Fast Estimation of Nonparametric Kernel Density Through PDDP, and its Application in Texture Synthesis

Arnab Sinha and Sumana Gupta
Department of Electrical Engineering
Indian Institute of Technology Kanpur,
Kapur-208016, U.P., India
arnab@iitk.ac.in and *sumana@iitk.ac.in*

Abstract

In this work, a new algorithm is proposed for fast estimation of nonparametric multivariate kernel density, based on principal direction divisive partitioning (PDDP) of the data space.The goal of the proposed algorithm is to use the finite support property of kernels for fast estimation of density. Compared to earlier approaches, this work explains the need of using boundaries (for partitioning the space) instead of centroids (used in earlier approaches), for better unsupervised nature (less user incorporation), and lesser (or atleast same) computational complexity. In earlier approaches, the finite support of a fixed kernel varies within the space due to the use of cluster centroids. It has been argued that if one uses boundaries (for partitioning) rather than centroids, the finite support of a fixed kernel does not change for a constant precision error. This fact introduces better unsupervision within the estimation framework. The main contribution of this work is the insight gained in the kernel density estimation with the incorporation of clustering algortihm and its application in texture synthesis.
Texture synthesis through nonparametric, noncausal, Markov random field (MRF), has been implemented earlier through estimation of and sampling from nonparametric conditional density. The incorporation of the proposed kernel density estimation algorithm within the earlier texture synthesis algorithm reduces the computational complexity with perceptually same results. These results provide the efficacy of the proposed algorithm within the context of natural texture synthesis.

Keywords: Nonparametric density estimation, Kernel density estimation, Principal Component Analysis, Vector quantization, Texture synthesis, Markov random field

1. INTRODUCTION

Multivariate density estimation is an important tool in computer vision for statistical decision making and pattern recognition tasks (Elgammel et al. [2003], Liu et al. [2007], Zhang et al. [2005] etc.). In density estimation, there are broadly three parallel worlds, parametric, semi-parametric and nonparametric, with their own advantages and disadvantages. In parametric approaches, one has to have a priori knowledge of underlying distribution. In semi-parametric approaches, finite mixture models can be found, where, the knowledge of the mixture distribution is assumed to be known a priori, and also the number of mixture components. A popular model is the Gaussian mixture model (GMM). These finite mixture models offer more flexibility than the parametric density estimation methods. Nonparametric density estimation methodologies assume less structure about the underlying distribution, which makes them a good choice for robust and more accurate analysis. There is another advantage of using nonprametric density estimation methodologies over semi-parametric ones, the estimation of the parameters for semi-parametric models is troublesome. A popular method for the parameter estimation of finite mixture models (e.g., GMM), is Expectation Maximization (EM) algorithm (Dempster et al. [1977]). In, Wu [1983], it has been shown that EM algorithm is not guranteed to converge to a global optimum, and *if the likelihood function is unimodal and a certain differentiability condition is satisfied, then any EM sequence*

converges to the unique maximum likelihood estimate. There are also other problems associated with the finite mixture models, such as, what should be the number of mixture components (Pernkopf and Bouchaffra [2005]) or what should be the component density model, or the slow convergence property of EM algorithm (Huang et al. [2005], Thiesson et al. [1999, Revised 2001]).

Among all nonparametric approaches, kernel-based density estimation (KDE) approach offer more flexibility. However, huge storage (required to store all available data vectors), and expensive computational (and time) complexity for the calculation of probability at each target data vector, makes nonparametric kernel density estimation (KDE) less popular than the semi-parametric approaches (such as GMM). Recently, a number of algorithms have been proposed for fast estimation of Kernel density, (Greengard and Strain [1991], Yang et al. [2003], Zhang et al. [2005] etc.). Moreover, the computational efficiency of modern day systems has also increased. As a result, the nonparametric KDE is gaining popularity in different image and video processing applications (Elgammel et al. [2003], Liu et al. [2007], Zhang et al. [2005]).

Natural texture synthesis is an important problem in terms of understanding the mathematical modeling structure of texture, which not only used for artificial textured object surface synthesis in computer graphics, but also in understanding the textures for further processing of images and videos (such as, classification, segmentation, retrieval, compression etc.). Nonparametric, noncausal, Markov Random Field (MRF) model (Paget and Longstaff [1998]), provides a theoretical background for synthesizing textures from a broad range of natural textures. The sampling process for synthesizing texture depends upon repeatitive estimation of Markovian conditional density. This process has a huge computational complexity (Sinha and Gupta [2007]), which does not favor real-time application. In this work, we compare the computational complexity of the original algorithm (Paget and Longstaff [1998]), with the fast sampling algorithm implemented through the proposed fast KDE (FKDE) algorithm. The resulting texture synthesis algorithm is tested visually with respect to a number of natural textures, taken from a standard database (Brodatz [1966]).

In section 2, the mathematical background of KDE is described and the problem is defined. A brief discussion of the earlier algorithms for fast KDE (FKDE) with their limitations are described in section 3. Section 4 present the proposed algorithm for FKDE, with its error bound and computational complexity. The mathematical model for texture synthesis is described briefly with the description of the practical problem in section 5. Section 6 describes the application of the proposed algorithm in natural texture synthesis. The paper is concluded in section 7.

2. KDE PROBLEM INTRODUCTION

Let $t_s \in \Re^d$ be a set of source data points, and $t_n \in \Re^d$ a target data point. In nonparametric Gaussian kernel density estimation, the probability of the target data point t_n is calculated as follows.

$$P(t_n) = \frac{1}{N} \sum_{s=1}^{N} K_H(t_s - t_n) \tag{1}$$

where, $K_H(x) = \frac{1}{(2\pi)^{d/2}(|H|)} \exp\{-(\sum_{i=1}^{d} x^2/2h_i^2)\}$ is the multivariate Gaussian product kernel. Here, N is the number of source data points, $H = \{h_1 \dots h_d\}^T$, $|H| = h_1 h_2 \dots h_d$, $K_H(x)$ is the Gaussian kernel, and $h_i = \sigma_i N^{-1/(d+4)}$, Scott [1992]. The computational complexity of equation 1 is $O(Nd)$ for each target data point. In general, kernels (multivariate or univariate) have numerically finite support, e.g., in case of Gaussian kernel , if the absolute value of the kernel argument becomes ≥ 700, the value of the kernel term becomes quite negligible, i.e., 9.86×10^{-305}. One can use this knowledge in kernel density estimation to minimize the required number of source data points, i.e., considering the source data points within a *radius* from the target data point, expressed as shown in equation 2.

$$P(t_n) = \frac{1}{N} \sum_{s \in R_n} K_H(t_s - t_n) \tag{2}$$

Here, $R_n \subset \Re^d$ is the subspace within a constant area corresponding to a constant radius (say R) around the data point t_n. Let $h_i = h, \forall i$; now, one can define R for Gaussian multivariate kernel as,

$R = \sum_{i=1}^{d} x_i^2 = 700h^2$. And the precision error would be

$$\begin{aligned} P_{err}(t_n) &= \frac{1}{N}\sum_{s \notin R_n} K_H(t_s - t_n) \\ &\leq N_n K_H(R) \text{ ; where } N_n = |\{s \notin R_n\}| \\ &\leq N K_H(R) \end{aligned} \tag{3}$$

In the equation 3, we have described a least upper error bound (LUEB) for the FKDE algorithms. In the rest of this paper, we will analyze the proposed FKDE algorithm with respect to the earlier algorithms in terms of computational complexity and unsupervisedness for attaining the least upper error bound described in equation 3 for a given value of R.

3. EARLIER APPROACHES AND MOTIVATION FOR FURTHER INVESTIGATION

In the following subsections 3.1 and 3.2, we will briefly describe two algorithms for FKDE with their limitations. Afterwards in sub-section 3.3, we will try to provide the seed of a possible solution to the limitations of the earlier algorithms.

3.1. Reconstructionhistogram

Zhang et al. [2005] has described an algorithm named as *reconstructionhistogram* for reducing the computational complexity of the KDE. The source data set $\{t_s\}$ is first clustered within a number of clusters (user defined), $\{Clust_i; i = 1 \ldots M\}$. Here $Clust_i$ describes the mean vector of the i^{th} cluster. Define n_i as the number of source data vectors within i^{th} cluster. Thereafter the probability of target data vector t_n is calculated as follows,

$$P(t_n) = \frac{1}{N}\sum_{i=1}^{M} K_H(t_n - Clust_i) n_i \tag{4}$$

Equation 4 can be thought of as KDE of t_n given the source data points at cluster centroids with a weight factor n_i/N. It can be also thought as GMM. Therefore, it does not have the flexibility of original KDE. Moreover, the error bound is not mentioned by Zhang et al. [2005].

3.2. Improved Fast Gauss Transform

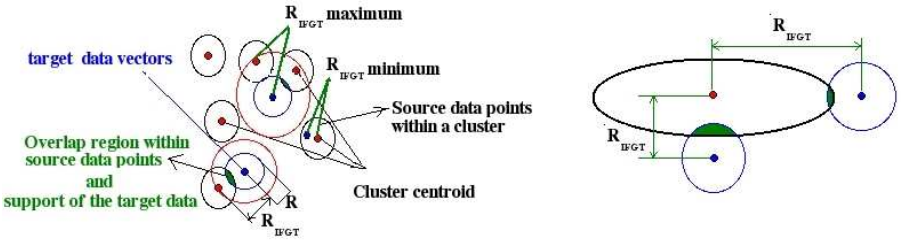
Fast Gauss transform (FGT) (Greengard and Strain [1991]), is a more general form of the Gaussian KDE, where the kernels have a Gaussian form. In FGT, the weights of the Gaussians (centered at each data point) can differ, whereas in case of (Gaussian) KDE the weights are equal. The speed is due to the grid structure assumed upon the total space. In the calculation of Gauss transform (or KDE), the effect of source data points within each multidimensional box of a grid structure, can be approximated with a finite series expansion around a modal vector (may be a centroid).

Yang et al. [2003] proposed an improved version of FGT (IFGT) algorithm for reducing the unnecessary computational complexity in multidimensional case through a prior clustering of the source data points. For the calculation of KDE, the distances between the target data point t_n and the cluster centroids are calculated. If the distance (between t_n and any cluster centroid) is more than a user-defined threshold, the source data points within the corresponding cluster are not considered for the calculation of KDE. Let us define the clusters as described in sub-section 3.1. Then the following equation describes the FKDE according to IFGT,

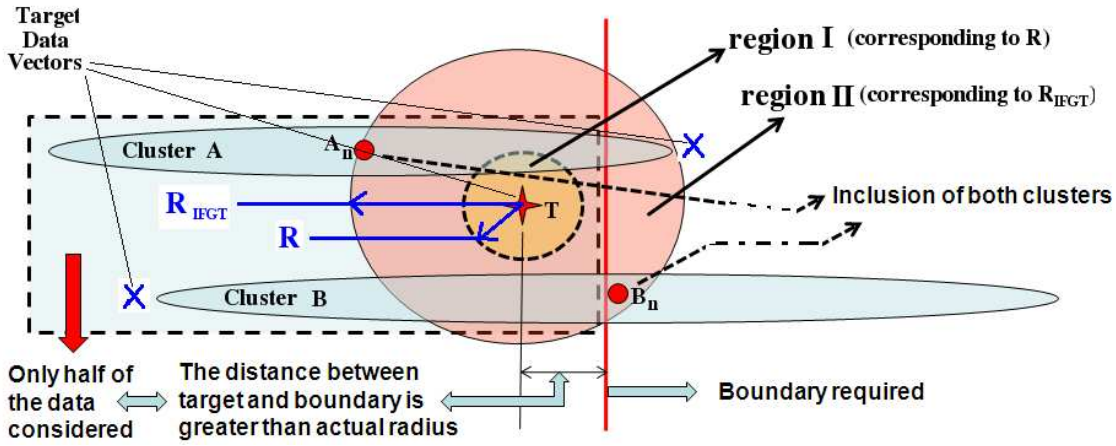
$$P(t_n) = \sum_{\|t_n - Clust_i\| \leq R_{IFGT}} K_H(t_n - Clust_i) f(t_n, Clust_i) \tag{5}$$

In equation 5, $f(t_n, Clust_i)$ is the approximation comes from truncated multivariate Taylor expansion (only possible for Gaussian kernel Yang et al. [2003]). If one wants to implement original KDE (for any kernel function) then the modified equation could be,

$$P(t_n) = \frac{1}{N}\sum_{\|t_n - Clust_i\| \leq R_{IFGT}} \sum_{s \in Clust_i} K_H(t_n - t_s) \tag{6}$$



(a) The R_{IFGT} varies with target data vector t_n and cluster shape



(b) Computational complexity may increase in IFGT

FIGURE 1: Limitations of IFGT Yang et al. [2003] and possible direction of solution

Here, R_{IFGT} is different from R (required for error bound described in equation 3). Again, R_{IFGT} varies depending upon the target data t_n and the cluster under consideration $Clust_i$. This can be best understood from figure 1a.

If one wants to fix the R_{IFGT} to a maximum value as $R_{IFGT} = R + max(R_{Clust})$, where, R_{Clust} is the maximum of maximum distances between any cluster to its source data points, the computational complexity of the algorithm may increase, as explained in figure 1b. Region I in figure 1b describes the area of radius R, fixed by the corresponding kernel. This region overlaps with the cluster A data set. Therefore, one needs to include the cluster A for the calculation of the density at target point T. But the cluster center A_n is far from the target data point. Therefore, a different radius threshold is needed, $R_{IFGT} > R$, and this defines the region II. Region II includes the sencond cluster centroid B_n, and therefore, the final calculation of the KDE will consider both cluster A and B. The inclusion of the redundant data points increases the computational complexity. The same problem will be encountered if one considers other target points marked as $\times$ in the figure 1b. Therefore, there are two main limitations (additionally with the truncation error) of IFGT, as follows,

1. optimal R_{IFGT} varies with both t_n (target data point) and cluster shape, figure 1a.
2. maximum possible R_{IFGT} can increase computational complexity of the FKDE algorithm, figure 1b.

3.3. Boundary Effect : A possible solution

Instead of cluster centroids, if one considers a boundary (d-dimensional hyperplane), partitioning the total space into two subspaces (as shown in figure 1b), the above mentioned problems will not arise for most of the target points. One can easily make decision based upon the distance between the target data point and the boundary. This distance is not required to change with respect to the target data point and therefore it will remain R. Therefore, for the fast kernel density estimation, collection of boundaries has the following advantages compared to a collection of centroids,

- The distance threshold (required for error bound equation 3) does not depend upon the cluster size or shape.
- The computational complexity could be lesser in boundary-based clustering, compared to centroid-based methodologies; because from figures 1a and 1b, it is clear that, $R_{IFGT} \gg R$.

4. FAST KERNEL DENSITY ESTIMATION (FKDE) THROUGH PDDP

We will first describe the basic clustering algorithm in subsection 4.1. Thereafter, we will use this clustering algorithm for fast kernel density estimation (FKDE), described in subsection 4.2. Finally, in subsection 4.3, the computational complexity is analysed.

4.1. Principal Directive Divisive Partitioning (PDDP)

In this section the basic clustering algorithm PDDP (Boley [1998]), is described. Here, the information extracted from the Principal Component Analysis (PCA) of a set of source data points, is used for defining the cluster boundary. The partitioning involves two basic steps as descrbed below;

- project each source data point within the present space onto the first principal direction (eigen vector corresponding to the largest eigen value).
- partition the present space into two sub-spaces with respect to the mean (or median) of the projected values.

One can build up a tree structure by recursively partitioning each sub-spaces. From this algorithm we construct source data index vectors for each leaf node, $\{J_i; i = 1, \ldots, M\}$, where $M = 2^q$ and q is the depth of the tree structure.

The above mentioned clustering algorithm, partitions the space into two sub-spaces along its first principal direction and with respect to its mean (or median). This implies that, a d-dimensional hyperplane is acting as a boundary to divide the space into two sub-spaces. This hyperplane is the matrix of $(d-1)$ number of eigen vectors (replacing the first eigen vector with all zeros). Therefore, at each node (except the leaf nodes) of the tree, one such boundary is created for partitioning the space constructed by the source data points of the present node.

4.2. FKDE

The formal pseudo-code for kernel density estimation is described in algorithm 1. The steps can be summarized as follows,

1. at current node, the target data point t_n is projected onto the first principal direction of the corresponding set of source data points,
2. the distance between projected t_n and the mean (or median) of the corresponding set is calculated,
3. if this distance is less than the pre-defined threshold R, consider both the children of the current set of source data points.
4. otherwise, consider left child if t_n belongs to left side of the boundary,
5. or else, consider the right child.

In short, first one finds the neighborhood leaf nodes corresponding to a target data point with a fixed distance threshold R. The source data points within these leaf nodes ($J = \bigcup_{i \in \aleph(t_n)} J_i$) are then used for final calculation of the kernel density of the target data point, as,

$$P(t_n) = \frac{1}{N} \sum_{s \in J} K_H(t_s - t_n) \tag{7}$$

Here, $\aleph(t_n)$ is the neighborhood area (subspace) around target data point t_n, defined as $\aleph(t_n) = \{t_s; \|t_s - t_n\| \leq R\}$.

Input:
$t_s \in \Re^d, s \in I_{in} = \{1 \ldots N\}$: The source data points
$t_n \in \Re^d$: The target data point
k: Depth of the tree
R: radius required for kernel computation
Output:
$P(t_n)$: Probability of t_n

begin
  Generate the required data structure through PDDP with t_s
  Initialize an index vector J as empty
  Define J_i as the source data indices of node i
  Initialize a temporary vector *temp_node* =root_node
  for $i = 1 \ldots |temp\_node|$ **do**
    r = depth of the current node
    if *temp_node*(i) *is not a leaf node* **then**
      $v_n = E_{r,i}^T t_n$
      if $abs(v_n - m_{r,i}) \leq R$ **then**
        remove i^{th} node from *temp_node*
        insert children nodes of the i^{th} node in *temp_node*
      end
      else
        remove the i^{th} node from *temp_node*
        if $v_n > m_{r,i}$ **then**
          insert right child node of i^{th} node within *temp_node*
        end
        else
          insert left child node of i^{th} node within *temp_node*
        end
      end
    end
    else
      $J = J \bigcup J_i$
      remove i^{th} node from *temp_node* vector
    end
  end
  $P(t_n) = \frac{1}{N} \sum_{s \in J} K_H(t_s - t_n)$
end

Algorithm 1: Fast kernel density estimation (FKDE) through PDDP

4.3. Computational Complexity

The computational complexity for KDE is $O(Nd)$, equation 1. The ideal computational complexity for FKDE is $O(|R_n|d)$, equation 2, where |.| denotes cardinality of the set. The problem is to know the source data points such that $s \in R_n$. As described in subsection 4.2, the search for leaf nodes J_i involve a path through the tree structure. At each node (steps described in subsection 4.2), one has to decide whether to search in the left child or in the right child or within both the children. This decision making process has a computational complexity of the order of d. Now, we have to consider the worst, best and average computational complexities of the FKDE algorithm.

In the worst case, one has to consider all the data points, i.e., all the leaf nodes. In this case the computational complexity becomes $O((2^{q+1} - 1)d + Nd) \cong O(Nd)$, since $N \gg 2^{q+1}$. This is very unlikely in practice. In the best case, only one leaf node has to be considered, and therefore, the minimum computational complexity becomes $O((q-1)d + |\{s \in J_i\}|d) \lll O(Nd)$,since $|\{s \in J_i\}| \cong \frac{N}{2^q} = \frac{N}{M}$. This can happen when t_n is well within a sub-space. In most cases, t_n will lie close to one or more boundaries, and in those cases, one has to consider more than one leaf nodes, say

this number is m ($\lll 2^q = M$). Since, the searching complexity is additive and $m(\lll 2^q)$ number of nodes will have approximately $\frac{mN}{M}$ ($M = 2^q$) number of source data points, on average the computational complexity would be approximately $O(\frac{mNd}{M}) \ll O(Nd)$.

5. NONPARAMETRIC MRF AND TEXTURE SYNTHESIS

MRF models have been used for different applications in different branches of science and engineering. In the present case, description of MRF is taken from the view point of lattice models. The lattice, Y, is a collection of random variables (r.v. henceforth) at the sites, $s = \{i, j\} \in S$, where, $i, j = 0, 1, \ldots, M-1$. The random variables are described as $Y_s \in \Lambda$, i.e., they belong to the same state space. The MRF assumption implies,

$$p\left(Y_s = y_s | Y_{(s)}\right) = p\left(Y_s = y_s | Y_s = \{y_r; r \in \aleph_s\}\right) \tag{8}$$

, which describes the fact that given a neighbor set, $\aleph_s$, the r.v. at s is independent of all other sites, $(s) = S - s$ and this conditional probability is termed as local conditional pdf (LCPDF). The neighborhood system is defined with the help of following two axioms,

- $s \notin \aleph_s$, and,
- $s \in \aleph_r \Leftrightarrow r \in \aleph_s$.

Let us now consider the nonparametric MRF model as described in Paget and Longstaff [1998] in the context of texture synthesis. Assume, S_{in} and S_{out} signify the input and output texture lattices respectively, and for simplicity we also assume that $X_s = \{y_r; r \in \aleph_s\}$ denote the neighborhood set of the pixel r.v. Y_s. For each pixel in the output lattice, $s \in S_{out}$, we estimate the LCPDF $P(Y_s|X_s)$, for $Y_s = 1, 2, \ldots, L$ (grey values) from the data $\left\{X_p, Y_p\right\}$ where, $p \in S_{in}$. This is generated from the input texture through Parzen window estimator using the product kernels as,

$$P(y_s|X_s) = \frac{\sum_{p \in S_{in}} \kappa_{h_y}(y_s - y_p) \prod_{j=1}^{d} \kappa_{h_j}(X_{s_j} - X_{p_j})}{\sum_{p \in S_{in}} \prod_{j=1}^{d} \kappa_{h_j}(X_{s_j} - X_{p_j})} \tag{9}$$

Here, $\kappa_h(.)$ is the Gaussian kernel function. In the following a brief description of the sampling process from the nonparametric conditional density, as described in Paget and Longstaff [1998], is given. Choose a new y_s, by sampling the estimated LCPDF, through either Gibbs sampler or ICM algorithm.

5.1. Local Simulated Annealing: Texture Synthesis Algorithm

For synthesizing textures from a random initialization we need local simulated annealing Paget and Longstaff [1998]. In this method the LCPDF (eq. 9) evolves with time. It starts from a marginal pdf and approaches to the actual nonparametric estimation of conditional density. In this subsection, we briefly discuss the algorithm of texture synthesis through the local simulated annealing.

Let the two random fields defined on the two lattices S_{in} and S_{out} be denoted as, $Y_{in} = \{Y_s; s \in S_{in}\}$ for the input texture, and $Y_{out} = \{Y_k; k \in S_{out}\}$ for output texture respectively. For local simulated annealing we require another random field defined on the output lattice structure. We define this random field as, $T_{out} = \{c_k; k \in S_{out}\}$. Paget and Longstaff [1998] has described T_{out} as the temperature field. In this paper, we use the term *confidence field*, to describe the behaviour and the requirement of the field in the process of texture synthesis. The random variables c_k reflect the confidence of synthesis of the corresponding random field Y_k at site $k \in S_{out}$. The range of the confidence random variables c_k varies from 0 to 1, where 1 represents complete confidence, and 0 none at all. The rule for updation of the confidence at site $k \in S_{out}$ is defined as,

$$c_k = min\{1, (\eta + \sum_{r \in \aleph_k} c_r)/d\} \tag{10}$$

, where $\eta \in [0, 1]$ is a random variable generated from uniform distribution. This equation describes the fact that the confidence of any pixel site will depend upon the confidence values of its

neighborhood sites. In the estimation of LCPDF, the argument within the kernel for neighborhood vector will change depending upon the confidence values at the corresponding neighborhood sites of the output lattice. Let us define a neighborhood vector of confidence random variables as, $W_k = \{c_r; r \in \aleph_k\}, \forall k \in S_{out}$. Similarly we define a confidence matrix at site k as, Φ_k, whose diagonal elements contain the random variables $c_r; r \in \aleph_k$, and the off diagonal elements are all zero. The estimation of LCPDF according to the confidence field is defined as,

$$p(Y_k|X_k, W_k) = \frac{\sum_{s \in S_{in}} K_h(Y_k - Y_s) K_h(X_k - X_s|W_k)}{\sum_{s \in S_{in}} K_h(X_k - X_s|W_k)} \tag{11}$$

where, X_k, and X_s correspond to the neighborhood vectors of output and input textures, respectively. The kernel for neighborhood vector can be written as,

$$K_h(X_k - X_s|W_k) = \frac{1}{n\sqrt{2\pi h^2}} \exp[\frac{1}{2h^2} \sum_{i=1}^{d} W_{k,i}(X_{k,i} - X_{s,i})^2] \tag{12}$$

or alternatively as,

$$K_h(X_k - X_s|W_k) = \frac{1}{n\sqrt{2\pi h^2}} \exp[\frac{1}{2h^2}(X_k - X_s)^t \Phi_k (X_k - X_s)] \tag{13}$$

where, $n = |S_{in}|$, $|.|$ is the cardinality, $(.)^t$ represents transpose, $W_{k,i}$, $X_{k,i}$ and $X_{s,i}$ are the i^{th} random variable of the random vectors W_k, X_k, and X_s, respectively.

From the equations 10, 12 and 13, it is clear that as we progress in time the LCPDF (equation 11) will approach the actual form starting from the marginal density $p(Y_s)$, as the values of confidence random variables approach one starting from zero. Since, we are approaching the true LCPDF through local simulated annealing, we can sample LCPDF according to independent conditional mode (ICM) algorithm as described by Besag [1986]. In ICM, we look for $Y_k = v, v \in \{0, 1, \ldots, L\}$ (where L is the number of gray levels), which has got maximum probability according to the given neighborhood X_k, i.e., maximum LCPDF. The advantage of ICM is fast convergence, whereas other sampling algorithms (such as, Gibbs sampler, Metropolis sampler) has a very slow convergence rate with almost sure convergence to the global optimum Paget and Longstaff [1998]. In this paper, we have worked with ICM algorithm.

5.2. Computational complexity of earlier texture synthesis algorithm

The computational complexity for the calculation of LCPDF is of the order of $O(N(d+1))$ (equation 11), where $N = |S_{in}|$ is the number of pixels within input texture sample and $(d + 1)$ is the dimensionality of the $\{X_p, Y_p\}$. Sinha and Gupta [2007] have tried to reduce this computational complexity by reducing the dimension of the neighborhood set X_p. In this work, the computational complexity is reduced by reducing N, which is equal to 128×128 in the current work. In the next section, the proposed algorithm for fast KDE (FKDE) is described, along with the reduction in computational complexity.

5.3. Texture synthesis with FKDE

The computational complexity is mainly due to the evaluation of equation 11 at each output pixel $k \in S_{out}$. Incorporation of proposed FKDE algorithm within the LCPDF estimation (equation 11), has two main problems,

1. how to include the effect of W_k (the temperature field) within the PDDP-based tree structure for the implementation of FKDE, and
2. there are two joint densities corresponding to $\{Y_k, X_k\}$ and X_k; therefore, it requires two FKDE structure, which is not computationally efficient.

W_k is required in equation 11 for modeling the local simulated annealing. When all the values of W_k are zero ($W_{k,i} = 0; i = 1 \ldots d$), then from equation 12 one can observe that $K_H(X_k - X_s|W_k)$

becomes constant irrespective of the value X_s and X_k, i.e., $P(X_k|W_k)$ becomes uniform. At other extreme, when all values of W_k are one ($W_{k,i} = 1; i = 1 \ldots d$), then $K_H(X_k - X_s|W_k) = K_H(X_k - X_s)$, i.e., $P(X_k|W_k) = P(X_k)$. To achieve these limiting effects within FKDE algorithm one has to change the radius threshold R, since R is responsible for the searching the leaf nodes required to calculate the final KDE. This new value of R (say R_{new}) can be explained as follows,

Local simulated annealing:

$$\begin{aligned}
\text{Starting} : \{W_{k,i} = 0; i = 1 \ldots d; k \in S_{out}\} &\Rightarrow P(X_k|W_k) = \text{ constant} \\
&\Rightarrow P(X_k) \text{ is uniform} \\
&\Rightarrow \text{Each } X_s \text{ has equal effect upon } X_k \\
&\Rightarrow \text{Every } X_s \text{ should be considered in the KDE} \\
&\Rightarrow R_{new} \text{ is very large} \\
\text{Ending} : \{W_{k,i} = 1; i = 1 \ldots d; k \in S_{out}\} &\Rightarrow P(X_k|W_k) = P(X_k) \\
&\Rightarrow R_{new} = R
\end{aligned}$$

Therefore, R_{new} should vary from a large value to R, as annealing process proceeds. We can achieve this effect by considering c_k (confidence value at site k) as shown in equation 14. This new radius threshold can be introduced within FKDE, at line number 10 of algorithm 1, as shown in equation in equations 15 and 16.

$$R_{new} = \frac{R}{c_k} \tag{14}$$

$$abs(v_n - m_{r,i}) \leq R_{new} \tag{15}$$

$$\Rightarrow abs(v_n - m_{r,i}) \leq \frac{R}{c_k}$$

$$\Rightarrow abs(v_n - m_{r,i})c_k \leq R \tag{16}$$

Now we will study the second problem mentioned above. If the KDE of the neighborhood vector (the denominator term of equation 11) $\sum_{s \in S_{in}} K_h(X_k - X_s|W_k)$ is not equal to zero, then one would be interested in the numerator term, otherwise LCPDF is equal to zero. Therefore, only when $K_h(X_k - X_s|W_k) \neq 0$, the calculation for $K_h(Y_k - Y_s)$ is required. In this work, the tree structure (sub-section 4.1) is built for the neighborhood vectors $\{X_s; s \in S_{in}\}$, i.e., collected from the input texture. The FKDE is applied to each element from the set $\{X_k; k \in S_{out}\}$, i.e., neighborhood vectors from the output texture. Therefore, the LCPDF described in equation 11 becomes,

$$P(Y_k|X_k, W_k) \cong \frac{\sum_{s \in S_{in,k}} K_h(Y_k - Y_s) K_h(X_k - X_s|W_k)}{\sum_{s \in S_{in,k}} K_h(X_k - X_s|W_k)} \tag{17}$$

where, $S_{in,k} \subset S_{in}$ describes the collection of pixels from input texture where $K_h(X_k - X_s|W_k) \neq 0$.

Therefore, with equations 17 and 16, one can evaluate the LCPDF (equation 11) with only one FKDE structure and with same computational complexity ($O(\frac{mN(d+1)}{M})$) of FKDE. This computational complexity is lesser than the earlier texture synthesis algorithm ($O(N(d+1))$), since $\frac{m}{M} \ll 1$. In the following section, we will test the proposed texture synthesis algorithm with a number of natural textures.

6. RESULTS

Natural textures range from near-regular to stochastic. We have tested the computationally efficient texture synthesis algorithm (sub-section 5.3) for these two extreme classes. The texture samples are taken from a standard database Brodatz [1966]. Three textures corresponding to near-regular textures are $D103$, $D20$, and $D22$. In case of stochastic textures we have taken two such cases $D12$ and $D15$. The texture $D15$ is not only stochastic in nature, it has also got a structural component. The *order* (required to define the neighborhood system of NNMRF) is kept fixed at 20 for all cases. The texture synthesis has been done at single resolution, since if the computationally efficient texture synthesis algorithm is working in single resolution it will also work in multi-resolution.

The results for near-regular and stochastic texture classes are shown respectively in figures 2 and 3.

From figure 3 it can be observed that the computationally efficient texture synthesis algorithm (with FKDE) produces better result than the earlier algorithm in cases of near-regular textures. Whereas, in case of stochastic texture class (figure 2) the results match closely. The results establish the effectiveness of incoporation of the proposed FKDE algorithm within texture synthesis algorithm.

7. CONCLUSION

The main contribution of this work is to analyse the nonparametric FKDE from the viewpoint of a strict upper error bound (equation 3), and to develop a new algorithm based upon this error bound. It has been shown that in FKDE, the boundaries (hyperplanes) instead of the cluster centroids for partitioning the space produce lesser computational complexity (section 3).

The computational complexity of earlier texture synthesis algorithm (Paget and Longstaff [1998]), is huge due to large dimensionality of the neighborhood vector (Sinha and Gupta [2007]) and the nonparametric kernel conditional density estimation. In this paper, the computational complexity of this earlier algorithm has been reduced through the incorporation of proposed FKDE. The proposed algorithm in texture synthesis has been tested with a number of textures taken from a standard database Brodatz [1966].

We are thankful to the invaluable comments of the reviewers. It is true that, the analysis is much similar to the kd-tree structure and in higher dimension kd-tree structure is not optimum. But, with the normal cluster techniques (as used in IFGT or ball-tree), the texture synthesis results are nowhere close to the original, as evidenced by us. Again, eigenvalue decomposition is a computationally expensive module, but it is the reason for faster texture synthesis compared to the kd-tree structure. A total analysis of the proposed algorithm in terms of kernel density estimation with error bound is in preparation.

References

Besag, J. E., 1986. On the statistical analysis of dirty pictures. Journal of Royal Statistical Society B-48, 259–302.

Boley, D. L., 1998. Principal direction divisive partitioning. Data Mining and Knowledge Discovery 2 (4), 325–344.

Brodatz, P., 1966. Textures:a photographic album for artists and designers. Toronto, Ont., Canada:Dover.

Dempster, A., Laird, N., Rubin, D., 1977. Maximum likelihood estimation from incomplete data via the em algorithm. Journal of Royal Statistic Society 30 (B), 1–38.

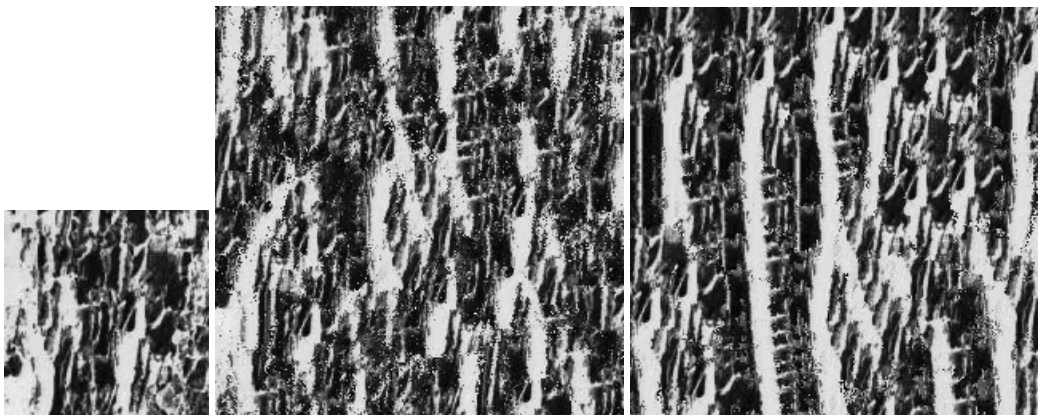
Elgammel, A., Duraiswami, R., Davis, L. S., 2003. Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (11), 1499–1504.

Greengard, L., Strain, J., 1991. The fast gauss transform. SIAM Journal on Scientific Statistical Computing 12 (1), 79–94.

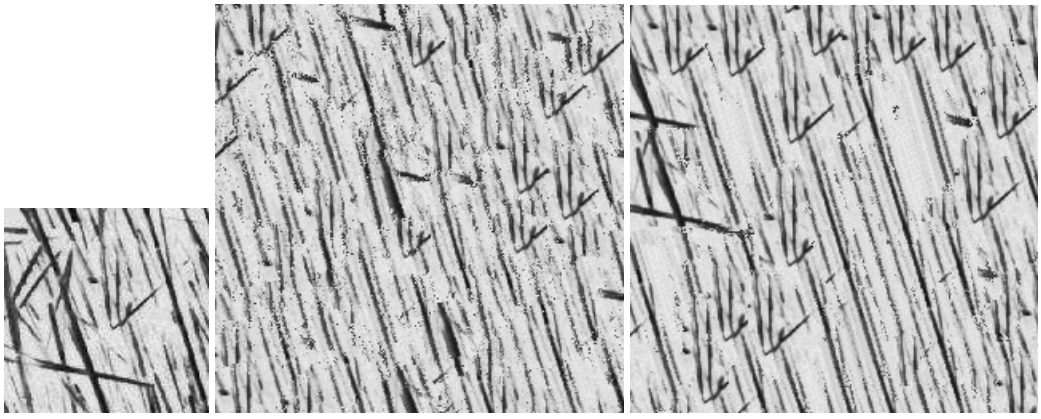
Huang, H., Bi, L.-P., Song, H.-T., Lu, Y.-C., August 2005. A variational em algorithm for large databases. In: Proc. of the Fourth International Conference on Machine Learning and Cybernatics. IEEE.

Liu, Z., Shen, L., Han, Z., Zhang, Z., 2007. A novel video object tracking approach based on kernel density estimation and markov random field. In: IEEE International Conference on Image Processing, ICIP.

Paget, R., Longstaff, I. D., Jun. 1998. Texture synthesis via a noncausal nonparametric multiscale markov random field. IEEE Transactions on Image Processing 7 (6), 925–931.



(a) D12



(b) D15

FIGURE 2: **Structural and Stochastic texture synthesis result**; Left column: original texture sample, middle column: synthesis result with earlier algorithm Paget and Longstaff [1998], last column: synthesis result with proposed algorithm

Pernkopf, F., Bouchaffra, D., 2005. Genetic-based em algorithm for learning gaussian mixture models. IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (8), 1344–1348.

Scott, D. W., 1992. Multivariate density estimation - theory, practice and visualization. Wiley interscience.

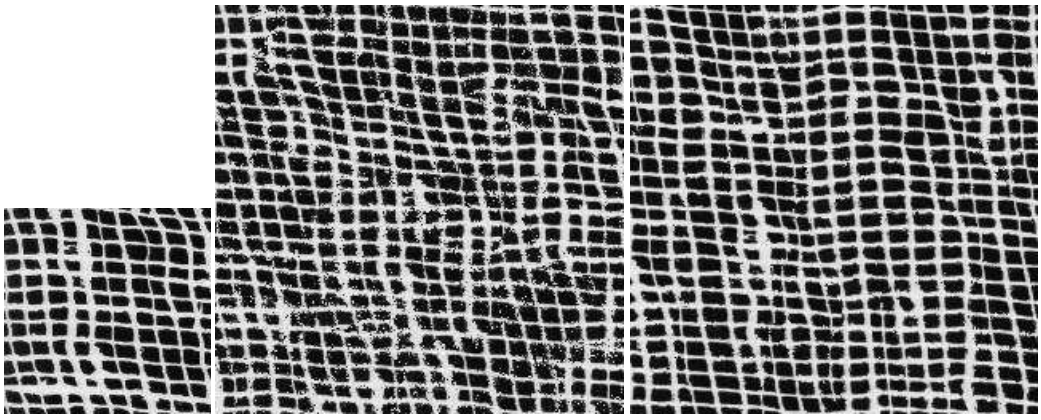
Sinha, A., Gupta, S., 2007. Approximation of conditional density of markov random field and its application to texture synthesis. In: IEEE International Conference on Image Processing, ICIP.

Thiesson, B., Meek, C., Heckerman, D., 1999, Revised 2001. Accelerating em for large databases. Tech. rep., Microsoft Research.

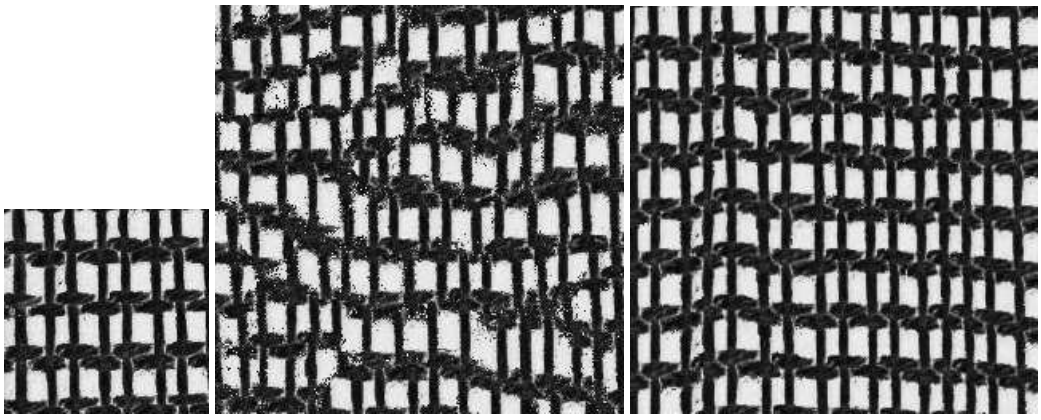
Wu, C. J., 1983. On the convergence properties of the em algorithm. The Annals of Statistics 11 (1), 95–103.

Yang, C., Duraiswami, R., Gumerov, N. A., Davis, L., 2003. Improved fast gauss transform and efficient kernel density estimation. In: Proceedings. Ninth IEEE International Conference on Computer Vision.

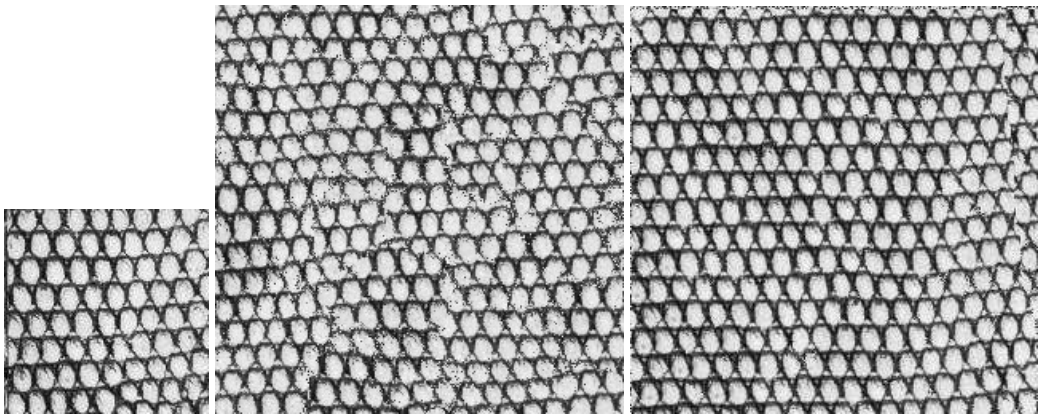
Zhang, K., Tang, M., Kwok, J. T., 2005. Applying neighborhood consistency for fast clustering and kernel density estimation. In: Proceedings of the 2005 Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Vol. 2.



(a) D103



(b) D20



(c) D22

FIGURE 3: **Near-regular texture synthesis result**; Left column: original texture sample, middle column: synthesis result with earlier algorithm Paget and Longstaff [1998], last column: synthesis result with proposed algorithm