Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Fast and accurate kernel density approximation using a divide-and-conquer approach*

Yan-xia JIN†1, Kai ZHANG[2], James T. KWOK[2], Han-chang ZHOU[3]

([1]*School of Electronics and Computer Science and Technology, North University of China, Taiyuan 030051, China*)

([2]*Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China*)

([3]*Key Laboratory of Instrumentation Science and Dynamic Measurement, North University of China, Taiyuan 030051, China*)

†E-mail: jinyanxia_730128@163.com

Received Nov. 3, 2009; Revision accepted Apr. 6, 2010; Crosschecked Aug. 2, 2010

Abstract: Density-based nonparametric clustering techniques, such as the mean shift algorithm, are well known for their flexibility and effectiveness in real-world vision-based problems. The underlying kernel density estimation process can be very expensive on large datasets. In this paper, the divide-and-conquer method is proposed to reduce these computational requirements. The dataset is first partitioned into a number of small, compact clusters. Components of the kernel estimator in each local cluster are then fit to a single, representative density function. The key novelty presented here is the efficient derivation of the representative density function using concepts from function approximation, such that the expensive kernel density estimator can be easily summarized by a highly compact model with very few basis functions. The proposed method has a time complexity that is only linear in the sample size and data dimensionality. Moreover, the bandwidth of the resultant density model is adaptive to local data distribution. Experiments on color image filtering/segmentation show that, the proposed method is dramatically faster than both the standard mean shift and fast mean shift implementations based on kd-trees while producing competitive image segmentation results.

Key words: Nonparametric clustering, Kernel density estimation, Mean shift, Image filtering
doi:10.1631/jzus.C0910668 **Document code:** A **CLC number:** TP391

1 Introduction

In recent years, there has been a lot of interest in the study of nonparametric clustering (Comaniciu and Meer, 2002; Chang and Yeung, 2008), which in turn is based on nonparametric density estimation (Bouezmarni and Rombouts, 2010). While parametric methods require assumptions about the unknown data distribution, nonparametric density estimation is largely model-free. In particular, one needs to specify neither the number of clusters nor their shapes. This flexibility in data modeling is particularly attractive in computer vision tasks such as image segmentation, where the image segments are usually non-Gaussian, and clustering assumptions are rarely warranted.

Among various nonparametric clustering techniques, the mean shift algorithm is particularly successful (Ozertem *et al*., 2008), and has been applied extensively in many vision and pattern recognition problems such as image segmentation (Rao *et al*., 2009), texture classification, tracking (Wang and Liu, 2009; Yu *et al*., 2009; Zivkovic *et al*., 2009), video processing (Ren *et al*., 2009), and more recently, multi-resolution community detection (Zhang *et al*., 2009).

The underlying nonparametric density estimator used in the mean shift algorithm is the well-known kernel density estimator (Mokkadem *et al*., 2009). Here, a 'kernel' function is centered at each sample and the density is estimated as a positive combination of all these kernel functions. The most commonly

* Project (No. 9140C1204060809) supported by the National Key Laboratory Foundation of China

used kernels for the mean shift algorithm are the Epanechnikov and Gaussian kernels. Under mild conditions, kernel density estimators approximate the underlying true density distribution asymptotically (Bouezmarni and Rombouts, 2010). As one kernel function is associated with each sample, however, kernel density estimation, and thus also the mean shift procedure, becomes very inefficient on large datasets. Given a set of N samples, computing the density at a single sample already takes $O(N)$ time, and the whole dataset thus takes $O(N^2)$ time (Georgescu *et al*., 2003).

Note that the kernel profile (given a kernel function K, its profile is the univariate function k: $[0, +\infty)\to\mathbb{R}$ such that $K(\boldsymbol{x})=k(\|\boldsymbol{x}\|)^2$) is typically a decreasing function. Consequently, the kernel density estimate at a sample is dominated by contributions from its neighbors, and the mean shift procedure often spends most of the time finding these neighbors (Georgescu *et al*., 2003). Another speedup paradigm thus aims at using spatial data structures to facilitate this so-called multidimensional range searching problem. The most prominent example is the kd-tree (Barbay *et al*., 2007), which has also led to significant speedups in *k*-means clustering (Chang *et al*., 2009) and mixture models.

From an optimization perspective, it can be shown that mean shift can be regarded as a bound optimization method (Fashing and Tomasi, 2005). Building on this connection, Shen and Brooks (2005) and Shen *et al.* (2007) proposed a fast mean shift procedure (called adaptive over-relaxed mean shift) using the technique of over-relaxing the step size in bound optimization algorithms (Ruslan and Sam, 2003). Experimentally, this can lead to a speedup factor of about 2 to 5 compared to the standard mean shift procedure.

In this paper, we adopt a well-established technique in the design of algorithms for large problems: divide-and-conquer. We first decompose the dataset into a number of small, compact clusters. The density component in each local cluster, which is a summation of kernel functions, is then replaced by a single representative kernel. While similar ideas have been explored, its main difficulty lies in computing this single kernel function efficiently. The expectation-maximization (EM) algorithm is prohibitively expensive on large datasets, such as those typically encountered in image processing. In comparison, our proposed algorithm has a low complexity that is only linear in the sample size and dimensionality. It thus scales much better than the EM algorithm, fast Gauss transform (FGT), or improved FGT (IFGT). Moreover, while the other scale-up methods are tailor-made for specific kernels, our method can be used with any kernel. Experimental results show that it can successfully capture the multi-modal structure of the data distribution with only a small number of kernels. Besides, its speedup relative to standard mean shift is much more dramatic than that of existing speedup methods.

2 Nonparametric clustering using mean shift

Nonparametric cluster analysis detects data clusters by the modes of its probability density. Given a set of samples $S=\{x_1, x_2, ..., x_N\}$ in $\mathbb{R}^d$ generated from an underlying probability density f, the kernel density estimate at point x is given by (Parzen , 1962)

$$\hat{f}(x)=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|H_i|^{1/2}}K_{H_i}(x-x_i), \tag{1}$$

where K is the kernel (which is usually radially symmetric and unimodal), $K_{H_i}(x-x_i)=K(H_i^{-1/2}(x-x_i))$, and H_i is a symmetric, positive definite bandwidth matrix associated with sample x_i. When all $H_i=H$, Eq. (1) reduces to the fixed bandwidth kernel density estimator.

The mean shift algorithm is an iterative, mode-seeking procedure based on the nonparametric estimate of the gradient ($\nabla f(x)$) of the underlying probability density. Suppose, for simplicity, that the bandwidth matrix is $H=h^2I$, where I is the identity matrix. For each sample x, it computes the so-called mean shift vector (Comaniciu and Meer, 2002) that points towards the steepest ascent direction of $\hat{f}(x)$. The algorithm then shifts $x^{(t+1)}\leftarrow x^{(t)}+m(x^{(t)})$ and repeats the process until convergence. Detailed analysis on the adaptive step length and convergence properties of the mean shift iteration can be found in Comaniciu *et al*. (2002). The mean shift algorithm has the key advantage that it can identify arbitrarily shaped clusters in the feature space. Note that there are N terms in the summations of Eqs. (1) and (2); therefore, it can also be prohibitively expensive due to

the resultant $O(N^2)$ complexity.

$$m(x)=\frac{\sum_{i=1}^{N}k'\left(\left\|(x-x_i)/h\right\|^2\right)x_i}{\sum_{i=1}^{N}k'\left(\left\|(x-x_i)/h\right\|^2\right)}-x_i. \qquad (2)$$

The high complexity of kernel density estimation (and thus mean shift) arises from the large number of terms (kernels) involved in Eq. (1). If the number of terms can be reduced without sacrificing the accuracy of $\hat{f}(x)$, the complexity can also be lowered. Hence, in this paper, the standard kernel density estimator in Eq. (1) is replaced by a simpler model. The primary issues are:

1. How to avoid the degradation of the simplified density estimator?

2. How to automatically determine the number of terms needed?

Otherwise, we will encounter the same model selection problem in parametric density estimation.

Recall that kernels are usually unimodal, radially symmetric functions. Therefore, when several kernels with very close centers are summed together, the resultant function will demonstrate significant similarity with the original kernel. This is illustrated in Fig. 1. In Fig. 1a, the Gaussian kernels are close together and their summation is very similar to a single Gaussian. In Figs. 1b and 1c, the kernels become more separated and the sum gradually deviates from a Gaussian. This suggests that by partitioning the set of samples into spatially local clusters, the kernel density component in each cluster can be well represented by a single kernel.

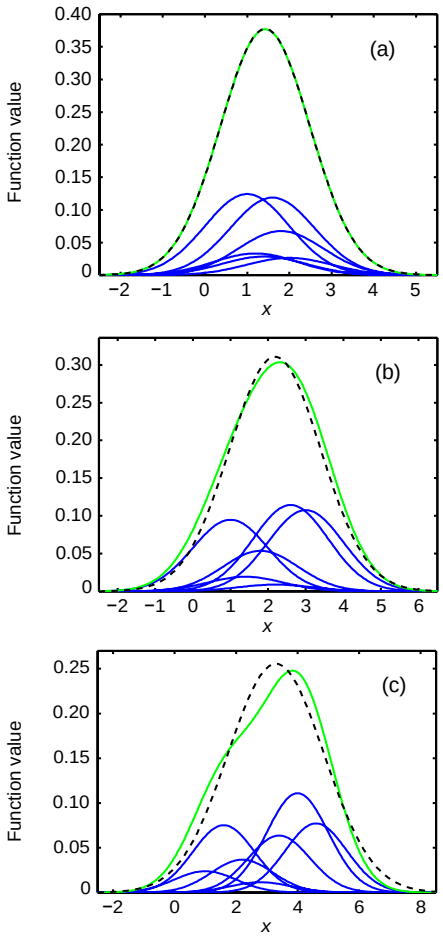

Fig. 1 The sum (solid line) of a group of nearby Gaussian kernels with random weights

A single Gaussian, with properly selected center and width, is also provided for reference (dashed line). (a) Close together; (b) More separated; (c) Far from each other

3 Fitting the kernel density estimator

Instead of involving all N kernel functions in Eq. (1), one approximates the kernel density estimate $\hat{f}(x)$ by a simpler model $f^M(x)$. The proposed nonparametric clustering algorithm follows the divide-and-conquer strategy and proceeds as follows.

1. Divide: partition the whole dataset $S=\{x_1, x_2, ..., x_N\}$ into m disjoint, local clusters $S_1, S_2, ..., S_m$, where $m<<N$. The complex kernel density estimator $\hat{f}$ in Eq. (1) is also decomposed accordingly into m small, 'local' estimates $\hat{f}_i^{\ell}(x)$, as

$$\hat{f}(x)=\frac{1}{N}\sum_{i=1}^{m}\hat{f}_i^{\ell}(x), \qquad (3)$$

where

$$\hat{f}_i^{\ell}(x)=\sum_{x_j\in S_i}\frac{1}{h_j^d}K\left(\frac{x-x_j}{h_j}\right). \qquad (4)$$

Instead of having N kernels in $\hat{f}$, each $\hat{f}_i^{\ell}$ involves only $|S_i|$ kernels and is much less complicated.

2. Conquer: approximate the local estimate $\hat{f}_i^{\ell}$ in each cluster S_i by a local 'model' involving only one kernel function Ψ:

$$f_i^M(x) = w_i \frac{1}{h_i^d} \psi\left(\frac{x - t_i}{h_i}\right). \quad (5)$$

Here, $w_i \in \mathbb{R}$ and h_i is the bandwidth of Ψ.

3. Clustering: the complete model is simply the sum of all the m local models:

$$f^M(x) = \frac{1}{N} \sum_{i=1}^{m} f_i^M(x). \quad (6)$$

This can then be used for density-based clustering algorithms such as the mean shift.

These three steps are discussed in detail in the following sections.

3.1 Divide: data partitioning

This step partitions the dataset into m disjoint, spatially local clusters. The focus in this step is on the efficiency of the partitioning algorithm and the resultant representation. Therefore, any fast algorithm that can produce a coarse partitioning (i.e., with a small m) will be sufficient. In the following, we focus on vector quantization (VQ) and sequential sampling (SS).

3.1.1 VQ and SS

A popular class of algorithms suitable for this task is VQ, of which the generalized Lloyd algorithm (GLA) is probably the most well-known. It partitions the feature space into m disjoint regions, and samples inside each local region are represented by a code vector. The GLA uses the k-means algorithm to iteratively update the code vectors and local clusters until the quantization error converges. Usually, the GLA solution is sensitive to the setting of the initial code vectors.

Besides, VQ can still be expensive on large datasets. Therefore, we introduce a simple but highly efficient method called SS. First, it randomly selects a sample from S and adds it to the set of cluster centers C. For each remaining x_i's in S, it is assigned to cluster j if the distance between x_i and $c_j \in C$ is smaller than a predefined threshold r. If no such c_j is found, x_i is added to C as a new cluster center. The algorithm terminates after all the samples in S are processed. Note that while VQ uses a number of iterations to refine the code vectors, SS determines them in one pass. Experimentally, SS produces a reasonably good partitioning and is insensitive to the choice of the initial sample. As an illustration, Fig. 2 shows the partitioning results on the 3D RGB color space of the 'hand' image. Different values of the threshold r, corresponding to different resolutions, are used. As can be seen, both VQ and SS obtain comparable results. In the sequel, whenever VQ is chosen for data partitioning, one will always initialize it with the SS solution.

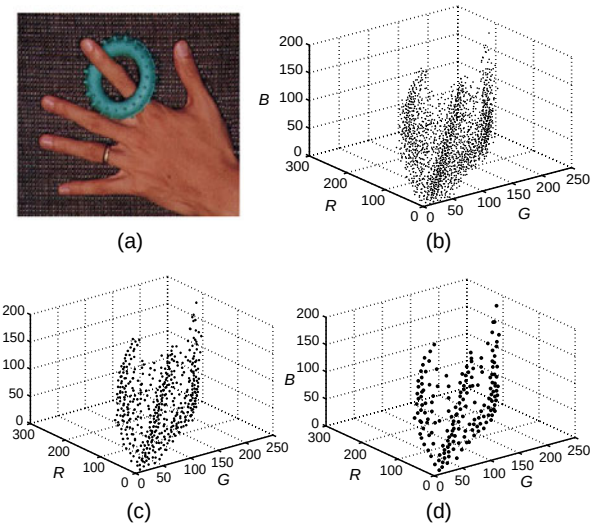

Fig. 2 Cluster centers obtained by partitioning the RGB color space of the 'hand' image using sequential sampling (SS) and vector quantization (VQ)

(a) The 'hand' image; (b) Samples in the RGB color space; (c) SS (predefined threshold r=10); (d) VQ (r=10)

While SS uses a subset of the samples as cluster centers, VQ is more flexible in that the cluster centers can move around. Thus, VQ, though slower than SS in practice, allows the use of a smaller m (and hence a more compact model) for approximation (Section 4.1). Moreover, as will be demonstrated in Section 3.2, VQ has the advantage of being able to assign more clusters to densely populated regions.

3.1.2 Resolution of the data partitioning

The quality of fit between the original kernel density estimate $\hat{f}$ in Eq. (3) and its approximation f^M in Eq. (6) can be measured by $\| \hat{f} - f^M \|^2$, where $\| \cdot \|$ denotes an appropriate norm in some function space. Using the decompositions in Eqs. (3) and (6), we have $\left\| \hat{f} - f^M \right\|^2 \le \frac{1}{N^2} \sum_{i=1}^{m} \left\| \hat{f}_i^{\ell} - f_i^M \right\|^2$.

Hence, if the approximation of $\hat{f}_i^{\ell}$ by f_i^M in each local region is good, the overall approximation of $\hat{f}$ by f^M will also be good. Because f_i^M always uses one single kernel function, the appropriateness of using f_i^M to approximate $\hat{f}_i^{\ell}$ clearly depends on the resolution of data partitioning, which is controlled by the threshold r. Intuitively, the coarser is the partitioning, the less accurate is the approximation but the larger is the computational saving. Our goal is to choose r such that significant speedup can be achieved while still ensuring a good approximation.

In practice, one chooses r to be proportional to the kernel bandwidth. For example, if we use the Gaussian kernel G with bandwidth h_G, then we set $r=\rho_G h_G$ for some constant ρ_G. As a rule of thumb, ρ_G is chosen to be smaller than 2.5. Extensive segmentation experiments in Section 4.2 show satisfactory results on using such a value.

3.2 Conquer: local approximation

In this section, the issue of how to obtain f_i^M at the ith local cluster $S_i=\{x_1, x_2, ..., x_{n_i}\}$ is dealt with. The main ideas are illustrated by first considering the special case of univariate feature space and the use of the Gaussian kernel for density estimation. This will then be followed by the general case and additional remarks.

3.2.1 Univariate case with Gaussian kernels

First, consider fixed-bandwidth kernel density estimation. Let the number of samples in S_i be n_i. As we assume the use of Gaussian kernels here, each $\hat{f}_i^{\ell}(x)$ in Eq. (4) becomes a mixture of n_i Gaussians. Similarly, each local model $f_i^M(x)$ in Eq. (5) is a single Gaussian.

$$\hat{f}_i^{\ell}(x)=\frac{1}{\sqrt{2\pi}h}\sum_{x_j\in S_i}\exp\left(-\frac{(x-x_j)^2}{2h^2}\right). \tag{7}$$

$$f_i^M(x)=\frac{w_i}{\sqrt{2\pi}h_i}\exp\left(\frac{-(x-t_i)^2}{2h_i^2}\right). \tag{8}$$

The task is to determine the unknown variables w_i, t_i, and h_i associated with $f_i^M(x)$. As is common in function approximation, we use the norm of the L^2 space as the error criterion.

$$\varepsilon_i=\int_{+\infty}^{-\infty}(\hat{f}_i^{\ell}(x)-f_i^M(x))^2\,\mathrm{d}x.$$

Plugging in the definitions of $\hat{f}_i^{\ell}(x)$ and $f_i^M(x)$, one has

$$\varepsilon_i=\frac{\sqrt{\pi}h_i}{2\pi}\left[\frac{w_i^2}{h_i^2}-2\frac{w_i}{h_i h}\sum_{x_j\in S_i}\sqrt{\frac{2h^2}{h^2+h_i^2}}\exp\left(-\frac{(x_j-t_i)^2}{2(h^2+h_i^2)}\right)\right]d+\text{const},$$

$$\text{const}=\frac{1}{2\pi h^2}\int_{+\infty}^{-\infty}\left[\sum_{x_j\in S_i}\exp\left(-\frac{(x-x_j)^2}{2h^2}\right)\right]^2\mathrm{d}x,$$

after simplifications. Here, const is independent of the corresponding partial derivatives of w_i, t_i, and h_i. To minimize ε_i with respect to w_i, t_i, and h_i, one can set the corresponding partial derivatives of ε_i to zero. This leads, however, to a nonlinear system that is quite difficult to solve.

Here, a method is proposed that decouples the relations among these three parameters and solves them one after another. First, observe that $\frac{\partial\varepsilon_i}{\partial t_i}=0$ implies

$$t_i=\frac{\sum_{x_j\in S_i}x_j\exp\left(-\frac{(x_j-t_i)^2}{2(h^2+h_i^2)}\right)}{\sum_{x_j\in S_i}\exp\left(-\frac{(x_j-t_i)^2}{2(h^2+h_i^2)}\right)}.$$

If h_i is known, this takes the form of an iterative contraction mapping, and one can obtain t_i using the fixed point method. To be more specific, one starts with an initial $t_i^{(0)}$ (in this paper, the mean of the samples in S_i is used as $t_i^{(0)}$), and then iterates until convergence.

$$t_i^{(k+1)}=\frac{\sum_{x_j\in S_i}x_j\exp\left(-\frac{(x_j-t_i^{(k)})^2}{2(h^2+h_i^2)}\right)}{\sum_{x_j\in S_i}\exp\left(-\frac{(x_j-t_i^{(k)})^2}{2(h^2+h_i^2)}\right)}. \tag{9}$$

The converged value of t_i can be subsequently substituted into the equation $\frac{\partial\varepsilon_i}{\partial w_i}=0$ to obtain w_i, as

$$w_i=\sqrt{\frac{2h_i^2}{h^2+h_i^2}}\sum_{x_j\in S_i}\exp\left(-\frac{(x_j-t_i)^2}{2(h^2+h_i^2)}\right). \tag{10}$$

The remaining question is how to determine h_i. Recall that f_i^M is a Gaussian while $\hat{f}_i^{\ell}$ is a mixture of Gaussians. Instead of obtaining h_i from $\frac{\partial \varepsilon_i}{\partial h_i}=0$, which is highly nonlinear, we equate the variances of f_i^M and $\hat{f}_i^{l}$ (after normalizing them as densities), i.e.,

$$h_i^2=\int_{-\infty}^{+\infty}\hat{f}_i^{\ell}(x)\left(x-E[\hat{f}_i^{\ell}(x)]\right)^2\mathrm{d}x.$$

After simplification, one obtains

$$h_i^2=h^2+\frac{1}{n_i}\sum_{x_j\in S_i}(x_j-\overline{x}_i)^2,\quad \overline{x}_i=\frac{1}{n_i}\sum_{x_j\in S_i}x_j, \quad (11)$$

where $\overline{x}_i$ is the mean of the local cluster S_i. Thus, all the parameters of f_i^M are now successfully recovered. The performance of this local approximation method will be illustrated experimentally in Section 4.1.

Note that Eq. (10) actually computes the strength of the local cluster, which can be deemed as an efficient representation of the density function without having to use a large number of samples. In Zhang *et al*. (2009), such density information was applied in scaling up the eigenvalue decomposition of the kernel matrix, hence, spectral clustering, which also demonstrates very encouraging performance there.

The above discussion considers a fixed-bandwidth kernel density estimator. For a variable-bandwidth kernel density estimator where sample x_j is associated with the bandwidth h_j, it can be shown that Eq. (11) will be changed to

$$h_i^2=\frac{1}{n_i}\sum_{x_j\in S_i}h_j^2+\frac{1}{n_i}\sum_{x_j\in S_i}(x_j-\overline{x}_i)^2$$

with analogous changes to the fixed-point iteration in Eq. (9).

3.2.2 General case

Now one examines the case where x is d-dimensional and a general kernel K is used in the $\hat{f}_i^{\ell}$ expansion in Eq. (4). Again, we will focus on fixed-bandwidth estimation. As is common in kernel density estimation, one assumes that the bandwidth matrix is diagonal, i.e., H=diag{$[h^{(1)}]^2$, $[h^{(2)}]^2$, …, $[h^{(d)}]^2$}, where $h^{(j)}$ is the bandwidth in the jth dimension. Moreover, one uses the notation $K_H(x-y)=k(\|x-y\|_H^2)=k((x-y)^{\mathrm{T}}H^{-1}(x-y))$.

For the ith cluster, its local density estimate is

$$\hat{f}_i^{\ell}(x)=|H|^{-\frac{1}{2}}\sum_{j=1}^{n_i}K_H(x-x_j), \quad (12)$$

while the corresponding local model is

$$f_i^M(x)=w_i|H_i|^{-\frac{1}{2}}K_{H_i}(x-t_i) \quad (13)$$

with center t_i, diagonal bandwidth matrix H_i=diag{$[h_i^{(1)}]^2$, $[h_i^{(2)}]^2$, …, $[h_i^{(d)}]^2$}, and weight w_i. Denote the covariance matrices of f_i^M and $\hat{f}_i^{\ell}$ (after normalization as densities) by $\sum_i^M$ and $\sum_i^l$, respectively. Similar to Section 3.2.1, one first determines the unknown H_i by requiring $[\sum_i^M]_{pp}=[\sum_i^l]_{pp}$; i.e., the corresponding diagonal entries are equalized.

$$h_i^{(p)}=\sqrt{h_p^2+\frac{1}{c_K}\mathrm{Var}[S_i]_{pp}}, \quad (14)$$

where c_K is a kernel-dependent constant, $\mathrm{Var}[S_i]_{pp}$ is the pth diagonal entry variance matrix of the local sample set S_i, and $1\le p\le d$. The L^2 approximation error ε_i can be written as

$$\varepsilon_i=\int\left(\frac{w_i^2}{|H_i|}K_{H_i}^2(x-t_i)-\frac{2w_i}{\sqrt{|H||H_i|}}\cdot\sum_{x_j\in S_i}K_H(x-x_j)K_{H_i}(x-t_i)\right)\mathrm{d}x.$$

Setting the partial derivative $\frac{\partial \varepsilon_i}{\partial t_i}$ to zero, we obtain the t_i from the iteration:

$$t_i=\frac{\sum_{x_j\in S_i}\int k\left(\left\|x-x_j\right\|_H^2\right)k'\left(\left\|x-t_i\right\|_{H_i}^2\right)x\mathrm{d}x}{\sum_{x_j\in S_i}\int k\left(\left\|x-x_j\right\|_H^2\right)k'\left(\left\|x-t_i\right\|_{H_i}^2\right)\mathrm{d}x}. \quad (15)$$

Finally, on setting $\frac{\partial \varepsilon_i}{\partial w_i}=0$, we obtain the weight w_i as

$$w_i = \frac{\sum_{x_j \in S_i} |H_i|^{\frac{1}{2}} \int K_H(x - x_j) K_{H_i}(x - t_i) \mathrm{d}x}{|H|^{\frac{1}{2}} \int K_{H_i}^2(x - t_i) \mathrm{d}x}. \quad (16)$$

3.2.3 Remarks

The proposed approach leads to a number of interesting observations. First, Eq. (9), which is used to determine the center of f_i^M, is indeed a mean shift iteration for finding the peak:

$$p(x) = \sum_{x_j \in S_i} \exp\left(-\frac{(x - x_j)^2}{2(h^2 + h_i^2)} \right).$$

Note that both p and $\hat{f}_i^{\ell}$ are kernel density estimates on S_i, but the bandwidth of p (i.e., $h^2+h_i^2$) is larger than that of $\hat{f}_i^{\ell}$ (i.e., h^2). In addition, Eq. (11) shows that the new Gaussian kernel in $\hat{f}_i^{\ell}(x)$ has a variance (h_i^2) which is equal to the sum of h^2 (variance of the Gaussian kernel used in the original density estimator) and the variance of the local sample set S_i. In other words, the bandwidths h_i's used in different S_i's are different and depend on the local data distributions. Thus, even when the original kernel density estimator $\hat{f}$ has a fixed bandwidth, our resultant model f^M will always be a variable-bandwidth estimator. Actually, this property is also related to the data partitioning step using VQ. As illustrated in Fig. 3, dense regions are more finely segmented by VQ. Thus, samples in the corresponding partitions have a smaller variance and contribute less to the h_i in Eq. (11). On the contrary, sparse regions are coarsely segmented. Therefore, the corresponding data covariances are large, leading to larger h_i's.

3.3 Clustering

After obtaining all the f_i^M's individually, combine them to form the complete model:

$$f^M(x) = \frac{1}{N} \sum_{i=1}^{m} |H_i|^{-\frac{1}{2}} w_i K_{H_i}(x - t_i),$$

which can then be used for clustering. For example, the mean shift iteration can be obtained by setting the gradient of f^M to zero, as

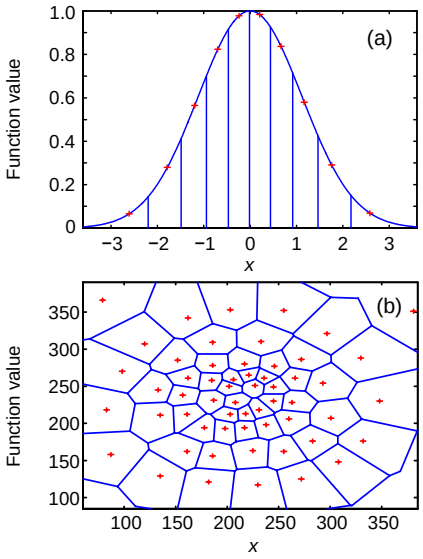

Fig. 3 Partitioning results by vector quantization (VQ) when the sample distribution is a 1D (a) or 2D (b) Gaussian

$$x^{(t+1)} = \tilde{H}^{(t)} \frac{\sum_{i=1}^{m} |H_i|^{-\frac{1}{2}} w_i k'\left(\left\| x^{(t)} - t_i \right\|_{H_i}^2 \right) H_i^{-1} t_i}{\sum_{i=1}^{m} |H_i|^{-\frac{1}{2}} w_i k'\left(\left\| x^{(t)} - t_i \right\|_{H_i}^2 \right)},$$

$$(\tilde{H}^{(t)})^{-1} = \frac{\sum_{i=1}^{m} |H_i|^{-\frac{1}{2}} w_i k'\left(\left\| x^{(t)} - t_i \right\|_{H_i}^2 \right) H_i^{-1}}{\sum_{i=1}^{m} |H_i|^{-\frac{1}{2}} w_i k'\left(\left\| x^{(t)} - t_i \right\|_{H_i}^2 \right)}. \quad (17)$$

3.4 Time complexities

The proposed method includes three steps: data partitioning, local approximation, and clustering.

1. For the first step, both VQ and SS have a time complexity of $O(Ndm)$. By using a hierarchical scheme, this can be further reduced to $O(Nd\log_2 m)$ (Xu and Xu, 2010). In practice, however, VQ is slower as additional iterations are used to refine the initial cluster centers.

2. The local approximation step involves determining the parameters of each local model f_i^M. It is easy to see that computing the bandwidth H_i in Eq. (14), center t_i in Eq. (15), and weight w_i in Eq. (16) all takes $O(n_i d)$ time. Hence, the total complexity for this step is: $\sum_{i=1}^{m} O(n_i d) = O(Nd)$.

3. For the last clustering step, one considers the use of the mean shift algorithm. Its complexity is then $O(mNd)$ since the model f^M in Eq. (17) involves only m kernel functions. If one further assumes that sam-

ples in the same cluster share the same class label, then the mean shift iteration needs only to be applied to the cluster centers (t_i's), and the complexity can be further reduced to $O(m^2d)$.

Summing up all the three steps, the overall complexity is

$$O(Nd\log_2 m)+O(Nd)+O(m^2d)=O(Nd\log_2 m+m^2d), \quad (18)$$

which is linear in both the sample size N and data dimensionality d. In image segmentation, m is in the tens/hundreds while N, the number of pixels, is in the hundreds of thousands (e.g., Table 2 in Section 4.2). Thus, the complexity is typically dominated by the $Nd\log_2 m$ term.

Alternatively, consider instead the use of the kd-trees to speed up density estimation (or the mean shift procedure). Each range query takes $O(N^{(d-1)/d}+p)$ time, where p is the number of samples found in the neighborhood (de Berg *et al*., 2008). For a set of m cluster centers, the total is plus $O(dN\log_2 N)$ time for building the kd-tree. In image segmentation, p is at least in the thousands or tens of thousands. Thus, this is much more expensive than that in Eq. (18). Experimental results in Section 4.2 also show that our proposed method is empirically faster than kd-tree based methods. In Zhang and Kwok (2007), a divide-and-conquer scheme was also used; however, the bandwidth used for local clusters were fixed, thus unadaptive to the data distribution and may lead to worse performance as will be empirically shown in the experiments (Fig. 5). In Zhang *et al*. (2005), a full covariance matrix was used for each local cluster, which leads to higher computational cost in case of high dimension.

Han *et al.* (2004) proposed a related density approximation approach for kernel-based object tracking. The modes of density distribution were detected by mean shift, and one Gaussian was placed around each mode to produce an approximate density model. As mean shift itself is quite expensive, however, they resorted to an incremental strategy that processes samples once at a time. This, unfortunately, also leads to an increase in the approximation error. Moreover, the covariance matrices of all the modes have to be updated as each sample is processed, causing a total number of $O(mN)$ matrix updates. This makes the approximation computationally inefficient.

3.5 Discussions

In this part we discuss the conditions under which our method can give desired scaling-up benefit with good accuracy. Our method is supposed to work well where there are relatively few clusters and a large number of data points. In particular, the larger the sample size, the more obvious the speedup. This is because our basic assumption is that close sample points can be combined and represented by a single kernel, as graphically shown in Fig. 1. If the number of clusters is large, then more partitions will be needed to discriminate between these clusters for a desired accuracy, which may hamper the efficiency. Actually, the original mean shift algorithm has been most commonly applied in such situations (i.e., large sample size and a few clusters), because it is based on the nonparametric density estimator, which usually requires a large number of samples to estimate the probability density function. For example, the most well-known application of mean shift is image segmentation, where the amount of sample points (pixels) is typically huge and the number of clusters (objects) is relatively small. In case there are a large number of clusters, the clustering problem becomes much more difficult, and our method may lead to degenerate performance. In this case supervised learning is more suitable.

4 Experiments

The performance of the proposed algorithm was studied in two aspects. First, from a density approximation viewpoint, we measured the approximation error between the simplified density model f^M and the standard kernel density estimator $\hat{f}$. In addition, one can also study how well clustering can be performed using the approximate density model f^M in real-world applications. The clustering algorithm was chosen as mean shift for color image segmentation.

4.1 Performance of local approximation

First, we studied the approximation of $\hat{f}_i^{\ell}$ by the local density model f_i^M in the ith cluster S_i. For illustration, consider the 1D case. Recall that the local cluster occupies a small, compact region. So, the density distribution was taken in the local cluster as

linear (of the form $ax+b$), i.e., a first-order approximation of the global density $f(x)$ in the domain of S_i. In the following, we used a=1, b=0.1, S_i=[0, 2], and the Gaussian kernel with bandwidth h=1. Using different choices of h_i, we obtained the values of w_i's and t_i's as discussed in Section 3.2, and the resultant errors ε_i's are plotted against h_i in Fig. 4. As can be seen, the minimum ε_i was obtained when h_i was equal to the proposed value in Eq. (11). On the other hand, setting h_i to either the bandwidth (h) of the original kernel or the standard deviation of the samples in S_i led to inferior results.

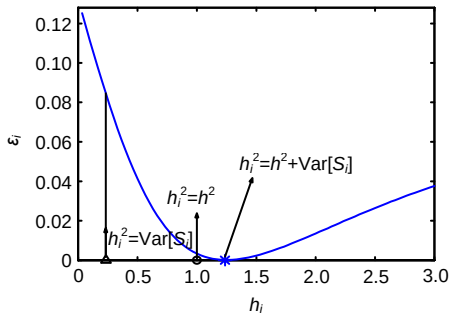

Fig. 4 The approximation error ε_i vs. the bandwidth h_i

To study the effect of different data partitioning (SS/VQ) and bandwidth selection schemes (fixed/variable) on the approximation quality, we performed experiments on another 1D set with 2000 samples drawn from the Gaussian mixture:

$$0.4N(-2.6, 0.09)+0.4N(-0.8, 0.36)+0.2N(1.7, 0.64),$$

where $N(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2. Again, the Gaussian kernel, with bandwidth h=0.4, was used for density estimation. As can be seen from Fig. 5, when the partitioning threshold was relatively small (r=0.5), all the four schemes achieved good approximation. When r was increased to 1.0, the density had to be modeled with fewer basis functions. The approximation qualities began to degrade, though the variable bandwidth scheme still performed better than the fixed bandwidth scheme. When r was further increased to r=1.5, only the variable bandwidth scheme provided satisfactory approximation. Moreover, generally speaking, using VQ for data partitioning led to better approximation results than using SS. Fig. 6 shows the total approximation log-error $\log_2\varepsilon$ versus the partitioning threshold r on this 1D dataset, where $\varepsilon=\sum_{i=1}^{m}\varepsilon_i$ is the sum of all the local approximation errors. As can be seen, the error decreased as the partitioning became finer (i.e., r gets smaller). However, the use of a smaller r also implies a larger value of m, and thus the resultant model f^M becomes more expensive. Moreover, the variable bandwidth scheme significantly reduced the approximation error compared to the fixed bandwidth scheme. Besides, as mentioned earlier, the use of VQ for data partitioning produced better approximation than SS. In other words, for the same level of accuracy, VQ can afford the use of a larger partitioning parameter r, though it runs slower than SS in practice.

4.2 Color image segmentation

In this subsection, experiments are reported on segmentation by running the (range-domain) mean shift clustering algorithm in the RGB color space. Each detected cluster in the RGB space corresponded to a homogeneous region (segment) in the image. Experiments were performed on a number of benchmark images used in Comaniciu and Meer (2002).

The Gaussian kernel was used with bandwidth matrix $H=h^2I$, where $h=\sqrt{150}$. As mentioned in Section 4.1, VQ can afford the use of a larger partitioning threshold r. Therefore, in the experiment, r was chosen to be much larger in VQ than in SS. For comparison, we also implemented the standard mean shift algorithm and its fast version using kd-trees. All the codes were implemented in C++ and run on a 2.26 GHz Pentium IV machine. As the 'true' segmentation of an image is subjective, only a visual comparison is intended here.

Segmentation results are shown in Figs. 7a–7d. Note that the segmentation results obtained using the proposed algorithm were comparable with those by standard mean shift, even though the partitioning threshold r has been set very large. Moreover, from Table 1, we see that the number of kernel functions in f^M was dramatically smaller than the sample size N. The use of VQ for data partitioning led to further reduction. This shows the success of our approximation scheme in maintaining the structure of the data distribution with very compact models. Segmentation was also very fast using the proposed method. On average, it can be thousands of times faster than the standard mean shift procedure, and hundreds of times

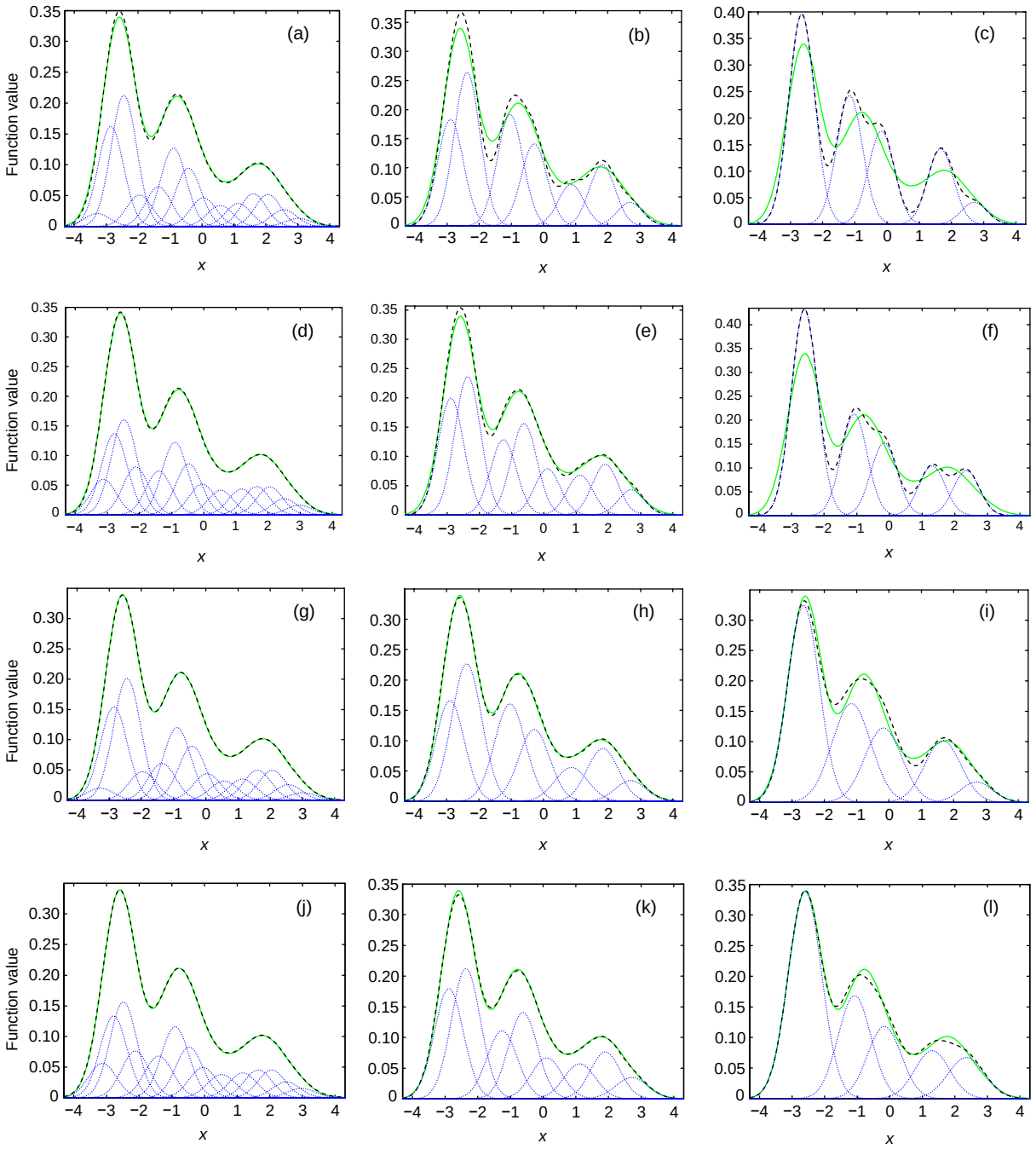

Fig. 5 Approximation results using different partitioning procedures (sequential sampling (SS) and vector quantization (VQ)) and bandwidth selection schemes (fixed/variable) at different predefined partitioning threshold *r*

Solid line: original kernel density estimator that is to be approximated; dashed line: approximation result; dotted line: basis functions. (a) SS+fixed bandwidth, r=0.5; (b) SS+fixed bandwidth, r=1.0; (c) SS+fixed bandwidth, r=1.5; (d) VQ+fixed bandwidth, r=0.5; (e) VQ+fixed bandwidth, r=1.0; (f) VQ+fixed bandwidth, r=1.5; (g) SS+variable bandwidth, r=0.5; (h) SS+variable bandwidth, r=1.0; (i) SS+variable bandwidth, r=1.5; (j) VQ+variable bandwidth, r=0.5; (k) VQ+variable bandwidth, r=1.0; (l) VQ+variable bandwidth, r=1.5

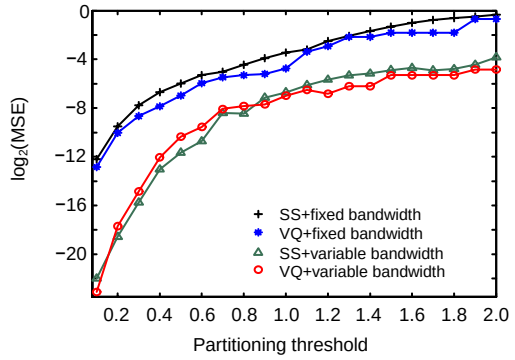

Fig. 6 Mean-squared errors obtained by different approximation schemes and partitioning thresholds
SS: sequential sampling; VQ: vector quantization

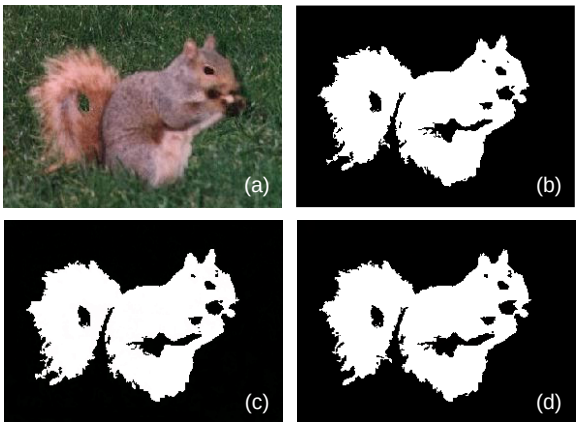

Fig. 7 The 'squirrel' image and the segmentation results
(a) Original image; (b) Standard mean shift; (c) Proposed method with sequential sampling (SS) and partitioning threshold $r=\sqrt{300}$; (d) Proposed method with vector quantization (VQ) and $r=\sqrt{800}$

Table 1 Number of kernel functions used in our method on the various segmentation tasks on the 'squirrel' image with a sample size (N) of 209×288=60 192 pixels

Method	Number of kernel functions
Sequential sampling*	81
Vector quantization**	26

* Partitioning threshold $r=\sqrt{300}$; ** $r=\sqrt{800}$

faster than the fast mean shift implementation using kd-trees (Table 2). More segmentation results on images from the 'starfish' segmentation benchmark dataset (with $h=\sqrt{100}$ and $r=\sqrt{300}$) are shown in Fig. 8.

Table 3 shows a detailed breakdown of the time consumed in the various steps of the mean shift procedure. As can be seen, the total time was often dominated by the clustering step, which involves the computations of the mean shift vectors. In both the standard and kd-tree-based implementations, this is very expensive because one kernel function is associated with each sample and the sample size is large. As kd-trees aim only at facilitating the range searching step, but not at reducing the large number of kernel functions, its speedup improvement was significant relative to the standard mean shift but still inferior to our proposed method.

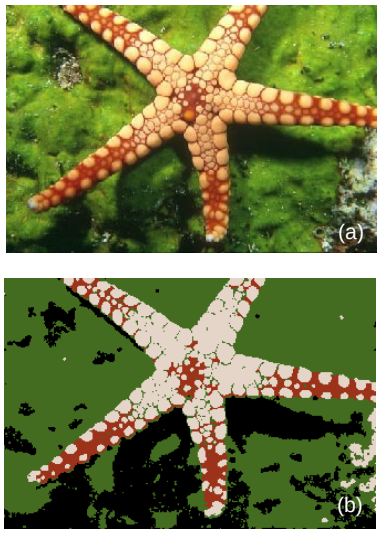

Fig. 8 Segmentation results on the 'starfish' image dataset
(a) Original image; (b) Segmentation result

Table 2 Total wall time on the various segmentation tasks on the 'squirrel' image

Segmentation task	Total wall time (s)
Mean shift	
Standard	1215.8
kd-tree	11.94
Our method	
SS, $r=\sqrt{300}$	0.05 (24 316)
SS, $r=\sqrt{800}$	0.27 (4486)

Number in brackets is the image size (in pixel). SS: sequential sampling

Table 3 Detailed breakdown of the time consumption on the 'squirrel' image as required by the kd-tree-based mean shift procedure and our method

Task	Time (s)	
	kd-tree	Our method*
Model building	0.21	0.03
Range sending	5.69	0.01
Clustering	6.04	0.01
Total time	11.94	0.05

* Sequential sampling, partitioning threshold $r=\sqrt{300}$

4.3 Demonstration of the effect of pre-clustering

In this part we demonstrate the usefulness of the pre-clustering (partition) step in mean shift clustering. We compared our method with the sub-sampling based method, where only a subset of samples were chosen on which the mean shift algorithm was performed. For a fair comparison, we performed sampling such that the two methods had roughly the same amount of computational cost. The results presented on the 'squirrel' image (Fig. 9) showed that the sub-sampling based method led to poor clustering performance (Fig. 9c). This is because to achieve a similar speedup, only a small subset of samples can be chosen, which is not sufficient for a reliable estimation of the underlying probability density function. As a result, the clustering based on the structure of this poor density estimator was also unsatisfactory. In comparison, our method successfully summarized the distribution information by computing the strength and covariance of local partitions (Fig. 9b).

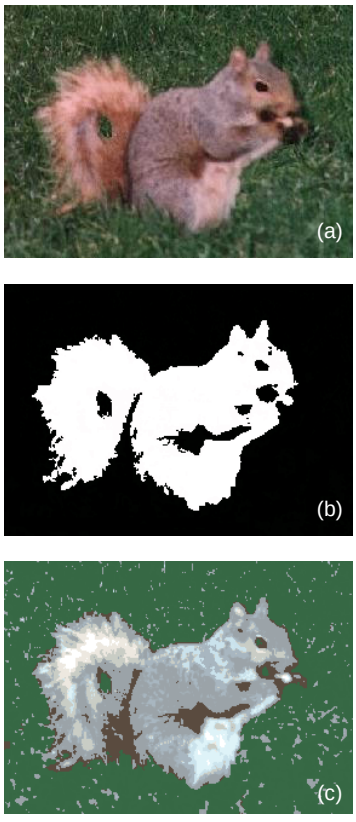

Fig. 9 The 'squirrel' image and the segmentation results by our method and the sampling-based method
(a) Original image; (b) Result by standard mean shift; (c) Result by the sampling-based method

5 Conclusions

In this paper, we propose an approximate density modeling approach that significantly reduces the complexity of density-based nonparametric clustering. Using ideas of divide-and-conquer and function approximation, we partition the dataset into spatially local clusters and derive an efficient method to summarize the density components in each cluster. The proposed method has a time complexity that is only linear in both the number of samples and data dimensionality. Moreover, the bandwidth of the resultant density model is adaptive to the local data distribution. Image segmentation experiments show that the proposed method achieves comparable segmentation results but is superior, in terms of both speed and size of the resultant density model, to both the standard and fast mean shift procedures based on kd-trees.

In the future, we will study the incorporation of traditional bias and variance reduction techniques, such as variable bandwidth selection schemes, into the proposed algorithm. Note that, while these schemes are often prohibitively expensive on large datasets, our simplified density model involves only a small number of basis functions and these schemes should then become more feasible. Besides, the technique of over-relaxing the step size in the mean shift procedure can be readily integrated with the proposed method, and we expect another speedup factor of 2 to 5 similar to those reported in Shen *et al*., (2007). Moreover, the proposed method can also be extended to speed up support vector machines on high-dimensional data. Another problem is to refine the method when the number of clusters in the datasets is large. In such case the incorporation of (semi-)supervised information is expected to help and will be a very interesting future direction.

References

Barbay, J., Golynski, A., Munro, J.I., Rao, S.S., 2007. Adaptive searching in succinctly encoded binary relations and tree-structured documents. *Theor. Comput. Sci.*, **387**(3):284-297. [doi:10.1016/j.tcs.2007.07.015]

Bouezmarni, T., Rombouts, J.V.K., 2010. Nonparametric density estimation for multivariate bounded data. *J. Statist. Plan. Infer.*, **140**(1):139-152. [doi:10.1016/j.jspi.2009.07.013]

Chang, D.X., Zhang, X.D., Zheng, C.W., 2009. A genetic algorithm with gene rearrangement for *k*-means

clustering. *Pattern Recogn.*, **42**(7):1210-1222. [doi:10.1016/j.patcog.2008.11.006]

Chang, H., Yeung, D.Y., 2008. Robust path-based spectral clustering. *Pattern Recogn.*, **41**(1):191-203. [doi:10.1016/j.patcog.2007.04.010]

Comaniciu, D., Meer, P., 2002. Mean shift: a robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**(5):603-619. [doi:10.1109/34.1000236]

de Berg, M., van Kreveld, M., Overmars, M., Cheong, O., 2008. Computational Geometry: Algorithms and Applications. Springer-Verlag, Berlin, Germany, p.105-120.

Fashing, M., Tomasi, C., 2005. Mean shift is a bound optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, **27**(3):471-474. [doi:10.1109/TPAMI.2005.59]

Georgescu, B., Shimshoni, I., Meer, P., 2003. Mean Shift Based Clustering in High Dimensions: a Texture Classification Example. Proc. 9th IEEE Int. Conf. on Computer Vision, p.456-463. [doi:10.1109/ICCV.2003.1238382]

Han, B., Comaniciu, D., Zhu, Y., Davis, L., 2004. Incremental Density Approximation and Kernel-Based Bayesian Filtering for Object Tracking. Proc. IEEE Computer Society Conf. on Computer Version and Pattern Recognition, p.638-644. [doi:10.1109/CVPR.2004.1315092]

Mokkadem, A., Pelletier, M., Slaoui, Y., 2009. The stochastic approximation method for the estimation of a multivariate probability density. *J. Statist. Plan. Infer.*, **139**(7):2459-2478. [doi:10.1016/j.jspi.2008.11.012]

Ozertem, U., Erdogmus, D., Jenssen, R., 2008. Mean shift spectral clustering. *Pattern Recogn.*, **41**(6):1924-1938. [doi:10.1016/j.patcog.2007.09.009]

Parzen, E., 1962. On estimation of a probability density function and mode. *Ann. Math. Statist.*, **33**(3):1065-1076. [doi:10.1214/aoms/1177704472]

Rao, S., de Martins Martins, A., Principe, J.C., 2009. Mean shift: an information theoretic perspective. *Pattern Recogn. Lett.*, **30**(3):222-230. [doi:10.1016/j.patrec.2008.09.011]

Ren, W., Singh, S., Singh, M., Zhu, Y.S., 2009. State of the art on spatio-temporal information based video retrieval. *Pattern Recogn.*, **42**(2):267-282. [doi:10.1016/j.patcog.2008.08.033]

Ruslan, S., Sam, R., 2003. Adaptive Overrelaxed Bound Optimization Methods. Proc. 20th Int. Conf. on Machine Learning, p.664-671.

Shen, C., Brooks, M., 2005. Adaptive Over-Relaxed Mean Shift. Proc. 8th Int. Symp. on Signal Processing and Its Applications, p.575-578. [doi:10.1109/ISSPA.2005.1581003]

Shen, C., Brooks, M., van den Hengel, A., 2007. Fast global kernel density mode seeking: application to localisation and tracking. *IEEE Trans. Image Process.*, **16**(5):1457-1469. [doi:10.1109/TIP.2007.894233]

Wang, X.H., Liu, J.L., 2009. Tracking multiple people under occlusion and across cameras using probabilistic models. *J. Zhejing Univ.-Sci. A*, **10**(7):985-996. [doi:10.1631/jzus.A0820474]

Xu, G., Xu, J.H., 2010. Efficient approximation algorithms for clustering point-sets. *Comput. Geom.*, **43**(1):59-66. [doi:10.1016/j.comgeo.2007.12.002]

Yu, S.Y., Wang, F.L., Xue, Y.F., Yang, J., 2009. Bayesian moving object detection in dynamic scenes using an adaptive foreground model. *J. Zhejing Univ.-Sci. A*, **10**(12):1750-1758. [doi:10.1631/jzus.A0820743]

Zhang, J., Zhang, K., Xu, X., Tse, C.K., Small, M., 2009. Seeding the kernels in graphs: towards multi-resolution community analysis. *New J. Phys.*, **11**(11):113003. [doi:10.1088/1367-2630/11/11/113003]

Zhang, K., Kwok, J.T., 2007. Simplifying Mixture Models Through Function Approximation. Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, MA, p.1577-1584.

Zhang, K., Tang, M., Kwok, J.T., 2005. Applying Neighborhood Consistency for Fast Clustering and Kernel Density Estimation. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, **2**:1001-1007. [doi:10.1109/CVPR.2005.73]

Zivkovic, Z., Cemgil, A.T., Krose, B., 2009. Approximate Bayesian methods for kernel-based object tracking. *Comput. Vis. Image Understand.*, **113**(6):743-749. [doi:10.1016/j.cviu.2008.12.008]