# Improved fast Gauss transform with variable source scales

VIKAS CHANDRAKANT RAYKAR and RAMANI DURAISWAMI
Perceptual Interfaces and Reality Laboratory
Department of Computer Science and Institute for Advanced Computer Studies
University of Maryland, CollegePark, MD 20783
{vikas,ramani}@cs.umd.edu

---

Evaluating sums of multivariate Gaussian kernels is a key computational task in many problems in computational statistics and machine learning. The computational cost of the direct evaluation of such sums scales as the product of the number of kernel functions and the evaluation points. The original fast Gauss transform due to Greengard and Strain (1991) can accelerate such sums in low dimensions. Yang et al. (2003) presented an extension of the fast Gauss transform (the improved fast Gauss transform or IFGT) that was suitable for higher dimensional problems, and applied it to some machine learning problems in Yang et al. (2004). However, this algorithm was restricted to the case of constant bandwidth Gaussians. In many applications performance is improved if variable bandwidth functions are used. We present an extension to the IFGT algorithm that allows variable bandwidth Gaussian kernels. Algorithm details, error bounds and numerical experiments are presented. For example for $N = M = 1,024,000$ while the direct evaluation takes around 2.6 days the fast evaluation requires only 4.65 minutes with an error of around $10^{-5}$.

[**CS-TR-4727/UMIACS-TR-2005-34**]: April 8, 2006

---

Contents

## 1. DISCRETE GAUSS TRANSFORM WITH VARIABLE SCALES

For each *target point* $\{y_j \in \mathcal{R}^d\}_{j=1,\ldots,M}$ the *discrete Gauss transform* with *variable source scales* is defined as,

$$G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2/h_i^2}, \tag{1}$$

where $\{q_i \in \mathcal{R}^+\}_{i=1,\ldots,N}$ are the *source weights*, $\{x_i \in \mathcal{R}^d\}_{i=1,\ldots,N}$ are the *source points*, i.e., the center of the Gaussians, and $\{h_i \in \mathcal{R}^+\}_{i=1,\ldots,N}$ are the *source scales* or *bandwidths*. In other words $G(y_j)$ is the total contribution at $y_j$ of $N$ Gaussians centered at $x_i$ with bandwidth $h_i$. Each Gaussian is weighted by the term $q_i$. The computational complexity of evaluating the discrete Gauss transform at $M$ target points is $\mathcal{O}(MN)$. This makes the computation for large scale problems prohibitively expensive. In many machine learning tasks data-sets containing more than $10^4$ points are already common and larger problems are of interest.

The *Fast Gauss Transform* (FGT) is an $\epsilon - exact$ approximation algorithm that reduces the computational complexity to $\mathcal{O}(M+N)$, at the expense of reduced precision $\epsilon$, which however can be arbitrary. The constant depends on the desired precision, dimensionality of the problem, and the bandwidths. Given any $\epsilon > 0$, it compute an approximation $\hat{G}(y_j)$ to $G(y_j)$ such that the maximum absolute error relative to the total weight $Q = \sum_{i=1}^{N} q_i$ is upper bounded by $\epsilon$, i.e.,

$$\max_{y_j} \left[ \frac{|\hat{G}(y_j) - G(y_j)|}{Q} \right] \leq \epsilon. \tag{2}$$

The fast Gauss transform (FGT) was first proposed in the seminal paper by Greengard and Strain [Greengard and Strain 1991]. The variable bandwidth case was handled in [Strain 1991]. However in their formulation the constant term increases exponentially (as $p^d$) with increasing dimensionality $d$. Further, the data structure used to cluster points was inefficient in high dimensions. Thus the algorithm is impractical for machine learning and statistical applications where the data encountered are usually high dimensional. Yang et al. [Raykar et al. 2005; Yang et al. 2003; Yang et al. 2005] proposed the improved fast Gauss Transform (IFGT) where the constant factor is reduced to asymptotically polynomial order. The reduction was achieved based on a multivariate Taylor expansion scheme combined with the efficient space subdivision using the $k$-center algorithm. The main contribution of the current paper is to extend the IFGT to handle variable bandwidths. While we follow the same space sub-division scheme as proposed in [Yang et al. 2003], the variable bandwidth case involves the factorization of two exponentials and and the error bound evaluation is more involved than the constant bandwidth case. We discuss the case of variable source bandwidths, though the approach presented here can also be extended to variable target bandwidths. Another novel contribution is the use of pointwise error bounds for each source point. This naturally leads to tighter error bounds and a good strategy for choosing the parameters. For each source point we choose different truncation numbers depending on its bandwidth and distance to the cluster center.

## 2. PRELIMINARIES

### 2.1 Multidimensional Taylor's Series

The factorization of the multivariate Gaussian and the evaluation of the error bounds are based on the multidimensional Taylor's series and Lagrange's evaluation of the remainder which we state it here without the proof.

THEOREM 2.1. *For any point $x_* \in \mathcal{R}^d$, let $I \subset \mathcal{R}^d$ be an open set containing the point $x_*$. Let $f : I \to \mathcal{R}$ be a real valued function which is $n$ times partially differentiable on $I$. Then for any $x = (x_1, x_2, \ldots, x_d) \in I$, there is a $\theta \in \mathcal{R}$ with $0 < \theta < 1$ such that*

$$f(x) = \sum_{k=0}^{n-1} \frac{1}{k!} \left[(x - x_*) \cdot \nabla\right]^k f(x_*) + \frac{1}{n!} \left[(x - x_*) \cdot \nabla\right]^n f(x_* + \theta(x - x_*)), \quad (3)$$

*where the operator* $\nabla = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_d}\right)$.

Based on the above theorem we state two corollaries which give the multivariate Taylor's series expansion of two exponential functions, which we will use in the IFGT algorithm. The first corollary gives the expansion for $e^{-\|x-x_*\|^2/h^2}$ and the second for $e^{2(x-x_*).(y-x_*)/h^2}$.

COROLLARY 2.1. *Let $B_r(x_*)$ be a open ball of radius $r$ with center $x_* \in \mathcal{R}^d$, i.e., $B_r(x_*) = \{x : \|x - x_*\| < r\}$. Let $h \in \mathcal{R}$ be a positive constant. For any $x \in B_r(x_*)$ and any non-negative integer $p_1$ the function $f(x) = e^{-\|x-x_*\|^2/h^2}$ can be written as*

$$f(x) = e^{-\|x-x_*\|^2/h^2} = \sum_{k=0}^{p_1-1} \frac{(-1)^k}{k!} \|(x - x_*)/h\|^{2k} + R_{p_1}(x), \quad (4)$$

*and the absolute value of the residual $R_{p_1}(x)$ can be bounded as follows.*

$$|R_{p_1}(x)| < \frac{2^{p_1} 1.09}{(2p_1)^{1/12}\sqrt{(2p_1)!}} \|(x - x_*)/h\|^{2p_1}. \quad (5)$$

PROOF. Consider the Taylor's series expansion of function $g(x) = e^{-\|x\|^2/h^2}$ around the origin $x_* = 0$ . From Theorem 2.1 for any $x \in B_r(0)$, there is a $\theta \in \mathcal{R}$ with $0 < \theta < 1$ such that

$$g(x) = \sum_{k=0}^{n-1} \frac{1}{k!} \left[x \cdot \nabla\right]^k g(0) + \frac{1}{n!} \left[x \cdot \nabla\right]^n g(\theta x). \quad (6)$$

It can be shown that,

$$\left[x \cdot \nabla\right]^k g(0) = \begin{cases} 0 & \text{if } k \text{ is odd} \\ (-1)^{k/2} \frac{k!}{(k/2)!} \|x/h\|^k & \text{if } k \text{ is even} \end{cases} \quad (7)$$

In fact it can be see that $[x \cdot \nabla]^k g(0) = H_k(0)\|x/h\|^k$, where $H_k(0)$ are the Hermite numbers, i.e., the Hermite polynomials [1] $H_k(y)$ evaluated at $y = 0$. Also,

$$[x \cdot \nabla]^n g(\theta x) = (-1)^n \|x/h\|^n H_n(\theta\|x/h\|) e^{-\theta^2\|x/h\|^2}. \quad (8)$$

Substituting Eqs. 7 and 8 in Eq. 6, we have

$$g(x) = \sum_{k=0}^{\lceil n/2\rceil - 1} \frac{(-1)^k}{k!}\|x/h\|^{2k} + \left[\frac{(-1)^n}{n!}\|x/h\|^n e^{-\theta^2\|x/h\|^2} H_n(\theta\|x/h\|)\right]. \quad (9)$$

Centering the function $g(x)$ around $x_*$ we have,

$$f(x) = e^{-\|x-x_*\|^2/h^2} = \sum_{k=0}^{p_1-1} \frac{(-1)^k}{k!}\|(x-x_*)/h\|^{2k} + R_{p_1}(x), \quad (10)$$

where,

$$R_{p_1}(x) = \frac{\|(x-x_*)/h\|^{2p_1}}{(2p_1)!} e^{-\theta^2\|(x-x_*)/h\|^2} H_{2p_1}(\theta\|(x-x_*)/h\|). \quad (11)$$

We bound the Hermite polynomial $H_n(y)$ as follows [Hille 1926],

$$|H_n(y)| < K 2^{n/2}\sqrt{n!} n^{-1/12} e^{y^2/2}, \quad (12)$$

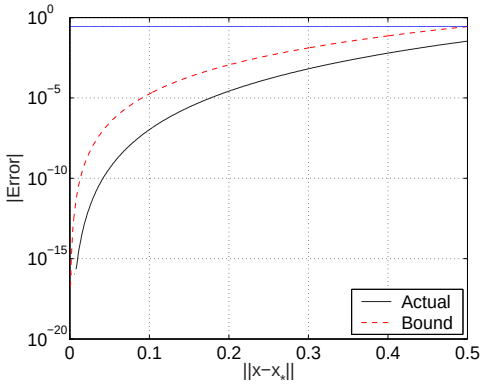
where the numerical constant $K < 1.09$. This is slightly tighter than the bound used in [Greengard and Strain 1991]. The residual $R_{p_1}(x)$ is bounded as follows,

$$\begin{aligned} |R_{p_1}(x)| &< \frac{\|(x-x_*)/h\|^{2p_1}}{(2p_1)!} e^{-\theta^2\|(x-x_*)/h\|^2} K 2^{p_1}\sqrt{(2p_1)!}(2p_1)^{-1/12} e^{\theta^2\|(x-x_*)/h\|^2/2}, \\ &< \frac{K 2^{p_1}(2p_1)^{-1/12}\|(x-x_*)/h\|^{2p_1}}{\sqrt{(2p_1)!}} e^{-\theta^2\|(x-x_*)/h\|^2/2}, \\ &< \frac{2^{p_1} 1.09}{(2p_1)^{1/12}\sqrt{(2p_1)!}}\|(x-x_*)/h\|^{2p_1}. \text{ [since } 0 < \theta < 1 \text{ and } K < 1.09] \end{aligned} \quad (13)$$

□

*Remark :* Fig. 1 shows the actual residual (solid line) and the bound (dashed line) given by Eq. 5 as a function of $\|x - x_*\|$ for $p_1 = 4$ and $h = 0.5$. The residual is minimum at $x = x_*$ and increases as $x$ moves away from $x_*$. It can be seen that the bound is pretty tight. It should be noted that we use the error bound which is a function of $\|x-x_*\|$. A consequence of this is that a lower truncation number $p_1$ can achieve a given error, depending on the magnitude of $\|x - x_*\|$. For points close to $x_*$ we need a very small truncation number compared to points far from the center. The original IFGT and the FGT algorithms used the same truncation number for all the points in the open ball. The truncation number was chosen based on the points at the boundary (the dotted line in Fig. 1). However our current approach adaptively chooses $p_1$ based on $\|x - x_*\|$. Also note that the error bound which we have in Eq. 5 is independent of the dimensionality $d$.

[1] The Hermite polynomials are set of orthogonal polynomials [Abramowitz and Stegun 1972]. The first few Hermite polynomials are $H_0(u) = 1$, $H_1(u) = u$, and$H_2(u) = u^2 - 1$.




(a)

Fig. 1. (a) The actual residual (solid line) and the bound (dashed line) given by Eq. 5 as a function of $\|x - x_*\|$. The residual is minimum at $x = x_*$ and increases as $x$ moves away from $x_*$. The bandwidth was $h = 0.5$ and the truncation number $p_1 = 4$.

COROLLARY 2.2. *Let $B_{r_x}(x_*)$ be a open ball of radius $r_x$ with center $x_* \in \mathcal{R}^d$, i.e., $B_{r_x}(x_*) = \{x : \|x - x_*\| < r_x\}$. Let $h \in \mathcal{R}^+$ be a positive constant and $y \in \mathcal{R}^d$ be a fixed point such that $\|y - x_*\| < r_y$. For any $x \in B_{r_x}(x_*)$ and any non-negative integer $p_2$ the function $f(x) = e^{2(x-x_*).(y-x_*)/h^2}$ can be written as*

$$f(x) = e^{2(x-x_*).(y-x_*)/h^2} = \sum_{k=0}^{p_2-1} \frac{2^k}{k!} \left[ \left( \frac{x - x_*}{h} \right) \cdot \left( \frac{y - x_*}{h} \right) \right]^k + R_{p_2}(x) \tag{14}$$

*and the residual $R_{p_2}(x)$ is bounded as follows.*

$$|R_{p_2}(x)| \leq \frac{2^{p_2}}{p_2!} \left( \frac{\|x - x_*\|}{h} \right)^{p_2} \left( \frac{\|y - x_*\|}{h} \right)^{p_2} e^{2\|x-x_*\|\|y-x_*\|/h^2}. \tag{15}$$

$$\tag{16}$$

PROOF. Let us define a new function $g(x) = e^{2[x.(y-x_*)]/h^2}$. Using the result

$$[(x - x_*) \cdot \nabla]^k g(x_*) = 2^k e^{2[x_*.(y-x_*)]/h^2} \left[ \left( \frac{x - x_*}{h} \right) \cdot \left( \frac{y - x_*}{h} \right) \right]^k \tag{17}$$

and Theorem 2.1, we have for any $x \in B_{r_x}(x_*)$ there is a $\theta \in \mathcal{R}$ with $0 < \theta < 1$ such that

$$\begin{aligned} g(x) = e^{2[x_*.(y-x_*)]/h^2} \Bigg\{ & \sum_{k=0}^{p_2-1} \frac{2^k}{k!} \left[ \left( \frac{x - x_*}{h} \right) \cdot \left( \frac{y - x_*}{h} \right) \right]^k \\ & + \frac{2^{p_2}}{p_2!} \left[ \left( \frac{x - x_*}{h} \right) \cdot \left( \frac{y - x_*}{h} \right) \right]^{p_2} e^{2\theta[(x-x_*).(y-x_*)]/h^2} \Bigg\}. \end{aligned} \tag{18}$$

Hence

$$\begin{aligned} f(x) &= e^{2(x-x_*)\cdot(y-x_*)/h^2} \\ &= \sum_{k=0}^{p_2-1} \frac{2^k}{k!} \left[ \left( \frac{x-x_*}{h} \right) \cdot \left( \frac{y-x_*}{h} \right) \right]^k + R_{p_2}(x), \end{aligned} \tag{19}$$

where,

$$R_{p_2}(x) = \frac{2^{p_2}}{p_2!} \left[ \left( \frac{x-x_*}{h} \right) \cdot \left( \frac{y-x_*}{h} \right) \right]^{p_2} e^{2\theta[(x-x_*)\cdot(y-x_*)]/h^2}. \tag{20}$$

Using the Cauchy-Schwartz inequality $x \cdot y \leq \|x\|\|y\|$ the remainder is bounded as follows.

$$\begin{aligned} R_{p_2}(x) &\leq \frac{2^{p_2}}{p_2!} \left( \frac{\|x-x_*\|}{h} \right)^{p_2} \left( \frac{\|y-x_*\|}{h} \right)^{p_2} e^{2\theta\|x-x_*\|\|y-x_*\|/h^2}, \\ &\leq \frac{2^{p_2}}{p_2!} \left( \frac{\|x-x_*\|}{h} \right)^{p_2} \left( \frac{\|y-x_*\|}{h} \right)^{p_2} e^{2\|x-x_*\|\|y-x_*\|/h^2} [\text{Since} \quad 0 < \theta < 1]. \end{aligned} \tag{21}$$

□

## 2.2 Multi-index Notation

A multi-index $\alpha = (\alpha_1, \alpha_2 \ldots, \alpha_d)$ is a $d$-tuple of nonnegative integers. The length of the multi-index $\alpha$ is defined as $|\alpha| = \alpha_1 + \alpha_2 + \ldots + \alpha_d$. The factorial of $\alpha$ is defined as $\alpha! = \alpha_1! \alpha_2! \ldots \alpha_d!$. For any multi-index $\alpha \in \mathbf{N}^d$ and $x = (x_1, x_2, \ldots, x_d) \in \mathbf{R}^d$ the $d$-variate monomial $x^\alpha$ is defined as $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_d^{\alpha_d}$. $x^\alpha$ is of degree $n$ if $|\alpha| = n$. The total number of $d$-variate monomials of degree $n$ is $\binom{n+d-1}{d-1}$. The total number of $d$-variate monomials of degree less than or equal to $n$ is $r_{nd} = \sum_{k=0}^{n} \binom{k+d-1}{d-1} = \binom{n+d}{d}$. Let $x, y \in \mathbf{R}^d$ and $v = x \cdot y = x_1 y_1 + \ldots + x_d y_d$. Then using the multi-index notation $v^n$ can be written as,

$$(x \cdot y)^n = \sum_{|\alpha|=n} \frac{n!}{\alpha!} x^\alpha y^\alpha. \tag{22}$$

## 2.3 Space sub-division

In the IFGT we will appropriately cluster source points and evaluate their contributions using an expression that involves the Taylor's series. Accordingly we need a strategy to choose a set of centers about which to expand the Taylor's series, i.e., we need to subdivide the space. We use an data adaptive space partitioning scheme based on $k$-center clustering as proposed in [Yang et al. 2003; Yang et al. 2003; Yang et al. 2005]. The $k$-center problem is defined as follows:

*Given a set of $N$ points in $d$ dimensions and a predefined number of the clusters $k$, find a partition of the points into clusters $S_1, \ldots, S_k$, and also the cluster centers $c_1, \ldots, c_k$, so as to minimize the cost function – the maximum radius of clusters,* $\max_i \max_{x \in S_i} \|x - c_i\|$.

The $k$-center problem is known to be NP-hard [Bern and Eppstein 1997]. Gonzalez [Gonzalez 1985] proposed a very simple greedy algorithm, called *farthest-point clustering*, and proved that it gives an approximation factor of 2. Initially pick

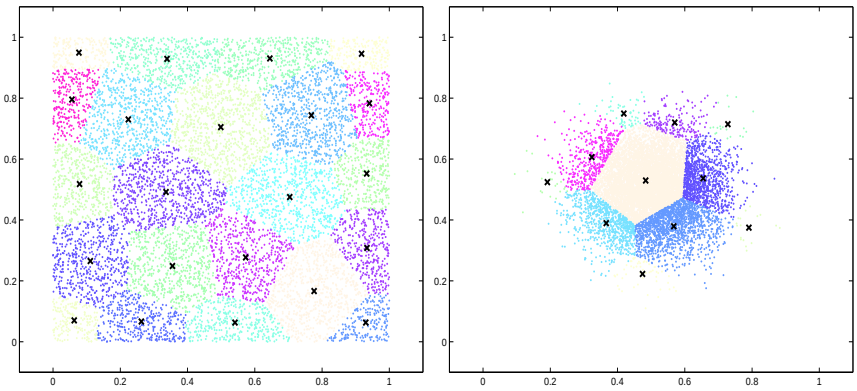



Fig. 2. (a) Using the farthest point clustering algorithm 10,000 points uniformly distributed in a unit square are divided into 22 clusters with the maximum radius of the clusters being 0.2. (b) 10,000 points normally distributed in a unit square are divided into 11 clusters with the maximum radius of the clusters being 0.2.

an arbitrary point $v_0$ as the center of the first cluster and add it to the center set $C$. Then for $i = 1$ to $k$ do the following: at step $i$, for every point $v$, compute its distance to the set $C$: $d_i(v, C) = \min_{c \in C} \|v - c\|$. Let $v_i$ be the point that is farthest from $C$, i.e., the point for which $d_i(v_i, C) = \max_v d_i(v, C)$. Add $v_i$ to set $C$. Report the points $v_0, v_1, \ldots, v_{k-1}$ as the cluster centers. Each point is assigned to its nearest center.

The direct implementation of farthest-point clustering has running time $\mathcal{O}(Nk)$. Feder and Greene [Feder and Greene 1988] give a two-phase algorithm with optimal running time $\mathcal{O}(N \log k)$. The first phase of their algorithm clusters points into rectangular boxes using Vaidya's [Vaidya 1986]'s *box decomposition*– a sort of quadtree in which cubes are shrunk to bounding boxes before splitting. The second phase resembles the farthest-point clustering on a sparse graph that has a vertex for each box. In practice, the initial point has little influence on the final approximation radius, if number of the points is sufficiently large.

Fig. 2 displays the results of farthest-point algorithm on a sample two dimensional data-set. After the end of the clustering procedure the center of each cluster is recomputed as the mean of all the points lying in each cluster. The farthest point algorithm is progressive. This means that if we have $k$ centers and we wish to compute the $(k+1)^{th}$ center, the first $k$ centers do not change.

## 3. IMPROVED FAST GAUSS TRANSFORM

Having discussed the Taylor series and the space subdivision we now present the actual algorithm. For any point $x_* \in \mathcal{R}^d$ the Gauss Transform at $y_j$ can be written

as,

$$\begin{aligned} G(y_j) &= \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2/h_i^2}, \\ &= \sum_{i=1}^{N} q_i e^{-\|(y_j - x_*) - (x_i - x_*)\|^2/h_i^2}, \\ &= \sum_{i=1}^{N} q_i e^{-\|x_i - x_*\|^2/h_i^2} e^{-\|y_j - x_*\|^2/h_i^2} e^{2(y_j - x_*)\cdot(x_i - x_*)/h_i^2}. \end{aligned} \tag{23}$$

In Eq. 23 the first exponential inside the summation $e^{-\|x_i - x_*\|^2/h_i^2}$ depends only on the source coordinates $x_i$ and the source scales $h_i$. However for the second and the third exponential the source and target are entangled. The second exponential $e^{-\|y_j - x_*\|^2/h_i^2}$ depends on the target coordinates $y_j$ and the source scales $h_i$, while the third exponential $e^{2(y_j - x_*)\cdot(x_i - x_*)/h_i^2}$ depends on the source coordinates $x_i$, the target coordinates $y_j$, and the source scales $h_i$. The crux of the algorithm is to separate this entanglement via the Taylor's series expansion of the exponentials.

### 3.1 Factorization

Using Corollary 2.1 the series expansion for $e^{-\|y_j - x_*\|^2/h_i^2}$ can be written as

$$e^{-\|y_j - x_*\|^2/h_i^2} = \sum_{m=0}^{p_1^i - 1} \frac{(-1)^m}{m!} \left(\frac{1}{h_i^2}\right)^m \|y_j - x_*\|^{2m} + error_{p_1^i}^1. \tag{24}$$

Similarly, using Corollary 2.2 the series expansion for $e^{2(y_j - x_*)\cdot(x_i - x_*)/h_i^2}$ can be written as,

$$e^{2(y_j - x_*)\cdot(x_i - x_*)/h_i^2} = \sum_{n=0}^{p_2^i - 1} \frac{2^n}{n!} \left(\frac{1}{h_i^2}\right)^n [(y_j - x_*)\cdot(x_i - x_*)]^n + error_{p_2^i}^2. \tag{25}$$

Using the multi-index notation (Eq. 22), this expansion can be written as,

$$e^{2(y_j - x_*)\cdot(x_i - x_*)/h_i^2} = \sum_{|\alpha| \le p_2^i - 1} \frac{2^\alpha}{\alpha!} \left(\frac{1}{h_i^2}\right)^\alpha (y_j - x_*)^\alpha (x_i - x_*)^\alpha + error_{p_2^i}^2. \tag{26}$$

The truncation numbers $p_1^i$ and $p_2^i$ for each source $x_i$ is chosen based on the prescribed error, the bandwidth $h_i$, and the distance from the expansion center. A strategy for choosing the truncation is discussed in a later section.

### 3.2 Regrouping

Ignoring the error terms $G(y_j)$ can be approximated as,

$$\begin{aligned}\hat{G}(y_j) &= \sum_{i=1}^{N} q_i e^{-\|x_i-x_*\|^2/h_i^2} e^{-\|y_j-x_*\|^2/h_i^2} e^{2(y_j-x_*)\cdot(x_i-x_*)/h_i^2} \\ &= \sum_{i=1}^{N} q_i e^{-\|x_i-x_*\|^2/h_i^2} \left[\sum_{m=0}^{p_1^i-1} \frac{(-1)^m}{m!}\left(\frac{1}{h_i^2}\right)^m \|y_j-x_*\|^{2m}\right] \\ &\quad \left[\sum_{|\alpha|\le p_2^i-1} \frac{2^\alpha}{\alpha!}\left(\frac{1}{h_i^2}\right)^\alpha (y_j-x_*)^\alpha (x_i-x_*)^\alpha\right]\end{aligned} \tag{27}$$

Let $p_1^{max} = \max_i p_1^i$ and $p_2^{max} = \max_i p_2^i$. Let $\mathbf{1}_{m\le p_1^i-1}$ be an indicator function for $m \le p_1^i - 1$ and $\mathbf{1}_{|\alpha|\le p_2^i-1}$ be an indicator function for $|\alpha| \le p_2^i - 1$, i.e.,

$$\mathbf{1}_{|\alpha|\le p_2^i-1} = \begin{cases} 1 & \text{if } |\alpha| \le p_2^i - 1 \\ 0 & \text{if } |\alpha| > p_2^i - 1 \end{cases}. \tag{28}$$

So now we have

$$\hat{G}(y_j) = \sum_{m=0}^{p_1^{max}-1} \sum_{|\alpha|\le p_2^{max}-1} C_{m\alpha} \|y_j-x_*\|^{2m} (y_j-x_*)^\alpha, \tag{29}$$

where,

$$C_{m\alpha} = \frac{(-1)^m 2^\alpha}{m!\alpha!} \sum_{i=1}^{N} q_i e^{-\|x_i-x_*\|^2/h_i^2} \left(\frac{1}{h_i^2}\right)^{m+|\alpha|} (x_i-x_*)^\alpha \mathbf{1}_{m\le p_1^i-1}\mathbf{1}_{|\alpha|\le p_2^i-1}. \tag{30}$$

The coefficients $C_{m\alpha}$ can be evaluated separately is $O(N)$. Evaluation of $\widehat{G}_r(y_j)$ at $M$ points is $O(M)$. Hence the computational complexity has reduced from the quadratic $O(NM)$ to the linear $O(N+M)$. Detailed analysis of the computational complexity will be provided later.

### 3.3 Space sub-divison

Thus far, we have used the Taylor's series expansion about a certain point $x_*$. However if we use the same $x_*$ for all the points we typically would require very high truncation numbers since the Taylor's series is valid only in a small open ball around $x_*$. We use an data adaptive space partitioning scheme like the farthest point clustering algorithm to divide the $N$ sources into $K$ clusters, $S_k$ for $k = 1, \ldots, K$ with $c_k$ being the center of each cluster. The Gauss transform can be written as,

$$\hat{G}(y_j) = \sum_{k=1}^{K} \sum_{m=0}^{p_1^{max}-1} \sum_{|\alpha|\le p_2^{max}-1} C_{m\alpha}^k \|y_j-c_k\|^{2m} (y_j-c_k)^\alpha, \tag{31}$$

where,

$$C^k_{m\alpha} = \frac{(-1)^m 2^\alpha}{m!\alpha!} \sum_{x_i \in S_k} q_i e^{-\|x_i - c_k\|^2/h_i^2} \left(\frac{1}{h_i^2}\right)^{m+|\alpha|} (x_i - c_k)^\alpha \, \mathbf{1}_{m \le p_1^i - 1} \mathbf{1}_{|\alpha| \le p_2^i - 1}. \tag{32}$$

### 3.4 Rapid decay of the Gaussian

Since the Gaussian decays very rapidly a further speedup is achieved if we ignore all the sources belonging to a cluster if the cluster is greater than a certain distance from the target point, $\|y_j - c_k\| > r_y^k$. The cluster cutoff radius depends on the desired precision $\epsilon$ and the bandwidth of the sources in the cluster. So now the Gauss transform is evaluated as

$$\hat{G}(y_j) = \sum_{\|y_j - c_k\| \le r_y^k} \sum_{m=0}^{p_1^{max}-1} \sum_{|\alpha| \le p_2^{max}-1} C^k_{m\alpha} \|y_j - c_k\|^{2m} (y_j - c_k)^\alpha, \tag{33}$$

where,

$$C^k_{m\alpha} = \frac{(-1)^m 2^\alpha}{m!\alpha!} \sum_{x_i \in S_k} q_i e^{-\|x_i - c_k\|^2/h_i^2} \left(\frac{1}{h_i^2}\right)^{m+|\alpha|} (x_i - c_k)^\alpha \, \mathbf{1}_{m \le p_1^i - 1} \mathbf{1}_{|\alpha| \le p_2^i - 1}. \tag{34}$$

## 4. COMPUTATIONAL AND SPACE COMPLEXITY

—The farthest point clustering has running time $\mathcal{O}(Nd \log K)$ [Feder and Greene 1988].

—Computing the cluster coefficients $C^k_{m\alpha}$ for all the clusters is of $\mathcal{O}(Np_1^{max} r_{p_2^{max} d})$, where $r_{p_2^{max} d} = \binom{p_2^{max}+d}{d}$ is the total number of $d$-variate monomials of degree less than or equal to $p_2^{max}$.

—Computing $\hat{G}(y_j)$ is of $\mathcal{O}(Mnp_1^{max} r_{p_2^{max} d})$ where $n$ if the maximum number of influential neighbor clusters for each target.

Hence the total computational complexity is

$$\mathcal{O}(dN \log K + Np_1^{max} r_{p_2^{max} d} + Mnp_1^{max} r_{p_2^{max} d}). \tag{35}$$

Assuming $M = N$, the total computational complexity can be written as

$$\mathcal{O}(\left[d \log K + (1+n) p_1^{max} r_{p_2^{max} d}\right] N). \tag{36}$$

The constant term depends on the dimensionality, the bandwidth, and the accuracy required. The constant $r_{p_2 d}$ is much smaller than $p_2^d$ in the original FGT. For example, when $p_2 = 10$ and $d = 10$ then, $r_{p_2 d} = 18,4756$ while $p_2^d = 10^{10}$. For $d \to \infty$ and moderate $p_2$, the number of terms is $\mathcal{O}(dp_2)$.

A different truncation number is chosen for each data point depending on its distance from the cluster center and the source bandwidth. A good consequence of this strategy is that only a few points at the boundary of the clusters will have high truncation numbers.

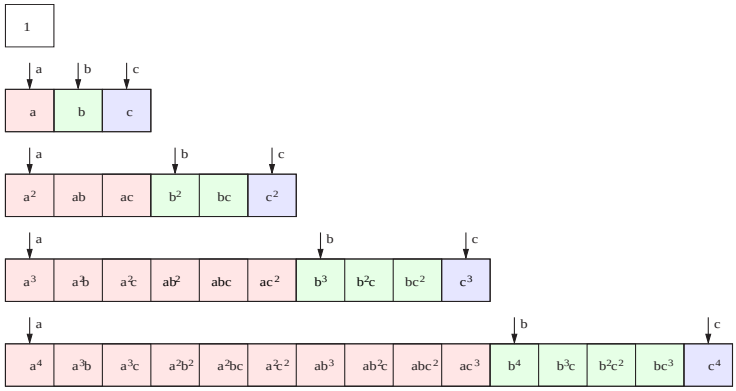



Fig. 3. Efficient expansion of multivariate polynomials using Horner's rule.

For each cluster we need to store $p_1^{max} r_{p_2^{max} d}$ coefficients. So the storage complexity is $O(K p_1^{max} r_{p_2^{max} d} + N + M)$.

## 5. EVALUATING MULTIVARIATE POLYNOMIALS USING HORNER'S RULE

Evaluating each $d$-variate monomial of degree $n$ directly requires $n$ multiplications. Hence direct evaluation of of all $d$-variate monomials of degree less than or equal to $n$ requires $\sum_{k=0}^{n} k\binom{k+d-1}{d-1}$ multiplications. The storage requirement is $r_{nd}$. However, efficient evaluation using the Horner's rule requires $r_{nd} - 1$ multiplications. The required storage is $r_{nd}$.

For a $d$-variate polynomial of order $n$, we can store all terms in a vector of length $r_{nd}$. Starting from the order zero term (constant 1), we take the following approach. Assume we have already evaluated terms of order $k-1$. We use an array of size $d$ to record the positions of the d leading terms (the simple terms such as $a^{k-1}$, $b^{k-1}$, $c^{k-1}$, . . . in Fig. 3) in the terms of order $k-1$. Then terms of order $k$ can be obtained by multiplying each of the $d$ variables with all the terms between the variables leading term and the end, as shown in the Fig. 3. The positions of the $d$ leading terms are updated respectively. The required storage is $r_{nd}$ and the computations of the terms require $r_{nd} - 1$ multiplications.

## 6. CHOOSING THE PARAMETERS

Given any $\epsilon > 0$, we want to choose the following parameters, $K$ (the number of clusters), $\{r_y^k\}_{k=1}^K$ (the cut off radius for each cluster), and $\{p_1^i, p_2^i\}_{i=1}^N$ (the truncation numbers for each source point $x_i$) such that for any target point $y_j$ we can guarantee that

$$\frac{|\hat{G}(y_j) - G(y_j)|}{Q} \leq \epsilon, \tag{37}$$

where $Q = \sum_{i=1}^{N} |q_i|$. Let us define $\Delta_{ij}$ to be the point wise error in $\widehat{G}(y_j)$ contributed by the $i^{th}$ source $x_i$. We now require that

$$|\hat{G}(y_j) - G(y_j)| = \left| \sum_{i=1}^{N} \Delta_{ij} \right| \leq \sum_{i=1}^{N} |\Delta_{ij}| \leq Q\epsilon = \sum_{i=1}^{N} |q_i|\epsilon. \tag{38}$$

One way to achieve this is to let

$$|\Delta_{ij}| \leq |q_i|\epsilon \ \forall i = 1, \ldots, N. \tag{39}$$

We choose this strategy because it helps us get tighter bounds. Also this strategy is very useful in our case where the bandwidths are varying. If we decide a common truncation number for all the sources then it has to be based on the minimum scale, for which the truncation number can be quite large. Sources with large bandwidths may not need such large truncation numbers.

Let $c_k$ be the center of the cluster to which $x_i$ belongs. There are two different ways in which a source can contribute to the error. The first is due to ignoring the cluster $S_k$ if it is outside a given radius $r_y^k$ from the target point $y_j$. In this case,

$$\Delta_{ij} = q_i e^{-\|y_j - x_i\|^2/h_i^2}. \tag{40}$$

The second source of error is due to truncation of the two Taylor's series. For all clusters which are within a distance $r_y^k$ from the target point the error is due to the truncation of the Taylor's series after order $p_1^i$ and $p_2^i$. From Equations 23, 24, and 26 the error can be written as,

$$\begin{aligned}
\Delta_{ij} &= q_i e^{-\|x_i - c_k\|^2/h_i^2} \left( e^{-\|y_j - c_k\|^2/h_i^2} - error^1_{p_1^i} \right) error^2_{p_2^i} \\
&+ q_i e^{-\|x_i - c_k\|^2/h_i^2} \left( e^{2(y_j - c_k)\cdot(x_i - c_k)/h_i^2} - error^2_{p_2^i} \right) error^1_{p_1^i} \\
&+ q_i e^{-\|x_i - c_k\|^2/h_i^2} error^1_{p_1^i} error^2_{p_2^i}.
\end{aligned} \tag{41}$$

Our strategy for choosing the parameters is as follows. The cutoff radius $r_y^k$ for each cluster is chosen based on Equation 40 and the radius of each cluster $r_x^k$. Given $r_y^k$ and $\|x_i - c_k\|$ the truncation numbers for each source is chosen based on Equation 41. Towards the end we suggest a strategy to choose the number of clusters $K$.

### 6.1 Automatically choosing the cut off radius for each cluster

We ignore all sources belonging to a cluster $S_k$ if $\|y_j - c_k\| > r_y^k$. $r_y^k$ should be chosen such that for all sources in cluster $S_k$ the error $|\Delta_{ij}| = |q_i| e^{-\|y_j - x_i\|^2/h_i^2} \leq |q_i|\epsilon$. This implies that

$$\|y_j - x_i\| > h_i \sqrt{\ln(1/\epsilon)} \tag{42}$$

Using the reverse triangle inequality, $\|a - b\| \geq \big|\|a\| - \|b\|\big|$, and the fact that $\|y_j - c_k\| > r_y^k$ and $\|x_i - c_k\| \leq r_x^k$, we have

$$\begin{aligned}
\|y_j - x_i\| = \|y_j - c_k + c_k - x_i\| &= \|(y_j - c_k) - (x_i - c_k)\|, \\
&\geq \big|\|(y_j - c_k)\| - \|(x_i - c_k)\|\big|, \\
&> \left| r_y^k - r_x^k \right|.
\end{aligned} \tag{43}$$

So in order that the error due to ignoring the faraway clusters is less than $q_i\epsilon$ we have to choose $r_y^k$ and $r_x^k$ such that,

$$\left| r_y^k - r_x^k \right| > h_i \sqrt{\ln(1/\epsilon)}. \tag{44}$$

If we choose $r_y^k > r_x^k$ then,

$$r_y^k > r_x^k + h_{max}^k \sqrt{\ln(1/\epsilon)}, \tag{45}$$

where $h_{max}^k = \max_{x_i \in S_k} h_i$ is the maximum source scale in cluster $S_k$. Let $R$ be the maximum distance between any source and target point. For example if the data were distributed in a $d$-dimensional hypercube of length $a$, then $R \leq \sqrt{d}a$, i.e., the length of the maximum diagonal. Hence,

$$r_y^k > r_x^k + \min\left(R, h_{max}^k \sqrt{\ln(1/\epsilon)}\right). \tag{46}$$

### 6.2 Automatically choosing the truncation numbers for each source

For all clusters which are within a distance $r_y^k$ from the target point the error is due to the truncation of the Taylor's series after order $p_1^i$ and $p_2^i$. From Equation 41 it can seen that the error consists of three components.

$$\Delta_{ij} = \Delta_{ij}^1 + \Delta_{ij}^2 - \Delta_{ij}^3, \tag{47}$$

where

$$\begin{aligned}
\Delta_{ij}^1 &= q_i e^{-\|x_i - c_k\|^2/h_i^2} e^{-\|y_j - c_k\|^2/h_i^2} error_{p_2^i}^2, \\
\Delta_{ij}^2 &= q_i e^{-\|x_i - c_k\|^2/h_i^2} e^{2(y_j - c_k)\cdot(x_i - c_k)/h_i^2} error_{p_1^i}^1, \\
\Delta_{ij}^3 &= q_i e^{-\|x_i - c_k\|^2/h_i^2} error_{p_1^i}^1 error_{p_2^i}^2.
\end{aligned} \tag{48}$$

For a given source $x_i$ we have to choose $p_1^i$ and $p_2^i$ such that

$$|\Delta_{ij}| = |\Delta_{ij}^1 + \Delta_{ij}^2 - \Delta_{ij}^3| \leq |\Delta_{ij}^1| + |\Delta_{ij}^2| + |\Delta_{ij}^3| \leq |q_i|\epsilon. \tag{49}$$

One way to achieve this is to let

$$|\Delta_{ij}^1| \leq |q_i|\epsilon/3 \text{ and } |\Delta_{ij}^2| \leq |q_i|\epsilon/3. \tag{50}$$

Before we proceed we show that if Equation 50 is satisfied then $|\Delta_{ij}^3| \leq |q_i|\epsilon/3$.

$$\begin{aligned}
|\Delta_{ij}^3| &= |q_i| e^{-\|x_i - c_k\|^2/h_i^2} |error_{p_1^i}^1| |error_{p_2^i}^2|, \\
&= (1/|q_i|) |\Delta_{ij}^1| |\Delta_{ij}^2| e^{\|y_j - x_i\|^2/h_i^2}.
\end{aligned} \tag{51}$$

Using the fact that $\|y_j - x_i\| < h_i \sqrt{\ln(1/\epsilon)}$ we have,

$$|\Delta_{ij}^3| < (1/\epsilon|q_i|) |\Delta_{ij}^1| |\Delta_{ij}^2| < |q_i|\epsilon/9 < |q_i|\epsilon/3. \tag{52}$$

From Corollary 2.1 and 2.2 we have,

$$|error_{p_1^i}^1| < \frac{2^{p_1^i} 1.09}{(2p_1^i)^{1/12}\sqrt{(2p_1^i)!}} \left(\frac{\|x_i - c_k\|}{h_i}\right)^{2p_1^i}, \tag{53}$$

$$|error_{p_2^i}^2| < \frac{2^{p_2^i}}{p_2^i!} \left(\frac{\|x_i - c_k\|}{h_i}\right)^{p_2^i} \left(\frac{\|y_j - c_k\|}{h_i}\right)^{p_2^i} e^{2\|x_i - c_k\|\|y_j - c_k\|/h_i^2}. \tag{54}$$

Hence we have

$$|\Delta_{ij}^1| < |q_i| \frac{2^{p_2^i}}{p_2^i!} e^{-(\|x_i-c_k\|-\|y_j-c_k\|)^2/h_i^2} \left(\frac{\|x_i-c_k\|}{h_i}\right)^{p_2^i} \left(\frac{\|y_j-c_k\|}{h_i}\right)^{p_2^i},$$
$$|\Delta_{ij}^2| < |q_i| \frac{2^{p_1^i} 1.09}{(2p_1^i)^{1/12}\sqrt{(2p_1^i)!}} e^{-\|x_i-c_k\|^2/h_i^2} e^{2\|y_j-c_k\|\|x_i-c_k\|/h_i^2} \left(\frac{\|x_i-c_k\|}{h_i}\right)^{2p_1^i}. \tag{55}$$

$|\Delta_{ij}^1|$ and $|\Delta_{ij}^2|$ depend both on distance between the source and the cluster center, i.e., $\|x_i - c_k\|$ and the distance between the target and the cluster center, i.e., $\|y_j - c_k\|$. The speedup is achieved because at each cluster $S_k$ we sum up the effect of all the sources. As a result we do not have a knowledge of $\|y_j - c_k\|$. So we will have to bound the right hand side of Equation 55, such that it is independent of $\|y_j - c_k\|$.

The error $|\Delta_{ij}^1|$ increases as a function of $\|y_j - c_k\|$, reaches a maximum and then starts decreasing. The maximum is attained at

$$\|y_j - c_k\| = \|y_j - c_k\|_* = \frac{\|x_i - c_k\| + \sqrt{\|x_i - c_k\|^2 + 2p_2^i h_i^2}}{2}. \tag{56}$$

Hence we choose $p_2^i$ such that,

$$|\Delta_{ij}^1| \big|_{\|y_j-c_k\|=\|y_j-c_k\|_*} \leq |q_i|\epsilon/3. \tag{57}$$

In case $\|y_j - c_k\|_* > r_y^k$ we need to choose $p_2^i$ based on $r_y^k$, since $\Delta_{ij}$ will be much lower there.

Hence out strategy for choosing $p_2^i$ is,

$$|\Delta_{ij}^1| \Big|_{\left[\|y_j-c_k\|=\min\left(\|y_j-c_k\|_*, r_y^k\right)\right]} \leq |q_i|\epsilon/3. \tag{58}$$

Similarly we choose $p_1^i$ such that

$$|\Delta_{ij}^2| \Big|_{\left[\|y_j-c_k\|=r_y^k\right]} \leq |q_i|\epsilon/3. \tag{59}$$

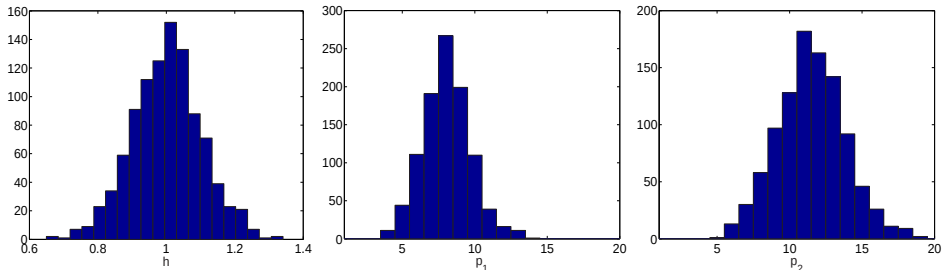
### 6.3 Automatically choosing the number of clusters

The only free parameter is the number of clusters $K$, which can be set to any reasonable value. If the source and the target points are uniformly distributed in a unit hypercube [2] then $r_x \sim K^{-1/d}$. Based on this we choose $K$ such that $rx$ is approximately equal to $h_{max}$. Hence,

$$K \sim \left\lceil (h_{max} + h_{min}/2)^{-d} \right\rceil. \tag{60}$$

Figure 4 demonstrates the computational advantage of using pointwise error bounds to choose the truncation numbers. The bandwidths were normally distributed with mean 1.0 and standard deviation 0.1 as shown in Figure 4(a). Figures 4(b) and (c) show the histogram of the truncation numbers $p_1$ and $p_2$ respectively. The

[2] If the data lies on a lower dimensional manifold, as usually is the case for structured data in high dimensions, we use the relation $r_x \sim K^{-1/d_{eff}}$. $d_{eff}$ is the actual intrinsic dimensionality of the data.




(a) (b) (c)

Fig. 4. The histogram of (a) bandwidths, (b) truncation number $p_1$, and (c) truncation number $p_2$.

truncation numbers are also roughly normally distributed. Only sources with small bandwidths will have a large truncation number.

## 7. NUMERICAL EXPERIMENTS

The algorithms were programmed in C++ and was run on a 1.6 GHz Pentium M processor with 512Mb of RAM. The code is available by contacting the first author for academic use.

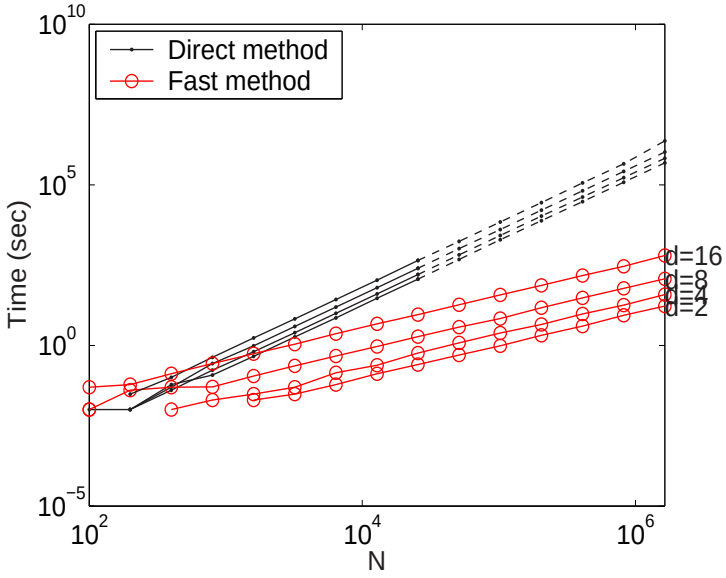
$N$ points were uniformly distributed in a unit hypercube. The weights $q_i$ were uniformly distributed between 0 and 1. The Gauss transform was evaluated at $M = N$ points uniformly distributed in the unit hypercube. The parameters were chosen such that the maximum absolute error relative to the total weight $Q$ was less than $10^{-3}$, which is a reasonable choice for most kernel density estimation in nonparametric statistics.

Table I shows the running time for the direct evaluation and the fast method, for different values of $N$ for $d = 3$. The source strengths $h_i$ were normally distributed with mean 2.0 and standard deviation 0.1. We see that the running time of the IFGT grows linearly as the number of sources and targets increases, while that of the direct evaluation grows quadratically. For example for $N = M = 1,024,000$ while the direct evaluation takes around 2.6 days the fast evaluation requires only 4.65 minutes with an error of around $10^{-5}$.

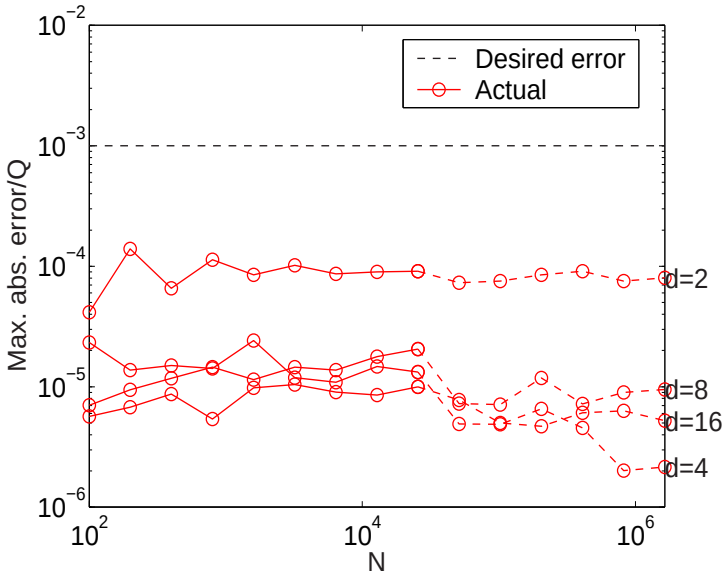
Fig. 5(a) shows the the running time in seconds for the direct and the fast methods as a function of $N$ for different dimensions $d$. The bandwidths were normally distributed with mean $d$ and variance 0.1. As the dimensionality increases the volume enclosed by a unit hypercube increases. As a result relative to the volume, the Gaussian appears to be at a smaller scale as the dimensionality of the space increases. Hence we have chosen scales to increase with dimension to fairly span the volume.

## 8. CONCLUSION

In this paper we extended the improved fast Gauss transform to handle variable source bandwidths. For each source point we choose different truncation numbers depending on its bandwidth and distance to the cluster center. Extremely good speedups were achieved for large bandwidths. For very small bandwidths the trun-




(a)




(b)

Fig. 5. (a) Comparison of the time taken by the direct and the fast method as a function of the number of points $N$ for different values of $d$. (b) The corresponding maximum absolute error relative to the total weight $Q$. The bandwidths were normally distributed with mean $h = d$ and variance 0.1. The target error was set to $10^{-3}$. The source and target points were uniformly distributed in a unit hypercube. The weights $q_i$ were uniformly distributed between 0 and 1. For $N > 25,600$ the timing results for the direct evaluation were obtained by evaluating the Gauss transform at $M = 100$ points and then extrapolating the results.

**Algorithm 1**: The improved fast Gauss transform with variable source scales.

**Input** :

$x_i \in \mathcal{R}^d \ i = 1, \ldots, N$ /* $N$ sources in $d$ dimensions. */
$q_i \in \mathcal{R} \ i = 1, \ldots, N$ /* source weights. */
$h_i \in \mathcal{R}^+ \ i = 1, \ldots, N$ /* source bandwidths. */
$y_j \in \mathcal{R}^d \ j = 1, \ldots, M$ /* $M$ targets in $d$ dimensions. */
$\epsilon > 0$ /* Desired error. */

**Output**: Computes an approximation $\hat{G}(y_j)$ to $G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2/h_i^2}$. such that the $|\hat{G}(y_j) - G(y_j)|Q \leq \epsilon$, where $Q = \sum_{i=1}^{N} |q_i|$.

**Step 0** *Define* $\delta_1(p,h,a,b) = \frac{1}{p!}\left(\frac{2ab}{h^2}\right)^p e^{-(a-b)^2/h^2}$, $b_*(p,h,a) = \frac{a+\sqrt{a^2+2ph^2}}{2}$, and $\delta_2(p,h,a,b) = \frac{1.09}{(2p)^{1/12}\sqrt{(2p)!}}\left(\frac{2a^2}{h^2}\right)^p e^{-(a^2-2ab)/h^2}$ ;

**Step 1** *Choose the number of clusters* $K \sim \left\lceil (h_{max} + h_{min}/2)^{-d} \right\rceil$ where $h_{max} = \max_i h_i$ and $h_{min} = \min_i h_i$;

**Step 2** *Divide the* $N$ *sources into* $K$ *clusters,* $\{S_k\}_{k=1}^K$, *using the Feder and Greene's farthest-point clustering algorithm. Let* $c_k$ *and* $r_x^k$ *be the center and radius respectively of the* $k^{th}$ *cluster* ;

**Step 3** *For each cluster* $S_k$ *with center* $c_k$ *compute the coefficients* $C_\alpha^k$.

$C_{m\alpha}^k =$
$\frac{(-1)^m 2^\alpha}{m!\alpha!} \sum_{x_i \in S_k} q_i e^{-\|x_i - c_k\|^2/h_i^2} \left(\frac{1}{h_i^2}\right)^{m+|\alpha|} (x_i - c_k)^\alpha \mathbf{1}_{m \leq p_1^i - 1} \mathbf{1}_{|\alpha| \leq p_2^i - 1}$.

*The truncation numbers* $p_1^i$ *and* $p_2^i$ *for each source are selected such that such that*

$\delta_1(p = p_2^i, h = h_i, a = \|x_i - c_k\|, b = \min\left[b_*(p_2^i, h_i, \|x_i - c_k\|), r^k + r_x^k\right]) \leq \epsilon/3$
$\delta_2(p = p_1^i, h = h_i, a = \|x_i - c_k\|, b = r^k + r_x^k) \leq \epsilon/3$
*where* $r^k = \min\left(R, h_{max}^k \sqrt{\ln(1/\epsilon)}\right)$ *and* $h_{max}^k = \max_{x_i \in S_k} h_i$;

**Step 4** *For each target* $y_j$ *the discrete Gauss transform is evaluated as*

$\hat{G}(y_j) = \sum_{\|y_j - c_k\| \leq r^k + r_x^k} \sum_{m=0}^{p_1^{max}-1} \sum_{|\alpha| \leq p_2^{max}-1} C_{m\alpha}^k \|y_j - c_k\|^{2m} (y_j - c_k)^\alpha$;

cation number required are pretty large. One way to handle very small bandwidths is to introduce a cutoff $h_c$. For all sources $h_i < h_c$ the nearest neighbor searching algorithms can be used to directly sum the Gaussians which have significant influence on the target point.

## REFERENCES

Abramowitz, M. and Stegun, I. A. 1972. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover. 5

Bern, M. and Eppstein, D. 1997. *Approximation algorithms for NP-hard problems*. PWS

Table I. The running times in seconds for direct evaluation v.s. the improved fast Gauss transform. The speedup achieved and the maximum absolute error relative to the total weight $Q$ are also shown. The source and target points were uniformly distributed in a unit cube. The weights $q_i$ were uniformly distributed between 0 and 1. The source strengths $h_i$ were normally distributed with mean 2.0 and standard deviation 0.1. For $N > 32,000$ the timing results for the direct evaluation were obtained by evaluating the Gauss transform at $M = 100$ points and then extrapolating the results. The target error was set to $10^{-3}$.

| Case | $N = M$ | Direct (sec.) | IFGT (sec.) | Speedup | Error |
|---|---|---|---|---|---|
| 1 | 1000 | 0.23 | 0.26 | 0.88 | 6.81e-005 |
| 2 | 2000 | 0.84 | 0.50 | 1.68 | 5.31e-005 |
| 3 | 4000 | 3.37 | 1.01 | 3.33 | 7.02e-005 |
| 4 | 8000 | 12.80 | 2.05 | 6.23 | 1.32e-005 |
| 5 | 16000 | 52.96 | 4.51 | 11.75 | 3.61e-005 |
| 6 | 32000 | 217.92 | 9.40 | 23.17 | 4.93e-005 |
| 7 | 64000 | 878.08 | 17.70 | 49.62 | 3.81e-005 |
| 8 | 128000 | 3512.32 | 35.64 | 98.55 | 3.63e-005 |
| 9 | 256000 | 14097.92 | 71.46 | 197.28 | 4.36e-005 |
| 10 | 512000 | 56453.12 | 152.41 | 370.41 | 3.45e-005 |
| 11 | 1024000 | 225607.68 | 279.00 | 808.63 | 3.64e-005 |
| 12 | 2048000 | 900976.64 | 1028.88 | 875.69 | 2.79e-005 |

Publishing Co., Boston, Chapter Approximation algorithms for geometric problems, 296–345. 7

Feder, T. and Greene, D. 1988. Optimal algorithms for approximate clustering. In *Proc. 20th ACM Symp. Theory of Computing*. 434–444. 8, 11

Gonzalez, T. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science 38*, 293–306. 7

Greengard, L. and Strain, J. 1991. The fast Gauss transform. *SIAM Journal of Scientific and Statistical Computing 12,* 1, 79–94. 3, 5

Hille, E. 1926. A class of reciprocal functions. *The Annals of Mathematics, 2nd Ser. 27,* 4 (Jun.), 427–464. 5

Raykar, V. C., Yang, C., Duraiswami, R., and Gumerov, N. 2005. Fast computation of sums of gaussians in high dimensions. Tech. Rep. CS-TR-4767, Department of Computer Science, University of Maryland, CollegePark. 3

Strain, J. 1991. The fast gauss transform with variable scales. *SIAM J. Sci. Stat. Comput. 12,* 5 (Sep.), 1131–1139. 3

Vaidya, P. M. 1986. An optimal algorithm for the all-nearest-neighbors problem. In *Proc. 27th IEEE FOCS*. 117–122. 8

Yang, C., Duraiswami, R., and Davis, L. 2005. Efficient kernel machines using the improved fast Gauss transform. In *Advances in Neural Information Processing Systems*. 1561–1568. 3, 7

Yang, C., Duraiswami, R., and Gumerov, N. 2003. Improved fast Gauss transform. Tech. Rep. CS-TR-4495, Dept. of Computer Science, University of Maryland, College Park. 3, 7

Yang, C., Duraiswami, R., Gumerov, N., and Davis, L. 2003. Improved fast Gauss transform and efficient kernel density estimation. In *IEEE International Conference on Computer Vision*. 464–471. 7