Sequential Kernel Herding: Frank-Wolfe Optimization for Particle Filtering

Simon Lacoste-Julien
INRIA - Sierra Project-Team
École Normale Supérieure, Paris, France

Fredrik Lindsten
Department of Engineering
University of Cambridge

Francis Bach
INRIA - Sierra Project-Team
École Normale Supérieure, Paris, France

Abstract

Recently, the Frank-Wolfe optimization algorithm was suggested as a procedure to obtain adaptive quadrature rules for integrals of functions in a reproducing kernel Hilbert space (RKHS) with a potentially faster rate of convergence than Monte Carlo integration (and "kernel herding" was shown to be a special case of this procedure). In this paper, we propose to replace the random sampling step in a particle filter by Frank-Wolfe optimization. By optimizing the position of the particles, we can obtain better accuracy than random or quasi-Monte Carlo sampling. In applications where the evaluation of the emission probabilities is expensive (such as in robot localization), the additional computational cost to generate the particles through optimization can be justified. Experiments on standard synthetic examples as well as on a robot localization task indicate indeed an improvement of accuracy over random and quasi-Monte Carlo sampling.

1 Introduction

In this paper, we explore a way to combine ideas from *optimization* with *sampling* to get better approximations in probabilistic models. We consider state-space models (SSMs, also referred to as general state-space hidden Markov models), as they constitute an important class of models in engineering, econometrics and other areas involving time series and dynamical systems. A discrete-time, nonlinear SSM can be written as

$$x_t \,|\, x_{1:(t-1)} \sim p(x_t|x_{t-1}); \quad y_t \,|\, x_{1:t} \sim p(y_t|x_t), \quad (1)$$

where $x_t \in \mathcal{X}$ denotes the latent state variable and $y_t \in \mathcal{Y}$ the observation at time t. Exact state inference in SSMs is possible, essentially, only when the model is linear and Gaussian or when the state-space $\mathcal{X}$ is a finite set. For solving the inference problem beyond these restricted model classes, sequential Monte Carlo methods, i.e. particle filters (PFs), have emerged as a key tool; see e.g., Doucet and Johansen (2011); Cappé et al. (2005); Doucet et al. (2000). However, since these methods are based on Monte Carlo integration they are inherently affected by sampling variance, which can degrade the performance of the estimators.

Particular challenges arise in the case when the *observation likelihood* $p(y_t|x_t)$ is computationally expensive to evaluate. For instance, this is common in robotics applications where the observation model relates the sensory input of the robot, which can comprise vision-based systems, laser rangefinders, synthetic aperture radars, etc. For such systems, simply evaluating the observation function for a fixed value of x_t can therefore involve computationally expensive operations, such as image processing, point-set registration, and related tasks. This poses difficulties for particle-filtering-based solutions for two reasons: *(1)* the computational bottleneck arising from the likelihood evaluation implies that we cannot simply increase the number of particles to improve the accuracy, and *(2)* this type of "complicated" observation models will typically not allow for adaptation of the proposal distribution used within the filter, in the spirit of Pitt and Shephard (1999), leaving us with the standard—but inefficient—*bootstrap proposal* as the only viable option. On the contrary, for these systems, the *dynamical model* $p(x_t|x_{t-1})$ is often comparatively simple, e.g. being a linear and Gaussian "nearly constant acceleration" model (Ristic et al., 2004).

The method developed in this paper is geared toward this class of filtering problems. The basic idea is that, in scenarios when the likelihood evaluation is the computational bottleneck, we can afford to spend additional computations to improve upon the sampling of

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

the particles. By doing so, we can avoid excessive variance arising from simple Monte Carlo sampling from the bootstrap proposal.

Contributions. We build on the optimization view from Bach et al. (2012) of kernel herding (Chen et al., 2010) to approximate the integrals appearing in the Bayesian filtering recursions. We make use of the Frank-Wolfe (FW) quadrature to approximate, in particular, mixtures of Gaussians which often arise in a particle filtering context as the mixture over past particles in the distribution over the next state. We use this approach within a filtering framework and prove theoretical convergence results for the resulting method, denoted as *sequential kernel herding* (SKH), giving one of the first explicit better convergence rates than for a particle filter. Our preliminary experiments show that SKH can give better accuracy than a standard particle filter or a quasi-Monte Carlo particle filter.

2 Adaptive quadrature rules with Frank-Wolfe optimization

2.1 Approximating the mean element for integration in a RKHS

We consider the problem of approximating integrals of functions belonging to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with respect to a *fixed* distribution p over some set $\mathcal{X}$. We can think of the elements of $\mathcal{H}$ as being real-valued functions on $\mathcal{X}$, with pointwise evaluation given from the reproducing property by $f(x) = \langle f, \Phi(x) \rangle$, where $\Phi : \mathcal{X} \to \mathcal{H}$ is the feature map from the state-space $\mathcal{X}$ to the RKHS. Let $\kappa : \mathcal{X}^2 \to \mathbb{R}$ be the associated positive definite kernel. We briefly review here the setup from Bach et al. (2012), which generalized the one from Chen et al. (2010). We want to approximate integrals $\mathbb{E}_p[f]$ for $f \in \mathcal{H}$ using a set of n points $x^{(1)}, \ldots, x^{(n)} \in \mathcal{X}$ associated with positive weights $w^{(1)}, \ldots, w^{(n)}$ which sum to 1:

$$\mathbb{E}_p[f] \approx \sum_{i=1}^{n} w^{(i)} f(x^{(i)}) = \mathbb{E}_{\hat{p}}[f], \tag{2}$$

where $\hat{p} := \sum_{i=1}^{n} w^{(i)} \delta_{x^{(i)}}$ is the associated empirical distribution defined by these points and $\delta_x(\cdot)$ is a point mass distribution at x. If the points $x^{(i)}$ are independent samples from p, then this Monte Carlo estimate (using weights of $1/n$) is unbiased with a variance of $\mathbb{V}_p[f]/n$, where $\mathbb{V}_p[f]$ is the variance of f with respect to p. By using the fact that f belongs to the RKHS $\mathcal{H}$, we can actually choose a better set of points with lower error. It turns out that the worst-case error of estimators of the form (2) can be analyzed in terms of their approximation distance to the *mean element* $\mu(p) := \mathbb{E}_p[\Phi] \in \mathcal{H}$ (Smola et al., 2007; Sriperumbudur et al., 2010). Essentially, by using Cauchy-Schwartz inequality and the linearity of the expectation operator, we can obtain:

$$\sup_{\substack{f \in \mathcal{H} \\ \|f\|_{\mathcal{H}} \leq 1}} |\mathbb{E}_p[f] - \mathbb{E}_{\hat{p}}[f]| = \|\mu(p) - \mu(\hat{p})\|_{\mathcal{H}} =: \mathrm{MMD}(p, \hat{p}), \tag{3}$$

and so by bounding $\mathrm{MMD}(p, \hat{p})$, we can bound the error of approximating the expectation for all $f \in \mathcal{H}$, with $\|f\|_{\mathcal{H}}$ as a proportionality constant. $\mathrm{MMD}(p, \hat{p})$ is thus a central quantity for developing good quadrature rules given by (2). In the context of RKHSs, $\mathrm{MMD}(p, q)$ can be called the *maximum mean discrepancy* (Gretton et al., 2012) between the distributions p and q, and acts a pseudo-metric on the space of distributions on $\mathcal{X}$. If κ is a *characteristic* kernel (such as the standard RBF kernel), then MMD is in fact a metric, i.e. $\mathrm{MMD}(p, q) = 0 \implies p = q$. We refer the reader to Sriperumbudur et al. (2010) for the regularity conditions needed for the existence of these objects and for more details.

2.2 Frank-Wolfe optimization for adaptive quadrature

For getting a good quadrature rule $\hat{p}$, our goal is thus to minimize $\|\mu(\hat{p}) - \mu(p)\|_{\mathcal{H}}$. We note that $\mu(p)$ lies in the *marginal polytope* $\mathcal{M} \subset \mathcal{H}$, defined as the closure of the convex-hull of $\Phi(\mathcal{X})$. We suppose that $\Phi(x)$ is uniformly bounded in the feature space, that is, there is a finite R such that $\|\Phi(x)\|_{\mathcal{H}} \leq R \; \forall x \in \mathcal{X}$. This means that $\mathcal{M}$ is a closed bounded convex subset of $\mathcal{H}$, and we could in theory optimize over it. This insight was used by Bach et al. (2012) who considered using the Frank-Wolfe optimization algorithm to optimize the convex function $J(g) := \frac{1}{2}\|g - \mu(p)\|^2_{\mathcal{H}}$ over $\mathcal{M}$ to obtain adaptive quadrature rules. The Frank-Wolfe algorithm (also called conditional gradient) (Frank and Wolfe, 1956) is a simple first-order iterative constrained optimization algorithm for optimizing *smooth* functions over *closed bounded convex* sets like $\mathcal{M}$ (see Dunn (1980) for its convergence analysis on general infinite dimensional Banach spaces). At every iteration, the algorithm finds a good feasible search *vertex* of $\mathcal{M}$ by minimizing the *linearization* of J at the current iterate g_k: $\bar{g}_{k+1} = \arg\min_{g \in \mathcal{M}} \langle J'(g_k), g \rangle$. The next iterate is then obtained by a suitable convex combination of the search vertex $\bar{g}_{k+1}$ and the previous iterate g_k: $g_{k+1} = (1 - \gamma_k) g_k + \gamma_k \bar{g}_{k+1}$ for a suitable step-size γ_k from a fixed schedule (e.g. $1/(k+1)$) or by using line-search. A crucial property of this algorithm is that the iterate g_k is thus a convex combination of the *vertices* of $\mathcal{M}$ visited so far. This provides a *sparse* expansion for the iterate, and makes the algorithm suitable

to high-dimensional optimization (or even infinite) – this explains in part the regain of interest in machine learning in the last decade for this old optimization algorithm (see Jaggi (2013) for a recent survey). In our setup where $\mathcal{M}$ is the convex hull of $\Phi(\mathcal{X})$, the vertices of $\mathcal{M}$ are thus of the form $\bar{g}_{k+1} = \Phi(x^{(k+1)})$ for some $x^{(k+1)} \in \mathcal{X}$. Running Frank-Wolfe on $\mathcal{M}$ thus yields $g_k = \sum_{i=1}^k w_k^{(i)} \Phi(x^{(i)}) = \mathbb{E}_{\hat{p}}[\Phi]$ for some weighted set of points $\{w_k^{(i)}, x^{(i)}\}_{i=1}^k$. The iterate g_k thus corresponds to a quadrature rule $\hat{p}$ of the form of (2) and $g_k = \mathbb{E}_{\hat{p}}[\Phi]$, and this is the relationship that was explored in Bach et al. (2012). Running Frank-Wolfe optimization with the step-size of $\gamma_k = 1/(k+1)$ reduces to the kernel herding algorithm proposed by Chen et al. (2010). See also Huszár and Duvenaud (2012) for an alternative approach with negative weights.

Algorithm 1 presents the Frank-Wolfe optimization algorithm to solve $\min_{g \in \mathcal{M}} J(g)$ in the context of getting quadrature rules (we also introduce the shorthand notation $\mu_p := \mu(p)$). We note that to evaluate the quality $\mathrm{MMD}(\hat{p}, p)$ of this adaptive quadrature rule, we need to be able to evaluate $\mu_p(x) = \int_{x' \in \mathcal{X}} p(x')\kappa(x', x)dx'$ efficiently. This is true only for specific pairs of kernels and distributions, but fortunately this is the case when p is a mixture of Gaussians and κ is a Gaussian kernel. This insight is central to this paper; we explore this case more specifically in Section 2.3. To find the next quadrature point, we also need to (approximately) optimize $\mu_p(x)$ over $\mathcal{X}$ (step 3 of Algorithm 1, called the FW vertex search). In general, this will yield a non-convex optimization problem, and thus cannot be solved with guarantees, even with gradient descent. In our current implementation, we approach step 3 by doing an exhaustive search over M random samples from p precomputed when FW-Quad is called. We thus follow the idea from the kernel herding paper (Chen et al., 2010) to choose the best N "super-samples" out of a large set of samples M. Thanks to the fact that convergence guarantees for Frank-Wolfe optimization can still be given when using an approximate FW vertex search, we show in Appendix B of the supplementary material that this procedure either adds a $O(1/M^{1/4})$ term or a $O(1/\sqrt{M})$ term to the worst-case $\mathrm{MMD}(\hat{p}, p)$ error.

In our description of Algorithm 1, a preset number N of particles (iterations) was used. Alternatively, we could use a variable number of iterations with the terminating criterion test $\|g_k - \mu(p)\|_{\mathcal{H}} \leq \epsilon$ which can be *explicitly computed during the algorithm* and provides the MMD error bound on the returned quadrature rule. Option (2) on line 5 chooses the step-size γ_k by analytic line-search (hereafter referred as the FW-LS version) while option (1) chooses the kernel herding step-size $\gamma_k = 1/(k+1)$ (herafter referred as the FW version) which always yields uniform weights: $w_k^{(i)} = 1/k$ for all $i \leq k$. A third alternative is to re-optimize $J(g)$ over the convex hull of the previously visited vertices; this is called the fully corrective version (Jaggi, 2013) of the Frank-Wolfe algorithm (hereafter referred as FCFW). In this case: $(w_{k+1}^{(1)}, \ldots, w_{k+1}^{(k+1)}) = \arg\min_{\boldsymbol{w} \in \Delta_{k+1}} \boldsymbol{w}^\top \boldsymbol{K}_{k+1} \boldsymbol{w} - 2\boldsymbol{c}_{k+1}^\top \boldsymbol{w}$, where Δ_{k+1} is the $(k+1)$-dimensional probability simplex, $\boldsymbol{K}_{k+1}$ is the kernel matrix on the $(k+1)$ vertices: $(\boldsymbol{K}_{k+1})_{ij} = \kappa(x^{(i)}, x^{(j)})$ and $(\boldsymbol{c}_{k+1})_i = \mu_p(x^{(i)})$ for $i = 1, \ldots, (k+1)$. This is a convex quadratic problem over the simplex. A slightly modified version of the FCFW is called the min-norm point algorithm and can be more efficiently optimized using specific purpose active-set algorithms — see Bach (2013, §9.2) for more details. We refer the reader to Bach et al. (2012) for more details on the rate of convergence of Frank-Wolfe quadrature assuming that the FW vertex is found with guarantees. We summarize them as follows: if $\mathcal{H}$ is infinite dimensional, then FW-Quad gives the same $O(1/\sqrt{N})$ rate for the MMD error as standard random sampling, for all FW methods. On the other hand, if a ball of non-zero radius centered at μ_p lies within $\mathcal{M}$, then faster rates than random sampling are possible: FW gives a $O(1/N)$ rate whereas FW-LS and FCFW gives exponential convergence rates (though in practice, we often see differences not explained by the theory between these methods).

2.3 Example: mixture of Gaussians

We describe here in more details the Frank-Wolfe quadrature when p is a mixture of Gaussians $p(x) = \sum_{i=1}^K \pi_i \mathcal{N}(x|\mu_i, \Sigma_i)$ for $\mathcal{X} = \mathbb{R}^d$ and κ is the Gaussian kernel $\kappa_\sigma(x, x') := \exp(-\frac{1}{2\sigma^2}\|x - x'\|^2)$. In this case, $\mu_p(x) = \sum_{i=1}^K \pi_i (\sqrt{2\pi}\sigma)^d \mathcal{N}(x|\mu_i, \Sigma_i + \sigma^2 I_d)$. We thus need to optimize a difference of mixture of Gaussian bumps in step 3 of Algorithm 1, a non-convex optimization problem that we approximately solve by exhaustive search over M random samples from p.

3 Sequential kernel herding

3.1 Sequential Monte Carlo

Consider again the SSM in (1). The joint probability density function for a sequence of latent states $x_{1:T} := (x_1, \ldots, x_T)$ and observations $y_{1:T}$ factorizes as $p(x_{1:T}, y_{1:T}) = \prod_{t=1}^T p(x_t|x_{t-1})p(y_t|x_t)$, with $p(x_1|x_0) := p(x_1)$ denoting the prior density on the initial state. We would like to do approximate inference in this SSM. In particular, we could be interested in computing the joint filtering distribution $r_t(x_{1:t}) := p(x_{1:t}|y_{1:t})$ or the joint predictive distribution $p_{t+1}(x_{t+1}, x_{1:t}) := p(x_{t+1}, x_{1:t}|y_{1:t})$. In parti-

Algorithm 1 FW-Quad(p, $\mathcal{H}$, N): Frank-Wolfe adaptive quadrature

Input: distribution p, RKHS $\mathcal{H}$ which defines kernel $\kappa(\cdot,\cdot)$ and state-space $\mathcal{X}$, number of samples N

1: Let $g_0 = 0$.
2: **for** $k = 0 \dots N-1$ **do**
3: Solve $x^{(k+1)} = \arg\min_{x\in\mathcal{X}} \langle g_k - \mu_p, \Phi(x)\rangle$
That is:
$$x^{(k+1)} = \arg\min_{x\in\mathcal{X}} \sum_{i=1}^{k} w_k^{(i)} \left(\kappa(x^{(i)}, x) - \mu_p(x)\right).$$
4: Option (1): Let $\gamma_k = \frac{1}{k+1}$.
5: Option (2): Let $\gamma_k = \frac{\langle g_k - \mu_p, g_k - \Phi(x^{(k+1)})\rangle}{\|g_k - \Phi(x^{(k+1)})\|^2}$ (LS)
6: Update $g_{k+1} = (1-\gamma_k) g_k + \gamma_k \Phi(x^{(k+1)})$
i.e. $w_{k+1}^{(k+1)} = \gamma_k$;
and $w_{k+1}^{(i)} = (1-\gamma_k) w_k^{(i)}$ for $i = 1 \dots k$
7: **end for**
8: **Return:** $\hat{p} = \sum_{i=1}^{N} w_N^{(i)} \delta_{x^{(i)}}$

Algorithm 2 Particle filter template (joint predictive distribution form) — SKH alg. by changing step 3

Input: SSM $p(x_t|x_{t-1})$,
$o_t(x_t) := p(y_t|x_t)$ for $t \in 1:T$.
Maintain $\hat{p}_t(x_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{x_{1:t}^{(i)}}(x_{1:t})$ during algorithm as approximation of $p(x_t, x_{1:(t-1)}|y_{1:(t-1)})$.

1: Let $\tilde{p}_1(x_1) := p(x_1)$
2: **for** t=1 …, T **do**
3: Sample: get $\hat{p}_t = \text{SAMPLE}(\tilde{p}_t, N)$
[For SKH, use $\hat{p}_t = \text{FW-Quad}(\tilde{p}_t, \mathcal{H}_t, N)$]
4: Include observation and normalize:
$\hat{W}_t = \mathbb{E}_{\hat{p}_t}[o_t]$; $\hat{r}_t(x_{1:t}) := \frac{1}{\hat{W}_t} o_t(x_t) \hat{p}_t(x_{1:t})$.
5: Propagate approximation forward:
$\tilde{p}_{t+1}(x_{t+1}, x_{1:t}) := p(x_{t+1}|x_t) \hat{r}_t(x_{1:t})$
6: **end for**
7: **Return** Filtering distribution $\hat{r}_T$; predictive distribution $\hat{p}_{T+1}$; normalization constants $\hat{W}_1, \dots, \hat{W}_T$.

cle filtering methods, we approximate these distributions with empirical distributions from weighted particle sets $\{w_t^{(i)}, x_{1:t}^{(i)}\}_{i=1}^N$ as in (2). We note that it is easy to marginalize $\hat{p}$ with a simple weight summation, and so we will present the algorithm as getting an approximation for the *joint* distributions r_t and p_t defined above, with the understanding that the marginal ones are easy to obtain afterwards. In the terminology of particle filtering, $x_t^{(i)}$ is the particle at time t, whereas $x_{1:t}^{(i)}$ is the *particle trajectory*. While principally the PF provides an approximation of the full joint distribution $r_t(x_{1:t})$, it is well known that this approximation deteriorates for any marginal of x_s for $s \ll t$ (Doucet and Johansen, 2011). Hence, the PF is typically only used to approximate marginals of x_s for $s \lesssim t$ (fixed-lag smoothing) or $s = t$ (filtering), or for prediction.

Algorithm 2 presents the bootstrap particle filtering algorithm (Gordon et al., 1993) from the point of view of propagating an approximate posterior distribution forward in time (see e.g. Fearnhead, 2005). We describe it as propagating an approximation $\hat{p}_t(x_{1:t})$ of the joint predictive distribution one time step forward with the model dynamics to obtain $\tilde{p}_{t+1}(x_{t+1}, x_{1:t})$ (step 5), and then randomly sampling from it (step 3) to get the new predictive approximation $\hat{p}_{t+1}(x_{t+1}, x_{1:t})$. As $\hat{p}_t$ is an empirical distribution, $\tilde{p}_{t+1}$ is a mixture distribution (the mixture components are coming from the particles at time t):

$$\tilde{p}_{t+1}(x_{t+1}, x_{1:t}) = \frac{1}{\hat{W}_t} \sum_{i=1}^{N} \underbrace{p(y_t|x_t^{(i)}) w_t^{(i)}}_{\text{mixture weight}} \underbrace{p(x_{t+1}|x_t^{(i)})}_{\text{mixture component}} \delta_{x_{1:t}^{(i)}}(x_{1:t}). \quad (4)$$

We denote the conditional normalization constant at time t by $W_t := p(y_t|y_{1:(t-1)})$ and the global normalization constant by $Z_t := p(y_{1:t}) = \prod_{u=1}^{t} W_u$. $\hat{W}_t$ is the particle filter approximation to W_t and is obtained by summing the un-normalized mixture weights in (4); see step 4 in Algorithm 2. Randomly sampling from (4) is equivalent to first sampling a mixture component according to the mixture weight (i.e., choosing a past particle $x_{1:t}^{(i)}$ to propagate), and then sampling its next extension state $x_{t+1}^{(i)}$ with probability $p(x_{t+1}|x_t^{(i)})$. The standard bootstrap particle filter is thus obtained by maintaining uniform weight for the predictive distribution ($w_t^{(i)} = \frac{1}{N}$) and randomly sampling from (4) to obtain the particles at time $t+1$. This gives an unbiased estimate of $\tilde{p}_{t+1}$: $\mathbb{E}_{\tilde{p}_{t+1}}[\hat{p}_{t+1}] = \tilde{p}_{t+1}$. Lower variance estimators can be obtained by using a different resampling mechanism for the particles than this multinomial sampling scheme, such as stratified resampling (Carpenter et al., 1999) and are usually used in practice instead.

One way to improve the particle filter is thus to replace the random sampling stage of step 3 with different sampling mechanisms with lower variance or better approximation properties of the distribution $\tilde{p}_{t+1}$ that we are trying to approximate. As we obtain the normalization constants W_t by integrating the observation probability, it seems natural to look for particle point sets with better integration properties. By replacing random sampling with a quasi-random number sequence, we obtain the already proposed sequential quasi-Monte Carlo scheme (Philomin et al., 2000; Ormoneit et al., 2001; Gerber and Chopin, 2014). The

main contribution of our work is to instead propose to use Frank-Wolfe quadrature in step 3 of the particle filter to obtain better (adapted) point sets.

3.2 Sequential kernel herding

In the sequential kernel herding (SKH) algorithm, we simply replace step 3 of Algorithm 2 with $\hat{p}_t = \text{FW-Quad}(\tilde{p}_t, \mathcal{H}_t, N)$. As mentioned in the introduction, many dynamical models used in practice assume Gaussian transitions. Therefore, we will put particular emphasis on the case when (more generally) $p(x_t|x_{1:(t-1)}, y_{1:(t-1)})$ is a mixture of Gaussians, with parameters for the mixture components that can be arbitrary functions of the state history $x_{1:(t-1)}, y_{1:(t-1)}$, and is thus still fairly general. We thus consider the Gaussian kernel for the FW-Quad procedure as then we can compute the required quantities analytically. An important subtle point is which Hilbert space $\mathcal{H}_t$ to consider. In this paper, we focus on the *marginalized* filtering case, i.e. we are interested in $p(x_t|y_{1:t})$ only. Thus we are only interested in functions of x_t, which is why we define our kernel at time t to only depend on x_t and not the past histories. For simplicity, we also assume that $\mathcal{H}_t = \mathcal{H}$ for all t (we use the same kernel for each time step). Even though the algorithm can maintain the distribution on the whole history $\hat{p}_t(x_{1:t})$, the past histories $x_{1:(t-1)}$ are marginalized out when computing the mean map, for example $\mu(\tilde{p}_t) = \mathbb{E}_{\tilde{p}_t(x_{1:t})}[\Phi(x_t)]$. During the SKH algorithm, we can still track the particle histories by keeping track from which mixture component in (4) x_t was coming from, but the past history is not used in the computation of the kernel and thus does not appear as a repulsion term in step 3 of Algorithm 1. We leave it as future work to analyze what kind of high-dimensional kernel on past histories would make sense in this context, and to analyze its convergence properties. The particle histories are useful in the Rao-Blackwellized extension that we present in Appendix A and use in the robot localization experiment of Section 4.3.

3.3 Convergence theory

In this section, we give sufficient conditions to guarantee that SKH is consistent as N goes to infinity. Let p_t here denote the *marginalized* predictive instead of the joint. Let F_t be the forward transformation operator on signed measures that takes the predictive distribution p_t on x_t and yields the unnormalized marginalized predictive distribution $F_t p_t$ on x_{t+1} in the SSM. Thus for a measure ν, we get $(F_t\nu)(\cdot) := \int_{\mathcal{X}_t} p(\cdot|x_t)p(y_t|x_t)d\nu(x_t)$. We also have that $p_{t+1} = \frac{1}{W_t} F_t p_t$.

For the following theorem, $\mathcal{F}_t$ is a function space on $\mathcal{X}_{t+1}$ defined (depending on $\mathcal{H}_{t+1}$) as all functions for which the following semi-norm is finite:[1]

$$\|f\|_{\mathcal{F}_t} := \sup_{\|h\|_{\mathcal{H}_{t+1}}=1} \left| \int_{\mathcal{X}_{t+1}} f(x_{t+1})h(x_{t+1})dx_{t+1} \right|.$$

Theorem 1 (Bounded growth of the mean map). *Suppose that the function $f_t : (x_{t+1}, x_t) \mapsto p(y_t|x_t)p(x_{t+1}|x_t)$ is in the tensor product function space $\mathcal{F}_t \otimes \mathcal{H}_t$ with the following defined nuclear norm: $\|f_t\|_{\mathcal{F}_t\otimes\mathcal{H}_t} := \inf \sum_i \|\alpha_i\|_{\mathcal{F}_t}\|\beta_i\|_{\mathcal{H}_t}$, where the infimum is taken over all the possible expansions such that $f_t(x_{t+1}, x_t) = \sum_i \alpha_i(x_{t+1})\beta_i(x_t)$ for all x_t, x_{t+1}. Then for any finite signed Borel measure ν on $\mathcal{X}_t$, we have:*

$$\|\mu(F_t\nu)\|_{\mathcal{H}_{t+1}} \leq \|f_t\|_{\mathcal{F}_t\otimes\mathcal{H}_t} \|\mu(\nu)\|_{\mathcal{H}_t}.$$

Theorem 2 (Consistency of SKH). *Suppose that for all $1 \leq t \leq T$, f_t is in $\mathcal{F}_t \otimes \mathcal{H}_t$ as defined in Theorem 1 and o_t is in $\mathcal{H}_t$. Then we have:*[2]

$$\|\mu(\hat{p}_T) - \mu(p_T)\|_{\mathcal{H}_T} \leq$$
$$\hat{\epsilon}_T + \left(R\frac{\|o_{T-1}\|_{\mathcal{H}_{T-1}}}{W_{T-1}} + \rho_{T-1}\right) \sum_{t=1}^{T-1} \chi_t \hat{\epsilon}_t \left(\prod_{k=t}^{T-2} \rho_k\right),$$

where $\rho_t := \frac{\|f_t\|_{\mathcal{F}_t\otimes\mathcal{H}_t}}{W_t}$, $\chi_t := \prod_{k=1}^{t-1} \frac{\hat{W}_k}{W_k}$ and $\hat{\epsilon}_t$ is the FW error reported at time t by the algorithm: $\hat{\epsilon}_t := \|\mu(\hat{p}_t) - \mu(\tilde{p}_t)\|_{\mathcal{H}_t}$.

We note that $\chi_t \approx 1$ as we expect the errors on W_k to go in either direction, and thus to cancel each other over time (though in the worst case it could grow exponentially in t). If $\hat{\epsilon}_t \leq \epsilon$ and $\rho_t \leq \rho$, we basically have $\|\mu(\hat{p}_T) - \mu(p_T)\| = O(\rho^T\epsilon)$ if $\rho > 1$; $O(T\epsilon)$ if $\rho = 1$; and $O(\epsilon)$ if $\rho < 1$ (a contraction). The exponential dependence in T is similar as for a standard particle filter for general distributions; see Douc et al. (2014) though for conditions to get a contraction for the PF.

Importantly, for a fixed T it follows that the rates of convergence for Frank-Wolfe in N translates to rates of errors for integrals of functions in $\mathcal{H}$ with respect to the predictive distribution p_T. Thus if we suppose that $\mathcal{H}$ is finite dimensional, that p_t has full support on $\mathcal{X}$ for all t and that the kernel κ is continuous, then by Proposition 1 in Bach et al. (2012), we have that the faster rates for Frank-Wolfe hold and in particular we could obtain an error bound of $O(1/N)$ with N particles. As far as we know, this is the first explicit faster rates of convergence as a function of the number

[1] In general, the integral on $\mathcal{X}_{t+1}$ should be with respect to the base measure for which the conditional density $p(x_{t+1}|x_t)$ is defined. All proofs are in the supplementary material.

[2] We use the convention that the empty sum is 0 and the empty product is 1.

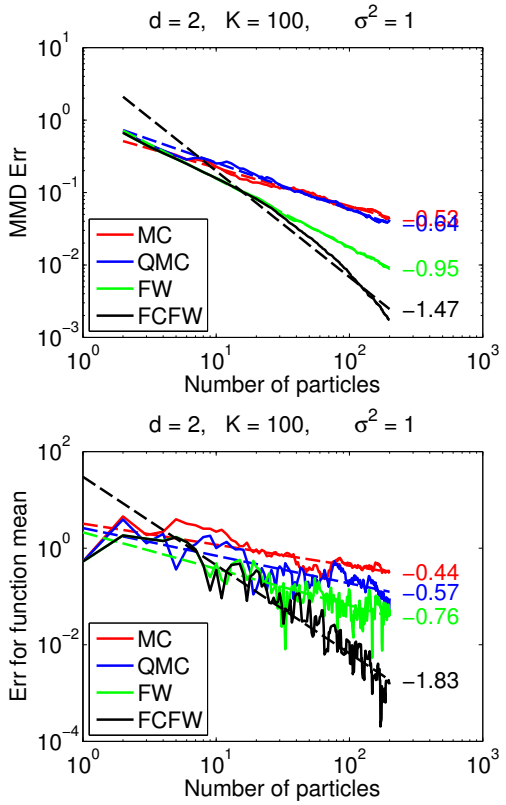

Figure 1: Top: MMD error for different sampling schemes where p is a mixture of 2d Gaussians with $K = 100$ components. Bottom: error on the mean estimate for the same mixture. The dashed lines are linear fits with slopes reported next to the axes.

of particles than the standard $O(\frac{1}{\sqrt{N}})$ for Monte Carlo particle filters. In contrast, Gerber and Chopin (2014, Theorem 7) showed a $o(\frac{1}{\sqrt{N}})$ rate for the randomized version of their SQMC algorithm (note the little-o).[3] Note that the theorem does not depend on how the error of ϵ is obtained on the mean maps of the distribution; and so if one could show that a QMC point set could also achieve a faster rate for the error on the *mean maps* (rather than on the distributions itself as is usually given), then their rates would translate also to the global rate by Theorem 2.[4]

4 Experiments

4.1 Sampling from a mixture of Gaussians

We start by investigating the merits of different sampling schemes for approximating mixtures of Gaussians, since this is an intrinsic step to the SKH algorithm. In Figure 1, we give the MMD error as well as the error on the mean function in term of the number of particles N for the different sampling schemes on a randomly chosen mixture of Gaussians with $K = 100$ components in $d = 2$ dimensions. Additional results as well as the details of the model are given in Appendix C.1 of the supplementary material. In our experiments, the number of FW search points is $M = 50{,}000$. We note that even though in theory all methods should have the same rate of convergence $O(1/\sqrt{N})$ for the MMD (as $\mathcal{H}$ is infinite dimensional), FCFW empirically improves significantly over the other methods. As d increases, the difference between the methods tapers off for a fixed kernel bandwidth σ^2, but increasing σ^2 gives better results for FW and FCFW than the other schemes.

In the remaining sections, we evaluate empirically the application of kernel herding in a filtering context using the proposed SKH algorithm.

4.2 Particle filtering using SKH on synthetic examples

We consider first several synthetic data sets in order to assess the improvements offered by Frank-Wolfe quadrature over standard Monte Carlo and quasi-Monte-Carlo techniques. We generate data from four different systems (further details on the experimental setup can be found in Appendix C.2):

Two linear Gaussian state-space (LGSS) models of dimensions $d = 3$ and $d = 15$, respectively.

A jump Markov linear system (JMLS), consisting of 2 interacting LGSS models of dimension $d = 2$. The switching between the models is governed by a *hidden* 2-state Markov chain.

A nonlinear benchmark time-series model used by, among others, Doucet et al. (2000); Gordon et al. (1993). The model is of dimension $d = 1$ and is given by:

$$x_{t+1} = 0.5x_t + 25\frac{x_t}{1+x_t^2} + 8\cos(1.2t) + v_t,$$
$$y_t = 0.05x_t^2 + e_t,$$

with v_t and e_t mutually independent standard Gaussian.

These models are ordered in increasing levels of difficulty for inference. For the LGSS models, the exact filtering distributions can be computed by a Kalman filter. For the JMLS, this is also possible by running a mixture of Kalman filters, albeit at a computational cost of 2^T (where T is the total number of time steps). For the nonlinear system, no closed form expressions

[3] The rate holds on the approximation of integrals of continuous bounded functions.

[4] We also note that a simple computation shows that for a Monte Carlo sample of size N, $\mathbb{E}\|\mu(\hat{p}) - \mu(p)\|_{\mathcal{H}}^2 \leq \frac{(R^2 - \|\mu(p)\|^2)}{N}$.

are available for the filtering densities; instead we run a PF with $N = 100{,}000$ particles as a reference.

We generate 30 batches of observations for $T = 100$ time steps from all systems, except for the JMLS where we use $T = 10$ (to allow exact filtering). We run the proposed SKH filter, using both FW and FCFW optimization and compare against a bootstrap PF (using stratified resampling (Carpenter et al., 1999)) and a quasi-Monte-Carlo PF based on a Sobol-sequence point-set. All methods are run with N varying from 20 to 200 particles. We deliberately use rather few particles since, as discussed above, we believe that this is the setting when the proposed method can be particularly useful.

To assess the performances of the different methods, we first compute the root-mean-squared errors (RMSE) for the filtered mean-state-estimates over the T time steps, w.r.t. the reference filters. We report the median RMSEs over the 30 *different* data batches, along with the 25% and 75% quantiles, and the minimum and maximum values in Figure 2. The SKH algorithms were run for three different values of $\sigma^2 \in \{0.01, 0.1, 1\}$. Here, we report the results for $\sigma^2 = 1$ for the LGSS models and the JMLS, and for $\sigma^2 = 0.1$ for the nonlinear benchmark model. The results for the other values are given in Appendix C.2. The improvements are somewhat robust to the value of σ^2, but in some cases significant differences were observed. As can be seen, both SKH methods improve significantly upon both QMC and the bootstrap PF. For the two LGSS models, we also compute the MMD (reported in the rightmost column in Figure 2).

4.3 Vision-based UAV Localization

In this section, we apply the proposed SKH algorithm to solve a filtering problem in field robotics. We use the data and the experimental setup described by Törnqvist et al. (2009). The problem consists of estimating the full six-dimensional pose of an unmanned aerial vehicle (UAV).

Törnqvist et al. (2009) proposed a vision-based solution, essentially tracking interest points in the camera images over consecutive frames to estimate the egomotion. This information is then fused with the inertial and barometer sensors to estimate the pose of the UAV. The system is modelled on state-space form, with a state vector comprising the position, velocity, acceleration, as well as the orientation and the angular velocity of the UAV. The state is also augmented with sensor biases, resulting in a state dimension of 22. Furthermore, the state is augmented with the three-dimensional positions of the interest points that are currently tracked by the vision system; this is a varying number but typically around ten.

To deal with the high-dimensional state-vector, Törnqvist et al. (2009) used a Rao-Blackwellized PF (see Appendix A) to solve the filtering problem, marginalizing all but 6 state components (being the pose, i.e., the position and orientation) using a combination of Kalman filters and extended Kalman filters. The remaining 6 state-variables were tracked using a bootstrap particle filter with $N = 200$ particles; the strikingly small number of particles owing to the computational complexity of the likelihood evaluation.

For the current experiment, we obtained the code and the flight-test data from Törnqvist et al. (2009). The modularity of our approach allowed us to simply replace the Monte Carlo simulation step within their setup with FW-Quad. We ran SKH-FW with $\sigma^2 = 10$ and SKH-FCFW with $\sigma^2 = 0.1$, as well as the bootstrap PF used in Törnqvist et al. (2009), and a QMC-PF; all methods using $N = 50$, 100, and 200 particles. We ran all methods 10 times on the same data; the variation in SKH coming from the random search points for the FW procedure, and in QMC for starting the Sobol sequence at different points. For comparison, we ran 10 times a reference PF with $N = 100{,}000$ particles and averaged the results. The median position errors for 100 seconds of robot time (there are 20 SSM time steps per second of robot time) are given in Figure 3. The UAV is assumed to start at a known location at time zero, hence, all the errors are zero initially. Note that all methods accumulate errors over time. This is natural, since there is no absolute position reference available (i.e., the filter is unstable) and the objective is basically to keep the error as small as possible for as long time as possible. SKH-FW here gives the overall best results, with significant improvements over the bootstrap PF and the QMC methods for small number of particles. SKH-FW even gives similar errors for the last time step with only $N = 200$ particles as one of the *reference* PFs (using $N = 100{,}000$ particles). See Appendix C.2.1 for a discussion of the role of σ^2 for FCFW.

Runtimes. In these experiments, we focused on investigating how optimization could improve the error per particle, as the gain in runtime depends on the exact implementation as well as the likelihood evaluation cost. We note that the FW-Quad algorithm scales as $O(NM)$ for N samples and M search points when using FW, by updating the objective on the M search points in an online fashion (we also empirically observed this linear scaling in N). On the other hand, FCFW scales as $O(N^2M)$ as the weights on the particles possibly change at each iteration, preventing the same online trick. SKH scales linearly with the number of time steps T (as a standard PF). For the UAV

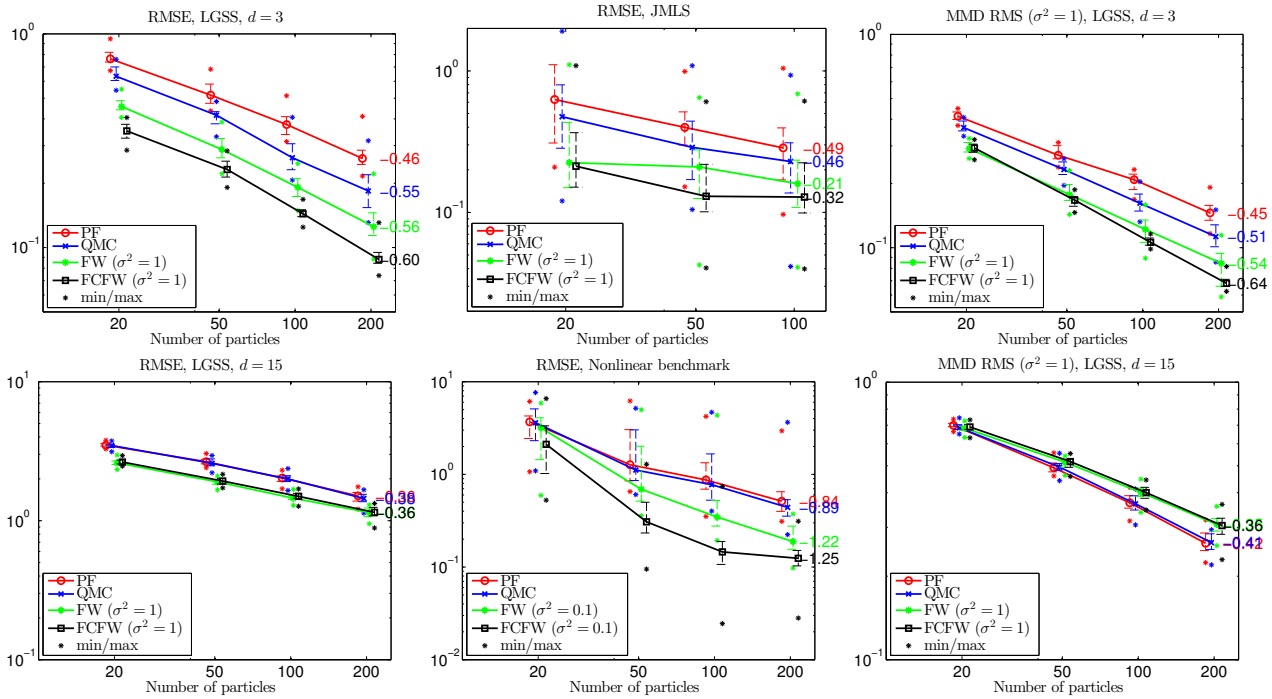

Figure 2: RMSEs (left and middle columns) for the four considered models and MMDs (right column) for the two LGSS models.

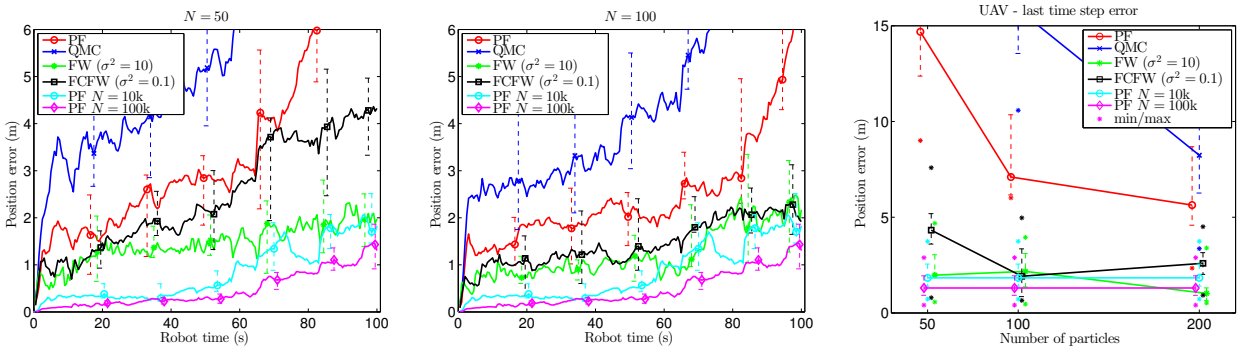

Figure 3: Median of position errors over 10 runs for each method. The errors are computed relative to the mean prediction over 10 runs of a PF with 100k particles (the variation of the reference PF is also shown for PF 100k). The error bars represent the [25%, 75%] quantile. The rightmost plot shows the error at the last time step as a function of N. 100 s of robot time represents 2,000 SSM time steps, it *does not* correspond to computation time.

application, the original Matlab code from Törnqvist et al. (2009) spent an average of 0.2 s per time step for $N = 50$ particles (linear in the number of particles as the likelihood evaluation is the bottleneck) on a XEON E5-2620 2.10 GHz PC. The overhead of using our Matlab implementation of FW-Quad with $N = 50$ is about 0.15 s per time step for FW and 0.3 s for FCFW; and 0.3 s for FW and 1.0 s for FCFW for $N = 100$ (we used $M = 10{,}000$ search points in this experiment). In practice, this means that SKH-FW can be run here with 50 particles in the same time as the standard PF is run with about 90 particles. But as Figure 3 shows, the error for SKH-FW with 50 particles is still much lower than the PF with 200 particles.

5 Conclusion

We have developed a method for Bayesian filtering problems using a combination of optimization and particle filtering. The method has been demonstrated to provide improved performance over both random sampling and quasi-Monte Carlo methods. The proposed method is modular and it can be used with different types of particle filtering techniques, such as the Rao-Blackwellized particle filter. Further investigating this possibility for other classes of particle filters is a topic for future work. Future work also includes a deeper analysis of the convergence theory for the method in order to develop practical guidelines for the choice of the kernel bandwidth.

Acknowledgements

We thank Eric Moulines for useful discussions. This work was partially supported by the MSR-Inria Joint Centre, a grant by the European Research Council (SIERRA project 239993) and by the Swedish Research Council (project *Learning of complex dynamical systems* number 637-2014-466).

References

F. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373, 2013.

F. Bach, S. Lacoste-Julien, and G. Obozinski. On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1359–1366, 2012.

O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005.

J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings Radar, Sonar and Navigation*, 146(1):2–7, 1999.

Y. Chen, M. Welling, and A. Smola. Supersamples from kernel herding. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2010.

R. Douc, E. Moulines, and J. Olsson. Long-term stability of sequential Monte Carlo methods under verifiable conditions. *Annals of Applied Probability*, 24 (5):1767–1802, 2014.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, S. J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

J. C. Dunn. Convergence rates for conditional gradient sequences generated by implicit step length rules. *SIAM Journal on Control and Optimization*, 18:473–487, 1980.

P. Fearnhead. Using random quasi-Monte-Carlo within particle filters, with application to financial time series. *Journal of Computational and Graphical Statistics*, 14(4):751–769, 2005.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

M. Gerber and N. Chopin. Sequential quasi-Monte Carlo. *arXiv preprint arXiv:1402.4039v5*, 2014.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, Apr. 1993.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13:723–773, 2012.

F. Huszár and D. Duvenaud. Optimally-weighted herding is Bayesian quadrature. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 377–385, 2012.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

D. Ormoneit, C. Lemieux, and D. J. Fleet. Lattice particle filters. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 395–402, 2001.

V. Philomin, R. Duraiswami, and L. Davis. Quasi-random sampling for condensation. In *Proceedings of the 6th European Conference on Computer Vision (ECCV)*, 2000.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: particle filters for tracking applications*. Artech House, London, UK, 2004.

A. Smola, A. Gretton, L. Song, and B. Schölkopf. A Hilbert space embedding for distributions. In *Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 99:1517–1561, 2010.

D. Törnqvist, T. B. Schön, R. Karlsson, and F. Gustafsson. Particle filter SLAM with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 55(4):249–266, 2009.