# Edge Detection based on Fast Adaptive Mean Shift Algorithm

Yong Zhu [1], Ruhan He [1], Naixue Xiong [2], Pu Shi [3], Zhiguang Zhang [4, *]
[1] *College of Computer Science,*
*Wuhan University of Science and Engineering,*
*Wuhan, 430073, China.*
[2] *Department of Computer Science,*
*Georgia State University,*
*Atlanta, GA, 30303, USA.*
[3] *Dept. of mathematics and computer science,*
*Emory University, USA.*
[4] *School of Computer Science,*
*Huazhong Univ. of Sci. & Tech.,*
*Wuhan, 430074, China.*
*Email: zy@wuse.edu.cn, [*] hustskyzhang@gmail.com*

## Abstract

*Edge detection is arguably the most important operation in low level computer vision. Mean shift is an effective iterative algorithm widely used in edge detection. But the cost of computation prohibits Mean shift algorithm for high dimensions feature space. In this paper, a fast adaptive mean shift algorithm is proposed for edge detection. It makes use of one approximate nearest neighbors search method, i.e. LSH (Locality-Sensitive Hashing) firstly, which dramatically reduces the computation of iterations in high dimensions. Moreover, the LSH procedure can help to determine the bandwidth of the kernel window adaptively. The experimental results show the effectiveness of the proposed approach.*

## 1. Introduction

Edge detection is arguably the most important operation in low level computer vision with a plethora of techniques. An edge is the boundary between an object and its background (the outline of the object). Edge detection must be efficient and reliable because the validity, efficiency, and possibility of the completion of subsequent processing stages (in computer vision for example) rely on it. This means that if the edges in an image can be identified accurately, objects in the image can be located and basic properties such as area, perimeter, and shape can be measured. A fundamental difficulty in edge detection processes is the possible extraction of spurious edges that arise from noise and minor intensity changes which are often non-meaningful and disturbing, and may subsequent processing stages degrade computational performance. Thus, a proper selection of the edges may be very important.

Numerous approaches have been dedicated for edge detection, among which the mean shift method is one of the most common methods. Mean-shift is a nonparametric density gradient estimator. The mean shift method presents an elegant way to locate these density maxima without having to estimate the density directly [1]. The mean shift vector always points to the direction of maximum increase in the density. The mean shift process is an iterative procedure that shifts each data point to these density maxima. In [2], it is employed to derive the object candidate that is the most similar to a given model while predicting the next object location. This method provides accurate localization, and it is computationally fast. However, the limitation of the approach based on mean shift is that it does not scale well with the dimension of the space. It was indicated that when the dimensionality is above 6, the analysis should be approached carefully [1].

Mean shift is a nonparametric density estimator which iteratively computes the nearest mode of a sample distribution. After its introduction in the literature [3], it has been adopted to solve various computer vision problems, such as line fitting [4], segmentation [5] and object tracking [6]. Despite its

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.226

promising performance, as discussed in various papers [7], [8]and [9], the traditional mean shift method has two main limitations, the first of which is the constancy of the kernel bandwidth. The change in the object scale requires an adjustment of the kernel bandwidth in order to consistently track the object. An intuitive approach to estimate the object scale is to search for the best scale by testing different kernel bandwidths and selecting the bandwidth which maximizes the appearance similarity [6]. Alternatively, after the object center is estimated, a mean shift procedure can compute the bandwidth of the kernel in the scale space, which is formed by convolving the image with a set of Gaussian kernels at various scales [8].

The second limitation of the traditional mean shift method is the use of radically symmetric kernels which are isotropic in shape. In the view of often anisotropic structure of the object, radically symmetric kernels inhibit robust image segmentation [9] and object tracking. For example, representing an elongated object with a circular kernel will bias the position estimation due to the non-object regions residing inside the kernel. This object can be better represented by an anisotropic symmetric kernel, such as an ellipse. The scale and orientation of a kernel representing an object can be computed by evaluating the second order moments from the object silhouette [10] or the posterior appearance probabilities [11]. Both of these methods, however, are computationally expensive compared to the mean shift tracking method. This observation resulted in the introduction of anisotropic symmetric kernels to the mean shift analysis. In particular, Wang et al. [9] has shown the improved performance of the mean shift segmentation when a circular kernel is replaced with an elliptical kernel. In their approach, in contrast to analyzing the local data distribution, the authors estimated the orientation and the scale of the elliptical kernel from images.

In this paper, we extend the traditional mean shift method by introducing an approximate nearest neighbor searching algorithm, i.e. LSH (Locality-Sensitive Hashing) [12]. We get the nearest neighbors for each pixel by an approximate neighbor search method, LSH. Then, we store the results and take it as the range of iteration, i.e. bandwidth. The pixels that do not promote the convergence of iteration are removed, which dramatically decreases the time cost for convergence of iteration. It makes use of one approximate nearest neighbors search method, i.e. LSH (Locality-Sensitive Hashing) firstly, which dramatically reduces the computation of iterations in high dimensions. Moreover, the LSH procedure can help to determine the bandwidth of the kernel window adaptively.

In the rest of this paper: Section 2 discusses the related works about the edge detection. In Section 3, our proposed approach that LSH-based mean shift algorithm is described. In Section 4, the experiments are done. Finally, we present our conclusion and future works.

## 2. Related works

A basic edge detection process usually involves the following stages: 1) Smoothing—required for noise reduction and regularization of the numerical differentiation. It depends on the regularization parameter (scale) which determines the compromise between noise elimination and image structure preservation. 2) Differentiation—an operation that evaluates the intensity variations in the image. 3) Labeling—the final decision stage that marks the identified edges. This stage usually involves a threshold parameter that separates true from false edges. This common detection process is based on evaluation of the strength of intensity transitions in the image. Another (complementary) approach to edge detection is based on evaluation of spatial properties of the image features [13] [14]. This approach (denoted saliency) states that points are more likely to be meaningful edges if they belong to longer, smoother, and continues curves. Lindenbaum and Berengolts [15] developed a saliency estimation mechanism which is based on probabilistically specified grouping cues and on length estimation. This mechanism produces a saliency map, in which higher values specify locations of pixels that belong to longer and smoother curves. Final edge detection was obtained by thresholding the saliency map using a previously selected threshold parameter.

The outcome of an edge detection process varies greatly with the choice of the detector parameters. Therefore, a prior step of parameter selection is necessary. Selection of the detector parameters is often done manually by a trial-and-error process. However, such a process is frequently non-efficient and tedious. Therefore, automatic techniques have been developed to select parameters of edge detectors [16, 17]. These methods have been concentrated on common specific parameters such as the smoothing scale [16] and the threshold [17]. However, other parameters may also be employed in edge detection. For example, a vision model-based edge detector developed recently uses an invariable threshold (an average contrast threshold of the visual system), but employs several band pass

filters that may be regarded as parameters [18].

## 3. The Proposed Edge Detection Algorithm

In the following, we firstly introduce the means shift algorithm and LSH algorithm respectively, and then give the hybrid algorithm.

### 3.1. Kernel density estimation and mean shift

The mathematical derivation of kernel density estimation theory was described in [1]. In pattern recognition, each sample is represented as a point in d-dimensional space, called feature space. Its dimension is determined by the number of parameters (such as intensity and coordinate loci on the genome for array CGH data) to describe the sample points. The feature space can be regarded containing an empirical probability density function (PDF) of the represented parameters. Given $n$ data points $x_i$ ($i = 1, \ldots, n$) in the d-dimensional space $R^d$, the multivariate kernel density estimator( $\hat{f}(x)$ ) at point x is computed with kernel $K(x)$ and a symmetric positive definite $d \times d$ bandwidth matrix **H**

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(x - x_i) \tag{1}$$

where

$$K_{\mathbf{H}}(x) = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2}x) \tag{2}$$

The bandwidth matrix is often chosen either as proportional to the identity matrix $\boldsymbol{H} = h^2\boldsymbol{I}$ or diagonal $\boldsymbol{H} = \boldsymbol{diag}\,[h_1^2, \ldots, h_d^2]$ in practice. For example, if we employ the former case, which provides one bandwidth parameter $h > 0$, we get

$$\hat{f}(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{3}$$

The kernel K(x) is a bounded function which must satisfies the following conditions:

$$\begin{aligned} &\sup_{x \in R^d} |K(x)| < \infty, \int_{R^d} |K(x)|\,dx < \infty \\ &\lim_{\|x\| \to \infty} \|x\| K(x) = 0, \int_{R^d} K(x)dx = 1 \end{aligned} \tag{4}$$

The radially symmetric kernel is a special case that satisfies $K(x) = c_{k,d}k(\|x\|^2)$ , where $k(x)$ is the profile of the kernel ($x \geq 0$). $c_{k,d}$ (assumed strictly positive) is the normalized constant, which makes $K(x)$ integrate to one. By introducing profile notation, the density estimator can be re-written as

$$\hat{f}(x) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k(\left\|\frac{x - x_i}{h}\right\|^2) \tag{5}$$

The first step in the analysis of the feature space with underlying density $f(x)$ is to find the modes of the density, which are among the zeros of the gradient $\nabla f(x) = 0$ . The mean shift method is an elegant way to locate these zeros without having to estimate the density [1]. By computing, using the chain rule, the gradient of $f(x)$ $\nabla f(x)$ , the formula (5) is changed to

$$\begin{aligned} \hat{\nabla}_{h,K} f(x) = \frac{2c_{k,d}}{nh^{d+2}} &\left[\sum_{i=1}^{n} g(\left\|\frac{x - x_i}{h}\right\|^2)\right] \\ &\cdot \left[\frac{\sum_{i=1}^{n} x_i g(\left\|\frac{x - x_i}{h}\right\|^2)}{\sum_{i=1}^{n} g(\left\|\frac{x - x_i}{h}\right\|^2)} - x\right] \end{aligned} \tag{6}$$

where $g(x) = -k'(x)$ using simplified notation. The kernel G(x) then is defined as $G(x) = c_{g,d}k(\|x\|^2)$, where $c_{g,d}$ is the corresponding normalization constant. In Equation (6), the first factor $\sum_{i=1}^{n} g(\left\|\frac{x - x_i}{h}\right\|^2)$ is assumed to be positive number. This condition is easy to satisfy for all the profiles in practice. The second factor in (6) is called the mean shift, which is the difference between the weighted mean (using the kernel $G$ for weight) and the center of the kernel $x$.

$$\mathbf{m}_{h,G}(x) = \frac{\sum_{i=1}^{n} x_i g(\left\|\frac{x - x_i}{h}\right\|^2)}{\sum_{i=1}^{n} g(\left\|\frac{x - x_i}{h}\right\|^2)} - x \tag{7}$$

It has been proven that the mean shift vector at location x computed with kernel $G$ is proportional to the normalized density gradient estimate obtained by kernel $K$ **[1]**. The mean shift vector always points toward the direction of the maximal increase in the density. The mean shift procedure is carried out by successive steps between the computation of the mean shift vector $\boldsymbol{m}_{h,G}(x)$ and the translation of window by $\boldsymbol{m}_{h,G}(x)$. It has been proven that this procedure is guaranteed to converge at a point nearby where the estimate has a zero gradient, if the kernel $K$ has a convex and monotonically decreasing.

Thus, the sequence of successive locations of kernel $G$, denoted by $y_j$ $(j=1,2,...)$ for each starting point $x_i$ ($y_1 = x_i$), can be computed as

$$y_{j+1} = \frac{\sum_{i=1}^{n} x_i g\left(\left\| \frac{y_j - x_i}{h} \right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\| \frac{y_j - x_i}{h} \right\|^2\right)} \quad (8)$$

Provided that the mean shift vector always points toward the direction of the maximal increase in density, the local mean is shifted toward the region in which the majority of the points are located. Consequently, the mean shift vector can define a path that leads to a stationary point of the estimated density, as it is aligned with the local gradient estimate. These stationary points are called the 'modes' of the estimated density. The mean shift procedure, obtained by consecutive computation of the mean shift vector $\boldsymbol{m}_{h,G}(y_j)$ and translation of the window $y_{j+1} = y_j + \boldsymbol{m}_{h,G}(y_j)$, is guaranteed to converge to a point where the gradient of density function is zero. The set of all locations that converge to the same mode defines "the basin of attraction" of that mode. The points which are in the same basin of attraction are associated with the same cluster .

### 3.2. LSH algorithm

There are numerous problems that involve finding similar items. These problems are often solved by finding the nearest neighbor to an object in some metric space. This is an easy problem to state, but when the database is large and the objects are complicated, the processing time grows linearly with the number of items and the complexity of the object. For very large databases of high dimensional items, LSH (Locality-Sensitive Hashing) [12] is a particularly valuable technique for retrieving items that are similar to a query item. In these searches it can drastically reduce the computational time, at the cost of a small probability of failing to find the absolute closest match.

Given a query point, we wish to find the points in a large database that are closest to the query. We wish to guarantee with a high probability equal to $1-\varepsilon$ that we return the nearest neighbor for any query point.

Conceptually, this problem is easily solved by iterating through each point in the database and calculating the distance to the query object. However, our database may contain billions of objects—each object described by a vector that contains hundreds of dimensions. Therefore, it is important that we find a solution that does not depend on a linear search of the database. Existing methods to accomplish this search include trees and hashes.

Several methods of this general nature have been proposed, and Locality-Sensitive Hashing (LSH) [12] has received considerable recent attention because it was shown that its runtime is independent of the dimension $D$ and has been put forth as a practical tool. Roughly speaking, a locality sensitive hashing function has the property that if two points are "close," then they hash to same bucket with "high" probability; if they are "far apart," then they hash to same bucket with "low" probability. Formally, a function family $H = \{h : S -> U\}$ is $(r1, r2, p1, p2)$ *-sensitive*, where $(r1 < r2, p1 > p2)$, for distance function $D$ if for any two points $p, q \in S$, the following properties hold:

1) if $p \in B(q, r1)$ ,then $\Pr_{h \in H}[h(q) = h(p)] \geq p1$, and

2) if $p \notin B(q, r2)$ ,then $\Pr_{h \in H}[h(q) = h(p)] \leq p2$, where $B(q, r)$ denotes a hypersphere of radius $r$ centered at $q$. By defining a LSH scheme, namely a $(r, r(1+\varepsilon), p1, p2)$ -sensitive hash family, the $(1+\varepsilon)$ -*NN* problem can be solved by performing a series of hashing and searching within the buckets. Applications have found $(1+\varepsilon)$ approximation to be useful, for example when the *k*-nearest-neighbor search is just one component in a large system with many parts, each of which can be highly inaccurate. In this paper we explore the extent to which the most successful exact search structures can be adapted to perform $(1+\varepsilon)$ approximate high-dimensional searches. A notable previous approach along this line is a simple modification of *k-d-trees* – ours takes the more powerful metric trees as a starting point. We next review metric trees, then introduce a variant, known as *spill trees*.

### 3.3. LSH-based Mean Shift algorithm

In the edge detection by standard Mean Shift algorithm, each pixel has a radius of neighbor window, i.e. the kernel function bandwidth. Every pixel in this neighbor window must be included in the iteration computation of means shift. However, some of the pixels is useless while consumes the time of circulation.

Therefore, we get the nearest neighbors for each pixel by an approximate neighbor search method, LSH. Then, we store the results and take it as the range of iteration, i.e. bandwidth. The pixels that do not promote the convergence of iteration are removed, which dramatically decreases the time cost for convergence of iteration.

Based on LSH algorithm and Mean shift, the LSH-based Mean shift algorithm is described as Table 1.

Table 1. ANN-based Mean Shift Algorithm

```
/* Step 1: input data point and construct the tree. */
   1)  Load data point;
   2)  Build Tree

/* Step 2: Set the search distance and search the Near
Neighbor for each data points. */
   For i=0 to nPointNumber
      Querypt = DataPoints[i];
      LSH(Querypt, SquaredRadius, Numreturn,
            IndexPointer, Distance , Error Bound);
      nCount = NumReturn;
      while(nCount = NumReturn)
         nCount =0;
         for j=0 to NumReturn
            if(IndexPointer[j]>-1)
              nCount= nCount +1;
         end for
         if(nCount < NumReturn)
            for j=0 to nCount
               store IndexPointer[j] to PindexArray (which is
                  the whole Nearest Neighbors Array of all the
                  data Points);
            end for
            break;
          end if
          Increase Radious
          LSH again;
      End while
   End for

/* Step 3: Iterative Computation of Mean Shift */
      For i=0 to nPointNumber
         IterPoint = DataPoints[i];
         Msv=MeanShiftVector(IterPoint, PindexArray)
         MsvMag= Caculate Msv magnitude squared;
         IterationCount = 1;
         While ( MsvMag <  && IterationCount <100)
            IterPoint = IterPoint + Msv;
            Msv=MeanShiftVector(IterPoint,
                                 PindexArray);
            MsvMag=Caculate Msv magnitude squared;
            IterationCount = IterationCount +1;
          End while
          Store the IterPoint's value as result;
      End for
```

## 4. Experiments

We have applied the proposed method to several hundreds of different images, and Fig. 1 shows some results. Figure 1(a) and Figure 2(a) are the original images for the famous "camera man" and "lena" respectively; Figure 1(b) are Figure 2(b) the result Edge Map by standard Mean shift with grad win.=2 and min. length=5; Figure 1(c) and Figure 2(c) are the result Edge Map by standard Mean shift with grad win.=5 and min. length=10; Figure 1(d) and Figure 2(d) are the results for edge detection by our method.

It can be seen from Figure 1 and Figure 2, the edges segmented by our method are almost same as the standard mean shift algorithm. Moreover, to some extent, it has more detail information than standard mean shift.

Table 2 lists the processing time for the two edge detection methods using a Pentium-IV 1.8 GHz PC with 1GB RAM for image sizes 512*512. The results show that the proposed method is faster than the conventional methods.

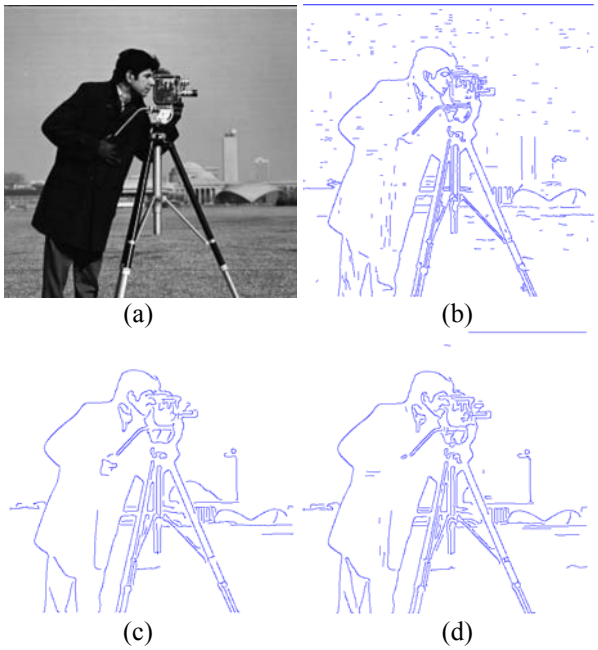


Figure 1. The edge diction results for "camera man" image. (a) Original image; (b) Edge Map by Mean shift with grad win.=2 and min. length=5; (c) Edge Map by Mean shift with grad win.=5 and min. length=10; (d) our approach.

(a) (b) (c) (d)

Figure 2. The edge diction results for "lena" image. (a) Original image; (b) Edge Map by Mean shift with grad win.=2 and min. length=5; (c) Edge Map by Mean shift with grad win.=5 and min. length=10; (d) our approach

Table 2. The time comparison

| Test image | Time for MS (s) | Time for proposed method (s) |
|---|---|---|
| Camera man | 8.23 | 2.21 |
| lena | 7.95 | 2.05 |
| fruit | 8.12 | 2.13 |

## 5. Conclusions

This paper proposes a new edge detection method that base on LSH and Mean Shift algorithm. The method takes the nearest neighbors as the iteration bandwidth of mean shift. The advantages of the proposed method are: 1) reducing the time for convergence of iterations; 2) having the adaptation for bandwidth. Therefore, the edge results can preserve more detail information of image. The experiment shows the effectiveness of the proposed method.

## Acknowledgment

This work is supported by Hubei Provincial Department of Education (Grant No: Q20091704), Natural Science Foundation of Hubei Province (Grant No. 2008CDB232 ).

## References

[1] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Trans. Pattern Recognition & Machine Intelligence*, vol. 24, pp. 603-619, 2002.

[2] D. Comaniciu, V. Ramesh, and P. Meer, "real-time tracking of non-rigid objects using mean shift," presented at IEEE Conf. Computer Vision and Pattern Recognition, South Carolina, 2000.

[3] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE IT*, vol. 21, pp. 32-40, 1975.

[4] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 790-799, 1995.

[5] D. Comaniciu and P. Meer, "Mean shift analysis and applications," presented at IEEE Int. Conf. on Computer Vision, 1999.

[6] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 564-575, 2003.

[7] D. Comaniciu, "An algorithm for data-driven bandwidth selection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 281-288, 2003.

[8] R. Collins, " Mean-shift blob tracking through scale space," presented at IEEE Conf. on Computer Vision and Pattern Recognition, 2003.

[9] J. Wang, B. Thiesson, Y. Xu, and M. Cohen, "Image and video segmentation by anisotropic mean shift," presented at European Conf. on Computer Vision, 2004.

[10] G. Bradski, "Computer vision face tracking for use in a perceptual user interface," presented at IEEE Workshop on Applications of Computer Vision, 1998.

[11] H. Tao, H. Sawhney, and R. Kumar, "Object tracking with bayesian estimation of dynamic layer representations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 75-89, 2002.

[12] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," presented at STOC, 1998.

[13] A. Sha'ashua and S. Ullman, "Structural saliency: the detection of globally salient structures using locally connected network," presented at ICCV-88, 1988.

[14] G. Guy and G. Medioni, "Inferring global perceptual contours from local features," *Int. J. Comput. Vis*, vol. 20, pp. 113-133, 1996.

[15] M. Lindenbaum and A. Berengolts, "A probabilistic interpretation of saliency network," presented at ECCV, 2000.

[16] T. Lindenberg, "Feature detection with automatic scale selection," *Int. J. Comput. Vis.*, vol. 30, pp. 77-116, 1998.

[17] S. Venkatesh and P. Rosin, "Dynamic threshold determination by local and global edge evaluation," *Comput. Vis. Graph. Image Process.*, vol. 57, pp. 146-160, 1995.

[18] E. Peli, " Feature detection algorithm based on a visual system model," in *IEEE 90*, 2002, pp. 78-93.