then the pixels in the neighbourhood are inserted into two priority queues, the insert-event one according to the minimal intensity value and the delete-event one according to the maximal intensity value. Both queues are sorted. All intensity values are processed and for each pixel, counter is incremented or decreased according to their properties. The average result (vector) is computed according to the resulting sums and the number of pixels that contributed to the computation. The computational time is roughly equivalent to the time of only one mean-shift iteration.

The fourth ”Fast Mean Shift” [33] was presented in 2014 and utilizes very simple idea of an enlargement of the mean-shift vector. This FMS was based on the idea that the direction of the movement of the processed point usually stays almost the same or it is similar in the following iterations. The mean-shift vector can be enlarged and because of that, the convergence can be faster. Authors proved that this addition should be in the interval \(-1 < t < 1\). Many tests were carried out and it was proved that ideal length of the mean-shift vector is 1.5-times the original one. If the multiplication factor is larger than that, the average time rises because of a higher oscillation. It achieved approx. 4-times better speed.

In 2015, the Improved Fast Mean Shift algorithm for remote sensing image segmentation [79] was presented. This method uses the superpixel structure in the first stage. The method uses adaptive range bandwidths that are set according to the difference between a luminance of the computed pixel and the average luminance of the clustered areas. It produces better segmentation results and it is also slightly faster than original MS (EDISON implementation).

Some papers [17] deal with high computational demands of the mean-shift method by using a powerful hardware and parallel implementation for this dedicated hardware. For example, the authors used an unrolling the outer loop, pipelining the inner loop and other techniques. The authors of this paper are talking about three decimal orders speedup.

In a few last years, new Fuzzy Flood Fill Mean Shift (FFFMS) was studied. This
method was firstly introduced in 2010 by H. Kang, S. H. Lee and J. Lee [35]. The fuzzy kernels and the flood fill technique are used. These kernels overcome the problem of oversegmentation. FFFMS uses fuzzy kernels that have a limitless contribution of very distant pixels as the spatial bandwidth is not set at all. Therefore, it is similar to the non-truncated kernels. The limit in range domain still exists. The method has the non-truncated kernel in the spatial domain but concurrently this kernel is truncated in the range domain. A very high speedup (up to two decimal orders) was also observed by the authors.

Many interesting improvements were developed in object tracking area [53]. For example, Adaptive Pyramid Mean-Shift was presented in [38]. It is one type of the hierarchical mean-shift methods which are working in video sequences. It uses a color histogram for a comparison of the tracked areas. Several different sizes of digital images are used and the algorithm seeks for the positions of objects in the coarse images. Then it performs the search in a more detailed image with a higher resolution. It is able to track the object globally and it is still able to set its position very precisely locally because of the images with a higher resolution in the second stage. Methods that are searching the attractors in the previous image [3] are fine for the video sequences, but they are not applicable for static images. There are methods working with 3D or depth data (from Microsoft Kinect or Intel RealSense cameras, for example) too, for example [76].

The results from previous image (backward tracking) can be used for an adaptive change of the bandwidth size [40] in order to be able to track objects that are moving closer to the camera. This new method can adaptively update the size of the window.

In recent years, Manifold Blurring Mean Shift [66] was presented. This method improves well known Blurring Mean Shift by using the additional step that removes a shrinkage of data along the manifold. The manifold structure is estimated by a local PCA and the parallel motion of the point is subtracted.

Although the general MS method works with the Euclidean spaces, Subbarao and Meer [56] presented Nonlinear Mean Shift that is able to cluster data even on analytic manifolds. It extends the MS method that was able to cluster data on Lie groups [59]. The paper [59] also did not use classical Euclidean distance and the authors used the cosine distance metric for diarizing telephone speech conversations.

Interesting approach was presented in Fast Integral Mean Shift [37]. The color space is discretized, therefore, integer values are used instead of the real ones. Because the paths are discrete, they can be easily memorized in the form of cubes. The algorithm uses a volume integral (voxels) and it greatly reduces the computational complexity. Instead of demanding computation of the mean, only few additions and subtractions are carried out. The preparation of the voxel cubes takes a lot of time but the algorithm itself is very fast. The complexity of MS was lowered from $O(n^2)$ to $O(n)$.

Although mean-shift methods are not usually supervised methods and belong to non-parametric ones, there is also the Semi-Supervised Kernel Mean Shift method [4]. This method emerged in 2014 and brought supervising for the mean-shift segmentation algorithm. The segments can be roughly marked by a user and the MS process will find the exact boundaries of these segments.
2.5 Hierarchical approaches applied to mean-shift method

The first mentions about the hierarchical mean-shift methods emerged in 2002 [18] but the hierarchical approaches are widely developed only last few years. This method was used for acceleration of the mean-shift method in video sequences. In the beginning, a small spatial bandwidth is used with the full dataset and the computation is fast because of the small searching window. The resulting segmentation is used as the input for the following computation (reduction of the dataset). Because of reduced dataset, the computation can also be fast. In this paper, data points were represented by 7D coordinates. Three dimensions are color components, two following dimensions are motion angle components and the last two represent motion position components. Therefore, the segments also have similar motion properties. This algorithm is suitable for solving the object tracking problem. A small increase of the kernel size between stages was used, mostly between 1.25 and 1.5 multiple.

A similar approach was presented by Vatturi [60]. Multiple stages of the mean-shift method, each with the different bandwidth, were also used. The initial bandwidth for the first stage is computed as the minimum non-zero distance between any two points in the input dataset. After each stage, the bandwidth is multiplied by a scalar value $k$. Choi and Hall [10] also presented the technique of carrying out computations that will make small clusters from original data and these clustered data will be used in various density estimators instead of the original one. Their method can be used also for orthogonal series, histosplines, singular integrals, and other methods.

The hierarchical approach using the oversegmented result as an input for the following stage is also used in the technique utilizing superpixel structures [46]. The superpixel algorithm originally used the Normalized Cuts algorithm. The algorithm used a moving, merging, or splitting the segments according to the texture and brightness similarity, contour energy, and curvilinear continuity. During recent years, many other techniques were incorporated with the Superpixel technique, for example the Watershed Transformation [39], the Quick Shift [24], and also the Mean Shift [65]. Many improvements for the superpixel method exist, for example [38].

In [74], multi-layered histogram back projection [58] is used. It divides an object (for example, a tracked human) into three parts in other to avoid color differences in each part and to aggregate the results of each mean-shift procedure.

The mean-shift algorithm can be also mixed with the Markov Random Fields [78]. There are more methods which are using a mean-shift method and the hierarchical approach. For example in [48], the centroids of the first stage were used as an input for the next one. Post-processing of a Watershed segmentation and a topological filtration were the main ideas. Markov Random Fields were used with the MS method in [25] too. In this paper, a hierarchical segmentation was done using the Tree-Structured Markov Random Fields.

Some kind of hierarchical approach was used in Boosted Mean Shift Clustering [48] that was presented in 2014. It uses the modified mean-shift method that works only with very small grid neighbourhood (area of 5-13 points) and creates a lot of small clusters that are called intermediate modes (iModes). The results from the first iterations of the mean-shift procedure are used in the next step that utilizes the DBSCAN method [19]. If the DBSCAN result is the same for three consecutive iterations, the algorithm is stopped. This method usually provides a smaller sensitivity to the bandwidth parameters.
The hierarchical approach can also be found in the paper from 2013 [4], where the already known hierarchical method was used for segmenting the population data. The paper [34] uses a variable mean-shift bandwidth selector to the mean-shift algorithm (superpixels were used). They added the hierarchical approach (superpixel technique) to Fast Adaptive Mean Shift [26] and tried even more combinations of various methods. In the paper, Spectral Gradient FAMS (SG-FAMS) and two Superpixel Mean Shift methods were presented (Per Superpixel MS called PSPMS and Full Superpixel MS called FSPMS).

Hierarchical versions of the mean-shift methods are also used for elastic image registration [71]. Corresponding points were estimated by maximizing the Bhattacharyya coefficients using the special version of the mean-shift method. This method is spatially based and computes spatial displacements of the control points.

### 2.6 Various mean-shift-like methods

Alongside the mean-shift method, there are many algorithms that use similar principles. One of the similar methods is called Medoid Shift that was presented in 1987 [36] and was revisited later in 2007 [51]. This method does not need explicit feature space, only a valid distance measure is required. After each iteration, the point is shifted and the closest data point to the new position of the processed point is selected as a new position for this data point. Therefore, the points are moving through the positions of points in the initial dataset. One can easily remember the path of points and an incremental clustering of the dataset can be easily obtained. There is still a huge drawback in the computational complexity of $O(N^3)$ using naive implementations of this algorithm. By using the matrix multiplication technique, the complexity can be lowered to $O(N^{2.38})$. The algorithm is finished when no point changes its position.

One of many variants of the medoid-shift method is the Quick Shift [61]. In this paper, the Medoid Shift was also deeply discussed and the authors mentioned Faster Euclidean Medoid Shift and Faster Kernel Medoid Shift. The first one uses an identity matrix in computation and the resultant matrix product of the following computations needs to carry out only $O(dN^2)$ operations. Therefore, the medoid-shift method can actually be faster than the mean-shift method. The authors presented an acceleration of the mean-shift method by initializing this by the medoid-shift method. The quick-shift algorithm constructs a tree, where each pixel is connected with the pixel that has a greater value of the density. Fulkerson and Soatto presented the GPU implementation of this algorithm [23] in 2010.

Medoid Shift has also its own hierarchical version that was developed further and Hierarchical Iconoid Shift (HIS) was revealed [69] in 2013. This algorithm uses similar approaches like the mean-shift and the medoid-shift methods but it is focused on architecture features recognition. The iconoid is the image of the feature which the algorithm is searching for and it should be the most central view of the feature. Hierarchical Iconoid Shift and the former version Iconoid Shift [68] use the medoid-shift method for searching for the features and buildings but the novelty is in a different distance measure (the homography overlap distance). This algorithm can also be easily parallelized.
Chapter 3

My Contributions to the Area

In this chapter, I present my contributions to the area of the mean shift methods. All the contributions presented here were published in international conferences. Further details can be found in the subsequent sections.

3.1 Hierarchical Mean-Shift Methods

A big part of my research was focused on hierarchical mean-shift methods. The hierarchical approach substantially improves the computational speed of the algorithm. Improving the speed was the main motivation for this activity since the usual mean shift algorithms are often difficult to use them in practice. In some cases, the segmentation results were even improved.

A large spatial bandwidth will cause very slow computation. On the other hand, larger bandwidths are necessary to obtain a smaller number of larger segments because the larger values prevent the oversegmentation in general. In contrast, if a small spatial bandwidth is used, the computation is carried out very quickly, but the result usually is oversegmented. The hierarchical mean-shift approach divides the computation into several stages, which improves the properties of the algorithm.

Computation is carried out several times in several stages (e.g. two or three). Each stage has its own spatial bandwidth and the output from each stage is used as an input for the following one. During the computation in each stage, the amount of data is reduced. The higher the reduction is in one stage, the faster the following stage can be. The general idea is to reduce the amount of data as much as possible in the first stage. For this reason, a small searching window (small spatial bandwidth, small kernel) is used on the original dataset in the first stage. If an extremely small bandwidth is used, the result is obtained very fast, but the reduction of dataset can be negligible and almost useless. Therefore, the bandwidth must be selected carefully because it is crucial for the next stage. That problem is discussed in Section 3.2.

The attractor of each resulting segment from one stage is considered as one point that is used in the next stage; its weight is proportional to the number of pixels the segment contains (i.e. the number of points/pixels that moved to the attractor). In the second stage, the spatial bandwidth can be enlarged. In spite of this, the computation can be still carried out quickly because this larger spatial bandwidth is not used with the original image, but it is used on a reduced number of points, which are the attractors created in the previous stage. In the introductory stages, the
speedup is achieved by the reduction of spatial bandwidth; in the following stages, the additional speedup is achieved by the reduction of dataset. Therefore, both the steps are much faster and the sum of their computational times can be still significantly lower than the computational time of mean-shift method carried out with only one final large spatial bandwidth.

An arbitrary number of stages can be used. Even though the number of stages is not limited, my experience shows that three stages are usually enough for the majority of images.

(a) The 1st stage filtration
(b) The 2nd stage filtration
(c) The 3rd stage filtration
(d) The 1st stage segmentation
(e) The 2nd stage segmentation
(f) The 3rd stage segmentation

Figure 3.1: Example of filtration and segmentation evolving after each stage of HBMS.

The only question is whether the time that was saved in the initial and in the final stage is or is not shorter than the time of the extra middle stage. The total time savings are significant because both stages saved lot of time and the middle stage is a very small computational burden only. Therefore, the 3-stage algorithms usually tend to be much faster then the the 2-stage algorithms, see Tab. 3.1.

Many experiments and evaluations have been carried out. I present five images in Fig. 3.2 that show the segmentation quality of the new algorithms.

First, the original algorithms (MS, BMS, and EMS) were used with the spatial bandwidth $\sigma_s = 25$. Then the new hierarchical approaches were used. The 2-stage versions of the algorithms are denoted by HMS2, HBMS2, and HEMS2. For the 3-stage versions, I use the notation HMS3, HBMS3, and HEMS3. If I am speaking about a hierarchical version in general, the notation HMS, HBMS, and HEMS without any numbers is used.

From the images, it is obvious that the general MS has the problem with over-
CHAPTER 3. MY CONTRIBUTIONS TO THE AREA

Figure 3.2: Row 1: the original image; rows 2/3/4: MS/BMS/EMS (spatial bandwidth $\sigma_s = 25$); rows 5/6/7: HMS2/HBMS2/HEMS2 ($\sigma_s = 4/25$); rows 8/9/10: HMS3/HBMS3/HEMS3 ($\sigma_s = 3/9/25$).
CHAPTER 3. MY CONTRIBUTIONS TO THE AREA

Table 3.1: Comparison of elapsed time and number of segments in general MS, HMS2 (HMS with 2 stages), and HMS3 (HMS with 3 stages). In all cases, the final bandwidth $\sigma_s = 25$ was used.

<table>
<thead>
<tr>
<th>Method</th>
<th>Segm.</th>
<th>t [s]</th>
<th>Segm.</th>
<th>t [s]</th>
<th>Segm.</th>
<th>t [s]</th>
<th>total [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>synthetic image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>410</td>
</tr>
<tr>
<td>HMS2</td>
<td>-</td>
<td>-</td>
<td>22833</td>
<td>9.0 s</td>
<td>242</td>
<td>210.2 s</td>
<td>220.4 s</td>
</tr>
<tr>
<td>HMS3</td>
<td>22232</td>
<td>3.7 s</td>
<td>1285</td>
<td>13.7 s</td>
<td>196</td>
<td>1.2 s</td>
<td>19.8 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>noisy synthetic image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>162</td>
</tr>
<tr>
<td>HMS2</td>
<td>-</td>
<td>-</td>
<td>13306</td>
<td>9.6 s</td>
<td>183</td>
<td>42.3 s</td>
<td>52.3 s</td>
</tr>
<tr>
<td>HMS3</td>
<td>15141</td>
<td>3.8 s</td>
<td>989</td>
<td>3.6 s</td>
<td>194</td>
<td>1.0 s</td>
<td>9.1 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>airplane image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>48</td>
</tr>
<tr>
<td>HMS2</td>
<td>-</td>
<td>-</td>
<td>13341</td>
<td>10.1 s</td>
<td>70</td>
<td>48.8 s</td>
<td>59.3 s</td>
</tr>
<tr>
<td>HMS3</td>
<td>14488</td>
<td>4.9 s</td>
<td>686</td>
<td>3.2 s</td>
<td>90</td>
<td>0.9 s</td>
<td>9.6 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>savana image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>45</td>
</tr>
<tr>
<td>HMS2</td>
<td>-</td>
<td>-</td>
<td>13502</td>
<td>9.9 s</td>
<td>87</td>
<td>56.7 s</td>
<td>66.9 s</td>
</tr>
<tr>
<td>HMS3</td>
<td>14660</td>
<td>4.8 s</td>
<td>752</td>
<td>3.7 s</td>
<td>103</td>
<td>0.9 s</td>
<td>10.2 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stones image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>97</td>
</tr>
<tr>
<td>HMS2</td>
<td>-</td>
<td>-</td>
<td>13849</td>
<td>9.1 s</td>
<td>79</td>
<td>110.3 s</td>
<td>119.8 s</td>
</tr>
<tr>
<td>HMS3</td>
<td>15642</td>
<td>4.1 s</td>
<td>681</td>
<td>4.5 s</td>
<td>102</td>
<td>0.8 s</td>
<td>9.9 s</td>
</tr>
</tbody>
</table>

segmentation in flat areas whereas other parts of the images are segmented well. Nevertheless, a higher number of iterations does not necessarily lead to much better results. The reason is that if there is not any gradient underneath the computed point, it cannot move. Blurring MS does not have this problem in the flat areas because the points near the borders of the flat areas are moving towards the center of the flat area, and sooner or later, they will influence the computation of the points in the center.

The results of the experiments are shown in Table 3.1 and 3.2. The tables present the speed gains of the hierarchical approach; the hierarchical approach was applied to several versions of the mean-shift method.

Blurring MS is much faster than general MS and it also gives better segmentation results. The problem with small 1-pixel segments does not exist here.

### 3.2 Setting the appropriate bandwidths in the hierarchical mean-shift methods

If any of the hierarchical mean-shift methods is used, it is usually desirable to use a larger final spatial bandwidth in order to make the resulting segments larger and to overcome the oversegmentation problem. It can be understood in such a way that the expected size of the final segments usually determines the value of the final bandwidth. However, there is still the problem of proper setting the initial spatial bandwidth in the first stage and in all the following internal stages.
### Chapter 3. My Contributions to the Area

#### Table 3.2: Comparison of elapsed time and number of segments in general BMS, HBMS2 (HBMS with 2 stages), and HBMS3 (HBMS with 3 stages).

<table>
<thead>
<tr>
<th>Method</th>
<th>Segm.</th>
<th>t [s]</th>
<th>Segm.</th>
<th>t [s]</th>
<th>Segm.</th>
<th>t [s]</th>
<th>total [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>synthetic image</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>106</td>
<td>157.9 s</td>
<td>158.3 s</td>
</tr>
<tr>
<td>HBMS2</td>
<td>-</td>
<td>-</td>
<td>1737</td>
<td>5.9 s</td>
<td>109</td>
<td>0.7 s</td>
<td>7.8 s</td>
</tr>
<tr>
<td>HBMS3</td>
<td>2965</td>
<td>4.9 s</td>
<td>515</td>
<td>0.3 s</td>
<td>127</td>
<td>0.3 s</td>
<td>7.3 s</td>
</tr>
<tr>
<td><strong>noisy synthetic image</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>102</td>
<td>80.5 s</td>
<td>81.0 s</td>
</tr>
<tr>
<td>HBMS2</td>
<td>-</td>
<td>-</td>
<td>1669</td>
<td>7.1 s</td>
<td>113</td>
<td>0.6 s</td>
<td>8.8 s</td>
</tr>
<tr>
<td>HBMS3</td>
<td>2787</td>
<td>3.4 s</td>
<td>489</td>
<td>0.2 s</td>
<td>122</td>
<td>0.1 s</td>
<td>5.6 s</td>
</tr>
<tr>
<td><strong>airplane image</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>40</td>
<td>89.2 s</td>
<td>89.6 s</td>
</tr>
<tr>
<td>HBMS2</td>
<td>-</td>
<td>-</td>
<td>1361</td>
<td>6.7 s</td>
<td>43</td>
<td>0.5 s</td>
<td>8.8 s</td>
</tr>
<tr>
<td>HBMS3</td>
<td>2379</td>
<td>3.0 s</td>
<td>299</td>
<td>0.2 s</td>
<td>49</td>
<td>0.1 s</td>
<td>5.1 s</td>
</tr>
<tr>
<td><strong>savanna image</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>52</td>
<td>97.9 s</td>
<td>98.3 s</td>
</tr>
<tr>
<td>HBMS2</td>
<td>-</td>
<td>-</td>
<td>1398</td>
<td>5.7 s</td>
<td>56</td>
<td>0.6 s</td>
<td>7.4 s</td>
</tr>
<tr>
<td>HBMS3</td>
<td>2488</td>
<td>3.3 s</td>
<td>353</td>
<td>0.2 s</td>
<td>66</td>
<td>0.2 s</td>
<td>5.1 s</td>
</tr>
<tr>
<td><strong>stones image</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>66</td>
<td>121.6 s</td>
<td>122.1 s</td>
</tr>
<tr>
<td>HBMS2</td>
<td>-</td>
<td>-</td>
<td>1408</td>
<td>6.4 s</td>
<td>68</td>
<td>0.7 s</td>
<td>8.8 s</td>
</tr>
<tr>
<td>HBMS3</td>
<td>2516</td>
<td>3.3 s</td>
<td>356</td>
<td>0.2 s</td>
<td>89</td>
<td>0.2 s</td>
<td>5.1 s</td>
</tr>
</tbody>
</table>

To clarify this, I have carried out several tests focused on the two-stage methods. I used the final bandwidths with the values $\sigma_s = 30, 50, 70,$ and 90. For each of these values, I tested many different initial bandwidths in the range from $\sigma_s = 2$ to $\sigma_s = 10$ with the step of 0.5. I measured the time that was needed to compute the initial stage, and the final stage.

The results of Hierarchical Blurring Mean Shift are presented in Fig. 3.3. We can see that the first stage takes approx. 2.5 seconds with the $\sigma_{s_1} = 2$ and it increases almost quadratically with the increasing size of the bandwidth. On the other hand, the second stage is slow for the small values of the initial bandwidth, and the computational times decrease dramatically as this value increases. After this dramatic fall down, the times of the second stage cannot be lower because they are getting very close to zero. Under these circumstances, the increasing times of the first stage tend to make the significant part of the whole computation time, and the algorithm is slowing down as a whole (Fig. 3.3).

The same was tested with general MS too. The results are presented in Fig. 3.4. General MS does not provide such good segmentation qualities as BMS and created more segments if the same conditions were set. Because of that, the input for the next stage was more oversegmented and not so much compressed. The second stage has to work with a bigger amount of data and, therefore, the whole algorithm was very slow. Since general MS does not reduce dataset so much, much larger initial bandwidth is needed in order to achieve relatively fast second stage.

Generally, I can say that the optimal initial bandwidth for the general mean shift
is approximately $\sqrt{\sigma_s}$. Similarly, the ideal initial bandwidth for the blurring mean shift is approximately $\frac{1}{2}\sqrt{\sigma_s}$.

Finally, I note that the different values of the initial bandwidth have only a small impact on the visual segmentation result. Thanks to this, I can optimise the value of the initial bandwidth with respect to speed only. Naturally, the final segmentation results greatly depend on the final spatial bandwidth.

### 3.3 Optimization of Evolving Mean Shift

The original Evolving Mean Shift (EMS) [77] that was presented had one issue I needed to solve in order to make Hierarchical EMS to be reasonably fast. Even though the number of iterations per pixel is very small, these iterations take a long time to process. The bottleneck is in searching for the point with the highest energy value. One has to go through the whole image to find such a value and it has to be done after a movement of each pixel. Therefore, the authors of Evolving Mean Shift used random sampling technique to accelerate this searching process. My goal was to find an algorithm that is able to speed up the searching task and to be precise at the same time.

My approach uses a hierarchic technique to overcome both speed and precision problems. There are two buffers for sorting and precise searching for the maximal energy in the dataset. I created a buffer that contains few percent of data points that are chosen precisely according to a precomputed threshold. Only a small amount of the data points with the highest energy values are chosen to the second buffer that is even smaller and has its own threshold. Then the point with the highest energy is searched
CHAPTER 3. MY CONTRIBUTIONS TO THE AREA

![Graphs showing computational times](image)

Figure 3.4: The computational times (vertical axis) of the Hierarchical General Mean Shift (HMS) in dependence on the value of the initial bandwidth (horizontal axis).

only in this very small buffer. During the computation, I delete or insert data points from/to the buffers according to their values in comparison with the thresholds. The demanding step of creating the buffer is carried out only when it has been emptied and that is not happening too often. Moreover, the second buffer is an additional improvement to EMS that significantly reduces the number of computations needed when the energy values are updated.

My experiments showed that the ideal threshold should pick up approx 4% of values to this first buffer. Therefore, my algorithm will use only \(2n + 0.04n \cdot \log(0.04n)\) steps instead of traditional \(n \cdot \log(n)\) because I do not need to sort the whole array of numbers. Then I pick up only few highest energy values from the sorted array to the second hierarchy level. The ideal size of this second buffer is \(\frac{1}{2} \sqrt{n}\) to \(\sqrt{n}\) according to my experiments and the lowest value from this buffer becomes the second threshold value. Because the second buffer has only the highest energy values of the whole dataset and very limited size, I can find the maximal energy extremely quickly. If an energy that was in the buffer changes, the buffer is updated (the value is deleted or inserted).

Recomputation of mean shift vectors (energy values) is another problem and this was the main problem that caused the original method to be slow. If one point is shifted, its shift can influence the energy of all the points that are in his surroundings (where it came from and where it was shifted). To solve this problem, I use an extra buffer that stores the number (contribution) of pixels that were involved in the computation of energy for each pixel. Therefore, I do not have to compute all the mean-shift vectors again when I need to update only one of them. Thanks to this buffer, I can very easily recompute the sum of energies of all pixels affected and to
3.4 Layered Mean-Shift Methods

I have developed a family of so called layered mean-shift algorithms that are focused mainly on oversegmentation reduction. The methods have been published in [82] and [81]. The trick is to carry out more mean-shift segmentations and compile their results together. All of them will find the true edges correctly. They will differ only in the areas without edges where the segments (edges) are unwanted but present due to the problem of oversegmentation.

The layered mean-shift algorithms will be demonstrated and explained on the image of a church (Fig. 3.5). The algorithm utilize the fact that the correct object boundaries are always detected in the same positions no matter how large value of the spatial bandwidth is used. The only difference is in false detections in flat areas (Fig. 3.6). If I stack these segmentation results together, I can clearly see that the object boundaries are the same (or almost the same) in all cases (Fig. 3.7(a)).

The advantage of the layered methods is that I can carry out several very fast mean-shift computations with small values of the spatial bandwidth, which gives the desired several segmentation results that can be stacked together afterwards. These computations can be very fast because all of them are carried out with a small spatial bandwidth. The total time of all the computations is comparable with the one non-layered MS computation.

According to the nature of image, 3-5 MS computation are usually enough. Not only the number of particular segmentations must be chosen, but also a merging rule must be determined. In my case, I use a merging threshold. This threshold (denoted by \( t \)) has to be lower than the number of computations (denoted by \( n \)).

If two pixels are at least \( t \) times in the same segment (of \( n \) segmentation results) it is said that they are in the same final segment. Because many boundaries in this segmentation result are oversegmented artifacts only, there are usually no boundaries in other segmentations in that same place. Therefore, one can say that this boundary was the false one (detected only once whereas other segmentation outputs did not detect it) and the segment label is unified with all the pixels from the second segment.
Figure 3.6: The phases of the LBMS method. The segmentations for different spatial bandwidths are shown in the above three figures.

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

Only the boundaries that were detected multiple times, remain.

### 3.5 Hierarchical Layered Mean-Shift Method

Hierarchical Layered Mean Shift (HLMS) is a significant improvement to the layered mean-shift method that was presented in the previous section. This new method was developed to overcome all remaining problems of stability and speed of layered approaches and utilizes both the hierarchical and the layered approach. The layered approach allows me to use small bandwidths and concurrently eliminate the oversegmentation problem with small improvement of speed, the hierarchical approach was especially focused on the significant speed improvement.

First, the hierarchical approach takes place and the first stage of the mean-shift method is carried out with a very small spatial bandwidth $\sigma_s$. This first stage creates an oversegmented image. Such an oversegmented image is far better in the role of the input for the following computation in comparison with classical layered approach that works with the whole input image.

Now, I use already known layered approach on this oversegmented image. I am not merging single pixels from the input image into segments but I am merging small segments into the larger segments. Whereas layered approach is testing how many times each pair of pixels was in the same segmentation, now it is testing how many times was each pair of segments in the same multi-segment.
CHAPTER 3. MY CONTRIBUTIONS TO THE AREA

23

Figure 3.8: Stages of Hierarchical Layer Blurring Mean Shift (the same approach applies also to general mean-shift and evolving mean-shift method).

Figure 3.9: Stacked segmentation and result of merging the segments using the Hierarchical LBMS method. The image 3.9(a) is a composition of hierarchy steps 3.8(b), 3.8(c) and 3.8(d).
Because of this method of computation, I usually do not have problem with the pixels that are on the boundaries and it is not clear which segment is the best for them. The algorithms were tested and the results are presented in Fig. 3.10.

![Figure 3.10: Hierarchical layered mean-shift methods. Original images are in the first row. Rows 2-3 show HLMS method in configurations 2/1 and 3/1. Notation 2/1 means that 2 segmentations were carried out and sub-segments that were at least once in the same segment (in the hierarchy step), are merged. If they were separated twice (of 2 segmentations), than this is considered as a boundary of the final segment. Rows 4-5 show HLBMS results and rows 6-7 show HLEMS result in the same configurations.](image)

Practical results show that two layered segmentation provide basic merging of the partial results but it is not sufficient for a high quality result. It is obvious that better results are obtained when three segmentations are carried out.

The Table shows the computational times of each stage of the hierarchical layered method. It is obvious that the largest amount of time is taken by the first stage, especially when the blurring and evolving mean-shift method is used. The following layer stages are very fast. That suggests we can use smaller bandwidth in the first stage to accelerate the algorithm and to avoid the bottleneck.
Table 3.3: Comparison of elapsed time of hierarchical methods. Hierarchical Layered Mean Shift is denoted by HLMS, Hierarchical Layered Blurring Mean Shift is denoted by HLBMS and HLEMS stands for Hierarchical Layered Evolving Mean Shift. The number 3/1 stands for 3 processed mean-shift segmentations and 1 is a merging rule (if two sub-segments are at least once in the same segment in the three segmentation outputs, the pixels are merged into one final segment. In all cases, the initial spatial bandwidth was set to $\sigma_s = 4$, range bandwidth was set to $\sigma_r = 24$. The multiplier value was 1.35. The measured times contain the overheads of each stage (these are not the times of MS computation only) and there is merging step that is included only in the final time.

### CHAPTER 3. MY CONTRIBUTIONS TO THE AREA

#### 3.6 Mean-Shift Method with Flatness Constraints

Another view on the mean shift method was presented in the paper [83] together with an improvement of the method. I have shown there that the mean-shift methods can also be viewed as a problem of maximising a certain functional. Based on this understanding of Mean Shift, it is also possible to improve it. The improvement is based on adding certain new terms into the functional that will influence the properties of the results. Namely, I tried to reduce the problem of oversegmentation.

Let me introduce the needed notation first. The dataset can be expressed by the set $\{(x_i, r_i)\}_{i=1}^n$, where $n$ is the number of data points (pixels in the image), $x_i$ are the spatial coordinates ($x_i \in \mathbb{R}^2$) and $r_i$ are the luminance values ($r_i \in \mathbb{R}$ in the grey-scale images, and $r_i \in \mathbb{R}^3$ in colour images). For each original point (pixel), its position of convergence (attractor) is computed. I denote the attractor of $x_i$ by $(\chi(x_i), \rho(x_i))$
saying where the positions of the attractor for each point is located. All the points that have the same attractor form a cluster (a segment). It is easy to see that the set \( \{(x_i, \rho(x_i))\}_{i=1}^{n} \) defines the filtered image. It also defines the result of segmentation.

In order to show that the mean-shift method can be viewed as a problem that can be solved by making use of the calculus of variations, let me consider the following functional

\[
F(\chi(x_i), \rho(x_i)) = \int_{\Omega} \sum_{i=1}^{n} k(||(x_i, r_i) - (\chi(x), \rho(x))||^2) dx
\]

\[
= \int_{\Omega} \sum_{i=1}^{n} k(||x_i - \chi(x)||^2 + ||r_i - \rho(x)||^2) dx .
\]  (3.1)

The area of spatial coordinates is given by \( \Omega \subseteq \mathbb{R}^2 \) and the function \( k() \) is a kernel (searching window). In order to solve the problem of finding the maximum value, I can easily write the Euler-Lagrange equations that correspond to Eq. (3.1). They have the following form

\[
\sum_{i=1}^{n} k'(||(x_i, r_i) - (\chi(x), \rho(x))||^2)[x_i - \chi(x)] = 0 ,
\]

\[
\sum_{i=1}^{n} k'(||(x_i, r_i) - (\chi(x), \rho(x))||^2)[r_i - \rho(x)] = 0 .
\]  (3.2)

Figure 3.11: Comparison of segmentation results carried out with the original MS method and our proposed method.

The whole process is still iterative and, therefore, the systems of equations have to be solved several times. Even though the size of the equation systems is really big and equal to the number of pixels in the input image, they are also sparse with only five non-zero elements in one row. Because of that, the equations can be solved effectively.

The result is visible in Fig. 3.11 where the proposed method is able to filter noisy flat areas completely and the edge between these two flat areas is also detected much better with less artifacts. The results for three real-life images are shown in Fig. 3.12.
Figure 3.12: The original image is on the left. The segmentation result of general MS is shown in the center. The new proposed method is shown in the right column.
CHAPTER 3. MY CONTRIBUTIONS TO THE AREA

Figure 3.13: Comparison of segmentation results of fast mean-shift method. If the step is 20 pixels (in x and y axis) then only each 400th pixel is involved in the computation. The white points represent the detected head.

3.7 Hierarchical Fast Blurring Mean-Shift for Head Position and Center Estimation in Depth Images

The mean-shift method is usually used for object tracking nowadays and we also use that in our automotive industrial project. It is a part of the gaze tracking algorithm. In order to be able to compute the gaze direction, we need to detect pupil and/or iris in the eye and we need to know where the geometrical center of the eye is. In order to speed up the whole computation, it is useful to know where the head is and where to search for the eyes. All these improvements for the blurring mean-shift method used with depth images were published [80] and I will discuss them in the next following pages.

The most precise method for detecting the eye center is to detect IR reflection in IR images but it also has some drawbacks. If the person is wearing glasses, many unwanted reflections emerge and it is hard to decide which reflection is the right one. There are also problems when the center of the eye is covered by lids. Therefore, we need to estimate the eye center using some different approach.

It is not hard to detect the face (for example, by using famous Viola and Jones Adaboost algorithm [63]) and eyes in RGB or IR images. If we have detected the eyes, we can easily find out where the middle position between the eyes is. Thanks to that information, one can estimate the tilting angle of the head if the geometric center of the head is known. The tricky part is to detect head in RGB images.

First, I will finish the explanation, why the head center is so important. If I am able to detect the head, its geometrical center and where the area between eyes (or nose) is, I also know which direction the head is heading. The above mentioned area between the eyes is usually the center of the face detected. If the head is tilted, the center of the face (area between eyes/eyebrows/nose) is not in the same place as the geometric head center. The bigger the difference (shift) is the higher tilting angle is and I can estimate the angle in which the head and all its parts are pointed. Therefore, I can also estimate the perspective shift of the eye center.

There are many approaches how to detect a head and its center. In my approach, I use the detection of the face in the first iteration of algorithm (in IR or RGB image) and then only depth data are used. I rely on position consistency, therefore, I assume that the head will be in a quite similar position in the next image.

I do not need extreme (per-pixel) precision of boundaries of the head but I need
CHAPTER 3. MY CONTRIBUTIONS TO THE AREA

quite a good estimation of the head center. The idea of the fast-mean shift algorithm [75] is that neighboring pixels usually have similar properties and converge to the same attractor. Therefore, one can skip some pixels and pick only few representatives that will “represent” their surroundings. Using this approach, I can greatly reduce the input dataset and concurrently improve the speed of the algorithm with acceptable results. In my testing case (see Fig. 3.13), I skip 20, 15 or 10 pixels in both axes.

The first simple blurring mean-shift method able to provide the result in 2.69 milliseconds when 20-pixel step was used and 7.08 milliseconds when 15-pixel step was used. The second version of fast blurring mean-shift method (FBMS) was using a weighting parameter. After each iteration of the mean-shift method, all pixels were checked if there is any new segment (if there are two pixels in the same position). If that happens, both pixels are merged, the first one has the weight increased by one, the second pixel has the weight set to zero. Therefore, in all the following iterations of the fast blurring mean-shift method, I can check if the weight is zero and in such a case, I can skip that pixel (I know that it was merged with another one already). That can improve the speed of the algorithm and I was able to achieve 1.96 milliseconds (20-pixel step), resp. 4.87 milliseconds (15-pixel step).

<table>
<thead>
<tr>
<th>Type</th>
<th>20-pixel step</th>
<th>15-pixel step</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBMS</td>
<td>2.69 ms</td>
<td>7.08 ms</td>
</tr>
<tr>
<td>FBMS with zero values</td>
<td>1.96 ms</td>
<td>4.87 ms</td>
</tr>
<tr>
<td>FBMS with relocation</td>
<td>1.75 ms</td>
<td>3.45 ms</td>
</tr>
</tbody>
</table>

Table 3.4: Elapsed time according to the optimization applied.

I was able to improve the algorithm further. Instead of setting the weight of the second merged pixel to zero, I replace this pixel in the list of points with the last one from the dataset. Then I declare that the size of dataset is decreased by one (that the list is shorter even though it is not, in fact). This number (the size of dataset) forms a barrier in the list. All the data points in front of the barrier are used in the computation of BMS and all others are skipped (they are zero already). In such a case, the $n$ in $O(n^2)$ is really decreasing because I have only the pixels with non-zero weights in the dataset. I do not need to check for the zero weight anymore. I managed to achieve the computational time 1.75 milliseconds when the 20-pixel step was used and 3.45 ms when the 15-pixel step was used. All these result are shown in Table 3.4.

There is also one optimization that should be considered. It is relatively common that new segments usually do not emerge in the very first iterations of the mean-shift method. Therefore, it is useless to check for the existence of new segments in these iterations. Because of that, I can skip this check in the beginning of the computation and carry out the check when it makes sense. On the other hand, I should not skip too many steps. Otherwise, my method will be changed to the unoptimized FBMS.

<table>
<thead>
<tr>
<th>Skipped iterations</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>1.89 ms</td>
<td>1.81 ms</td>
<td>1.75 ms</td>
<td>1.78 ms</td>
<td>1.91 ms</td>
</tr>
</tbody>
</table>

Table 3.5: Comparison of elapsed time according to the number of skipped iterations where the correspondence check (emerging of attractors) was not used.

The best result was achieved when two iterations were skipped and when the check
for the existence of two pixels that converged to the same place was carried out in the third and following iterations.

I have improved my approach further by the hierarchical approach. It is needed because even this improved fast blurring mean-shift is still either slow and precise or fast and imprecise. In the first stage, small spatial bandwidth is used in order to minimize the data points involved in computation and to speed up the algorithm. Because the fast mean-shift technique is used, the dataset is already shrunk by neglecting some of the data points and a small spatial bandwidth is used on already reduced dataset. The resulting small segments are used as the input in the second stage with higher spatial bandwidth. This stage is very fast even though a larger bandwidth is used because the dataset was already twice reduced (by neglecting the points by FMS reduction of input and by the first stage of the algorithm). Because of these reductions, the overheads can play quite significant role in computation and we need to implement the algorithm very carefully.

The first idea is to use supportive lattice (see Fig. 3.15) that divides the images to tiles. All the data points that are in the area covered by a tile are stored in lists. Therefore, if I examine the neighbourhood of the computed pixel, I need to examine only the points under the neighbouring tiles. Choosing the appropriate tiles is much faster operation (only two very short for loops) than the check of distances between all the points because I have to examine all of them otherwise.

The tiles improve the speed when checking for the emerging segments too. It is obvious that potential segments (a formation of the attractor) have to be under the same tile. Therefore, I need to inspect only a very limited number of the points to find out if they have converged to the same position. Instead of $O(n^2)$, the complexity is $O(m^2)$ only where $m = n/l$ is the number of points under the lattice ($n$ is number of data points in the whole dataset, $l$ is the number of tiles). All of these optimizations can be used in ordinary fast blurring mean-shift too.

I have compared all versions of Blurring Mean Shift in Fig. 3.16. In the first row, there is the original version of the method, I also present the result from Hierarchical BMS (my method that was presented in 2011). The results from Fast Blurring Mean Shift are presented in the second row. It is obvious that skipping 4 pixels in each direction still gives a quite acceptable result. For our purposes, even FBMS8
Figure 3.15: A lattice structure. The black circle represents the data point that is going to be processed. The red circle is the kernel window given by the $\sigma_s$ parameter. It is obvious that data points that are covered by the searching window can be only in lists of data points represented by the green tiles.

can be used and will give reasonable head dimensions and the position of the head center. The last row presents Hierarchical Fast Blurring Mean Shift that combines my hierarchical approach with the improved lattice structure and accelerated merging of already converged data points. These results are almost the same as the FBMS ones. One can see that the visible differences given by the hierarchical approach are almost negligible.

On the other hand, what really differs, is the computational time. The results are presented in Table 3.6. Computational times of original BMS and basic hierarchical approach are not enough for real-time purposes. If Fast Blurring Mean Shift is used, one need to skip 8 pixels in each direction (only 1/64th of original data is used) in order to achieve 22.6 ms. Such a time is quite short but I need to take into account that there are following steps in the complete gaze direction algorithm. The hierarchical version (HFBMS) needs only 9 milliseconds when the same conditions are used and, therefore, a lot of time remains for other parts of the gaze direction algorithm.

The mean-shift method is very suitable for these tasks as the last presented papers show. Gaze recognition using the help of the mean-shift method was also mentioned in [62] where the MS was entropy oriented.
Figure 3.16: Comparison of the results of various blurring mean-shift methods. The white lines represent the detected head. The last number in the name of the method represents the number of skipped points in fast mean-shift methods. BMS stands for Blurring MS, HBMS is abbreviation for Hierarchical BMS, FBMS is Fast BMS and HFBMS is Hierarchical FBMS (my new proposed method).
Table 3.6: The time comparison of various blurring mean-shift methods. In all cases, the final spatial bandwidth was set to 80. BMS-NL is a blurring mean-shift method with no lattice. BMS is using this supportive lattice with the tile size of 20 × 20 (this tile size is used for all the following tests). HBMS stands for Hierarchical BMS with the supportive lattice structure, the initial spatial bandwidth \( \sigma_s \) was set from 2 to 6. FBMS is Fast BMS where the dataset is reduced prior the computation because of the skipped data points (pixels). The number “x” in FBMSx denotes the number of skipped pixels in each direction. Therefore, FBMS8 uses each 8th pixels in each coordinate. That means only 1/64th of data points is used. HFBMS denotes Hierarchical Fast BMS (our proposed method) that not only skips the pixels but also uses the hierarchical approach and lattices structure. The initial spatial bandwidth for the first stage was set to 1.5 times the distance of non-skipped pixels.
Chapter 4

Conclusion

In this work, I tried to bring some improvements to the well known mean-shift method. It is used for image filtering, image segmentation, object tracking and many other purposes. I dealt especially with image segmentation problem and how to make it faster with similar or even better segmentation results. Nevertheless, some of the proposed improvements can also be used for other purposes of mean-shift method like the filtering or object tracking.

In the first part of my work, I provided a deeper study of hierarchical approaches applied to the mean-shift method. Although some papers dealt with the hierarchical approach more than ten years ago, it was more developed in recent few years by me and some other authors (especially since 2012 year). I presented Hierarchical Blurring Mean Shift that provides quite good segmentation results in a fraction of original time. The same was achieved by creating the Hierarchical Evolving Mean Shift that is also much faster than the original method. In this particular case, I also provided the hierarchical recomputation of energy values (mean-shift vectors) that can be applied to the original EMS and also its hierarchical version HEMS. The purpose of this recomputation step is to improve the speed with no impact on the segmentation quality and stability. My approach is not only faster, but it is also absolutely precise in comparison with the original algorithm that used random sampling in order to improve the speed.

Later, the Layered Mean Shift was developed. This method used the idea that many different MS computations (with different bandwidths) will give different results that will usually have the same boundaries around the real objects and they will differ only in the flat regions that suffer from oversegmentation. One can consider the oversegmentation as a noise that should be removed and more measurements (more MS computations) can eliminate this noise. Stacking more layers together will keep only the true boundaries (the boundaries that are the same in all obtained results). This method gives nice results using BMS and EMS as its base method. On the other hand, because of computation of many MS segmentations (even though they are carried out with small spatial bandwidths), this method is not significantly faster than the original one.

The speed problem was solved by Hierarchical Layered Mean-Shift that combined both above mentioned approaches, the hierarchical and layered one. First, the initial MS segmentation is carried out with a small spatial bandwidth, which makes it possible to obtain the heavily oversegmented MS result quickly. Although it is oversegmented, the number of segments is reduced enough to make the following stages reasonably
CHAPTER 4. CONCLUSION

35

fast as same as in the hierarchical methods. Instead of making more hierarchical steps using this result, I carry out the layered mean-shift method on this preprocessed data. In other words, I am only deciding which of the detected boundaries are important (boundaries that emerge in all the layered results) and which are only the segmentation noise. This method proved to be very efficient using blurring MS and evolving MS method as its base method and it was very fast and precise in all cases. That is also an advantage in comparison with the classical layered approach that did not work very well with the general mean-shift method. The hierarchical layered version is able to give nice results even when MS is used.

I also developed the mean-shift method with flatness constraints that formulates the mean shift method as a problem of maximising a functional; two new terms were added that should minimise the negative influence of large segments to detail resolution and luminance changes.

The last proposed method uses already known so called fast mean-shift algorithm that skips some data points in order to speed up the computation. In my case, blurring MS method as its basic function was used and I have brought some improvements in order to speed it up further. My version works with the depth images obtained from Intel RealSense camera and it is used in industrial automotive projects for head detection and gaze direction estimation. I have developed the acceleration method that greatly reduces already reduced dataset in order to speed-up the algorithm. Therefore, the segmentation of the the head is a matter of few milliseconds and I can estimate the head boundaries very fast. I can also find out the position of the head center that is used for gaze detection afterward and the head tilting angle. These properties can be used for estimating the real geometrical eye center. This center is the starting point for the gaze direction vector and I can use the proposed approach if the IR reflection that represents the eye center more precisely is not found.

The tasks was to find the head dimensions and its center effectively. My new method Hierarchical Fast Blurring Mean Shift was focused on the speed improvement in order to provide fast results that should be used for the following steps of the algorithm that estimate the eye center location. The fast blurring technique was improved by the hierarchical approach and the supportive lattice structure that improves seeking for data points that should be involved in the computation, it also effectively accelerates the merging of the converged data points. My algorithm is able to segment the head in depth data in few milliseconds or tens of milliseconds even if a very good precision is needed. Therefore, it can be used in real-time applications and a multi-threaded optimization will be considered in the future.
Bibliography


Chapter 5

Author’s Bibliography

5.1 Publications Related to the Thesis

5.1.1 Conference Proceedings


5.1.2 PhD Workshops


5.2 Publications Unrelated to the Thesis

5.2.1 Conference Proceedings


