Efficient Particle Filtering via Sparse Kernel Density Estimation

Amit Banerjee, Senior Member, IEEE, and Philippe Burlina, Senior Member, IEEE

Abstract—Particle filters (PFs) are Bayesian filters capable of modeling nonlinear, non-Gaussian, and nonstationary dynamical systems. Recent research in PFs has investigated ways to appropriately sample from the posterior distribution, maintain multiple hypotheses, and alleviate computational costs while preserving tracking accuracy. To address these issues, a novel utilization of the support vector data description (SVDD) density estimation method within the particle filtering framework is presented. The SVDD density estimate can be integrated into a wide range of PFs to realize several benefits. It yields a sparse representation of the posterior density that reduces the computational complexity of the PF. The proposed approach also provides an analytical expression for the posterior distribution that can be used to identify its modes for maintaining multiple hypotheses and computing the MAP estimate, and to directly sample from the posterior. We present several experiments that demonstrate the advantages of incorporating a sparse kernel density estimate in a particle filter.

Index Terms—Bayesian filtering, machine learning, particle filters, support vectors, tracking.

I. INTRODUCTION

Bayesian filtering algorithms seek a recursive estimate of the posterior distribution \( p(x_k | z_{1:k}) \) of the state vector \( X \) at time \( k \) given all of the observations \( Z_{1:k} \) through the following prediction and correction steps:

\[
p(x_k | z_{1:k-1}) = \int dx_{k-1} p(x_k | x_{k-1}) p(x_{k-1} | z_{1:k-1})
\]

\[
p(x_k | z_{1:k}) \propto p(z_k | x_k) p(x_k | z_{1:k-1}).
\]

The prediction step uses the propagation density \( p(x_k | x_{k-1}) \) to compute the prior distribution \( p(x_k | z_{1:k-1}) \). The correction step uses the measurement function \( p(z_k | x_k) \) to update the posterior according to the observation at time \( k \).

When the noise distributions are Gaussian and the evolution and measurement functions are linear, the exact analytical solution is given by the Kalman Filter [13]. Monte Carlo techniques for sequential Bayesian filtering, such as particle filters [6], have received significant attention for their ability to accommodate non-Gaussian conditions and nonlinear systems. Particle filters represent \( p(x_k | z_{1:k}) \) through a set of random particles and their associated weights. The particles are sampled from the state space and the posterior is updated by propagating the particles and updating their weights based upon the observations’ likelihoods. When implementing a particle filter, a key consideration is the tradeoff between accuracy and speed of the algorithm. While an increase in the number of samples improves the estimate of the posterior distribution, it also adds to the computational burden. This problem is exacerbated by the dimensionality of the state space.

In general, generating samples directly from a nontrivial posterior distribution is a difficult problem. Indeed, particle filter algorithms can be classified according to how they draw samples from the state space. Examples of sampling strategies include importance sampling (IS) [1], [7], [12], [14], factored sampling [11] and rejection sampling [8]. The aim of the popular IS methods is to sample from the peaks of the distribution, especially for sparse, high-dimensional data which lie mostly near the modes. This is achieved by employing a proposal distribution \( q(x) \) that is easier to draw samples from and whose peaks lie near the modes of the posterior \( p(x_k | z_{1:k}) \) [7]. The main consideration of IS methods is the choice of the proposal distribution \( q() \). A common solution is to use the sequential importance sampling (SIS) method to iteratively generate the proposal distribution based upon previous estimates of the posterior [21]. The main drawback of IS methods is the degeneracy problem [1]: if \( q() \) does not match the shape of \( p(x_k | z_{1:k}) \), many of the weights will be uneven and negligible after a few iterations. Thus, significant computation is wasted on particles with low weights. Factored sampling, used in Condensation, often suffers from the sample impoverishment problem, where multiple samples fall at the same point in the state space [1].

To address these issues, we propose a particle filtering approach that exploits a support vector method for density estimation. Given a set of sampled particles, the prior distribution \( p(x_k | z_{1:k-1}) \) is computed using a sparse kernel density estimation method. Examples of such kernel density estimators include the support vector data description (SVDD) and relevance vector machines (RVMs) [24]. In this paper, we use the SVDD algorithm that yields a linear mixture of kernels (LMK) expression for the distribution (a linear combination of kernel functions). This model is subsequently updated using the observation to produce a closed-form expression for the posterior state distribution.
Combining the SVDD density estimate within a PF has the following advantages:

- **Increasing the speed of the algorithm.** The main computational burden of most particle filters lies in evaluating the weights of the particles. The SVDD provides a sparse representation of the distribution by identifying the particles that are most significant. This avoids the need to compute the weights of the remaining samples, thereby reducing the computational burden.

- **Providing an analytical expression for the posterior state density \( p(X_k | Z_k) \).** The SVDD provides a continuous pdf, instead of a probability mass function (pmf), expression for the posterior. The expression is more amenable for further processing, such as finding the local maxima, maintaining multiple hypotheses, and generating samples from the posterior pdf.

This work investigates the benefits of the sparse density estimate for a broad class of PFs. The computational complexity of the SVDD is examined and shown to be insignificant compared to the weighting step of a PF. Experimental results demonstrate that the sparse SVDD density estimate significantly reduces the processing time of a general PF while maintaining similar tracking performance.

The paper is organized as follows. Section II gives a brief overview of related methods to improve particle filters. The generic particle filter is reviewed in Section III. The SVDD algorithm for sparse density estimation is presented in Section IV and Section V examines the computational complexity of the SVDD density estimate. Section VI details how the SVDD density estimate is incorporated within a generic PF, and Section VII examines the performance characteristics of the estimator. Experimental results on synthetic and real data are provided in Section VIII. Finally, Section IX concludes the paper.

II. RELATED WORK

An exhaustive summary of previous work on particle filtering is beyond the scope of this section. Interested readers are referred to [1], [2], [6], [36], and [37] for a broad overview. In the following, we highlight recent developments in PFs for the purpose of visual tracking.

Several related algorithms have been recently introduced that incorporate density estimation methods into a particle filter [3], [35], [38]. In [5], Cham and Rehg employ a piecewise Gaussian (PWG) model to represent the measurement function and the prediction density. While this allows for multihypothesis tracking, sampling from and computing the posterior distribution using a PWG is a nontrivial problem. In [18], Vermaak *et al.* directly model the posterior density as a Gaussian mixture model (GMM) and show that the distribution can be propagated by filtering each component of the mixture independently. By allocating samples to each component, the algorithm is able to maintain multiple hypotheses. A kernel-based Bayesian filter is introduced in [3]. Assuming a GMM, an unscented transform is used to compute the prior density \( p(X_k | Z_{1:k-1}) \). An analytical form for the measurement function \( p(Z_k | X_{1:k}) \) is estimated from the data using multistage sampling and density interpolation methods. After multiplying the two functions, a smooth approximation for the posterior density is found using a variable-width mean-shift technique. Finally, a previous example of using support vector regression within a particle filter framework is described [35].

While these methods exploit an analytical form to maintain multiple hypotheses or simplify the sampling problem, PFs suffer from a high computational cost, especially when large numbers of particles are required to model the posterior distribution. To reduce the computational burden, several modifications of the particle filter have been developed. In [25], Deutcher *et al.* interleave a simulated annealing step in the traditional Condensation filter. The simulated annealing sampling allows to sample from a posterior distribution which is a smoothed version of the posterior. The combination allows for the generation of samples that are closer to the actual modes of the posterior distribution and avoids the problem of getting locked generating samples from local minima. The authors provide favorable comparisons with the traditional Condensation filter to recursively estimate the high-dimensional state vector of articulated body motion. In [26], Choo and Fleet replace the often-used importance sampling method with a Monte Carlo Markov chain (MCMC) method to sample from the posterior. The rationale is that by using a suitable transition distribution, the MCMC will converge to the posterior distribution and will allow generation of posterior samples in a much faster and more efficient way (with fewer number of particles) when compared to the traditional Condensation filter. Pitt and Shephard [27] note two of the drawbacks of particle filters, namely that they are inefficient in high-dimensional spaces and do not approximate the tails of the multimodal posterior. They indicate that these problems are general to the PF whether it uses important sampling, rejection sampling or MCMC. They instead propose to extend the state space with one additional auxiliary dimension made up of an additional index to improve posterior sampling efficiency. They show through use of simulated data and comparison with the Condensation filter an improvement in performance.

III. GENERIC PARTICLE FILTER

In this section, a generic particle filter algorithm is outlined. The outline will serve to illustrate how a sparse kernel density estimate can be used within the PF framework in Section VI. We review how algorithms such as the bootstrap and Condensation PFs are derived as special cases of the generic PF.

Particle filters model a distribution by a set of samples and associated weights at time \( k \), \( S_k = \{s_i^k, w_i^k\}, i = 1, \ldots, L \), with \( \sum_j w_j^k = 1 \). Each iteration of the algorithm emulates the prediction/update steps of a Bayesian filter. Adopting the notation used in [2], the pseudo-code for the filter is given in the following.
Generic Particle Filter

\{s_{k-1}^j, u_{k-1}^j\}_{j=1}^L = \text{ParticleFilter}([s_{k-1}^j, u_{k-1}^j]_{j=1}^L, Z_k)

1) Generate samples from the proposal distribution (or importance function) \(q(\cdot)\),
\[ s_k^j \sim q(x_k | x_{k-1} = s_{k-1}^j, z_k), j = 1, \ldots, L. \]
2) Assign weights to the particles according to
\[ w_k^j \propto w_{k-1}^j p\left( x_k = s_k^j | x_{k-1} = s_{k-1}^j \right) / q\left( x_k = s_k^j | x_{k-1} = s_{k-1}^j, z_k \right). \]

This rule makes the samples properly weighted to the posterior \( p(x_k | z_k) \), even when the particles are sampled from the proposal density \( q(\cdot) \).
3) Normalize the weights so that they sum to one
\[ w_k^j = w_k^j / \left( \sum_{j=1}^L w_k^j \right). \]
4) Compute the effective sample size of this weighted set of particles as \( N_{\text{eff}} = \left( \sum_{j=1}^L w_k^j \right)^2 / \sum_{j=1}^L w_k^j \).
5) If the effective sample size is less than a threshold, \( N_{\text{eff}} < \tau \), then resample to remove particles with small weights and concentrate on particles with higher weights.
   - Construct the CDF of the particle weights \( c_i = \sum_{j=1}^i c_j + w_k^j, i = 1, \ldots, L. \)
   - FOR \( j = 1: L. \)
     a) Generate a uniformly distributed random sample \( u_j \sim U[0, c_L] \).
     b) Find \( m \), such that \( c_m \leq u_j < c_{m+1} \).
     c) Assign sample \( s_k^j = s_k^m \), with weight \( w_j^* = 1/L. \)
   - End FOR.

From this generic algorithm, it is straightforward to derive various PFs that are commonly used [2]. For example, the bootstrap PF [9] is a special case of the generic algorithm if the proposal density is chosen as \( q(x_k | x_{k-1}, z_k) \equiv p(x_k | x_{k-1}) \). If the resample step (step 5) is never performed, the resulting algorithm is the sequential importance sampling (SIS) filter. Conversely, if the resampling step is always performed, and the state transition density is selected as the importance function \( q(\cdot) \), the PF corresponds to the well-known sampling importance resampling (SIR) or Condensation algorithm.

The PF is a Monte Carlo method that offers the flexibility to model nonlinear observations and non-Gaussian, multimodal distributions. However, the representation using the set of samples and weights is not amenable for further processing, such as locating the modes or peaks of the distribution which correspond to the maximum likelihood (ML) or maximum a posteriori (MAP) estimates of the state vector. As a consequence, many particle filters use the moments of the distribution (such as the mean), which are easier to compute. Although the mean is optimal in the MMSE sense, it may not be useful when the posterior is multimodal. In such cases, the MAP criteria may be more appropriate, where the estimate of the local maxima (or modes) of the posterior is required [33], [34]. Accurate estimates of the MAP solution can be obtained via gradient ascent by exploiting a smooth analytical form of the distribution obtained using kernel methods. The SVDD, described in the following section, is an example of such a method.

IV. SVDD DENSITY ESTIMATE

Given a set of data points \( \{x_i, i = 1, \ldots, N\} \), we use a kernel-based density estimate of the underlying distribution of the data \( f(x) \), expressed as a linear mixture of kernels (LMK)
\[ f(x) = \sum_{i=1}^N \alpha_i K(x, x_i) \]  
where the \( \alpha_i \) are nonnegative weights that sum to one, and \( K \) is a nonnegative kernel function which integrates to one. When \( \alpha_i = 1/N \), this results in the well-known Parzen density estimator [19]. However, computing the value of the Parzen density estimate is often time-consuming, since the kernel function must be evaluated for every training sample. This is especially important for particle filtering, since the density function is evaluated many times for each iteration of the filter.

To improve the speed of kernel density estimators, machine learning approaches such as support vector machines (SVMs) and RVMs [24] have been recently developed. These are sparse methods that are alternate means of finding the weights \( \alpha_i \) to approximate \( f(x) \). A full review of these algorithms and their respective performance compared to the Parzen methods is provided in [28].

One possibility is to use RVMs, which are based upon a Bayesian formalism to estimate the LMK expression for problems such as classification, vector quantization, and density estimation [24]. It assumes a probabilistic model for the parameters of the LMK and utilizes a ML or MAP criteria to estimate their values. Sparsity is achieved by imposing a hyperprior for the weights that encourages many of the \( \alpha_i \) to go to zero. Although RVMs often provide a more sparse representation, the procedure to estimate the RVM parameters is computationally complex. Thus, incorporating an RVM density estimate in a PF framework is not practical for real-time processing.

Instead, this work utilizes the SVDD, which was developed by Tax and Duin [15], [16], as a support vector machine for one-class classification. It can be shown that given the set of training points \( \{x_i\} \), the SVDD provides a sparse LMK expression for the shape or boundary of the region of support for the training points. Using kernels, the SVDD has the flexibility to model the support of arbitrary complex, non-Gaussian, and multimodal data.

Rather than using a probabilistic framework, the SVDD uses a geometric approach to estimate the shape of the underlying pdf. The SVDD seeks to minimize the radius \( R \) of the \( D \)-dimensional sphere centered at \( \mathbf{a}, S = \{x : ||x - a||^2 < R^2\} \) enclosing the entire set of training exemplars \( T = \{x_i \in \mathbb{R}^D, i = 1, \ldots, N\} \). This constrained optimization problem is solved by minimizing the following Lagrangian cost function:
\[ J = R^2 + \sum_{i=1}^N \lambda_i ||x_i - \mathbf{a}||^2. \]  
The parameters are estimated by taking the partial derivatives and using quadratic programming. Based upon the Kuhn-Tucker theorem, it can be shown that the following statements hold [32]:
1) Only a small fraction of the Lagrangian multipliers, \( \alpha_i \), are nonzero.
2) The sample vectors corresponding to nonzero \( \alpha_i \), called support vectors, lie at the sphere’s boundary and form a sparse representation of the training data distribution.
3) The Lagrange multipliers sum to one: \( \sum_i \alpha_i = 1 \).
4) The center of the sphere is the center of mass of the support vectors: \( \mathbf{a} = \sum_i \alpha_i \mathbf{x}_i \).

Given the optimal weights \( \alpha_i \), the linear SVDD function is expressed as

\[
SVDD_{\text{linear}}(\mathbf{x}) = R^2 - \left\| \mathbf{x} - \sum_i \alpha_i \mathbf{x}_i \right\|^2 \\
= R^2 - \mathbf{x}^T \mathbf{x} + 2 \sum_i \alpha_i (\mathbf{x}^T \mathbf{x}_i) \\
- \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j). 
\]  

The minimum volume hypersphere boundary containing the training points occurs when \( SVDD(\mathbf{x}) = 0 \). Different values for \( SVDD(\mathbf{x}) \) correspond to other isocontours of the pdf. The SVDD method can be generalized to allow for nonspherical support regions by using the well-known “kernel-trick” [32], where the SVDD function is now expressed as

\[
SVDD(\mathbf{x}) = R^2 - K(\mathbf{x}, \mathbf{x}) + 2 \sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) \\
- \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\
= \sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) \\
- 1/2 \left( K(\mathbf{x}, \mathbf{x}) - \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \\
\alpha \sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b_0. 
\]

Equation (5) shows that the LMK expression found by the SVDD describes the boundary of the density function. As with the linear SVDD, different values of the function correspond to different isocontours of the pdf. Since the \( \alpha_i \) are nonnegative and sum to one, and the kernel function integrates to one, the SVDD’s LMK estimate is a proper density estimate.

It can be shown that the performance of this estimate is similar to the Parzen method [28]. However, unlike Parzen, since many of the \( \alpha_i \) are zero, \( SVDD(\mathbf{x}) \) results in a sparse representation that allows for fast evaluation of the underlying distribution. A comparison of the performance of several kernel density estimators in [28] illustrates that the SVDD density estimate and the Parzen method provide similar quantitative performance. Further evaluation of density estimation performance can be found in [28].

In this paper, we use the Gaussian radial basis function (RBF) as the kernel function, as is common practice. The RBF has one free parameter, which is the scale parameter \( \sigma \). This parameter affects the tightness-of-fit for the training data. Smaller values for \( \sigma \) generate more support vectors and model the distribution of the data tightly (and can lead to data over-fitting). Larger values for \( \sigma \) yield fewer support vectors, and the sparser density representation may possibly underfit the data [15], [16]. By varying the scale parameter of the RBF, the SVDD can determine multiple regions of support for a dataset (see [15]). This allows the SVDD to model non-Gaussian, multimodal distributions. An example of a multimodal SVDD density estimate is given in Fig. 1. In Fig. 2, a quantitative evaluation of several kernel density estimators shows that they provide similar results as the Parzen method.
V. SVDD FUNCTION OPTIMIZATION

Computing the SVDD density estimate requires the following optimization:

$$\min_{\alpha} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \text{ subject to } 0 \leq \alpha_i \leq 1$$

$$\sum_{i} \alpha_i = 1.$$  \hspace{1cm} (6)

Traditional support vector algorithms use standard quadratic programming (QP) techniques such as active set and conjugate gradient methods to minimize the function. These methods have a complexity of $O(N^3)$, where $N$ is the number of training samples. However, given a large number of training samples, standard QP techniques cannot solve the problem efficiently. They involve a matrix whose size scales quadratically with the number of active constraints (non-zero $\alpha_i$’s) and therefore require a lot of memory and a large number of iterations to arrive at a solution. For particle filtering, this is a critical issue since many samples are needed to accurately model high-dimensional PDFs.

To address this issue, a “chunking” method was introduced by Vapnik [32] to solve the dense SVM QP problem. The chunking algorithm uses the fact that the value of the quadratic function is the same if you remove the rows and columns of the matrix that correspond to zero Lagrange multipliers. Therefore, the large QP problem can be broken down into a series of smaller QP problems, and the size of the matrix is approximately the number of nonzero Lagrange multipliers squared (instead of the number of training samples). However, given a large dataset, chunking still requires a considerable amount memory which causes a noticeable reduction in algorithm speed.

In [31], a new class of QP algorithms was proposed by Osuna, et al. They demonstrate that the large QP problem can be broken down into a series of smaller QP subproblems. Sequential minimal optimization (SMO) [30] is an example of such a fast method to find the solution for a very large QP optimization problem. SMO breaks this large QP problem into a series of QP problems involving only two examples. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The amount of memory required for SMO is linear in $N$, which allows SMO to handle very large training sets.

The results presented in this work utilize the SVDD implementation found in the NTU SVM software package [23]. This implementation uses the SMO minimization method to find the minimum volume hypersphere. To illustrate the increase in speed when using the SMO method, the SVDD algorithm is run using both the standard QP and SMO methods on the same data with the same value for $\sigma$ while varying the number of training samples. Plots of the run-times as a function of the number of samples are given for both methods in Fig. 3. The plots show that the runtime for the standard QP increases exponentially with the number of training sample, while SMO increases linearly. Furthermore, SMO is an order-of-magnitude faster. This is a key benefit to using SMO for sequential Bayesian filtering, since the SVDD density estimate is computed for each iteration of the filter.

VI. SVDD-PF ALGORITHM

This section outlines the generic PF algorithm that incorporates the sparse SVDD kernel density estimate. Our method expands on the approach introduced in [10], where a kernel density estimate is used to model the prior distribution $p(x_k | z_{1:k-1})$. This approach offers a simple method for updating the distribution to estimate the posterior $p(x_k | z_{1:k})$, in contrast to the algorithms outlined in Section II. The algorithm presented in the following improves upon the work in [10] by using a sparse density estimate to increase the speed of the algorithm. Furthermore, since a wide range of PFS (i.e., SIS, SIR, bootstrap) are special cases of the generic PF [2], the SVDD can be incorporated in those algorithms, as well.

Generic Particle Filter with SVDD

$$\{s^j_k, w^j_k\}_{j=1}^L = \text{PF-SVDD}[\{s^j_{k-1}, w^j_{k-1}\}_{j=1}^L; Z_k]$$

1) **Generate samples** from the proposal distribution (or importance function) $q()$,

$$s^j_k \sim q(x_k | z_k), j = 1, \ldots, L.$$  \hspace{1cm} (7)

2) Using equally weighted samples $\{s^j_k\}, j = 1, \ldots, L$, compute the sparse SVDD kernel density estimate $q(X_k = x | z_k) = \sum_{j=1}^L \alpha_j K(x, s^j_k)$.  \hspace{1cm} (8)

3) **Assign weights** only to those particles with nonzero $\alpha_j$ according to:

$$w^j_k = \frac{p(X_k = s^j_k | z_{1:k})}{q(X_k = s^j_k | z_{1:k})} = \frac{p(z_k | X_k = s^j_k) p(X_k = s^j_k | X_{k-1} = s^j_{k-1})}{q(X_k = s^j_k | z_{1:k}).}$$

This rule makes the samples properly weighted to $p(x_k | z_{1:k})$, and the resulting estimate for the posterior distribution is

$$p(X_k = x | z_k) = \sum_{j=1}^L w^j_k \alpha_j K(x, s^j_k) = \sum_{j=1}^L \pi_j K(x, s^j_k).$$  \hspace{1cm} (9)
Note that since many of the $\alpha_j$ are equal to zero, the algorithm only needs to compute the weights for the particles with nonzero $\alpha_j$.

4) **Normalize the weights** so that they sum to one: 
\[
\pi^*_k = \frac{\pi_k}{\sum_{i=1}^L \pi^*_i}.
\]

5) **Compute the MMSE or MAP estimate** for the state vector. Given the LMK expression in (8), use gradient ascent to detect the modes of the posterior.

6) **Resample** to generate new particles with equal weights. Since many of the particles will have zero weight due to the sparse kernel density estimate, it will be necessary to resample at each iteration to avoid the sample degeneracy problem. **Resample Algorithm**:
   - Generate samples near the modes of the posterior found in Step 5.
   - Or sample directly from the LMK model in (8).

In the following section, it is shown that the sparse LMK expression in (8) using the SVDD provides a nearly unbiased estimate for the posterior. The bias of the estimate gives insight into the tradeoff between speed and accuracy of the SVDD particle filter. Furthermore, it is shown that the traditional pdf representation for the posterior is a special case of (8) when the bandwidth of the kernel goes to zero. The closed-form expression for the MMSE estimate is also derived.

Note that the use of the LMK model for the posterior is similar to the regularized PF [29]. However, the LMK is utilized only for the resampling step. In the proposed algorithm, the LMK is used throughout to realize additional advantages, including:

- **Computing the ML or MAP estimate** of the state vector using gradient ascent to find the peaks of $p(x_k | z_{1:k})$. Unlike most particle filters that use moments of the distribution to estimate the state vector, the generic SVDD-PF can quickly find the modes of the posterior using gradient ascent. When using the RBF kernel, the closed-form expressions for the gradient and Hessian of the distribution are given in [4]. Since these expressions are simple to evaluate, the peaks of the distribution can be found quickly and robustly. For further details on the gradient ascent iterations, the reader is referred to [4].

- **Increase in algorithm speed.** As most of the $\alpha_j$ are zero, we only need to compute $p(z_k | x_k = s_j)$ for the small fraction of nonzero $\alpha_j$. In general, evaluating the weights for the samples is the most time-consuming part of a particle filter. Therefore, using the sparse representation where most of the weights are zero offers significant reduction in processing time. The improvement is especially significant for high-dimensional state spaces, since the number of required particles increases exponentially, while the number of support vectors increases approximately linearly.

Choosing an appropriate value for the RBF scaling parameter $\sigma$ for a particular application is an ongoing research topic. Existing methods include cross-validation and leave-one-out methods [15], [17] to estimate the type one error as a function of the scaling parameter. For Bayesian filtering, we use the observation that, for a given set of training data, the value of sigma determines the number of support vectors [15], [17]. Since the number of support vectors affects the speed and accuracy of the filter, one can choose the value of sigma that satisfies their operating constraints in terms of processing time and tracking performance.

### VII. Analysis of LMK Estimate

In this section, the validity of the SVDD LMK estimate in (8) for the posterior distribution $p(x_k | z_{1:k})$ is examined. In particular, we show that:

1) the SVDD LMK density estimate is properly weighted;
2) on average, SVDD LMK density estimate for the posterior is simply a smoothed version of the true posterior density: $E_q[p(x | z_{1:k})] = p(x | z_{1:k}) \ast K(x)$;
3) the SVDD LMK density estimate for the posterior is unbiased-up to the first order term of its Taylor series expansion.

These properties illustrate that the LMK density estimate in (8) is a nearly unbiased estimate for the posterior even when the samples are drawn from the proposal density.

To show that the estimate for the posterior in (8) is properly weighted, we start by deriving the MMSE estimate for the state vector under the SVDD density estimate

\[
\hat{x}_{\text{MMSE}} = E[x | z_{1:k}] = \int x p(x | z_{1:k}) \, dx
\]

where the equality in the last line holds since $K$ is a symmetric, nonnegative function centered at $s_j$ that sums to one (e.g., RBF, uniform, and Epanechnikov kernels) for most density estimation techniques.

Since the $\{s_j\}$ are sampled from $q()$, the expected value of the estimator is

\[
E_q[\hat{x}_{\text{MMSE}}] = E_q \left[ \sum_{j=1}^L \alpha_j w_k^j s_j^i \right] = \sum_{j=1}^L \alpha_j E_q \left[ w_k^j s_j^i \right]
\]

where

\[
E_q[\hat{x}_{\text{MMSE}}] = \sum_{j=1}^L \alpha_j \int p(x | z_{1:k}) \, dx \approx \sum_{j=1}^L \alpha_j \int p(x | z_{1:k}) \, dx \approx \sum_{j=1}^L \alpha_j \int p(x | z_{1:k}) \, dx
\]

and

\[
E_q[\hat{x}_{\text{MMSE}}] = \sum_{j=1}^L \alpha_j \int p(x | z_{1:k}) \, dx \approx \sum_{j=1}^L \alpha_j \int p(x | z_{1:k}) \, dx
\]

and

\[
E_q[\hat{x}_{\text{MMSE}}] = \sum_{j=1}^L \alpha_j \int p(x | z_{1:k}) \, dx \approx \sum_{j=1}^L \alpha_j \int p(x | z_{1:k}) \, dx
\]
\[ \sum_{j=1}^{L} \alpha_j \int s_k^j p(\mathbf{s}_k^j | \mathbf{z}_{1:k}) d\mathbf{s}_k^j = \sum_{j=1}^{L} \alpha_j E_p[\mathbf{x}] = E_p[\mathbf{x}] \] (10)

since the \( \alpha_j \) sum to one. Hence, using properly weighted samples, the SVDD density estimator and the true posterior densities provide similar MMSE estimates for the state vector.

To evaluate the bias of the sparse SVDD density estimator, we follow the steps outlined in Chapter 6 of [39]. Note that the LMK can be written as a convolution

\[ \hat{p}(\mathbf{x}_k = \mathbf{x} | \mathbf{z}_{1:k}) = \sum_{j=1}^{L} \alpha_j u_k^j K\left(\mathbf{x} - \mathbf{s}_k^j\right) \]
\[ = \int \sum_{j=1}^{L} \alpha_j u_k^j \delta(\mathbf{y} - \mathbf{s}_k^j) K(\mathbf{x} - \mathbf{y}) d\mathbf{y} \]
\[ = \left[ \sum_{j=1}^{L} \alpha_j u_k^j \delta(\mathbf{x} - \mathbf{s}_k^j) \right] * K(\mathbf{x}). \] (11)

The pdf estimate \( \hat{p}(\mathbf{x}_k = \mathbf{x} | \mathbf{z}_{1:k}) \) is a function of the samples \( \{s_k^j\} \), where each sample is randomly and independently generated from the proposal density, i.e., \( s_k^j \sim q(\cdot | \mathbf{z}_{1:k}) \). Hence, the density estimators are themselves random variables and their performance can then be evaluated by considering their statistical properties. To characterize the estimator’s bias we examine its expected value under the proposal distribution

\[ E_q[\hat{p}(\mathbf{x} | \mathbf{z}_{1:k})] = \int E_q \left[ \sum_{j=1}^{L} \alpha_j u_k^j \delta(\mathbf{y} - \mathbf{s}_k^j) \right] K(\mathbf{x} - \mathbf{y}) d\mathbf{y}. \] (12)

Looking specifically at the expectation term

\[ E_q \left[ \sum_{j=1}^{L} \alpha_j u_k^j \delta(\mathbf{y} - \mathbf{s}_k^j) \right] \]
\[ = \int E_q \left[ \sum_{j=1}^{L} \alpha_j u_k^j \delta(\mathbf{y} - \mathbf{s}_k^j) \right] q(\mathbf{s}_k^j | \mathbf{z}_{1:k}) d\mathbf{s}_k^j \]
\[ = \int \sum_{j=1}^{L} \alpha_j p(\mathbf{s}_k^j | \mathbf{z}_{1:k}) \delta(\mathbf{y} - \mathbf{s}_k^j) q(\mathbf{s}_k^j | \mathbf{z}_{1:k}) d\mathbf{s}_k^j \]
\[ = \sum_{j=1}^{L} \alpha_j p(\mathbf{s}_k^j | \mathbf{z}_{1:k}) \delta(\mathbf{y} - \mathbf{s}_k^j) d\mathbf{s}_k^j \]
\[ = \sum_{j=1}^{L} \alpha_j \delta(\mathbf{y} | \mathbf{z}_{1:k}) \]
\[ = p(\mathbf{y} | \mathbf{z}_{1:k}) \] (14)

where we make use of the fact that \( \sum_{j=1}^{L} \alpha_j = 1 \) in the last step.

Finally, using this result in (12) we find that

\[ E_q[\hat{p}(\mathbf{x} | \mathbf{z}_{1:k})] = \int p(\mathbf{y} | \mathbf{z}_{1:k}) K(\mathbf{x} - \mathbf{y}) d\mathbf{y} = p(\mathbf{x} | \mathbf{z}_{1:k}) * K(\mathbf{x}). \] (15)

This shows that, on average, the SVDD LMK density is simply a smoothed version of the true posterior density. As explained in [39], the use of a smoothing kernel function \( K \) introduces some bias, which is true of all kernel density estimators, including the Parzen method. Using Taylor series expansion, it can be shown [39] that the estimate is only unbiased to the first order Taylor series expansion, i.e., the first order terms of the Taylor series expansion of the mean under \( q(\cdot) \) of the kernel estimate and the actual density are equal. Using a second-order approximation, the expected value of the estimate is [39]

\[ E_q[\hat{p}(\mathbf{x} | \mathbf{z}_k)] \approx p(\mathbf{x}) \left[ 1 + \frac{1}{2} g(\mathbf{x}) \sigma^2 \right] \] (16)

where \( \sigma \) is the kernel width parameter, \( g(\mathbf{x}) = N(\nabla^2 p(\mathbf{x})/(p(\mathbf{x}))) \), and \( N \) is the dimensionality of \( \mathbf{x} \), for the Gaussian or RBF kernel. This expression shows that large values of \( \sigma \) produce sparse density estimates with greater bias. Conversely, smaller values of \( \sigma \) lead to a less biased estimate while generating a less sparse density estimate. This represents the tradeoff between speed and accuracy of the filter and is directly influenced by the choice of the kernel parameter. In fact, as \( \sigma \to 0 \), the kernel function converges to a Dirac delta function and the SVDD weights \( \alpha_j = 1/L \) for all \( j \) [15], [16]. The resulting estimate for the posterior is the traditional pdf representation \( p(\mathbf{x} | \mathbf{z}_{1:k}) = \sum_{j=1}^{L} u_k^j \delta(\mathbf{x} - \mathbf{s}_k^j) \).

VIII. EXPERIMENTAL RESULTS

Several sets of experiments are presented to demonstrate the benefits of incorporating a sparse kernel density estimate into a PF. A simulated experiment with known plant and measurement equations is used to demonstrate the ability of the proposed method to track in the presence of nonlinear dynamics, observations, and non-Gaussian noise. Contour tracking and appearance-based tracking examples are presented to show how the proposed method exploits the sparse density estimate to reduce the processing time of the filter while maintaining similar tracking performance.

To simplify notation, the particle filter that incorporates the SVDD kernel density estimate within a SIR filter is denoted as the SIR-SVDD PF. Likewise, the IS-SVDD PF refers to the algorithm that combines the SVDD with the importance sampling PF. For the experiments described in the following, the proposal function \( q(\cdot) \) for the SIR and SIR-SVDD PFs is the state transition density \( p(\mathbf{x}_k | \mathbf{x}_{k-1}) \). For the IS and IS-SVDD methods, the proposal is a Gaussian PDF whose mean is the MAP estimate from the previous iteration of the filter. Finally, the resampling technique for the SIR and IS algorithms is described in step 5 of the generic particle filter. For the SIR-SVDD and IS-SVDD PFs, resampling is achieved by directly sampling from the mixture of Gaussians distribution obtained when using the RBF kernel for the LMK (see (8)).
Recall from Section IV that the value for \( \sigma \) and the edges in the image. The dynamics of both algorithms.

**Fig. 5.** (Left) Plots comparing the true (solid line) and estimated (dashed line) values plotted in red. Note the bimodal nature of the posterior distribution.

**Fig. 4.** Plot of marginal distribution of the state vector parameters with the true values plotted in red. Note the bimodal nature of the posterior distribution.

**Simulated Experiment** To generate synthetic tracking data, we use the prediction and measurement update equations used in [10]

\[
X_{k+1} = \frac{X_k}{2} + \frac{25}{1 + \frac{X_k}{X_k}} + 8 \cos(1.2 k) + v_k,
\]

\[
v_k \sim N(0, 10 * I)
\]

\[
Z_k = \frac{X_k^T X_k}{20} + w_k, \quad w_k \sim N(0, 1).
\]

The dynamics of the system are clearly nonlinear, and the resulting posterior distribution is multimodal due to the square term in the measurement equation (see Fig. 4). Using a 7-D state-space, we compared the SIR PF with the SIR-SVDD filter with varying values for the RBF kernel parameter \( \sigma \). Fig. 5 compares the accuracy of the filters by plotting the average error and error variance with respect to the number of particles.

The statistics are computed over 100 runs of simulated data, where the length of each run is 500 time steps \((k = 0, \ldots, 499)\). For the SIR PF, the estimated state vector is the mean of the weighted samples. For the SIR-SVDD, the MMSE and the MAP estimates for the state vector are computed, as described in Section VI. It can be inferred from the plot that the SIR and SIR-SVDD PFs provide similar performance when using the mean estimate. In this multimodal example, the average error of the MAP estimate is lower, since it is able to detect the modes and select the mode with the highest posterior value.

The true and estimated state parameters are compared in Fig. 5 for different values of \( \sigma \). The differences between the true and estimated values are noticeably greater for \( \sigma = 80 \) than for \( \sigma = 10 \). Recall from Section IV that the value for \( \sigma \) determines the number of support vectors used to approximate the density function. Larger values of \( \sigma \) yield a sparser representation of the pdf with fewer support vectors. Conversely, using a smaller value for \( \sigma \) yields more support vectors to generate a more accurate pdf for the set of training data. Since the SIR-SVDD method evaluates the measurement likelihood \( p(z_k | x_k = s_i) \) for the support vectors only, the choice for \( \sigma \) represents a tradeoff between speed and accuracy of the algorithm. In summary, choosing a large value for \( \sigma \) provides a faster particle filter implementation, while a smaller value leads to a more accurate filter at the cost of increased processing speed. This is shown in Fig. 5, where the mean square error and the error variance for the SIR-SVDD and the SIR filters are plotted with respect to the number of particles.

**Contour Tracking** A contour tracking example is given to demonstrate the ability of the proposed method to significantly increase the speed of particle filters. The SIR and SIR-SVDD PFs are used to recursively estimate the 3-D planar motion of a building rooftop observed by a camera from a small UAV platform.

A B-spline describes the contour of the rooftop, and an 8-D motion model (the parameters of a planar homography matrix) serves as the state vector of the system

\[
x_{k+1}(i) = x_{k+1}(i) + u_i^2
\]

\[
p(z_k | x_k = s_k) \propto \exp(-h(s_k, z_k)/25)
\]

where \( h(s_k, z_k) \) denotes the Chamfer distance between the hypothesized contour \( s_k \) and the edges in the image. The dynamics of the camera motion are assumed to be unknown and, thus, a random walk model is adopted for the state transition density.

To compare the speeds, the SIR and SIR-SVDD PFs are given 1000 particles to approximate the distribution at each iteration. An equal number of particles was chosen to ensure both filters achieve similar tracking performance. For the SIR-SVDD, the scale parameter of the RBF kernel is selected so that an average of 275 support vectors is found. As can be seen in Table I, the total running time for the SIR-SVDD filter is noticeably shorter than the SIR particle filter. The reduction in processing time is primarily due to the fewer number of likelihood measurements needed to compute the weights of the particles, as shown in Table I. This is a key benefit of the SVDD—since it provides a sparse density estimate, only the samples with nonzero weights require a likelihood measurement and update. The sparse density estimate provides a nearly 40% increase in speed for nearly similar tracking performance (as shown in Figs. 6 and 7). Also note that the total time devoted to computing the SVDD density estimate is approximately 0.67% of the total time, which is negligible when compared to the total time to process all 300 frames. We use the LIBSVM [23] software package which provides a fast implementation of the SVDD based upon the SMO algorithm.
Appearance-based Tracking Next, a color tracking example is presented to demonstrate the improvement in speed when using the SVDD density estimate. The IS and IS-SVDD filters are used to track two players in the football sequence, as shown in Fig. 8. Each player is tracked independently by separate filters (i.e., 2 IS and 2 IS-SVDD filters). The objects are denoted by a bounding box, and a 4-D parameterization of the bounding box (the $x$- and $y$-location of its centroid and its height and width) serves as the state vector of the system

$$x_{k+1}(i) = x_{k+1}(i) + w_i^2$$

$$i = 1, \ldots, 4, \quad w_i \sim N \left(0, \psi_i^2 \right)$$

$$p(z_k | X = s_k) \propto \exp(- \text{bhatt}(s_k, z_k) / \sigma^2) \quad (19)$$

The color distribution of the pixels within the bounding box is used to characterize the objects’ appearance. In this example, the weight of each particle is the negative exponentiation of the Bhattacharya distance between the normalized color histogram of the object and the color histogram of its hypothesized location in the image. As stated previously, the dynamics of the players is unknown and modeled as a random walk.

The two players are tracked simultaneously using two independent IS or IS-SVDD filters. Each filter uses 500 particles, for a total of 1000 particles. For the IS-SVDD, the scale parameter of the RBF kernel is chosen so that an average of 100 support vectors is found per filter while maintaining the same qualitative performance. An equal number of particles was chosen to ensure both filters achieve similar tracking performance, as can be seen in Fig. 8. As before, the results in Table II show that the total processing time for the IS-SVDD filter is noticeably less than the IS particle filter. The primary reason for the reduction in total processing time stems from the reduced number
TABLE I
COMPARING THE PROCESSING TIMES FOR THE SIR AND SIR-SVDD PARTICLE FILTERS FOR 300 FRAMES FROM THE SMALL UAV SEQUENCE.
ALL TIMES ARE ACCORDING TO THE MATLAB PROFILER

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total SVDD Time</th>
<th>No. of Likelihood Measurements</th>
<th>Time to Calculate Particle Weights</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIR</td>
<td>-</td>
<td>$1000 \times 300 = 300,000$</td>
<td>368.494 sec</td>
<td>480.943 sec</td>
</tr>
<tr>
<td>SIR-SVDD</td>
<td>2.008 sec</td>
<td>82,572</td>
<td>109.398 sec</td>
<td>301.241 sec</td>
</tr>
</tbody>
</table>

Fig. 8. Football sequence. (top) Tracking two players using IS filter. (bottom) Tracking two players using the IS-SVDD filter. (d) Frame 41, (e) Frame 68, and (f) Frame 95.

TABLE II
COMPARING THE PROCESSING TIMES FOR THE IS AND IS-SVDD PARTICLE FILTERS FOR 96 FRAMES OF THE FOOTBALL SEQUENCE.
ALL TIMES ARE ACCORDING TO THE MATLAB PROFILER

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total SVDD Time</th>
<th>No. of Likelihood Measurements</th>
<th>Time to Calculate Particle Weights</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>-</td>
<td>$1000 \times 96 = 96,000$</td>
<td>240.408 sec</td>
<td>305.135 sec</td>
</tr>
<tr>
<td>IS-SVDD</td>
<td>0.681 sec</td>
<td>19,342</td>
<td>49.152 sec</td>
<td>157.572 sec</td>
</tr>
</tbody>
</table>

IX. CONCLUSION

We introduce a new development for particle filters which exploits a sparse kernel density estimate to increase speed of the algorithm while preserving tracking performance. The improvement in speed is due to the sparse representation of the density that uses a smaller number of particles to model the posterior distribution during the filter correction step. The approach additionally provides an explicit analytical form for the posterior PDF which allows for 1) simple determination of the MAP estimate via gradient ascent, 2) multimodal modeling for multi-hypothesis tracking, and 3) several strategies for sampling from the posterior.

ACKNOWLEDGMENT

The authors would like to thank the associate editor, Dr. L. Younes and the reviewers for their thorough reviews and insightful comments to help improve the paper.

REFERENCES


