Real-Time Kernel-Based Tracking in Joint Feature-Spatial Spaces

Changjiang Yang, Ramani Duraiswami, Ahmed Elgammal[†] and Larry Davis
Perceptual Interfaces and Reality Laboratory, UMIACS, University of Maryland, College Park, MD
[†] Dept. of Computer Science, Rutgers University, Piscataway, NJ
{yangcj,ramani,lsd}@umiacs.umd.edu [†]elgammal@cs.rutgers.edu

Abstract

An object tracking algorithm that uses a novel simple symmetric similarity function between spatially-smoothed kernel-density estimates of the model and target distributions is proposed and tested. The similarity measure is based on the expectation of the density estimates over the model or target images. The density is estimated using radial-basis kernel functions which measure the affinity between points and provide a better outlier rejection property. The mean-shift algorithm is used to track objects by iteratively maximizing this similarity function. To alleviate the quadratic complexity of the density estimation, we employ Gaussian kernels and the fast Gauss transform to reduce the computations to linear order. This leads to a very efficient and robust nonparametric tracking algorithm. The proposed algorithm is tested with several image sequences and shown to achieve robust and reliable real-time tracking.

1 Introduction

Object tracking is a common vision task to find and follow moving objects between consecutive frames, which is important for many computer vision applications such as human-computer interaction, surveillance, smart rooms and medical imaging. A verity of tracking algorithms have been proposed and implemented to overcome difficulties that arise from noise, occlusion, clutter, and changes in the foreground objects being tracked or in the background environment. Region-based methods typically align the tracked regions between the successive frames by minimizing a cost function [17, 2, 16]. Feature-based approaches extract features (such as intensity, colors, edges, contours) and use them to establish correspondence between model images and target images [18, 12, 8]. Model-based tracking algorithms incorporate *a priori* information about the tracked objects to develop representations such as projected shape, skin complexion, body blobs, kinematic skeleton and silhouette [31, 29, 5, 26, 6]. Appearance-based approaches apply recognition algorithms to learn the objects either in the eigenspace or in the kernel space. The trained systems are used to search for the targets in image sequences [4, 1, 28].

Many of these approaches employ a statistical description of the region or the pixels to perform the tracking. The tracked objected can be described using either parametric or nonparametric representations. In a parametric framework, the objects or persons are typically fitted by Gaussian models or via a mixture of Gaussians [29]. A nonlinear estimation problem has to be solved to obtain the number of Gaussians and their parameters. However, the common parametric forms rarely fit the multimodal complex densities in practice, and are problematic when the fitted distributions are multidimensional. In contrast, nonparametric density estimation techniques [20, 10] allow representation of complex densities just by using the data. They have been successfully applied to object tracking [8, 11]. The conceptually simplest density estimation approach is to build a histogram and use it to establish the correspondences between the model image and the target image [12, 8]. The histogram is very flexible and robust for tracking deformable and nonrigid objects. However histogramming is only suitable for low dimensional spaces because as the number of dimensions increase, the number of bins grow exponentially. In contrast, given sufficient samples, kernel density estimation works well both in low and high dimensions, and has successfully been applied to tracking [11].

To build a matching of the objects across frames, many tracking algorithms use measures of "similarity" or "distance" between the two regions, feature vectors, or distributions. The sum of squared differences (SSD) assumes "constant brightness" from frame to frame [17, 16], which is liable to fail with noise, deformation or occlusion. The Kullback-Leibler divergence, Hellinger's distance and other probabilistic distance functions are employed to measure the similarity between frames [8, 11]. All these information-theoretic distance measures require an estimate of the conditional probability density function and its numerical integration. When such measures are used by the mean shift algorithm or other gradient based methods, the evaluation of their gradient functions is often involved, which is numerically unstable and computationally expensive, especially in high dimensions.

The mean shift algorithm, originally invented by Fukunaga and Hostetler [13], was successfully applied to computer vision applications by Comaniciu [7, 8]. It is an effec-

tive gradient-based optimization technique for finding the target location but has two difficulties. First, the kernel-based densities are expensive to evaluate. Second, the classically used similarity measures between the distributions in the model and target images are unwieldy, and computationally even more expensive to evaluate than the density.

In this paper we address these difficulties by presenting an object tracking algorithm that uses a simple symmetric similarity function between kernel density estimates of the model and target distributions. In our formulation we use the joint spatial-feature formulation of [11], and consider both feature vectors and pixel locations as probabilistic random variables. The density is estimated in the joint feature-spatial space using radial-basis kernel functions which measure the affinity between points and provide a better outlier rejection property. The joint feature-spatial spaces impose a probabilistic spatial constraint on the tracked region and provide an accurate representation of the tracked objects. The similarity measure we use is symmetric and is the expectation of the density estimates centered on the model (target) image over the target (model) image. The mean shift algorithm is used to track objects by iteratively maximizing this similarity function. To alleviate the quadratic complexity of the density estimation, we employ Gaussian kernels and the improved fast Gauss transform (FGT) [30] to reduce the computations to linear order.

2 Image Representation

The distribution of features and pixels of the tracked objects are represented as probability distribution functions over joint feature-spatial spaces. Pixels in the spatial domain are mapped into points in a multidimensional feature space. Such a mapping is used to characterize the tracked objects and is usually nonlinear. A good feature space will greatly relieve difficulties in distinguishing objects from the background and provide tolerance to the noise [25]. The most commonly used features are image intensity, colors, edges, texture, wavelet filter response, etc.. The associated spatial space enhances the feature space by imposing the constraint of spatially continuity in a statistical way.

Suppose we are given two images, with one designated as the "model image" that includes the tracked objects, while the other is the "target image" in which we need to find the objects. The sample points in the model image are denoted by $I_x = \{\mathbf{x}_i, \mathbf{u}_i\}_{i=1}^N$, where $\mathbf{x}_i$ is the 2D coordinates and $\mathbf{u}_i$ is the corresponding feature vector. The sample points in the target image are $I_y = \{\mathbf{y}_j, \mathbf{v}_j\}_{j=1}^M$, encoding the the 2D coordinates and the corresponding feature vector.

The structure of the joint feature-spatial spaces is generally complex and can be analyzed only by nonparametric methods. The probability density function of the joint feature-spatial spaces can be estimated from the sample points by the kernel density estimation [20, 10]. In pattern recognition and computer vision, the following radial-basis function (RBF) kernel (symmetric, positive-definite) is widely used [20, 21, 24, 22]:

$$k(\mathbf{x}, \mathbf{x}') = k(\Big\| \frac{\mathbf{x} - \mathbf{x}'}{h} \Big\|^2), \tag{1}$$

where $k(x)$ is the *profile* of the kernel, and h is the *bandwidth*. The important RBF kernel — Gaussian kernel in d dimensions is

$$k(\mathbf{x}, \mathbf{x}') = \frac{1}{(2\pi)^{d/2} h^d} e^{-\|\mathbf{x}-\mathbf{x}'\|^2/2h^2}, \tag{2}$$

which is supported by many results from psychology and learning theory [21, 22].

Given the sample points and the RBF kernel function $k(x)$, the probability density function of the model image can be estimated in the feature space as

$$\hat{p}_x(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} k(\Big\| \frac{\mathbf{u} - \mathbf{u}_i}{h} \Big\|^2). \tag{3}$$

Usually the exterior points of a region are less reliable than the interior points. To combat noise and improve robustness, we regularize the probability density function (3) by smoothing it with another RBF kernel $w(x)$ in the spatial domain [8]. Then the spatially-smoothed probability density function of the model image centered at $(\mathbf{x}, \mathbf{u})$ can be estimated in the joint feature-spatial space as

$$\hat{p}_x(\mathbf{x}, \mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} w(\Big\| \frac{\mathbf{x} - \mathbf{x}_i}{\sigma} \Big\|^2) k(\Big\| \frac{\mathbf{u} - \mathbf{u}_i}{h} \Big\|^2). \tag{4}$$

Similarly the spatially-smoothed probability density function of the target image centered at $(\mathbf{y}, \mathbf{v})$ can be estimated as

$$\hat{p}_y(\mathbf{y}, \mathbf{v}) = \frac{1}{M} \sum_{j=1}^{M} w(\Big\| \frac{\mathbf{y} - \mathbf{y}_j}{\sigma} \Big\|^2) k(\Big\| \frac{\mathbf{v} - \mathbf{v}_j}{h} \Big\|^2), \tag{5}$$

where σ and h are the bandwidths in the spatial and feature spaces. We also absorb the normalization constants into the kernels for convenience.

3 Similarity Between Distributions

Once we have the probability density functions of two distributions, we need a similarity (or dissimilarity) function to measure the affinity between groups of points or distributions. There are many similarity measures between distributions proposed in the statistics and pattern recognition [9, 27, 23]. A conceptually simple similarity measure is the sum of squared differences (SSD) [17, 16]. Several probabilistic distance measures have been proposed [9, 27]

and some have been applied to tracking. In [8], the Bhattacharyya coefficient is employed as the similarity measure. The Kullback-Leibler divergence is used as similarity measure in [11]. All of these information-theoretic distance measures require an estimate of the probability density function and its numerical integration. Their gradient functions are often involved and numerically unstable, especially in high dimensions.

In this paper, we define the similarity between two distributions as the expectation of the spatially-smoothed density estimates over the model or target image. Suppose we have two distributions with samples $I_x = \{\mathbf{x}_i, \mathbf{u}_i\}_{i=1}^N$ and $I_y = \{\mathbf{y}_j, \mathbf{v}_j\}_{j=1}^M$, where $\mathbf{x}_i$ and $\mathbf{y}_j$ are 2D coordinates, $\mathbf{u}_i$ and $\mathbf{v}_j$ are feature vectors, the center of sample points in the model image is $\mathbf{x}_*$, and the current center of the target points is $\mathbf{y}$, the spatially-smoothed similarity between I_x and I_y is

$$J(I_x, I_y) = \frac{1}{M}\sum_{j=1}^{M} w(\left\|\frac{\mathbf{y}-\mathbf{y}_j}{\sigma}\right\|^2)\hat{p}_x(\mathbf{x}_*, \mathbf{v_j}), \quad (6)$$

which can be rewritten as

$$J(I_x, I_y) \quad = \quad (7)$$
$$\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M} w(\left\|\frac{\mathbf{x}_*-\mathbf{x}_i}{\sigma}\right\|^2)k(\left\|\frac{\mathbf{u}_i-\mathbf{v}_j}{h}\right\|^2)w(\left\|\frac{\mathbf{y}-\mathbf{y}_j}{\sigma}\right\|^2).$$

The similarity function (7) can be interpreted as the expectation of the spatially-smoothed density estimates over the model image.

We normalize the data along each dimensions and use fixed bandwidth for simplicity. Variable and adaptive bandwidth can be applied to the similarity function (7) and will give better performance. The spatial smoothing can also be improved by considering the background information and the shape of the region.

The similarity measure (7) is symmetric and bounded by zero and one, but violates the triangle inequality which means the similarity measure is non-metric. Often distance functions that are robust to outliers or to noise disobey the triangle inequality [19].

If we set $\sigma \to 0$, then $w(x)$ becomes a delta function. The similarity function reduces to the affinity between $\mathbf{x}_*$ and $\mathbf{y}$ in the feature space. If we set $h \to 0$, then $k(x)$ becomes a "look-up" table, in the sense that only the exactly matched pairs in the feature space are counted in the similarity function.

The similarity measure (7) is directly computed from the sample points. The affinities between all pairs of sample points are considered based on their distances and exact correspondence is not necessary, which is more robust than the template matching or sum of squared differences (SSD). Furthermore, the sample points are sparse in the high dimensional feature space. It is difficult to get an accurate density estimation or histogram which will cause the similarity measures such as Kullback-Leibler divergence and Bhattacharyya coefficient to become unstable. The effectiveness of similarity measure (7) in high dimensional space is well explained by the theories developed for support vector machines [24, 22].

The similarity function (7) is non-metric. However, it can be shown that its negative natural logarithm

$$L(I_x, I_y) = -\log J(I_x, I_y) \quad (8)$$

is a probabilistic distance, provided we have sufficient samples, so that the kernel density estimate converges to the true probability density function [9].

4 Mean-Shift Based Target Localization

Once we have the similarity measure between the model image and target image, we can find the target location in the target image by maximizing the similarity measure (7) or equivalently minimizing the distance (8) with respect to the variable $\mathbf{y}$. There are many techniques for searching for the optimal solution. Since the similarity function (7) is smooth and differentiable, and the displacement between the successive frames is small, we adopt the mean-shift algorithm [7] which has already proved successful in many computer vision applications [7, 8].

The gradient of the distance function (8) with respect to the vector $\mathbf{y}$ is

$$\nabla L(\mathbf{y}) = -\frac{\nabla J(\mathbf{y})}{J(\mathbf{y})}, \quad (9)$$

where

$$\nabla J(\mathbf{y}) = \frac{2}{MN\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M}(\mathbf{y}-\mathbf{y}_j)w_i k_{ij} w'(\left\|\frac{\mathbf{y}-\mathbf{y}_j}{\sigma}\right\|^2), \quad (10)$$

and $w_i = w(\left\|\frac{\mathbf{x}_*-\mathbf{x}_i}{\sigma}\right\|^2)$ and $k_{ij} = k(\left\|\frac{\mathbf{u}_i-\mathbf{v}_j}{h}\right\|^2)$.

The *mean shift* of the smoothed similarity function $\mathbf{m}(\mathbf{y})$ is

$$\nabla L(\mathbf{y}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\mathbf{y}_j w_i k_{ij} g(\left\|\frac{\mathbf{y}-\mathbf{y}_j}{\sigma}\right\|^2)}{\sum_{i=1}^{N}\sum_{j=1}^{M} w_i k_{ij} g(\left\|\frac{\mathbf{y}-\mathbf{y}_j}{\sigma}\right\|^2)} - \mathbf{y}, \quad (11)$$

where $g(x) = -w'(x)$ is also the profile of a RBF kernel.

Given the sample points $\{\mathbf{x}_i, \mathbf{u}_i\}_{i=1}^N$ centered at $\mathbf{x}_*$ in the model image, and $\{\mathbf{y}_j, \mathbf{v}_j\}_{j=1}^M$ centered at the current position $\hat{\mathbf{y}}_0$ in the current target image, the object tracking based on the mean-shift algorithm is an iterative procedure which recursively moves the current position $\hat{\mathbf{y}}_0$ to the new position $\hat{\mathbf{y}}_1$ until reaching the density mode according to

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\mathbf{y}_j w_i k_{ij} g(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{y}_j}{\sigma}\right\|^2)}{\sum_{i=1}^{N}\sum_{j=1}^{M} w_i k_{ij} g(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{y}_j}{\sigma}\right\|^2)}. \quad (12)$$

From equation (12), we observe that the new position is the weighted centroid of the sample points $\{\mathbf{y}_j\}_{j=1}^M$. The weights consist of three parts: the first part is the weight from the kernel function $w(x)$ which assigns smaller weights to points farther away from center point $\mathbf{x}_*$ in the model image; the second part is the weight from the kernel function $k(x)$ which encourages pairs of similar vectors in feature space and penalizes mismatched pairs; the third part is the weight from kernel function $g(x)$ which favors neighboring points of $\hat{\mathbf{y}}_0$ and vanishes at the peripheral points.

Since the kernel functions we used are convex and smooth RBFs, it can be proved that the above mean-shift procedure converges and that the similarity measure (7) monotonically increases as in [8].

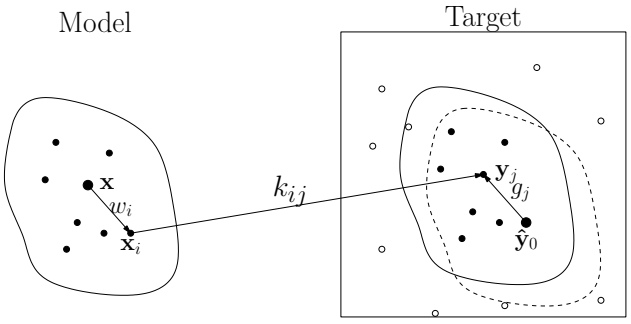

Figure 1: The mean-shift based tracking procedure. At each step of the mean-shift procedure, the new location of the target is the weighted centroid of the points within the old region (dashed line). The weight is a combination of w_i, k_{ij} and g_j.

5 Speedup by the Improved FGT

The computational complexity per frame in the above algorithm is $O(PMN)$, where P is the average number of iterations per frame, M and N are the number of sample points in target image and model image respectively. Typically the average number of iterations per frame P is less than ten and $M \approx N$. Then the order of the computational complexity is quadratic. While the above simple algorithm runs at real-time frame rate when the number of points N is small, say up to 100, it will slow down quadratically with the number of sample points.

From now on, we use the Gaussian kernel (2) in the above tracking algorithm. We apply the fast Gauss transform (FGT) [15, 30] to the tracking algorithm to reduce its computational complexity from quadratic order to linear order.

Since the derivative of Gaussian kernel is still a Gaussian, the mean shift based object tracking with the Gaussian kernel is

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^N \sum_{j=1}^M \mathbf{y}_j w_i k_{ij} e^{-\|\hat{\mathbf{y}}_0 - \mathbf{y}_j\|^2/\sigma^2}}{\sum_{i=1}^N \sum_{j=1}^M w_i k_{ij} e^{-\|\hat{\mathbf{y}}_0 - \mathbf{y}_j\|^2/\sigma^2}}. \tag{13}$$

For the sake of convenience, we drop the constant 2 in the Gaussian kernel and absorb it into the bandwidth. By exchanging the order of the summations in (13), we have

$$\hat{\mathbf{y}}_1 = \frac{\sum_{j=1}^M \mathbf{y}_j f(\mathbf{y}_j)}{\sum_{j=1}^M f(\mathbf{y}_j)}, \tag{14}$$

where

$$f(\mathbf{y}_j) = \sum_{i=1}^N e^{-\|\mathbf{x}_i - \mathbf{x}_*\|^2/\sigma^2} e^{-\|\mathbf{u}_i - \mathbf{v}_j\|^2/h^2} e^{-\|\mathbf{y}_j - \hat{\mathbf{y}}_0\|^2/\sigma^2} \tag{15}$$

is a *discrete Gauss transform* of $\mathbf{y}_j$ for $j = 1, \ldots, M$. The vectors $\mathbf{u}_i$ are called "sources" and $\mathbf{v}_j$ are called "targets".

The computational complexity of a direct evaluation of the discrete Gauss transform (15) requires $O(MN)$ operations. In low-dimensional spaces, the computational complexity has been reduced by Greengard and Strain [15] to $C \cdot (M+N)$ using the fast Gauss transform, where constant factor C depends on the precision required and dimensionality.

The fast Gauss transform is based on a divide-and-conquer strategy. The source points are subdivided into uniform boxes. The contributions from the sources are collected to the centers of the boxes by means of Hermite expansions and Taylor series. Then the contributions are distributed to each target point from the box centers by consolidating the expansions at each target point.

Although the fast Gauss transform achieved great success in low dimensions, the performance in higher dimensions is poor. The reason is that the fast Gauss transform is originally designed for solving the heat equation whose dimension is up to three. There are two major drawbacks in the original FGT. One is that the number of boxes in FGT grows exponentially with dimensionality. The other is that the number of terms in the expansions grows exponentially with the dimensionality, too. So the performance of the FGT degrades exponentially with the dimensionality.

An improved version of fast Gauss transform was presented in [30] to deal with the above serious drawbacks of the FGT in higher dimensions. First, multivariate Taylor expansions are used to replace the Hermite expansion, which reduces the number of expansion terms from $O(p^d)$ to $O(d^p)$ asymptotically, where p is the truncation order and d is the dimensionality. In higher dimensions, p typically is small, so this is a substantial reduction.

Another technique used in [30] is the use of the k-center algorithm [14, 3] which adaptively partitions the space according to the distribution of the points and obtains a much more compact space subdivision than the uniform box scheme in the original FGT [15].

We implemented the improved FGT (IFGT) algorithm of [30] and observed that it achieves success in spaces of

dimension up to ten (see Figure 2), which is sufficient for our tracking applications. We applied this method as a black box to evaluate the Gauss transform (15).

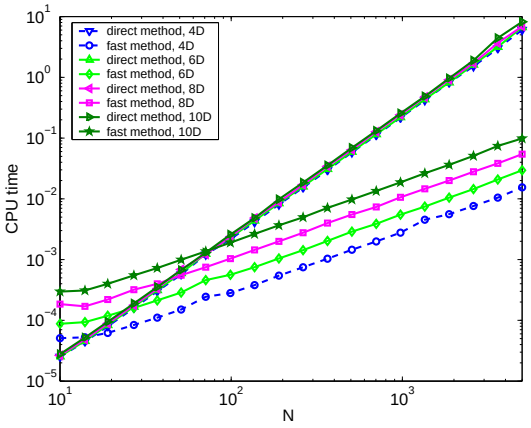

Figure 2: The CPU time in seconds of the improved FGT with single center and $p = 5$. The source and target points are uniformly generated in a unit hypercube in dimensions $4, 6, 8, 10$ and weights are uniformly distributed between 0 and 1. The bandwidth is set to 1 and maximum absolute errors are under 10^{-5}. For all dimensions the IFGT is faster than direct evaluation for $N \sim 60$.

6 Experimental Results

In this section, we present some real-time object tracking results using the proposed algorithm. In the first two experiments, the RGB color space is used as the feature space, and in the third one, the RGB color space plus 2D image gradient is used. The 2D spatial domain is combined to the feature space. The Gaussian kernel (2) is used in all the experiments. The algorithm is implemented in C++ with Matlab interface and runs on a 900MHZ PIII PC.

We first compare results on 2 clips that were used in [8]. The first clip is the *Football* sequence which has 154 frames of size 352×240. The tracking algorithm is initialized with a manually selected region in frame 0 of size 60×60. The bandwidth in the feature space is $h = 20$ and in the spatial domain is $\sigma = 10$. The algorithm tracks the player reliably with partial occlusion, clutter, blurring and compression noise (see Figure 3). The number of mean-shift iterations is shown in Figure 4. The average number of the iterations is 2.8609 per frame and the average processing time per frame: $0.0291s$. The number of iterations required in each frame for this sequence are shown in Figure 4. The number of iterations required in [8] for each corresponding frame (see Figure 2 in [8]) is larger. This shows that our similarity measure (7) functions is as good or better than the Bhattacharyya coefficient used in [8].

The second experiment uses the *Ball* sequence. If we blindly apply the tracking algorithm, it will either track the background if a large region is used, or lose the ball if the tracking region is small and the movement is large. We utilize the background information and assume a mask about the tracked object is available. We initialize the model with a region in frame 3 size of 48×48. The bandwidths are $(h, \sigma) = (18, 12)$. We only keep the foreground pixels in the model and run the algorithm as in the previous experiment. The algorithm reliably and accurately tracks the ball with average number of iteration 2.7679 and average processing time per frame $0.0169s$. In contrast, to successfully track this sequence, in [8] a background-weighted histogram was employed. The tracking results shown in Figure 5 are more accurate than those in [8].

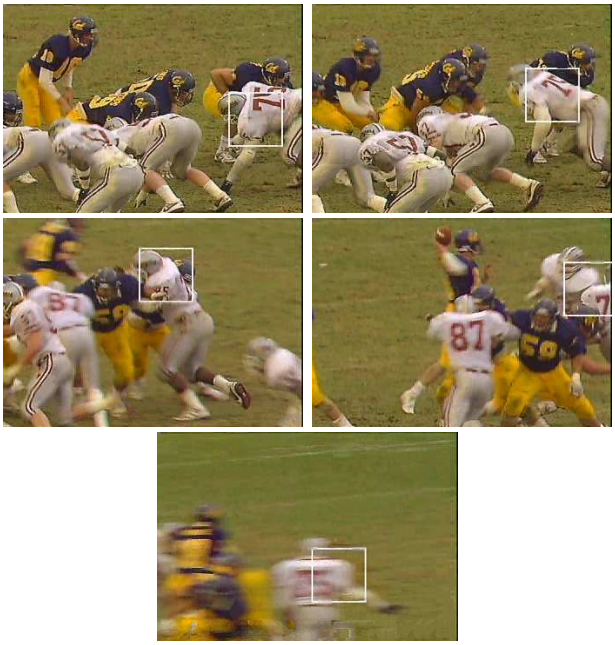

Figure 3: Tracking results of the *Football* sequence. Frames 30, 75, 105, 140 and 150 are displayed.

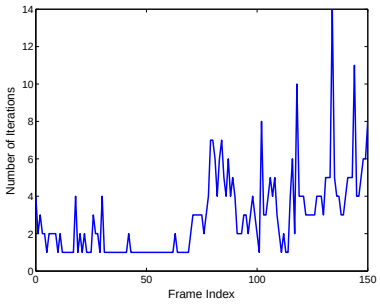

Figure 4: The number of mean-shift iterations *w.r.t.* the frame index for the *Football* sequence.

If more features are available, we can conveniently integrate the feature information into high dimensional feature-spatial spaces. In the third experiment a more complex clip is taken. In order to track a face with changing appearance and complex background, we use both the RGB color space and 2D image gradients as features. The image gradients are the horizontal and vertical image gradients of the grayscale image obtained using the Sobel operator. We ini-

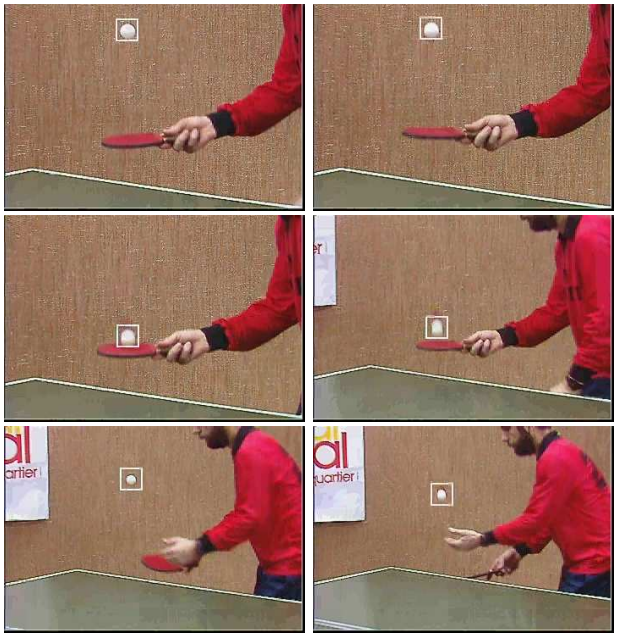

Figure 5: Tracking results of the *Ball* sequence. Frames 3, 16, 26, 40, 48 and 51 are displayed.

tialize the model with a region in frame 0 size of 24×24. The bandwidths are $(h, \sigma) = (25, 12)$. The average number of iteration per frame is 2.1414 and average processing time per frame is $0.0044s$. The algorithm reliably tracks the face and results are shown in Figure 6.

Figure 6: Tracking results of the *Walking* sequence. Frames 4, 19, 50, 99, 166 and 187 are displayed.

The algorithm has been implemented in a real-time system that runs on a laptop. If accepted, it will be demonstrated at the conference.

7 Discussion and Conclusions

In this paper we proposed a novel simple symmetric similarity function between spatially-smoothed kernel-density estimates of the model and target distributions for object tracking. The similarity measure is based on the expectation of the density estimates over the model or target image. The well-known radial-basis kernel functions are used to measure the affinity between points and provide a better outlier rejection property. To track the objects, the similarity function is maximized using the mean-shift algorithm to iteratively find the local mode of the function. The tracking algorithm based on this similarity function is very simple and we attach the actual Matlab code for tracking in the Appendix (without the fast Gauss transform).

Since the similarity measure is an average taken over all pairs of the pixel between two distributions, the computational complexity is quadratic. To alleviate the quadratic complexity, we employ Gaussian kernels and the fast Gauss transform to reduce the computations to linear order. This leads to a very efficient and robust nonparametric tracking algorithm. It also very convenient for integration of the background information and generalization to high dimensional feature space. The similarity is directly based on the kernel density estimation, there is no stability and singularity problems which perplex the information-theory based distance measures. In this paper we use a fixed bandwidth which by no means is optimal for the performance. The variable and adaptive bandwidth selection will be studied in the future work.

Appendix: Matlab Code for Tracking

Attached below is actual Matlab code that implements the tracking algorithm with the similarity function (7). Note that this Matlab code does not include the improved fast Gauss transform (IFGT). This code achieved tracking speeds of about $2.5s$ per frame for a region of size 12×12. With the inclusion of the IFGT the tracking speeds are substantially faster.

```
function  [newpos, nits] = mspos(initimg, newimg, ...
          sig, h, initpos, oldpos, epsilon, maxits)
%   Copyright 2003 by author.
%   $Revision: 1.2$ $Date: Tue Nov 18 17:47:38 EST 2003$
[ix1,ix2] = inddisk(initimg,initpos,sig);
sig2 = 2*sig*sig; h2 = 2*h*h;
y = oldpos;
for k = 1:maxits,
    [jy1,jy2] = inddisk(newimg,y,sig);
    y0 = y; sumxyuv = 0.0; sumyxyuv = zeros(size(y));
    for i = 1:length(ix1),
        dx = initpos - [ix1(i) ix2(i)];  dx2 = dx*dx.';
        ui = initimg(ix2(i),ix1(i),:); ui = ui(:);
        for j = 1:length(jy1),
            jy = [jy1(j) jy2(j)];
            dy = y - jy; dy2 = dy*dy.';
            vj = newimg(jy2(j),jy1(j),:); vj = vj(:);
            duv = ui - vj; duv2 = duv.'*duv;
            wt = exp(-((dx2+dy2)/sig2 + duv2/h2));
```

```
            sumxyuv = sumxyuv + wt;
            sumyxyuv = sumyxyuv + jy*wt;
        end
    end
    y = sumyxyuv / sumxyuv;
    if norm(y - y0) < epsilon, break; end
end
newpos = y;
nits = min(k,maxits);
return;

function  [ix,iy] = inddisk(img,pos,h)
siz = size(img);
[XX,YY] = meshgrid(1:siz(1),1:siz(2));
[ix,iy] = find((XX-pos(2)).^2 + (YY-pos(1)).^2 < h^2);
return;
```

References

[1] S. Avidan. Support vector tracking. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume I, pages 184–191, Kauai, HI, 2001.

[2] B. Bascle and R. Deriche. Region tracking through image sequences. In *Proc. Int'l Conf. Computer Vision*, pages 302–307, 1995.

[3] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 8, pages 296–345. PWS Publishing Company, Boston, 1997.

[4] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int'l Journal of Computer Vision*, 26(1):63–84, 1998.

[5] G. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2), 1998.

[6] G. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume I, pages 77–84, Madison, WI, 2003.

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603 – 619, May 2002.

[8] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–577, May 2003.

[9] P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall International, London, 1982.

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

[11] A. Elgammal, R. Duraiswami, and L. Davis. Probabilistic tracking in joint feature-spatial spaces. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume I, pages 781–788, Madison, WI, 2003.

[12] P. Fieguth and D. Terzopoulos. Color based tracking of heads and other mobile objects at video frame rates. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 21–27, Puerto Rico, 1997.

[13] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Inform. Theory*, 21:32–40, 1975.

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

[15] L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Statist. Comput.*, 12(1):79–94, 1991.

[16] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(10):1025–1039, 1998.

[17] M. Irani and S. Peleg. Motion analysis for image enhancement: Resolution, occlusion, and transparency. *JVCIP*, 4:324–335, Dec. 1993.

[18] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. Computer Vision*, pages 343–356, Cambridge, UK, 1996.

[19] D. Jacobs, D. Weinshall, and Y. Gdalyahu. Class representation and image retrieval with non-metric distances. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(6):583–600, 2000.

[20] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 33(3):1065–1076, 1962.

[21] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

[22] T. Poggio and S. Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50(5):537–544, 2003.

[23] J. Puzicha, J. Buhmann, Y. Rubner, and C. Tomasi. Emperical evaluation of dissimilarity measures for color and texture. In *Proc. Int'l Conf. Computer Vision*, pages 1165–1172, Kerkyra, Greece, 1999.

[24] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, MA, 2002.

[25] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 593–600, Seattle, WA, 1994.

[26] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3D human tracking. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume I, pages 69–76, Madison, WI, 2003.

[27] A. R. Webb. *Statistical Pattern Recognition*. John Weley & Sons, UK, 2nd edition, 2002.

[28] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *Proc. Int'l Conf. Computer Vision*, pages 353–360, Nice, France, 2003.

[29] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):780–785, 1997.

[30] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proc. Int'l Conf. Computer Vision*, pages 464–471, Nice, France, 2003.

[31] J. Yang and A. Waibel. A real-time face tracker. In *Proceedings of WACV*, pages 142–147, Sarasota, FL, 1996.