Applying Neighborhood Consistency for Fast Clustering and Kernel Density Estimation

Kai Zhang † Ming Tang* James T. Kwok†

† Department of Computer Science, Hong Kong University of Science and Technology, Kowloon Hong Kong. {twinsen, jamesk}@cs.ust.hk

* NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, 100080 tangm@nlpr.ia.ac.cn

Abstract

Nearest neighborhood consistency is an important concept in statistical pattern recognition, which underlies the well-known k-nearest neighbor method. In this paper, we combine this idea with kernel density estimation based clustering, and derive the fast mean shift algorithm (FMS). FMS greatly reduces the complexity of feature space analysis, resulting satisfactory precision of classification. More importantly, we show that with FMS algorithm, we are in fact relying on a conceptually novel approach of density estimation, the fast kernel density estimation (FKDE) for clustering. The FKDE combines smooth and non-smooth estimators and thus inherits advantages from both. Asymptotic analysis reveals the approximation of the FKDE to standard kernel density estimator. Data clustering and image segmentation experiments demonstrate the efficiency of FMS.

1. Introduction

In many computer vision problems, the analysis of the feature space that exhibits complex multimodal structures is realized through nonparametric kernel estimation techniques. Among them, the mean shift iterative procedure of mode seeking is a highly successful one. It was originally developed by Fukunaga and Hostetler [8] based on kernel density gradient estimation, and later applied to many computer vision tasks such as tracking [2, 15, 11], image segmentation [4, 5], information fusion [1], clustering and classification [9, 3], and video processing [6].

A brief introduction of the mean shift algorithm is presented as follows. Given a set of data points $\{\mathbf{x}_i \mid i = 1, 2, \ldots, N\}$ in the d-dimensional feature space, the kernel density estimator with symmetric kernel function $K(\cdot)$ and fixed bandwidth h can be written as [13]

$$f_K(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h^d}k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right), \tag{1}$$

where $k(\cdot)$ is the *profile* of kernel K such that $K(\mathbf{x}) = ck(\|\mathbf{x}\|^2)$, and c is a normalization constant. When the derivative of $k(\cdot)$ exists, $g(\cdot) = -k'(\cdot)$ can be used as a *profile* to define a new kernel $G(\mathbf{x})$ such that $G(\mathbf{x}) = c'g(\|\mathbf{x}\|^2)$ with normalization constant c'. Take the gradient of (1), we can obtain

$$m(\mathbf{x}) = C\frac{\hat{\nabla} f_K(\mathbf{x})}{\hat{f}_G(\mathbf{x})}, \tag{2}$$

where

$$m(\mathbf{x}) = \frac{\sum_{i=1}^{N}\frac{1}{h^{d+2}}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{N}\frac{1}{h^{d+2}}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x} \tag{3}$$

is called the *mean shift vector*, and C is a constant. From (2), we can see that $m(\mathbf{x})$ points towards the steepest ascent direction of the density function $\hat{f}_K(\mathbf{x})$, therefore the mean shift iteration

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + m(\mathbf{x}^{(k)}), \quad k = 1, 2, \ldots \tag{4}$$

is a hill climbing process to the nearest maximum of $\hat{f}_K(\mathbf{x})$.

Although mean shift algorithm has been applied extensively in computer vision, the complexity $O(N^2)$ [3] is high when large data sets are involved, and various methods have been proposed to speed up the algorithm. One category is aimed at improving the efficiency of the most expensive operation in mean shift, i.e., finding the nearest neighbors of a point. kD-tree is a popularly adopted structure to achieve this goal. In [9], a different technique called locality-sensitive hashing (LSH) is employed for approximate neighborhood searching in variable bandwidth mean

0-7695-2372-2/05/$20.00 (c) 2005 IEEE

shift. The optimal parameters of LSH are determined by a pilot learning procedure.

The other category focuses on applying function expansion techniques. For instance, the fast Gauss transform (FGT) [10] has been introduced for efficient modelling of the color distribution of homogeneous regions in object tracking [7]. In [14], an improved fast Gauss transform (IFGT) is proposed to accelerate the calculation of the mean shift vector (3) in higher dimensionality.

In this paper, we adopt the idea of "nearest neighbor consistency" in mean shift, and develop a fast mean shift (FMS) algorithm that significantly reduces the complexity of feature space analysis. The FMS algorithm is presented in section 2, and the resultant "fast kernel density estimation" (FKDE) is discussed in section 3. In section 4, asymptotic analysis of the FKDE demonstrates its approximation to common kernel density estimators. Section 5 gives clustering and image segmentation results using FMS algorithm. And conclusions are made in section 6.

2. Fast Mean Shift Algorithm

Nearest neighbor consistency is an important concept in statistical pattern recognition. Given a training set with known class labels, the nearest neighbor rule determines the class label of a point according to the label of its nearest neighbor. But due to the lack of labelled samples, it is not easy to apply the neighborhood consistency to completely unsupervised environments. In this section, we adopt the nearest neighbor rule to mean shift, an unsupervised clustering method introduced in section 1, and derive the fast mean shift (FMS) algorithm that significantly accelerates the feature space analysis.

2.1. Apply Neighborhood Consistency in Mean Shift

Usually, the mean shift vector $m(\mathbf{x})$ is calculated using the whole data set, as shown in (3). Inspired by the neighborhood consistency, however, we can simplify the calculation of $m(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \frac{\sum_{j=1}^{m}\sum_{\mathbf{x}_i\in S_j}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{j=1}^{m}\sum_{\mathbf{x}_i\in S_j} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x} \\ &\simeq \frac{\sum_{j=1}^{m} n_j\mathbf{c}_j g\left(\left\|\frac{\mathbf{x}-\mathbf{c}_j}{h}\right\|^2\right)}{\sum_{j=1}^{m} n_j g\left(\left\|\frac{\mathbf{x}-\mathbf{c}_j}{h}\right\|^2\right)} - \mathbf{x} \qquad (5) \\ &= m_F(\mathbf{x}). \end{aligned}$$

where the original data set $\{\mathbf{x}_i \mid i = 1, 2, \ldots, N\}$ is decomposed into a number of "local subsets", $S_1, S_2, \ldots, S_m$, each with size n_j and center $\mathbf{c}_j = \frac{1}{n_j}\sum_{\mathbf{x}_i\in S_j}\mathbf{x}_i$. $m_F(\mathbf{x})$ is called the "fast mean shift vector". The approximation in (5) reveals two basic elements of the fast mean shift algorithm (FMS). First, the samples of each local subset S_j are treated as a whole in describing the density distribution. Second, the samples of each S_j are assumed to come from the same class (neighborhood consistency), hence only the representative ($\mathbf{c}_j$) needs to be labelled for each S_j. The complete FMS algorithm is described as follows:

Fast Mean Shift algorithm (FMS)

I. PARTITION
Divide $\{\mathbf{x}_i \mid i = 1, 2, \ldots, N\}$ into m local subsets $S_j (j = 1, 2, \ldots, m)$, each with size n_j and center $\mathbf{c}_j$. Let $C = \{\mathbf{c}_j \mid j = 1, 2, \ldots, m\}$ be the cluster centers.

1. Initialize C by randomly selecting a sample as $\mathbf{c}_1$. Let $C = \{\mathbf{c}_1\}$. Then, at the ith iteration, $i = 1, 2, \ldots, N$, do the following.
2. Calculate the distances between $\mathbf{x}_i$ and $\mathbf{c}_j$'s, ($\mathbf{c}_j \in C$). Once if $\|\mathbf{x}_i - \mathbf{c}_j\| \le r$, assign $\mathbf{x}_i$ to S_j, let $i = i + 1$, and go to the next iteration.
3. If $\|\mathbf{x}_i - \mathbf{c}_j\| > r$ for all $\mathbf{c}_j \in C$, add $\mathbf{x}_i$ to C as a new subset center, and assign $\mathbf{x}_i$ to this new subset. Let $i = i + 1$ and go to the next iteration.
4. On termination, count n_j, the number of samples in S_j, and update each $\mathbf{c}_j \in C$ as $\mathbf{c}_j = \frac{1}{n_j}\sum_{\mathbf{x}_i\in S_j}$.

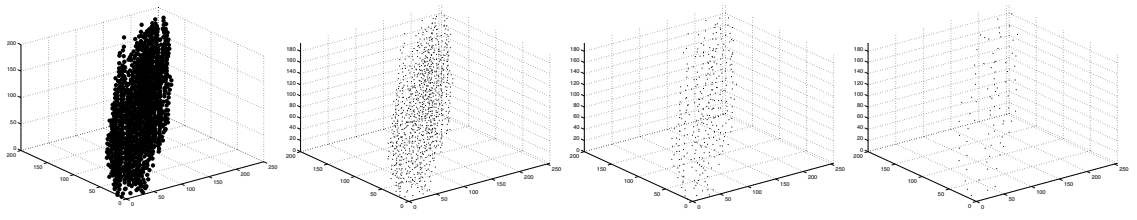
II. CLUSTERING.
Run the mean shift iterative procedure in (4) on C, and determine the class assignments of $\mathbf{x}_i$'s according to the convergent result of $\mathbf{c}_j$'s.

1. For each $\mathbf{c}_j$, start the iteration

$$\mathbf{c}_j^{(k+1)} = \frac{\sum_{p=1}^{m} n_p \cdot \mathbf{c}_p \cdot g(\|\frac{\mathbf{c}_j^{(k)}-\mathbf{c}_p}{h}\|^2)}{\sum_{p=1}^{m} n_p \cdot g(\|\frac{\mathbf{c}_j^{(k)}-\mathbf{c}_p}{h}\|^2)}$$

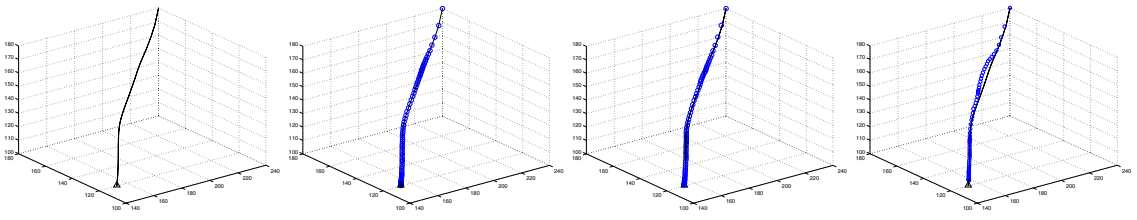
   with $\mathbf{c}_j^{(0)} = \mathbf{c}_j$. Record the convergent point as μ_j^C.
2. For $\forall\mu_a^C, \forall\mu_b^C, (1 \le a, b \le m, a \ne b)$, if $\|\mu_a^C - \mu_b^C\| \le \varepsilon$, then assign elements of S_a and S_b to the same class, or else S_a and S_b belong to different classes.

Note that the volume of the local subsets S_j's created in the PARTITION step in the d-dimensional feature space will never exceed the volume of the d-dimensional hypercube with edge $2r$. Unlike some clustering-orientated algorithms, most of which are based on the minimization of certain objective function, the PARTITION step here is not aimed at clustering, but rather to find a simplified description of the original sample set by partitioning. Moreover,



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

Figure 1. PARTITION of the "squirrel" sample set under different parameters. See text for details.



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

Figure 2. Comparison of the convergent routes of Mean Shift and FMS iterations. See text for details.

the number of local sets, m, is automatically determined by the PARTITION parameter r, and is therefore totally data-driven.

Figures 1 and 2 give some examples of the FMS procedures. Figure 1(a) plots the sample set of the "squirrel" image (Fig.6, the 3rd one of the leftmost column) in the L*u*v feature space. Using the PARTITION procedure with different parameters ($r = 4, 8, 12$), we obtain different cluster center sets (C_1, C_2, C_3) in Figures 1(b)–(d). Starting from point $(240, 180, 180)$, convergent routes of the standard mean shift are shown in Figures 2(a)–(d) in dots; and the convergent routes of FMS (on C_1, C_2, and C_3) are plotted in circles in Figures 2(b)–(d). The Gaussian kernel with bandwidth $h = 20$ is always used. Note that even though C is sparse due to the coarse PARTITION procedure, the convergent route of FMS still approximates that of the standard mean shift very well.

2.2. Approximation Error of the mean shift vector

In this subsection, we will study a sufficient condition on the error between $m_F(\mathbf{x})$ in (5) and $m(\mathbf{x})$ in (3), and show that it is related to the PARTITION parameter r.

Given a d-dimensional data set, define the *basic partition* that divides the feature space into a number of equal-sized hypercubes with edge r_R, where r_R is the resolution, i.e., the minimum non-zero distance between two samples on the coordinate axis. Let $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$, be the underlying true density. $f(\mathbf{x})$ is proportional to the number of samples falling in the small hypercube centered at $\mathbf{x}$. Then, the mean shift vector $m(\mathbf{x})$ can be regarded as a numerical solution to

$$I = \frac{\int_{R^d} \mathbf{y} f(\mathbf{y}) g(\|\frac{\mathbf{x}-\mathbf{y}}{h}\|^2) d\mathbf{y}}{\int_{R^d} f(\mathbf{y}) g(\|\frac{\mathbf{x}-\mathbf{y}}{h}\|^2) d\mathbf{y}} - \mathbf{x}$$

by using the rectangular method on the *basic partition*. Similarly, the "fast mean shift vector" $m_F(\mathbf{x})$ can also be regarded as another numerical solution of I on a coarser partition, whose bin-width is bounded by $2r$ (r is the PARTITION parameter). Suppose $\|m(\mathbf{x}) - I\| < \epsilon_1$, and $\|m_F(\mathbf{x}) - I\| < \epsilon_2$, then $\|m(\mathbf{x}) - m_F(\mathbf{x})\| < \epsilon_1 + \epsilon_2$. Actually, ϵ_1 and ϵ_2 are associated with r_R and r, respectively. Because the numerator and denominator of I are both integrals, their numerical approximation errors using the rectangular method are $O(q''(\xi)b^2)$ in univariate case, where $q(\cdot)$ is the integration function and b is the bin-width. Similar conclusion can also be drawn in multivariate case. Therefore, the smaller the r_R and r, the smaller the numerical integration error of the numerator and denominator of I for $m(\mathbf{x})$ and $m_F(\mathbf{x})$, and hence the smaller are ϵ_1 and ϵ_2. Since the *basic partition* (or r_R) is fixed, therefore the smaller the PARTITION parameter r, the smaller $\|m(\mathbf{x}) - m_F(\mathbf{x})\|$.

2.3. Complexity

The PARTITION step takes $O(mN)$ time (supremum). However, this can be significantly reduced by using a hierarchical scheme. First, use a large parameter r_0 to divide the original set into very few local subsets. Then, hierarchically divide the existing local subsets by adopting the PARTITION procedure with gradually decreased r's, until a threshold r_T is reached. In this way, the number of local subsets increases exponentially with the depth

of the hierarchy, and the process terminates in $\log m$ levels. Therefore the complexity of PARTITION is reduced to $O(N \log m)$, where m is the number of local subsets ultimately obtained. The CLUSTERING step, on the other hand, takes $O(m^2)$ time. So the overall complexity for FMS is $O(N \log m) + O(m^2)$. Notice that m is inversely associated with PARTITION parameter r_T, and is usually much smaller than N.

2.4. FMS with More Efficient Partitions

We use "targets" and "sources" to represent the set of points to be classified through mean shift, and the set of points on which the mean shift vector is calculated, respectively. In FMS, both are chosen to be the same $C = \{\mathbf{c}_j \mid j = 1, 2, \ldots, m\}$. Often, it is more desirable to choose them separately, since the sources determine the underlying density estimate of FMS, while the targets determine the class assignment of $\mathbf{x}_i$'s.

Usually, to capture the important characteristics of a distribution, a more delicate partition is required to form the sources compared with the partition for the targets. In practice, we first perform a fine hierarchical partitioning on the original data set by using a relatively small threshold r_T, and choose the sources as the centers $\mathbf{c}_j$'s of the local subsets $S_j (j = 1, 2, \ldots, m)$. Then, we fuse some of the centers $\mathbf{c}_j$'s to form the targets. This fusion operation is defined as follows. Perform mean shift on the centers $\mathbf{c}_j$s. After an iteration, if two updated centers $\mathbf{c}_a$ and $\mathbf{c}_b$ are close enough, then S_a and S_b are fused to form a larger local set, whose center replaces $\mathbf{c}_a$ and $\mathbf{c}_b$ as a new target. The fusion operation efficiently decreases the number of targets, and consequently the complexity of FMS.

3. A Novel Fast Density Estimator

The convergent terminals of the mean shift iterations are determined by the local maxima of the estimated density (1) [4]. This also holds for FMS. In analyzing the corresponding density estimate of the FMS algorithm, we find that it's actually a novel density estimation technique, which we call "fast kernel density estimation" (FKDE). The procedures and advantages of FKDE are described as follows.

1. Divide the sample set into a number of local subsets $S_j (j = 1, 2, \ldots, m)$ (with PARTITION parameter r), each with size n_j and center $\mathbf{c}_j$. This is similar to building a multivariate histogram of variable bin-width bounded by $2r$.

2. The FKDE at $\mathbf{x}$ is defined by

$$\hat{f}_F(\mathbf{x}) = \frac{1}{Nh^d} \sum_{j=1}^{m} k\left(\left\| \frac{\mathbf{x} - \mathbf{c}_j}{h} \right\|^2 \right) n_j, \qquad (6)$$

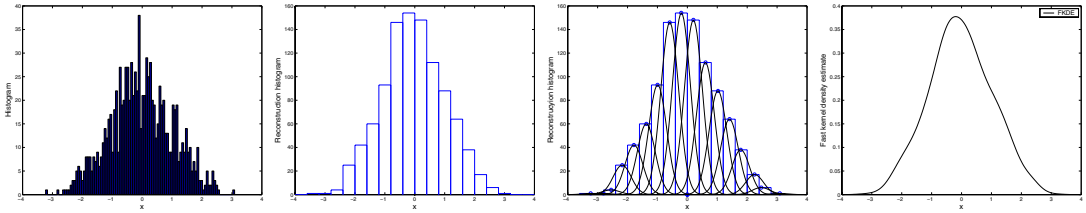
   which can be regarded as the weighted sum of m kernels centered at the mean of each bin in the histogram, with bin-heights as the weights, and $N = \sum_{j=1}^{m} n_j$. Therefore, the histogram in step 1 is also called the "reconstruction histogram".

If we use the hierarchical partition scheme described in Section 2.4 to build the "reconstruction histogram", step 1 of the FKDE will only cost $O(N \log m)$ time. Then, we only have to spend $O(m)$ time to estimate the density at each sample, and $O(Nm)$ time altogether. In contrast, standard kernel density estimation takes $O(N^2)$ time to obtain the estimates at all samples. This means a speedup ratio of about $\gamma = N^2/(N \log m + Nm) \approx N/m$ is obtained. In image segmentation experiments, N is usually of $10^4 \sim 10^5$ and m is of $10^2 \sim 10^3$, so the FKDE can be hundreds of times faster.

The FKDE is illustrated in Figure 3, where a set of 1000 1-D samples is drawn from the standard normal distribution. Figure 3(a) shows its histogram with bin-width 0.15, while Figure 3(b) is the reconstruction histogram with a much larger bin-width $w = 0.4$. When a number of Gaussian kernels with bandwidth $h = 0.3$, centered at the mean of each bin and weighted by the bin height, are added together (Figure 3(c)), we obtain the fast kernel density estimate $\hat{f}_F(x)$ (Figure 3(d)).

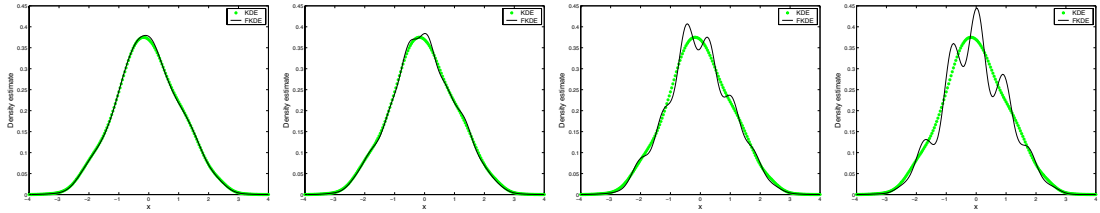
Figure 4 compares $\hat{f}_F(x)$ with the standard density estimator $\hat{f}(x)$ when the bin-width of the reconstruction histogram w varies (both of them use the Gaussian kernel of bandwidth $h = 0.3$). When $w < 0.5$, there is nearly no difference between $\hat{f}(x)$ and $\hat{f}_F(x)$, so we only plot the result when $w = 0.5$ (Figure 4(a)), where the thick green curve is $\hat{f}(x)$ and the thin black curve is $\hat{f}_F(x)$. When $w = 0.6$, a slight difference occurs (Figure 4(b)). When w increases to 0.7 and 0.9 (Figures 4(c)–(d)), $\hat{f}_F(x)$ is significantly different from $\hat{f}(x)$. In practice, we will not use such coarse partitions. The choice of proper partition parameter will be discussed in section 4.

From (6) we can see that the buildup of the FKDE novelly involves both smooth and non-smooth estimators. The non-smooth "reconstruction histogram" first provides a "skeletonized" view of the whole distribution, and then smooth kernel functions are placed on the center of each bin to further improve the estimate. By doing this, FKDE successfully inherits advantages from both kinds of estimators. As we know, the histogram estimator is easily constructed, but may fail to expose the true modes of distribution due to its non-smooth nature. And its bias is $O(h)$, where h is the bandwidth. Smooth kernel estimators, on the other hand, better reveals the structure of the underlying distribution, with a bias $O(h^2)$. Of course its computation is much more expensive. By combining smooth and non-smooth estimators, the FKDE can provide a good density estimate (whose bias is proved to be $O(h^2) + O(r^2)$ in the next section) with



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

Figure 3. Illustration of the fast kernel density estimation (FKDE). See text for details.



(a) $w = 0.5$. (b) $w = 0.6$. (c) $w = 0.7$. (d) $w = 0.9$.

Figure 4. Comparison of standard kernel density estimate (KDE) and the fast kernel density estimate (FKDE) when the bin-width w of the reconstruction histogram varies.

much less computation compared with the smooth density estimators.

4. Asymptotic Approximation Property of the Fast Density Estimator

In this section, we investigate the asymptotic properties of the FKDE $\hat{f}_F(\mathbf{x})$, which provides more intuition about the way FMS algorithm behaves. Compared with the error analysis of the fast Gauss transform in the special case of Gaussian kernels, our analysis is more generalized.

In estimation theory, a global discrepancy measure between the estimated and target densities is the *mean integrated square error* (MISE), which is obtained by integrating the *mean square error* (MSE) over the whole domain. The MSE is defined as:

$$\begin{aligned} MSE(\mathbf{x}) &= E\{\hat{f}(\mathbf{x}) - f(\mathbf{x})\}^2 \\ &= Var\left(\hat{f}(\mathbf{x})\right) + \left(Bias\left(\hat{f}(\mathbf{x})\right)\right)^2, \end{aligned} \quad (7)$$

which is decomposed into the bias and variance [13]. Next we will show how to link the MSE of standard kernel density estimator with that of FKDE. Consider first the bias. In standard kernel density estimation, the bias is [13]

$$\begin{aligned} Bias\left(\hat{f}(\mathbf{x})\right) &= E[\hat{f}(\mathbf{x})] - f(\mathbf{x}) \\ &= \frac{1}{h^d}\int_{R^d} K\left(\frac{\mathbf{x}-\mathbf{y}}{h}\right) f(\mathbf{y})d\mathbf{y} - f(\mathbf{x}). \end{aligned} \quad (8)$$

For the bias of the FKDE, notice that (6) can be written as

$$\hat{f}_F(\mathbf{x}) = \frac{1}{Nh^d}\sum_{j=1}^{m} n_j K\left(\frac{\mathbf{x}-\mathbf{c}_j}{h}\right) = \frac{1}{Nh^d}\sum_{i=1}^{N} K\left(\frac{\mathbf{x}-c(\mathbf{x}_i)}{h}\right),$$

where $c(\mathbf{x}_i)$ maps $\mathbf{x}_i$ to the mean of the samples in S_j by $c(\mathbf{x}_i) = \frac{1}{n_j}\sum_{\mathbf{x}_p \in S_j} \mathbf{x}_p$, and $\mathbf{x}_i \in S_j$. Asymptotically, when the sample size $N \to \infty$, summation becomes integration. So $c(\mathbf{x}_i)$ becomes $c(\mathbf{x}_i) = \int_{I_j} \mathbf{x} f(\mathbf{x})d\mathbf{x}$, where I_j is the region in the feature space that surrounds S_j. Hence, $c(\mathbf{x})$ is constant for all $\mathbf{x} \in I_j$, denoted as $\mathbf{y}_j$. Then the bias of $\hat{f}_F(\mathbf{x})$ can be written as

$$\begin{aligned} Bias\left(\hat{f}_F(\mathbf{x})\right) &= E\left[\hat{f}_F(\mathbf{x})\right] - f(\mathbf{x}) \\ &= \frac{1}{h^d}\int_{R^d} K\left(\frac{\mathbf{x}-c(\mathbf{y})}{h}\right) f(\mathbf{y})d\mathbf{y} - f(\mathbf{x}) \\ &= \frac{1}{h^d}\sum_{j=1}^{m} K\left(\frac{\mathbf{x}-\mathbf{y}_j}{h}\right)\int_{I_j} f(\mathbf{y})d\mathbf{y} - f(\mathbf{x}). \end{aligned} \quad (9)$$

The $Bias(\hat{f}_F(\mathbf{x}))$ in (9) can be viewed as an approximation to $Bias(\hat{f}(\mathbf{x}))$ in (8) by replacing $K(\cdot)$ with discrete values. Since $f(\mathbf{y})$ is bounded, it is easy to see that

$$\begin{aligned} & \left|Bias\left(\hat{f}(\mathbf{x})\right) - Bias\left(\hat{f}_F(\mathbf{x})\right)\right| \\ = & \left|\sum_{i=1}^{m}\frac{1}{h^d}\int_{I_j}\left[K\left(\frac{\mathbf{x}-\mathbf{y}}{h}\right) - K\left(\frac{\mathbf{x}-\mathbf{y}_j}{h}\right)\right] f(\mathbf{y})d\mathbf{y}\right| \\ \leq & \max[f(\mathbf{y})]\left|\sum_{i=1}^{m}\frac{1}{h^d}\int_{I_j}\left[K\left(\frac{\mathbf{x}-\mathbf{y}}{h}\right) - K\left(\frac{\mathbf{x}-\mathbf{y}_j}{h}\right)\right] d\mathbf{y}\right| \\ = & \rho\mathcal{E}, \end{aligned}$$

where $\rho = \max[f(\mathbf{y})]$, and $\mathcal{E}$ is the numerical integration error of $\int \frac{1}{h^d}K\left(\frac{\mathbf{x}-\mathbf{y}}{h}\right)d\mathbf{y}$ using rectangular method. This means the bias of the FKDE is bounded by $|Bias(\hat{f}_F(\mathbf{x})| \leq |Bias(\hat{f}(\mathbf{x}))| + \rho\mathcal{E}$. In the univariate case, $\mathcal{E} = O(r^2)$, where r is the PARTITION parameter, and the bias of the

standard kernel estimate is $|Bias(\hat{f}(\mathbf{x}))| = O(h^2)$ [13]. Therefore the bias of the FKDE can be ultimately obtained as $|Bias(\hat{f}_F(\mathbf{x}))| = O(h^2) + O(r^2)$. This explains why we should not set r too large compared with h, or else the approximation would be poor. Similar conclusions can be drawn on the variance of the FKDE as it can also be written in form of integration [13].

In practice, the selection of r is a trade-off between speed and accuracy. If $r = 0$, the proposed FMS/FKDE and the original MS/KDE algorithms are identical and there is no speedup. The larger is r, the faster is the proposed algorithm and the larger is the error. But the error can be well-controlled if r is reasonably smaller than the bandwidth h. This guarantees that the possible negative effect of pre-clustering (the PARTITION step) can be weakened greatly. In experiments, we choose r to be $0.2h - 0.7h$. The results are quite satisfactory with significant speedup.

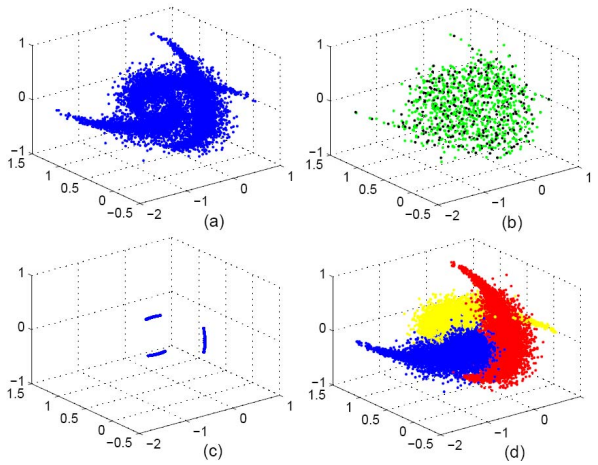


Figure 5. Illustration of FMS clustering.

5. Experiments

We apply FMS to clustering and image segmentation. All the codes are written in C++ and run on a 2.26GHz Pentium-III PC. The first experiment is to cluster a synthetic data set with 32640 3-D points (Figure 5(a)). Standard unsupervised procedure such as ISODATA [12] will fail on this data set. Comaniciu and Meer [3] proposed running mean shift on a set of m randomly selected points, where the distance between any two neighbors is at least h, and the points should not lie in sparsely distributed regions. The resultant complexity in the clustering step is $O(mN)$, much larger than our $O(m^2)$ (strictly speaking, $O(m_s m_t)$, where m_s and m_t are number of sources and targets, respectively).

We use hierarchical PARTITION with initial $r_0 = 1$ and a 30% decrease of r on each level of hierarchy, until the threshold $r_T = 0.15$ is reached. Then, we fuse the local set centers (Section 2.5), and obtain 710 sources and 331 targets (represented as light and dark points in Figure 5(b), respectively). The Gaussian kernel with bandwidth $h = 0.15$ is used. Figure 5(c) plots the convergent targets, and Figure 5(d) is the clustering result. Table 1 gives the run time and error rate of the FMS clustering algorithm when the PARTITION threshold r_T varies.

We also perform segmentation through clustering in the L*u*v color space, with the feature values normalized to $[0, 1]$. Figure 6 shows the experimental results. The left-most column is the original image, and the second column is the segmentation results by using the standard mean shift procedure, which is time consuming. The three right-most columns are the FMS segmentation results with different PARTITION parameters. Both the mean shift and fast mean shift algorithms use a Gaussian kernel with bandwidth $h = 0.08$. The run time of FMS is listed in Table 2.

The improved fast Gauss transform [14], which is applied to speed up the calculation of the mean shift vector, has complexity $O(N \log K) + O(Nr_{pd}) + O(Nnr_{pd})$, where N is the sample size, K is the number of partitioned clusters, n is the maximum number of the neighbor clusters for each target, $r_{pd} = \binom{p+d}{d}$, d is the dimension, and p is a parameter for controlling the error. Clearly, FMS has lower complexity, in the mean time, its segmentation results are quite competitive.

6. Conclusion

In many computer vision problems with large sample size, exploring the underlying modes of the distribution based on the whole data set would be quite uneconomical and unnecessary. To alleviate the heavy burden of distribution description and classification, we combine the nearest neighbor rule with density based clustering to develop the fast mean shift (FMS) algorithm, which significantly accelerates the feature space analysis. The underlying density estimate of the FMS algorithm, FKDE, inherits advantages from both smooth and non-smooth estimators, and asymptotically approximates the standard kernel density estimators. Clustering and image segmentation experiments demonstrate that FMS produces satisfactory results with very high speed.

Acknowledgment

This work is partially funded by NSFC (Grant No. 60318003). And the first two authors gratefully acknowledge the support of K. C. Wong Education Foundation, Hong Kong.

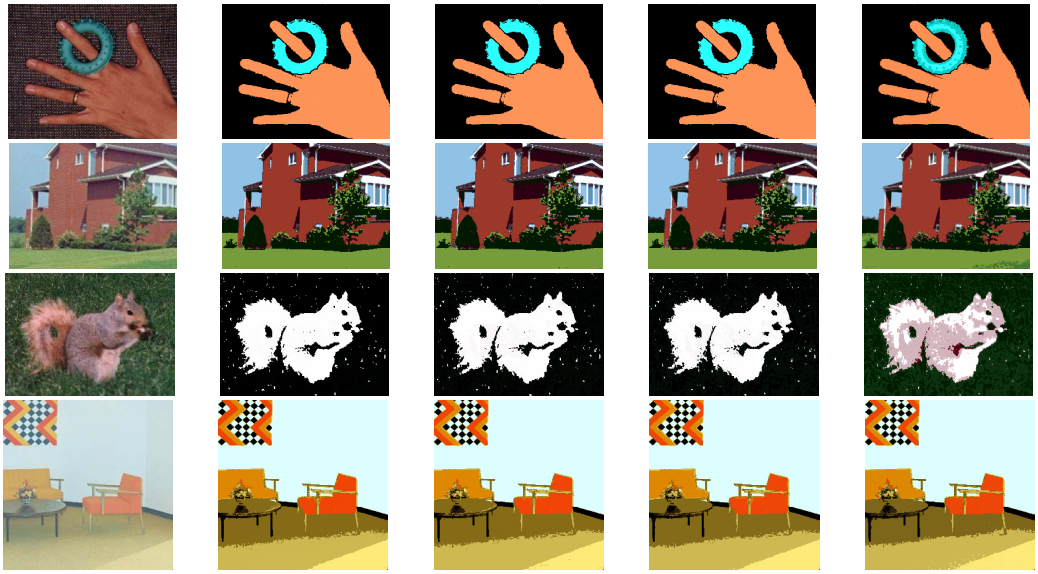


Figure 6. Comparison of mean shift (MS) and fast mean shift (FMS) Segmentation results.

Table 1. Clustering performance of FMS.

r_T	0.05	0.1	0.15	0.2	0.25	0.3
error(%)	0.55	0.89	1.06	1.22	1.82	3.04
time(sec)	14.1	3.01	0.731	0.25	0.12	0.08

Table 2. Segmentation time of FMS (in sec).

image	size	FMS (r_T=0.015)	FMS (r_T=0.03)	FMS (r=0.05)
house	192×255	0.971	0.25	0.08
hand	243×302	1.432	0.361	0.2
room	256×256	0.741	0.21	0.11
squirrel	209×288	1.26	0.23	0.07

References

[1] H. Chen and P. Meer. Robust fusion of uncertain information. In *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, pages 16–22, 2003.

[2] R. Collins. Mean-shift blob tracking through scale space. In *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, pages 234–240, 2003.

[3] D. Comaniciu and P. Meer. Distribution free decomposition of multivariate data. *Pattern Analysis and Applications*, 2(1):22–30, 1999.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, May 2002.

[5] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *Proc. Int'l Conf. Computer Vision*, pages 438–445, 2001.

[6] D. Dementhon. Spatial-temporal segmentation of video by hierarchical mean shift analysis. In *Proc. Statistical Methods in Video Processing Workshop*, 2002.

[7] A. Elgammal, R. Duraiswami, and L. Davis. Efficient nonparametric adaptive color modeling using fast Gauss transform. In *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, pages 563–570, 2001.

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

[9] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: A texture classification example. In *Proc. IEEE Int'l Conf. Computer Vision*, pages 456–463, 2003.

[10] L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Computing*, 12(1):79–94, 1991.

[11] B. Han, D. Comaniciu, Y. Zhu, and L. Davis. Incremental density approximation and kernel-based bayesian filtering for object tracking. In *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, pages 638–644, 2004.

[12] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, USA, 1988.

[13] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[14] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proc. Int'l Conf. Computer Vision*, pages 464–471, 2003.

[15] Z. Zivkovic and B. Kröse. An EM-like algorithm for color-histogram-based object tracking. In *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, pages 798–803, 2004.