# Efficient Non-parametric Adaptive Color Modeling Using Fast Gauss Transform

Ahmed Elgammal Ramani Duraiswami Larry S. Davis

Computer Vision Laboratory
The University of Maryland, College Park, MD 20742
{*elgammal,ramani,lsd*}*@umiacs.umd.edu*

## Abstract

*Modeling the color distribution of a homogeneous region is used extensively for object tracking and recognition applications. The color distribution of an object represents a feature that is robust to partial occlusion, scaling and object deformation. A variety of parametric and non-parametric statistical techniques have been used to model color distributions. In this paper we present a non-parametric color modeling approach based on kernel density estimation as well as a computational framework for efficient density estimation. Theoretically, our approach is general since kernel density estimators can converge to any density shape with sufficient samples. Therefore, this approach is suitable to model the color distribution of regions with patterns and mixture of colors. Since kernel density estimation techniques are computationally expensive, the paper introduces the use of the Fast Gauss Transform for efficient computation of the color densities. We show that this approach can be used successfully for color-based segmentation of body parts as well as segmentation of multiple people under occlusion.*

## 1 Introduction

Modeling the color distribution of a homogeneous region has a variety of applications for object tracking and recognition. The color distribution of an object represents a feature that is robust to partial occlusion, scaling and object deformation. It is also relatively stable under rotation in depth in certain applications. Therefore color distributions have been used successfully to track non-rigid bodies [17, 2, 14, 6] with applications like tracking heads [1, 6, 13, 14], hands [11] and other body parts against cluttered backgrounds from stationary or moving platforms. Color distributions have also been used for object recognition.

A variety of parametric and non-parametric statistical techniques have been used to model the color distribution of a homogeneous colored regions. In [17] the color distribution of a region (blob) was modeled using a single Gaussian in the three dimensional $YUV$ space. The use of a single Gaussian to model the color of a blob restricts it to be of a single color which is not a general enough assumption to model regions with mixtures of colors. For example, people's clothing and surfaces with texture usually contain patterns and mixture of colors. Fitting a mixture of Gaussians using the EM algorithm provides a way to model color blobs with a mixture of colors. This technique was used in [13, 14] for color based tracking of a single blob and was applied to tracking faces. The mixture of Gaussians technique faces the problem of choosing the right number of Gaussians for the assumed model (model selection). Non-parametric techniques using histograms have been widely used for modeling the color of object for different applications to overcome the previously mentioned problems with parametric models. Color histograms have been used in [12] for people tracking. This work used 3-dimensional adaptive histograms in RGB space to model the color of the whole person. Color histograms have also been used in [11] for tracking hands, in [2] for color region tracking and in [10] for skin detection. The major drawback with color histograms is the lack of convergence to the right density function if the data set is small. Another major drawback with histograms, in general, is that they are not suitable for higher dimensional features.

A particular nonparametric technique that estimates the underlying density, avoids having to store the complete data, and is quite general is the kernel density estimation technique. In this technique the underlying probability density function is estimated as

$$\hat{f}(x) = \sum_i \alpha_i K(x - x_i) \tag{1}$$

where $K$ is a "kernel function" (typically a Gaussian) centered at the data points, $x_i, i = 1..n$ and $\alpha_i$ are weighting coefficients (typically uniform weights are used, i.e., $\alpha_i = 1/n$). Note that choosing the Gaussian as a kernel function is different from fitting the distribution to a Gaussian model. Here, the Gaussian is only used as a function

that is weighted around the data points. Theoretically, kernel density estimators can converge to any density function with enough samples [15, 3].

The major drawback of kernel density estimation techniques is their computational cost. The use of such an approach requires a way to efficiently evaluate the estimate $\hat{f}(x)$ at any new target point $x$. In general, given $N$ original data samples and $M$ target points at which the density need to be evaluated, the complexity is $O(NM)$ evaluations of the kernel function, multiplications and additions. For many applications in computer vision, where both realtime operation and generality of the classifier are desired, this complexity can be a significant barrier to use of these density estimation techniques. In this paper we present the application of the kernel density estimation technique to the problem of modeling the color distribution of homogeneous regions. The paper introduces the application of the Fast Gauss Transform (FGT) algorithm [8, 9] for efficient estimation of color densities. The algorithm improves the complexity to $O(N + M)$ operations instead of $O(NM)$. We use this approach to segment foreground regions corresponding to people into major body parts, and also to segment foreground regions corresponding to multiple people into individuals.

The outline of the paper is as follows: Section 2 presents the use of kernel density estimation for color modeling. Section 3 describes the fast Gauss algorithm. Application of the approach to body part segmentation is presented in section 4. Evaluations of the performance of the algorithm are presented in section 5.

## 2 Color Density Estimation

Given a sample $S = \{x_i\}_{i=1..N}$ from a distribution with density function $p(x)$, an estimate $\hat{p}(x)$ of the density at $x$ can be calculated using

$$\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N} K_\sigma(x - x_i)$$

where $K_\sigma$ is a kernel function with a bandwidth (scale) $\sigma$ such that $K_\sigma(t) = \frac{1}{\sigma}K(\frac{t}{\sigma})$. The kernel function $K$ should satisfy $K(t) \geq 0$ and $\int K(t)dt = 1$. A variety of kernel functions with different properties have been used in the literature. Typically the Gaussian kernel is used for its continuity, differentiability and locality properties. A good discussion of kernel estimation techniques can be found in [15].

Given a sample $S = \{x_i\}$ taken from an image region, where $i = 1...N$, and $x_i$ is a $d$-dimensional vector representing the color, we can estimate the density function at any point $y$ of the color space directly from $S$ using the product of one dimensional kernels [15] as

$$\hat{P}(y) = \frac{1}{N\prod_{j=1}^{d}\sigma_j}\sum_{i=1}^{N}\prod_{j=1}^{d} K\left(\frac{y_j - x_{ij}}{\sigma_j}\right), \quad (2)$$

where the same kernel function is used in each dimension with a different bandwidth $\sigma_j$ for each dimension of the color space. Usually in color modeling two or three dimensional color spaces are used. Two dimensional chromaticity spaces, e.g., $r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}$ and $a, b$ from the $Lab$ color space, are used when it is desired to make the model invariant to illumination geometry. Three dimensional color spaces are widely used because of their better discrimination since the brightness information is preserved.

Using kernel density estimation for color modeling has many motivations. Unlike histograms, even with a small number of samples, kernel density estimation leads to a smooth, continuous and differentiable density estimate. Kernel density estimation does not assume any specific underlying distribution and, theoretically, the estimate can converge to any density shape with enough samples [15, 3]. Therefore, this approach is suitable to model the color distribution of regions with patterns and mixture of colors. If the underlying distribution is a mixture of Gaussians, kernel density estimation converges to the right density with a small number of samples. Unlike parametric fitting of a mixture of Gaussians, kernel density estimation is a more general approach that does not require the selection of the number of Gaussians to be fitted. One other important advantage of using kernel density estimation is that the adaptation of the model is trivial and can be achieved by adding new samples.

Let us assume that we are using a three dimensional color space and that the color space variables are $a, b, c$. Using Gaussian kernels, i.e., $K_\sigma(t) = \left(\sqrt{2\pi}\sigma\right)^{-1} e^{-1/2(t/\sigma)^2}$ with different bandwidth in each dimension, the density estimation can be evaluated as a sum of Gaussians as

$$\hat{p}(a,b,c) = \frac{1}{N}\sum_{i=1}^{N} e^{-\frac{1}{2}(\frac{a-a_i}{\sigma_a})^2} e^{-\frac{1}{2}(\frac{b-b_i}{\sigma_b})^2} e^{-\frac{1}{2}(\frac{c-c_i}{\sigma_c})^2} \quad (3)$$

A weighted version is practically useful where more weight is given to samples inside the region and less weight for samples from the boundary since they are expected to be noisy.

$$\hat{p}(a,b,c) = \sum_{i=1}^{N} w_i e^{-\frac{1}{2}(\frac{a-a_i}{\sigma_a})^2} e^{-\frac{1}{2}(\frac{b-b_i}{\sigma_b})^2} e^{-\frac{1}{2}(\frac{c-c_i}{\sigma_c})^2}$$

where $\sum_i w_i = 1$. The use of different bandwidths for kernels in different color dimensions is desirable since the variances are different in each color dimension. For example,

the luminance variable usually has more variance than the chromaticity variables and therefore wider kernels should be used in the luminance dimension.

Typically, it is required to evaluate this estimate of the density function at many different points $(a_j, b_j, c_j)$ in the color space corresponding to the color at many different locations in the image. For color-based tracking applications, the process of density estimation is repeated at each new frame for a different set of values, which is a very expensive computational task. The complexity of evaluating the summation in Equation(3) at $M$ different locations in $O(NM)$ with $N$ and $M$ typically very large. The application of the fast Gauss transform, as described in the next section, to this problem reduces the complexity to $O(N+M)$ and allows a very efficient framework for this computation.

# 3 Fast Gauss Transform (FGT)

The FGT was introduced by Greengard and Strain [8, 9] for the rapid evaluation of sums of the form

$$S(t_i) = \sum_{j=1}^{N} f_j \, e^{-\frac{(s_j - t_i)^2}{\sigma^2}}, \qquad i = 1, \ldots, M. \quad (4)$$

Here $\{s_j\}_{j=1..N}$ are $d$-dimensional centers of the Gaussians and are called "sources". Each $t_i$ is a location where the effect of these Gaussians need to be calculated. These locations are called "targets" while $\sigma$ is a scalar and $f_j$ are source strengths. They showed that using their fast algorithm this sum could be computed in $O(N+M)$ operations. They also showed results from 1-D and 2-D tests of the algorithm. It was extended by Strain [16] to sums where $\sigma$ in equation 4 varied with the position of the target or the source, i.e., for the case where $\sigma$ depends on the source location the sum is

$$S(t_i) = \sum_{j=1}^{N} f_j \, e^{-\frac{(s_j - t_i)^2}{\sigma_j^2}}, \qquad i = 1, \ldots, M. \quad (5)$$

The first application of the FGT algorithm to problems in computer vision was made in [**?**].

A generalization of the original algorithm is needed to handle the case where $\sigma$ is different in each dimension, i.e., summations of the form,

$$\begin{aligned} S(t_i) &= \sum_{j=1}^{N} f_j e^{-\sum_{k=1}^{d} \left( \frac{(t_i - s_j)_k}{\sigma_k} \right)^2} \\ &= \sum_{j=1}^{N} f_j e^{-\left[ \left( \frac{(t_i - s_j)_1}{\sigma_1} \right)^2 + \ldots + \left( \frac{(t_i - s_j)_d}{\sigma_d} \right)^2 \right]} \end{aligned} \quad (6)$$

where the subscript $k$ indicates the component along the $k$th coordinate axis, i.e., the covariance matrix is diagonal. Color density estimation, as introduced in section 2 uses Gaussian sums of this form since it is always desired to use different bandwidth for kernels in each color dimension since the variation in each color dimension is different.

## 3.1 Overview of the Algorithm

The shifting identity that is central to the algorithm is a re-expansion of the exponential in terms of a Hermite series by using the identity

$$\begin{aligned} e^{-\left(\frac{t-s}{\sigma}\right)^2} &= e^{-\left(\frac{t-s_0-(s-s_0)}{\sigma}\right)^2} \\ &= e^{-\left(\frac{t-s_0}{\sigma}\right)^2} \sum_{n=0}^{\infty} \frac{1}{n!} \left( \frac{s-s_0}{\sigma} \right)^n H_n \left( \frac{t-s_0}{\sigma} \right) \end{aligned} \quad (7)$$

where $H_n$ are the Hermite polynomials. This formula tells us how to evaluate the Gaussian field $\exp\left(-\left(\frac{t-s}{\sigma}\right)^2\right)$ at the target $t$ due to the source at $s$, as an Hermite expansion centered at any given point $s_0$. Thus a Gaussian centered at $s$ can be shifted to a sum of Hermite polynomials times a Gaussian, all centered at $s_0$. The series converges rapidly and for a given precision, only $p$ terms need be retained. The quantities $t$ and $s$ can be interchanged to obtain a Taylor series around the target location as

$$\begin{aligned} e^{-\left(\frac{t-s}{\sigma}\right)^2} &= e^{-\left(\frac{t-t_0-(s-t_0)}{\sigma}\right)^2} \\ &\simeq \sum_{n=0}^{p} \frac{1}{n!} h_n \left( \frac{s-t_0}{\sigma} \right) \left( \frac{t-t_0}{\sigma} \right)^n \end{aligned} \quad (8)$$

where the Hermite functions $h_n(t)$are defined by

$$h_n(t) = e^{-t^2} H_n(t). \quad (9)$$

Thus, the Gaussian centered at $s$ evaluated at $t$ can be expressed as a Taylor series about a nearby target, $t_o$, where the coefficient of that series is the Hermite function evaluated at $t_o$

The algorithm achieves its gain in complexity by avoiding evaluating every Gaussian at every target (which leads to $O(NM)$ operations). Rather, equivalent $p$ term series are constructed about a small number of source cluster-centers using Equation 7 (for $O(Np^d)$ operations). These series are then shifted to target cluster-centers, and evaluated at the $M$ targets in $O(Mp^d)$ operations. Here the number of terms in the series evaluations, $p$, is related to the desired level of precision $\epsilon$, and is typically small as these series converge quickly. Generally, in computer vision applications there is no need for very precise evaluation.

The process is illustrated in Figure 1. The sources and targets are divided into clusters by dividing the space into uniform boxes. This permits the division of the Gaussians according to their locations. The domain is scaled to be of $O(1)$, and the box size at dimension $k$ is chosen to be of size $r\sqrt{2}\sigma_k$.

| Targets \ Sources | Small | Large |
|---|---|---|
| Small | Direct Evaluation | -- Transform sources into Hermit expansion<br>-- Evaluate the expansion at all targets. |
| Large | -- Convert sources into a local Taylor series<br>-- Evaluate the series at each target. | -- Transform sources into Hermit expansion,<br>-- Convert the expansion to a local Taylor series.<br>-- Evaluate at each target |

**Table 1. The four different algorithms used in the computation based on the number of sources and targets in each pair of boxes**

Since Gaussians decay rapidly, sources in a given box will have no effect (in terms of the desired accuracy) to targets relatively far from there sources (in terms of distance scaled by the standard deviation $\sigma$). Therefore, the effect of sources in a given box need to be computed only for target in close-by boxes. Given the sources in one box and the targets in a neighboring box, the computation is performed using one of the following four methods depending on the number of sources and targets in these boxes: Direct evaluation is used if the number of sources and targets are small. If the sources are clustered in a box then they can be transformed into Hermite expansion about the center of the box using equation 7. This expansion is directly evaluated at each target in the target box if the number of the targets is small. If the targets are clustered then the sources or their expansion are converted to a local Taylor series (equation 8) which is then evaluated at each target in the box. These four methods are shown in table 1. The break-even point when using the expansion is more efficient than direct evaluation (small/large break point) is $O(p^{d-1})$. The number of terms to be retained in the series, $p$, depends on the required precision, the box size scale parameter $r$ and the standard deviation $\sigma$. Further details may be obtained from [9]. The clustering operation is aided by the use of appropriate data structures.

## 4 Color-based Body Part Segmentation

We use the color modeling approach described in section 2 as well as the FGT computation to segment foreground regions corresponding to tracked people in upright pose into major body parts. The foreground regions are detected using background subtraction [5]. People can be dressed in many different ways, but generally they are dressed in a way that leads to a set of major color regions aligned vertically (shirt, T-shirt, jacket etc., on the top and pants, shorts, skirts etc., on the bottom) for people in upright pose. We consider the case where people are dressed in a top-bottom manner which yields a segmentation of the person into a head, torso and bottom. Generally, a person in an

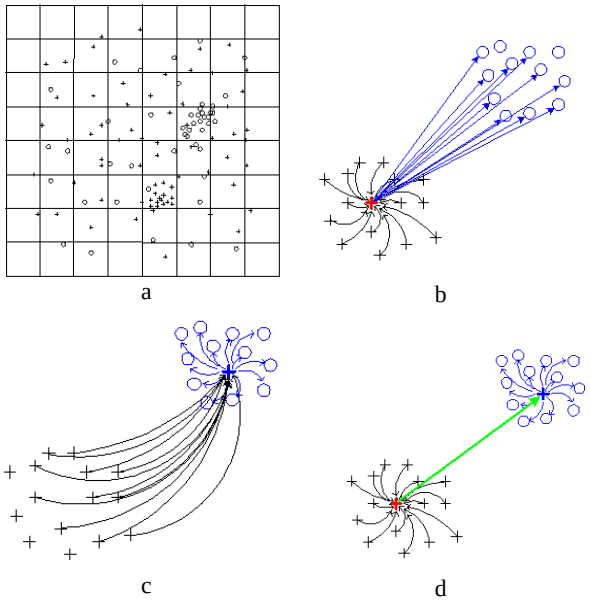


**Figure 1. a) space subdivision. b) Clustered sources are converted into a Hermit expansion that is evaluated at each target. c) sources are converted to a Taylor series near a target cluster. d) Clustered sources are converted to a Hermit expansion and transformed into a Taylor series near a target cluster. (a plus represents a source, a circle represents a target)**

upright pose is modeled as a set of vertically aligned blobs $M = \{A_i\}$ where a blob $A_i$ models a major color region along the vertical axis of the person representing a major part of the body as the torso, bottom or head. Each blob is represented by its color distribution as well as its spatial location with respect to the whole body. Since each blob has the same color distribution everywhere inside the blob, and since the vertical location of the blob is independent of the horizontal axis, the joint distribution of pixel $(x, y, c)$ (the probability of observing color $c$ at location $(x, y)$ given blob $A$) is a multiplication of three density functions

$$P_A(x, y, c) = f_A(x)g_A(y)h_A(c),$$

where $h_A(c)$ is the color density of blob $A$ and the densities $g_A(y)$,$f_A(x)$ represents the vertical and horizontal location of the blob respectively.

Estimates for the color density $h_A(c)$ can be calculated using kernel density estimation as in (2). We represent the color of each pixel as a 3-dimensional vector $X = (r, g, s)$ where $r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}$ are two chromaticity variables and $s = (R + G + B)/3$ is a lightness variable and the three variables are scaled to be on the range 0 to 1. Given a sample of pixels $S_A = \{X_i = (r_i, g_i, s_i)\}$ from

blob $A$, an estimate $\hat{h}_A(\cdot)$ for the color density $h_A(\cdot)$can be calculated as

$$\hat{h}_A(r,g,s) = \frac{1}{N}\sum_{i=1}^{N} K_{\sigma_r}(r-r_i)K_{\sigma_g}(g-g_i)K_{\sigma_s}(s-s_i),$$

Using Gaussian kernels with different bandwidth in each dimension, the density estimation can be evaluated as a sum of Gaussians as

$$\hat{h}_A(r,g,s) = \frac{1}{N}\sum_{i=1}^{N} e^{-\frac{1}{2}(\frac{r-r_i}{\sigma_r})^2} e^{-\frac{1}{2}(\frac{g-g_i}{\sigma_g})^2} e^{-\frac{1}{2}(\frac{s-s_i}{\sigma_s})^2}.$$

Given a set of samples $S = \{S_{A_i}\}$ corresponding to each blob and initial estimates for the position of each blob $y_{A_i}$, each pixel is classified into one of the three blobs based on maximum likelihood classification assuming that all blobs have the same prior probabilities

$$\begin{aligned} X \in A_k \text{ s.t. } k &= \arg_k \max P(X \mid A_k) \\ &= \arg_k \max g_{A_k}(y) h_{A_k}(c) \end{aligned} \quad (10)$$

where the vertical density $g_{A_k}(y)$ is assumed to have a Gaussian distribution $g_{A_k}(y) = N(y_{A_k}, \sigma_{A_k})$. Since the blobs are assumed to be vertically above each other, the horizontal density $f_A(x)$ is irrelevant to the classification.

At each new frame, it is desired to estimate the color density $h_{A_k}(r,g,s)$ corresponding to each blob, $A_k$, at each pixel in the foreground. The FGT algorithm is used to efficiently compute these probabilities. Here the sources are the sample $S_A$ while the targets are the vectors $(r,g,s)$ at each evaluation location. Since both the sources and the targets are clustered in the color space, the FGT algorithm gives a significant speedup. The estimation of the bandwidth for each dimension is done offline by considering batches of regions with a single color distribution taken from images of people's clothing and estimating the variance in each color dimension.

A horizontal blob separator is detected between each two consecutive blobs by finding the horizontal line that minimizes the classification error. Given the detected blob separators, the color model is recaptured by sampling pixels from each blob. Blob segmentation is performed, and blob separators are detected in each new frame as long as the target is isolated and tracked. Adaptation of the color model is achieved by updating the sample (adding new samples and ignoring old samples) for each blob model.

Model initialization is done automatically by taking three samples $S = \{S_H, S_T, S_B\}$ of pixels from three confidence bands corresponding to the head, torso and bottom. The locations of these confidence bands are learned offline as follows: A set of training data[1] is used to learn the location of blob separators (head-torso, torso-bottom) with respect to the body for a set of people in upright pose where these separators are manually marked. Based on these separator location estimates, we can determine the confidence bands proportional to the height where we are confident that they belong to head, torso and bottom and use them to capture initial samples $S = \{S_H, S_T, S_B\}$.

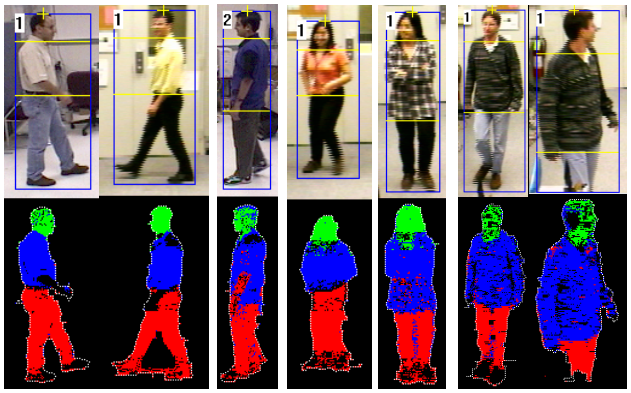

**Figure 2. Example results for blob segmentation**

Figure 2 illustrates some blob segmentation examples for various people. Notice that the segmentation and separator detection is robust even under partial occlusion of the target as in the rightmost result. Also, in some of these examples the clothes are not of a uniform color. In [4] we showed how this representation can be used to segment foreground regions corresponding to multiple people in occlusion. The segmentation is achieved by searching for the best arrangement for the people, in terms of 2D translation, that maximizes the likelihood of the foreground. In this application the computation of color probabilities corresponding to different blobs is performed once each frame for each foreground pixel. The search for the best arrangement does not involve re-computation of these probabilities. For more details refer to [4].

## 5 Experimental Results

In this section we present some experimental results that show the speed up that can be achieved using the FGT algorithm for both color modeling applications and for general kernel density estimation. The first experiment compares the performance of the FGT with different choices of the required precision $\epsilon$ with direct evaluation for a 2D problem. Table 2 shows the CPU time using direct evaluation versus that using FGT for different precisions, $\epsilon = 10^{-4}, 10^{-6}$, and $10^{-8}$ for sources with $\sigma = 0.05$. The box size scale parameter $r$ was set to 0.5. [2] The sources and targets were

[1] The training data consists of 90 samples of different people in upright pose from both genders in different orientations.

[2] The software was written in Visual C++ and the results were obtained on a 700MHz Intel Pentium III PC with 512 MB RAM.

uniformly distributed in the range [0,1] and the strength of the sources were random between 0 and 1. Table 3 shows the division of work between the different components of the FGT algorithm for the $\epsilon = 10^{-6}$ case. From the division of work we notice that for relatively small numbers of sources and targets only direct evaluations are performed. Although, in this case, the algorithm performs only direct evaluations, it is five to ten times faster when compared with direct evaluation because of the way the algorithm divides the space into boxes and the locality of the direct evaluation based on the desired precision. i.e., the FGT algorithm does a smart direct evaluation. This is comparable to the speed-ups reported in [7] where heuristic truncation was performed. As the number of sources and targets increases and they become more clustered, other evaluation decisions are made by the algorithm, and the algorithm starts to show the linear ($O(N+M)$) performance. For very large numbers of sources and targets, the computations are performed through Hermite expansions of sources transformed into Taylor expansions as described above, and this yields a significant speedup. For example, for $N = M = 10^5$, the algorithm gives more than 800 times speedup over direct evaluation for $10^{-4}$ precision.

From the figures we also note that the FGT starts to outperform direct evaluation for numbers of sources and targets as low as 60-80, based on the desired accuracy. This breakeven point can be pushed further down and more speedup can be achieved by increasing the box size scale parameter, $r$. This will enhance the performance of the algorithm for small $N, M$ but will worsen the asymptotic performance.

| N=M | Direct Evaluation | *Fast Gauss* $\epsilon = 10^{-4}$ | $\epsilon = 10^{-6}$ | $\epsilon = 10^{-8}$ |
|---|---|---|---|---|
| 50 | 0.7 | 0.8 | 0.9 | 1.1 |
| 100 | 2.7 | 1.3 | 1.7 | 2.0 |
| 200 | 10.8 | 2.9 | 3.7 | 4.6 |
| 400 | 43 | 8.2 | 10.8 | 14 |
| 800 | 174 | 27 | 36 | 46 |
| 1600 | 689 | 96 | 130 | 163 |
| 3200 | 2754 | 319 | 493 | 640 |
| 6400 | 11917 | 660 | 1281 | 1935 |
| 12800 | 58500 | 942 | 2022 | 3277 |
| 25600 | 234x $10^3$ | 1429 | 3210 | 5113 |
| 51200 | 936x $10^3$ | 2405 | 5602 | 8781 |
| 102400 | 3744x $10^3$ | 4382 | 10410 | 16181 |
| 204800 | 14976x $10^3$ | 8428 | 20191 | 31106 |
| 1024000 | 374x $10^6$ | 43843 | 100263 | 153791 |

**Table 2. Run time in milliseconds for direct evaluation vs. FGT with different precision**

Since for the color modeling application the sources and/or the targets are usually clustered in the space, we need to study the performance of the algorithm for these configurations. Figure 3 shows the performance of the algorithm for different configurations of sources and targets. The direct evaluation is compared with the FGT for three configurations of sources and targets: in the first case, the sources and targets are uniformly distributed between 0 and 1. In the second case the sources are clustered inside a circle of

| *Division of work (%) - Uniform sources and targets* | | | | |
|---|---|---|---|---|
| N=M | Direct Evaluation | Taylor Expansion | Hermit Expansion | Hermit + Taylor Expansion |
| $\leq$ 800 | 100 | 0 | 0 | 0 |
| 1600 | 96.6 | 1.6 | 1.8 | 0 |
| 3200 | 65.7 | 15.6 | 15 | 3.7 |
| 6400 | 5.3 | 18.3 | 16.9 | 59.5 |
| 12800 | 0 | 0.3 | 0.3 | 99.4 |
| $\geq$ 25600 | 0 | 0 | 0 | 100 |

**Table 3. Division of work between different computational methods for uniformly distributed random sources and targets**

radius 0.1 and the targets are uniformly distributed between 0 and 1. In the third case, both the sources and the targets are clustered inside a circle of radius $0.1$. For all three cases the desired precision was set to $10^{-6}$, the sources have a scale $\sigma = 0.05$ and a random strength between 0 and 1. The division of work for the three cases are shown in Tables 3, 4, and 5. From the figures we note that for the cases where sources and/or targets are clustered the computation shows linear time behavior for number of sources and targets as low as 100, which yields a significant speedup. For a very large number of sources and targets, the computations are performed through Hermite expansions of sources transformed into Taylor expansions.

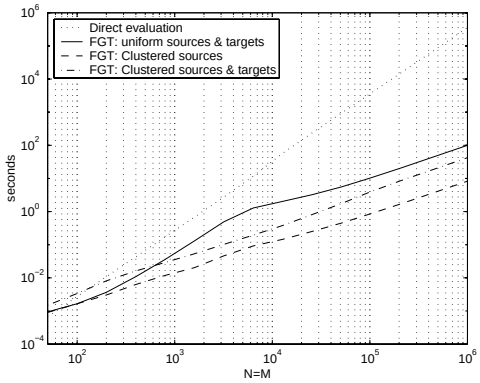


**Figure 3. Run time for FGT with different configuration of sources and targets layout**

| *Division of work (%) - Clustered sources, Uniform targets* | | | | |
|---|---|---|---|---|
| N=M | Direct Evaluation | Taylor Expansion | Hermit Expansion | Hermit + Taylor Expansion |
| 100 | 83.7 | 0 | 16.3 | 0 |
| 200 | 65.2 | 0 | 34.8 | 0 |
| 400 | 32.7 | 0 | 67.3 | 0 |
| 800 | 21.3 | 0 | 78.7 | 0 |
| 1600 | 17.1 | 0.3 | 81.4 | 1.2 |
| 3200 | 9.2 | 2.7 | 69.2 | 18.9 |
| 6400 | 1.8 | 5.6 | 22.5 | 70.1 |
| 12800 | 0 | 2.6 | 0.4 | 97 |
| $\geq$ 25600 | 0 | 0 | 0 | 100 |

**Table 4. Division of work between different computational methods for clustered sources and uniformly distributed targets**

Figure 4 shows the same experiment with the three con-

| *Division of work (%) - Clustered sources and targets* | | | | |
|---|---|---|---|---|
| N=M | Direct Evaluation | Taylor Expansion | Hermit Expansion | Hermit + Taylor Expansion |
| 100 | 69.4 | 13.9 | 13.9 | 2.8 |
| 200 | 36.9 | 28.7 | 19.3 | 15.1 |
| 400 | 12.7 | 21.6 | 24.4 | 41.3 |
| 800 | 4.8 | 16.8 | 17.4 | 61.0 |
| 1600 | 2.8 | 15.1 | 13.0 | 69.1 |
| 3200 | 2.2 | 10.3 | 15.3 | 72.2 |
| 6400 | 0.8 | 6.8 | 9.3 | 83.1 |
| 12800 | 0.1 | 2.4 | 2.4 | 95.1 |
| $\geq$ 25600 | 0 | 0 | 2.4 | 97.6 |

**Table 5. Division of work between different computational methods for clustered sources and targets**

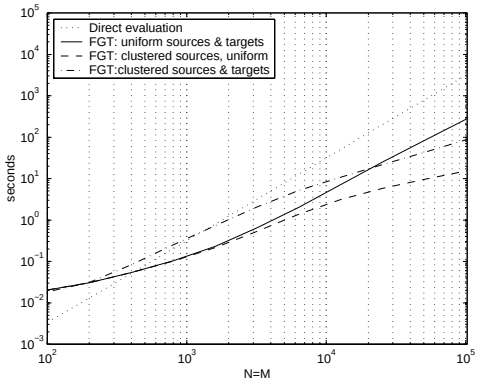


**Figure 4. Run time for 3D - FGT with different configuration of sources and targets layout**

figurations of sources and targets for the 3D case with precision set to $10^{-6}$ and $r = 0.5$. Note that the algorithm starts to utilize computations using Hermite and/or Taylor expansion only when the number of sources and/or targets in a box exceeds a break point of order $p^{d-1}$, which is higher in the 3D case. This causes the algorithm to do mostly direct evaluation for the uniform sources and targets case while Hermite and Taylor expansion computations are utilized for large number of clustered sources and/or targets.

Figure 5 shows effect of the source scale on the run time of the FGT algorithm. The figure shows the run time for three cases where sources have scale $\sigma = 0.1, 0.05, 0.01$. Typically, for color modeling applications, a suitable bandwidth is between 0.01 and 0.05. For all the cases, the sources and targets were uniformly distributed between 0 and 1. The box size scale parameter was set to $r = 0.5$ for all the cases. The run time in all the cases converges asymptotically to the linear behavior.

To show the speedup that can be achieved in the color-based body part segmentation application, we used the FGT computation framework to segment foreground regions corresponding to the two people shown in figure 7. In this experiment, there are six different color blobs corresponding to the head, torso, and bottom of each of the two people being tracked. The number of samples for each blob is re-

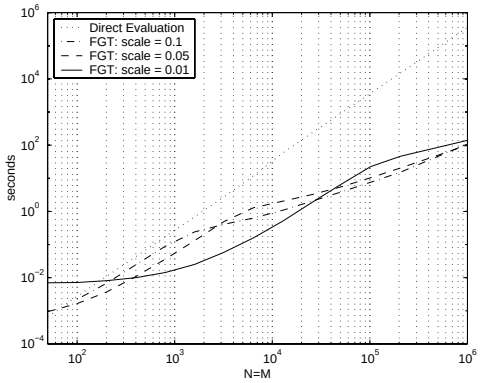


**Figure 5. Run time for 2D - FGT with uniformly distributed sources and targets with different scales.**

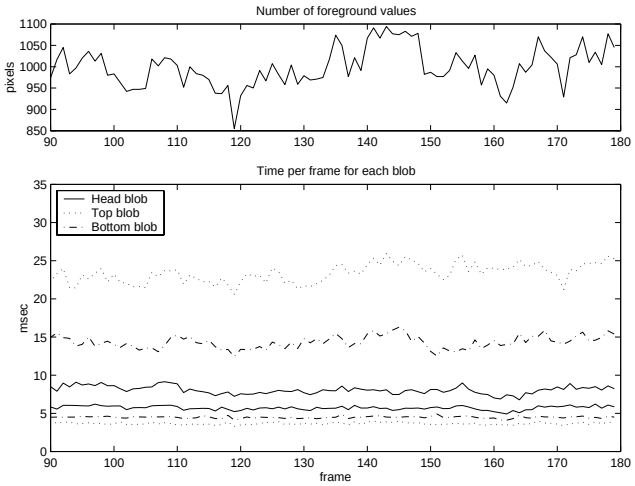


**Figure 6. Top: Number of unique foreground pixels per frame. Bottom: Run time for evaluating each of six different blobs**

stricted to 200 samples. At each new frame, the color of each pixel in the foreground is evaluated using the six blobs' color models and the segmentation is achieved as explained in section 4. There were about 7000-10000 pixels in the foreground at each frame containing about 1000 different color values each frame. Therefore, at each new frame we have six different computations, each involving 200 Gaussian sources and about 1000 targets. Figure 6 shows the run-time in milliseconds for each of color models. On the same hardware, the direct evaluation required about 65 milliseconds for each blob. On average, the speedup achieved is between 3 to 17 times per color model. Notice that although the six color models have the same number of samples, the run time is different since it depends on the way the sources and targets are clustered in the space. Figure 7 shows some of the segmentation results.

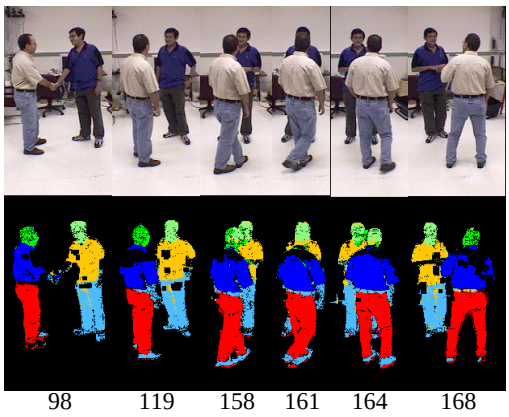


**Figure 7. Example results: Top: Original image. Bottom: Blob segmentation.**

## 6 Discussion

In this paper we presented an efficient non-parametric approach based on kernel density estimation for modeling the color distribution of a region. Kernel density estimation results in a smooth, continuous and differentiable estimate of the color density even with a small number of sample. Unlike fitting mixture of Gaussian, this approach does not assume a specific underlying distribution. We utilized the fast Gauss transform algorithm for efficient computation of color densities. The algorithm achieves a significant speedup because, typically, the color samples are clustered in the color space as well as the evaluation points (color of image pixels). This nature of the problem is the main motivation behind the use of the fast Gauss transform for computation. To adapt the algorithm to color modeling applications, a generalization of the original algorithm was developed and used to handle the case where the bandwidth is different in each color dimension.

An important reason for the lack of use of the FGT algorithm in applications is probably the fact that it is inconvenient to do so. First, it is not immediately clear what the cross-over point is when the algorithm begins to manifest its superior asymptotic complexity and offsets the preprocessing overhead. While the nominal complexity of the algorithm is $O(M + N)$, the constant multiplying it is $O(p^d)$, where $p$ is the number of retained terms in a polynomial approximation. This makes it unclear if the algorithm is useful for higher dimension applications seen in statistical pattern recognition. The fact that there is no readily available implementation to test these issues acts as a further barrier to its wide adoption.

## Acknowledgments

This work was supported in part by ARDA Video Analysis and Content Exploitation project, contract MDA 90400C2110. This work was also supported in part by NSF award number 9987944, "Textual Information Access for the Visually Impaired".

## References

[1] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Jun 1998.

[2] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 142–149, Jun 2000.

[3] R. O. Duda, D. G. Stork, and P. E. Hart. *Pattern Classification*. Wiley, John & Sons,, 2000.

[4] A. Elgammal and L. S. Davis. Probabilistic framework for segmenting people under occlusion. In *Proc. of IEEE 8th International Conference on Computer Vision*, 2001.

[5] A. Elgammal, D. Harwood, and L. S. Davis. Nonparametric background model for background subtraction. In *Proc. of 6th European Conference of Computer Vision*, 2000.

[6] P. Fieguth and D. Terzopoulos. Color-based tracking of heads and other objects at video frame rates. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Jun 1997.

[7] J. Fritsch and I. Rogina. The bucket box intersection (bbi) algorithm for fast approximative evaluation of diagonal mixture gaussians. In *Procedings of the ICASSP 96, May 2-5, Atlanta, Georgia USA*, 1996.

[8] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA, 1988.

[9] L. Greengard and J. Strain. The fast gauss transform. *SIAM J. Sci. Comput., 2*, pages 79–94, 1991.

[10] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[11] J. Martin, V. Devin, and J. Crowley. Active hand tracking. In *Proc. IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.

[12] S. J. McKenna, S. Jabri, Z. Duric, and A. Rosenfeld. Tracking groups of people. *Computer Vision and Image Understanding*, (80):42–56, 2000.

[13] Y. Raja, S. J. Mckenna, and S. Gong. Colour model selection and adaptation in dynamic scenes. In *Proc. 5th European Conference of Computer Vision*, 1998.

[14] Y. Raja, S. J. Mckenna, and S. Gong. Tracking colour objects using adaptive mixture models. *Image Vision Computing*, (17):225–231, 1999.

[15] D. W. Scott. *Mulivariate Density Estimation*. Wiley-Interscience, 1992.

[16] J. Strain. The fast gauss transform with variable scales. *SIAM J. Sci. Comput.*, 12:1131–1139, 1991.

[17] C. R. Wern, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of human body. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1997.