Efficient Kernel Density Estimation Using the Fast Gauss Transform with Applications to Color Modeling and Tracking

Ahmed Elgammal[1],

Department of Computer Science,

Rutgers University

Piscataway, NJ 08854, USA

Ramani Duraiswami, Larry S. Davis

Computer Vision Laboratory

The University of Maryland

College Park, MD 20742, USA

IEEE Transaction on Pattern Analysis and Machine Intelligence

[1]Corresponding Author

Abstract

Many vision algorithms depend on the estimation of a probability density function from observations. Kernel density estimation techniques are quite general and powerful methods for this problem, but have a significant disadvantage in that they are computationally intensive. In this paper we explore the use of kernel density estimation with the fast Gauss transform (FGT) for problems in vision. The FGT allows the summation of a mixture of M Gaussians at N evaluation points in $O(M + N)$ time as opposed to $O(MN)$ time for a naive evaluation, and can be used to considerably speed up kernel density estimation. We present applications of the technique to problems from image segmentation and tracking, and show that the algorithm allows application of advanced statistical techniques to solve practical vision problems in real time with today's computers.

key words

Statistical methods. Kernel Density Estimation. Fast Gauss Transform. Color Modeling. Tracking.

1 Introduction

Many problems in computer vision involve obtaining the probability density function (pdf) describing an observed random quantity. In general, the forms of the underlying density functions are not known. While classical parametric densities are mostly unimodal, practical computer vision problems involve multimodal densities. Further, high-dimensional densities cannot often be simply represented as the product of one-dimensional density functions. While mixture methods partially alleviate this problem, they require knowledge about the problem to choose both the form of the model (type and number of mixture functions) and the model parameters. This usually involves the solution of difficult estimation problems.

An attractive feature of nonparametric procedures is that they can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known. However, most nonparametric methods require that all of the samples be stored, or that the designer have extensive knowledge of the problem. A particular nonparametric technique that estimates the

underlying density and is quite general is the kernel density estimation technique (KDE) [1, 2]. 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, $x_i, i = 1, \cdots, n$ and α_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 mixture of Gaussian model. Here, the Gaussian is only used as a function that weights the data points. A good discussion of kernel estimation techniques can be found in [2].

The use of such an approach requires a way to efficiently evaluate the estimate $\hat{f}(x_j)$ in Equation (1) at any new point x_j. In general, given N original data samples and M points at which the density must be evaluated, the complexity is $O(NM)$ evaluations of the kernel function, multiplications and additions. For many applications in computer vision, where both real-time operation and generality of the classifier are desired, this complexity can be a significant barrier to the use of these density estimation techniques. In this paper we discuss the application of an algorithm, the Fast Gauss Transform (FGT) [3, 4], that improves the complexity of this evaluation to $O(N + M)$ operations, and apply it to computer vision problems.

The FGT is one of a class of very interesting and important new families of fast evaluation algorithms that have been developed over the past dozen years, and are revolutionizing numerical analysis. These algorithms use the fact that all computations are required only to a certain accuracy (which can be arbitrary) to speed up calculations. These algorithms enable rapid calculation of *approximations of arbitrary accuracy* to matrix-vector products of the form **Ad** where $a_{ij} = \phi(|x_i - x_j|)$ and ϕ is a particular special function. These sums first arose in applications that involve potential fields where the functions ϕ were spherical harmonics, and go by the collective name "fast multipole" methods [3]. The basic idea is to cluster the sources and target points using appropriate data structures, and to replace the sums with smaller summations that are equivalent to a given level of precision. An algorithm for evaluating Gaussian sums using this technique was developed by Greengard and Strain [4]. We will be concerned with this algorithm and its extensions in this paper. Related work also includes the work by Lambert *et al* [5] for efficient on-line computation of kernel density estimation.

Section 2 introduces some details of the Fast Gauss Transform, and our improvements to it. We present empirical demonstrations of the improvement in complexity and various tradeoffs in section 3. In section 4 we introduce the use of KDE to model the color of homogeneous regions as an example of a computer vision problem where likelihood estimation it required. In section 5 we show how the fast Gauss algorithm can be used to efficiently compute estimates for density gradient and use this for tracking people.

2 Fast Gauss Transform (FGT)

The FGT was introduced by Greengard and Strain [3, 4] for the rapid evaluation of sums of the form

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

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 σ 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 [6] to sums where σ in Equation (2) varied with the position of the target or the source, i.e., for the case where σ depends on the source location

$$S(t_i) = \sum_{j=1}^{N} f_j \, e^{-\frac{\left(s_j - t_i\right)^2}{\sigma_j^2}}, \qquad i = 1, \ldots, M. \tag{3}$$

The first application of the FGT algorithm to problems in computer vision was made in [7].

2.1 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

$$\begin{aligned} e^{-\left(\frac{t-s}{\sigma}\right)^2} &= e^{-\left(\frac{t-s_o-(s-s_o)}{\sigma}\right)^2} \\ &= e^{-\left(\frac{t-s_o}{\sigma}\right)^2} \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{s-s_o}{\sigma}\right)^n H_n\left(\frac{t-s_o}{\sigma}\right) \end{aligned} \tag{4}$$

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_o. Thus a Gaussian centered at s can be shifted to a sum of Hermite polynomials times a Gaussian, all centered at s_o. 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

$$\begin{aligned} e^{-\left(\frac{t-s}{\sigma}\right)^2} &= e^{-\left(\frac{t-t_o-(s-t_o)}{\sigma}\right)^2} \\ &\simeq \sum_{n=0}^{p} \frac{1}{n!} h_n\left(\frac{s-t_o}{\sigma}\right)\left(\frac{t-t_o}{\sigma}\right)^n \end{aligned} \tag{5}$$

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

$$h_n(t) = e^{-t^2} H_n(t). \tag{6}$$

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 centered 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 (4) in $O(Np^d)$ 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 ϵ, 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 $r\sqrt{2}\sigma_k$ where r is a box size scale parameter.

Since Gaussians decay rapidly, sources in a given box will have no effect (at a given accuracy) on targets relatively far from there sources (in terms of distance scaled by the standard deviation σ). Therefore, the effect of sources in a given box needs to be computed only for targets in close-by boxes where this evaluation window depends on the choice ofϵ and r. 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

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

they can be transformed into Hermite expansion about the center of the box using Equation (4). 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 (5)) which is then evaluated at each target in the box. These four methods are shown in table 1. The small/large break-even point (when using the expansion is more efficient than direct evaluation) 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 σ. Further details may be obtained from [4, 8]. The clustering operation is aided by the use of appropriate data structures.

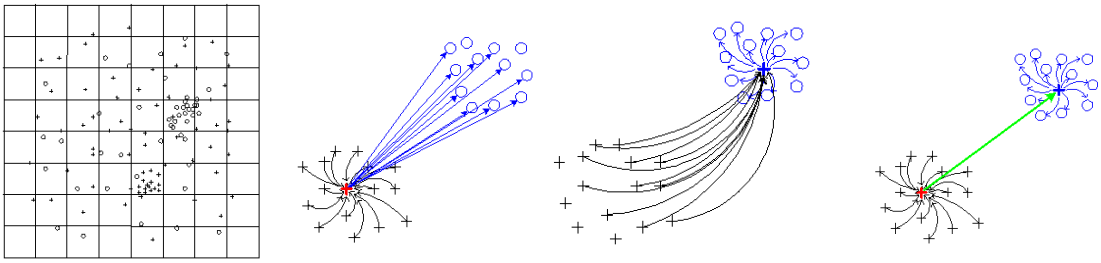


Figure 1: a) space subdivision. b) Clustered sources are converted into a Hermit expansion that is evaluated at each target. c) sources are converted to a Taylor series near a target cluster. d) Clustered sources are converted to a Hermit expansion and transformed into a Taylor series near a target cluster.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

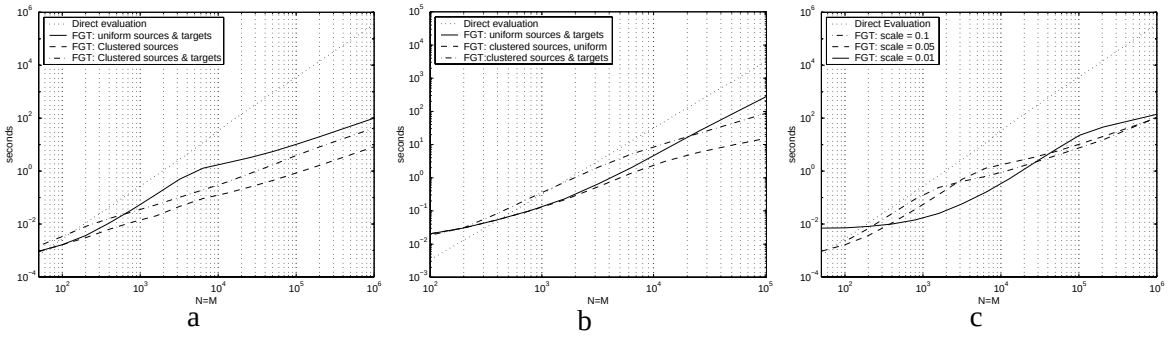


Figure 2: Run time for FGT with different configurations of sources and targets layout. (a): 2D case. (b): 3D case. (c) Effect of different scales (2D case)

3 FGT Performance Analysis

In this section we present some simulation experiments to show the performance of the FGT algorithm and the speed-up that can be achieved as well as the tradeoffs involved when using such approach for general kernel density estimation. In computer vision applications the sources and/or the targets are typically clustered in feature spaces. Therefore, we need to study the performance of the algorithm for these configurations. Figure 2 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 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 these cases are shown in Tables 2. For the uniform case, 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 and the locality of the direct evaluation based on the desired precision. 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 speed-up. For example, for $N = M = 10^5$, the algorithm gives more than 800 times speedup over direct evaluation for 10^{-4} precision. For the case of clustered sources and targets, the computation shows linear time behavior for number of sources and targets as low as 100, which yields a significant speedup.

In all these configurations, the FGT starts to outperform direct evaluation for numbers of sources and targets as low as 60-80, based on the desired accuracy. This break-even 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. This choice could be made in a problem-dependent way.

Division of work (%) - Uniform sources and targets

N=M	Direct Eval.	Taylor Expan.	Hermit Expan.	Hermit +Taylor
≤ 800	100.0	0.0	0.0	0.0
1600	96.6	1.6	1.8	0.0
3200	65.7	15.6	15	3.7
6400	5.3	18.3	16.9	59.5
12800	0.0	0.3	0.3	99.4
≥ 25600	0.0	0.0	0.0	100.0

Division of work (%) - Clustered sources and targets

N=M	Direct Eval.	Taylor Expan.	Hermit Expan.	Hermit +Taylor
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
≥ 25600	0.0	0.0	2.4	97.6

Table 2: Division of work between different computational methods for different configurations of sources and targets layout

Figure 2-b shows the same experiment with the three configurations 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 2-c 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.

4 Nonparametric Color Modeling

In this section, we investigate the use of FGT for nonparametric estimation of probability likelihood. In particular, we show the use of this approach in statistical color modelling as an example of a computer vision problem that requires likelihood estimation.

Modeling the color probability 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 [9, 10, 11, 12] with applications like tracking heads [13, 12, 14, 11], hands [15] 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. A single Gaussian model is typically used if the region is single-colored, for example for blob modeling in [9]. Fitting a mixture of Gaussians using the EM algorithm provides a way to model regions with a mixture of colors. This technique was used in [14, 11] 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 [16, 15, 10, 17] used for modeling the color of objects for different applications to overcome the previously mentioned problems with parametric models. 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. In this section we present the application of the KDE technique to the problem of modeling the color distribution of homogeneous regions and show the use of FGT for the efficient computation of color density functions.

4.1 Color Density Estimation

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 [2] as

$$\hat{P}(y) = \frac{1}{N \prod_{j=1}^{d} \sigma_j} \sum_{i=1}^{N} \prod_{j=1}^{d} K\left(\frac{y_j - x_{ij}}{\sigma_j}\right), \tag{7}$$

where the same kernel function is used in each dimension with a different bandwidth σ_j for each dimension of the color space denoted by the subscript j. Usually in color modeling two or three dimensional color spaces are used. Two dimensional chromaticity spaces, e.g., $r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}$ or 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 properties since the brightness information is preserved.

Using KDE for color modeling has many motivations. Unlike histograms, even with a small number of samples, KDE leads to a smooth, continuous and differentiable density estimate. KDE does not assume any specific underlying distribution and, theoretically, the estimate can converge to any density shape with enough samples [2, 1]. 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, KDE converges to the right density with a small number of samples. Unlike parametric fitting of a mixture of Gaussians, KDE 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 KDE is that 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

$$\hat{p}(a, b, c) = \frac{1}{N} \sum_{i=1}^{N} e^{-\frac{1}{2}(\frac{a-a_i}{\sigma_a})^2} e^{-\frac{1}{2}(\frac{b-b_i}{\sigma_b})^2} e^{-\frac{1}{2}(\frac{c-c_i}{\sigma_c})^2} \tag{8}$$

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 the noise are expected to be higher there.

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

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

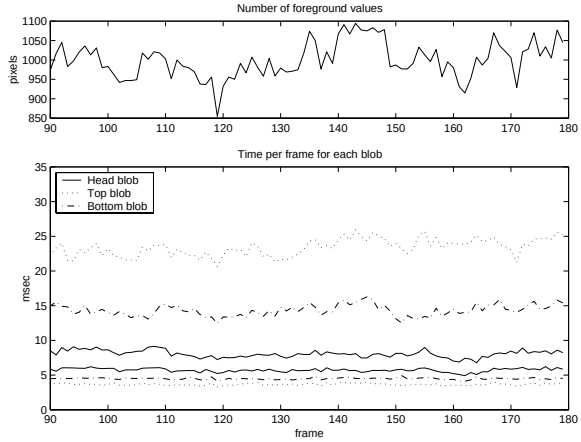
variable usually has more variance than the chromaticity variables and therefore wider kernels should be used in this dimension.

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 for each new frame for a different set of values, which is a very expensive computational task. The complexity of evaluating the summation in Equation(8) at M different locations in $O(NM)$ with N and M typically very large. The application of the fast Gauss transform, as described section 2, to this problem reduces the complexity to $O(N + M)$ and allows a very efficient framework for this computation.

4.2 Color Modeling Experiments

We used the color modeling approach described in section 4.1 as well as the FGT computation to segment foreground regions corresponding to tracked people. Each blob is modelled by acquiring a sample to represent the color distribution and the segmentation is performed by maximum likelihood classification. The details about the tracking and the segmentation approach can be found in [18, 19]. To show the speedup that can be achieved in this application, we present the results of tracking experiment where we used the FGT computation framework to segment foreground regions corresponding to the two people shown in figure 3. In this experiment, there are six different color blobs corresponding to the head, torso, and bottom of each of the two people being tracked. 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. Bandwidths for the kernels were set to 0.01, 0.01, 0.05 for each of the color space dimensions respectively. The number of samples for each blob is restricted to 200 samples. At each new frame, the likelihood of the color of each pixel in the foreground is evaluated given the six blobs' color models and the segmentation is achieved using maximum likelihood classification. There were about 7000-10000 pixels in the foreground of each frame containing about 1000 different color values each frame. Therefore, for each new frame we have six different computations, each involving 200 Gaussian sources and about 1000 targets. The precision was set to $\epsilon = 1e - 4$. Figure 3-a shows the run-time in mil-

liseconds 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 3-b shows some of the segmentation results.



(a)

(b)

Figure 3: (a) Top: Number of unique foreground pixels per frame. Bottom: Run time for evaluating each of six different blobs. (b)Example results: Top: Original image. Bottom: Blob segmentation.

5 Density Gradient Computation

In this section we show how the FGT algorithm can be used to efficiently compute estimates for the gradient of the pdf as well as estimates for the density itself. Estimation of the density gradient is useful in applications which use gradient based minimization such as the mean-shift algorithm [20, 21, 10, 22].

In many computer vision application it is desired to compute the likelihood $P(x \mid h)$ of a certain observation x given a hypothesis h of the state of the system. For example, in a Bayesian tracking framework, it is desired to model the observation as a probabilistic function of the tracked object state and, given different hypothesis about the state, it is desired to evaluate these hypothesis by estimating the observation likelihood given each hypothesis. Typically in such probabilistic framework, the observation likelihood computations involve an objective function of the form

$$\psi(h) = \int_R \phi(P(x \mid h))dx \tag{9}$$

where the integration is performed over a set of observation points, x (e.g., a region in the image, an object contour, a set of tracked feature points, etc.) In such frameworks, the optimization of such objective function involves computation of the gradient ∇P of the likelihood function. Given a nonparametric representation of the density function on the form of Equation (1), estimation of the density gradient ∇P would involve computations of summations of the form

$$\sum_i x_i K(x - x_i) \tag{10}$$

This form of summation appears also in many computer vision applications which use the mean shift algorithm to reach the density mode, for example for clustering [20, 21, 22] and for tracking [10]. Efficient computation of this form of summation can be achieved using the fast Gauss transform algorithm. In this case the the term x_i in the summation can be represented as the strength associated with the source i as in Equation (2).

5.1 Tracking Application Experiments

To illustrate the use of FGT in this line of application, we use it in a tracking application that involves computations of the likelihood function and its gradient. The tracked object appearance is modelled by acquiring sample pixels from the object to represent the joint color-spatial distribution. Given this representation, the likelihood of any observation is estimated using KDE in the joint color-spatial space. Given a hypothesis h for object location in the image, the likelihood of this hypothesis is evaluated using an objective function similar to Equation (9) where we used $\phi(\cdot) = \log(\cdot)$, i.e., log likelihood function. The object tracking is formalized as finding the maxima of the objective function at each new frame. To perform this optimization, a gradient based approach is used where at each iteration it is required to perform many summations of the form of Equation (1) to estimate the likelihood function and summations of the form of Equation (10) to evaluating the gradient of the objective function. The use of the FGT facilitates efficient computation of such summations which leads to a real-time performance of the tracker.

To achieve efficient implementation, we developed a two phase version of the fast Gauss algorithm where in the first phase all the source data structures and expansions are calculated from the samples. Then, for each new frame, these structures and expansions are reused to evaluate the new targets. Since targets (evaluation points) are expected to be similar from frame to frame, the results of each evaluation are kept in a look-up table to avoid repeating computations.

Figure 4: Tracking result

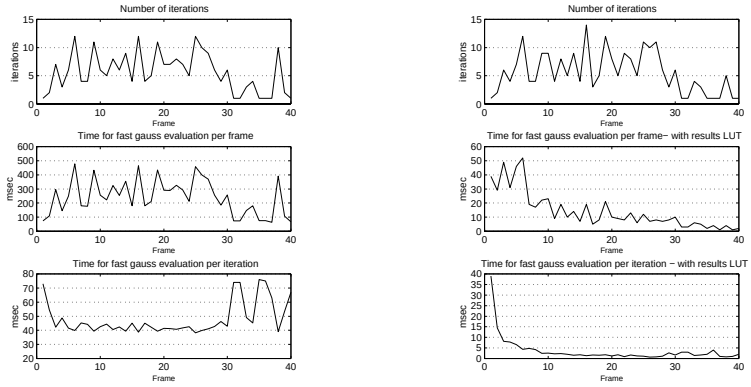


Figure 5: Performance of the tracker using Fast Gauss. Top: number of iterations per frame. Middle: run time per frame. Bottom: run time per iteration. Right: Performance using FGT with results look up tables as described in the text.

Figure 4 shows four frames from the tracking result for a sequence. The sequence contains 40 frames recorded at about 5 frames per second. The tracker successfully locates the target at this low frame rate. Notice that the target changes his orientation during the tracking. Figure 5-left shows the performance of the tracker on this sequence using Fast Gauss only (no look-up tables). The average run-time per frame is about 200-300 msec. Figure 5-right shows the performance of the tracker with look up tables used in conjunction with fast Gauss transform to save repeated computation. In this case, the run time decreases with time and the average run time per frame is less than 20 msec. This is more than a 10 times speed up per iteration, and is suitable for video frame-rate implementation.

6 Conclusion

In this paper we investigate the use of Fast Gauss Transform for efficient computation of KDE techniques for computer vision applications. The FGT exploits the clustering of the data to achieve efficient computation of Gaussian sums. This property makes it suitable for computer vision appli-

cations by allowing the use of KDE techniques to estimate probability density functions, since in such applications the data and evaluation points are usually naturally clustered. We presented the use of FGT to efficiently compute nonparametric estimate of probability densities and their gradient and used this in color modelling and tracking problems. In this application, the use of FGT results in real-time performance. The FGT can be useful to many vision problems that use kernel density estimation, and as such our paper serves to introduce the algorithm to this community. Kernel density estimation is a better way to estimate the densities in comparison with classical parametric methods in cases where the underlying distribution are not known, or when it is hard to specify the right model. Also kernel density estimation is preferred over the use of histogram techniques in certain applications, since a smooth and continuous estimate of the density is obtained. The version of the FGT algorithm that we presented here is a slight generalization of the original algorithm, as it uses a diagonal covariance matrix instead of a scalar variance [3, 4].

References

[1] Richard O. Duda, David G. Stork, and Peter E. Hart, *Pattern Classification*, Wiley, John & Sons,, 2000.

[2] David W. Scott, *Mulivariate Density Estimation*, Wiley-Interscience, 1992.

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

[4] Leslie Greengard and John Strain, "The fast gauss transform," *SIAM J. Sci. Comput., 2*, pp. 79–94, 1991.

[5] C.G. Lambert, S.E. Harrington, C.R. Harvey, and A. Glodjo, "Efficient on-line nonparametric kernel density estimation," *Algorithmica*, , no. 25, pp. 37–57, 1999.

[6] John Strain, "The fast gauss transform with variable scales," *SIAM J. Sci. Comput.*, vol. 12, pp. 1131–1139, 1991.

[7] Ahmed Elgammal, Ramani Duraiswami, and Larry S. Davis, "Efficient computation of kernel density estimation using fast gauss transform with applications for segmentation and tracking," in *Proc. of IEEE 2nd Int. workshop on Statistical and Computational Theories of Vision, Vancouver, CA, July 2001*, 2001.

[8] Leslie Greengard and Xiaobai Sun, "A new version of the fast gauss transform," *Documenta Mathematica, Extra Volume ICM*, vol. III, pp. 575–584, 1998.

[9] Christopher Richard Wern, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland, "Pfinder: Real-time tracking of human body," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1997.

[10] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Real-time tracking of non-rigid objects using mean shift," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2000, vol. 2, pp. 142–149.

[11] Yogesh Raja, Stephen J. Mckenna, and Shaogang Gong, "Tracking colour objects using adaptive mixture models," *Image Vision Computing*, , no. 17, pp. 225–231, 1999.

[12] Paul Fieguth and Demetri 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.

[13] Stan Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Jun 1998.

[14] Yogesh Raja, Stephen J. Mckenna, and Shaogang Gong, "Colour model selection and adaptation in dynamic scenes," in *Proc. 5th European Conference of Computer Vision*, 1998.

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

[16] Stephen J. McKenna, Sumer Jabri, Zoran Duric, and Azriel Rosenfeld, "Tracking groups of people," *Computer Vision and Image Understanding*, , no. 80, pp. 42–56, 2000.

[17] Michael J. Jones and James M. Rehg, "Statistical color models with application to skin detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[18] Ahmed Elgammal and Larry S. Davis, "Probabilistic framework for segmenting people under occlusion," in *Proc. of IEEE 8th International Conference on Computer Vision*, 2001.

[19] Ahmed Elgammal, Ramani Duraiswami, and Larry S. Davis, "Efficient non-parametric adaptive color modeling using fast gauss transform," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Dec 2001.

[20] Yizong Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, Aug 1995.

[21] Yizong Cheng and K.S. Fu, "Conceptual clustering in knowledge organization," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 592–598, 1985.

[22] Dorin Comaniciu and Peter Meer, "Mean shift analysis and applications," in *IEEE 7th International Conference on Computer Vision*, Sep 1999, vol. 2, pp. 1197–1203.