Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces

Marinho Z.
Robotics Institute, CMU
zmarinho@cmu.edu

Dragan A.
Robotics Institute, CMU
adragan@cs.cmu.edu

Byravan A.
University of Washington
barun@uw.edu

Srinivasa S.
Robotics Institute, CMU
siddh@cs.cmu.edu

Gordon G.
Robotics Institute, CMU
ggordon@cs.cmu.edu

Boots B.
Georgia Institute of Technology
bboots@cc.gatech.edu

Abstract

We introduce a functional gradient descent based trajectory optimization algorithm for robot motion planning in arbitrary Reproducing Kernel Hilbert Spaces (RKHSs). Functional gradient algorithms are a popular choice for motion planning in complex many degree-of-freedom robots. In theory, these algorithms work by directly optimizing continuous trajectories to avoid obstacles while maintaining smoothness. However, *in practice*, functional gradient algorithms *commit to a finite parametrization* of the trajectories, often as a finite set of waypoints. Such a parametrization limits expressiveness, and can fail to produce smooth trajectories despite the inclusion of smoothness in the objective. As a result, we often observe practical problems such as slow convergence and the requirement to choose an inconveniently small step size. Our work generalizes the waypoint parametrization to arbitrary RKHSs by formulating trajectory optimization as minimization of a cost functional. We derive a gradient update method that is able to take larger steps and achieve a locally minimum trajectory in just a few iterations. Depending on the selection of a kernel, we can directly optimize in spaces of *continuous* trajectories that are *inherently smooth*, and that have a *low-dimensional*, adaptively chosen parametrization. Our experiments illustrate the effectiveness of the planner for two different kernels, RBFs and B-splines, as compared to the standard discretized waypoint representation.

1 Introduction

Motion planning is a critical aspect of robotics. For a given task, motion planning algorithms ensure that robots are able to safely move from a start to goal configuration without colliding with obstacles. *Trajectory optimizers* used in motion planning focus on finding feasible trajectories that are also *efficient*. Recently, trajectory optimization approaches to motion planning have demonstrated great success in a number of high-dimensional real-world problems [1–5]. Some of these algorithms rely on gradient information of a cost function to perform a gradient-based approach to motion planning, that searches for smooth, collision-free trajectories through a manipulator's configuration space [2, 6]. In this work we exploit the same gradient optimization techniques for ensuring collision free trajectories in configuration space, with a novel approach of trajectory representation. Previous work is derived for continuous trajectories in Hilbert spaces, *in practice* they must commit to a parametrization of the trajectories in order to instantiate a gradient update [2, 13, 14]. Until now, the only parametrization supported was a large but finite set of discretized points over time, waypoints. The number of waypoints chosen to represent a trajectory trades off between computational complexity and trajectory expressiveness. Our work frees the optimizer from a discrete parametrization, enabling it to perform gradient descent on a much more general trajectory parametrization: Reproducing Kernel Hilbert Spaces (RKHSs) [7–9], of which waypoint parametrizations are merely one instance. RKHSs impose just enough structure on generic Hilbert spaces to enable a concrete and implementable gradient update rule, while leaving the choice of a parametrization flexible: different kernels lead to different parametrizations. Our contribution is two-fold. Our *theoretical* contribution is the formulation of functional gradient descent motion planning in RKHSs: we formulate trajectory optimization as the minimization of a regularized cost functional. Regularizing the norm in the RKHS is a common way to ensure smoothness in function approximation [10], and we apply the same idea to trajectories.The norm of the associated space is able to quantify different features of trajectories, such as any n-th order derivative [11]. A regularization with respect to a norm in RKHS is able to constrain the solution to a smooth trajectories in the sense of low velocity, acceleration, jerk.

Our *practical* contribution is the ability to perform motion planning in inherently smooth spaces of trajectories with low-dimensional parametrizations. Unlike discretized parametrizations, which require many points to produce smooth trajectories, RKHSs represent smooth trajectories with only a few point evaluations. The inherent smoothness of such spaces increases efficiency because it allows the optimizer to take large steps at every iteration without breaking trajectory smoothness, therefore converging to a collision-free trajectory faster. Our experiments (Section 4) illustrate these advantages of RKHSs over the waypoint parametrization, and compare different choices of kernels.

2 Previous Work

Our RKHS planning algorithm draws on CHOMP [2]. CHOMP is a functional gradient based optimization algorithm that minimizes a cost functional to produce a smooth collision-free trajectory. It operates in a Hilbert space of trajectories Ξ, with elements $\xi : [0,1] \to \mathcal{C}$, mapping time to robot configurations. The Hilbert space Ξ is equipped with an inner product $\langle \xi_1, \xi_2 \rangle_\Xi = \langle \xi_1, A\xi_2 \rangle$. For example A is the Laplacian operator ∇^2 for $\Xi = \mathcal{L}_2^d$, the space of d-dimensional trajectories whose components are square-integrable. In the waypoint representation, A is typically the Hessian matrix over points in the trajectory, which makes the norm in Ξ penalize unsmooth and inefficient trajectories. CHOMP finds the trajectory that minimizes this functional cost $\mathcal{U}$ by performing a line search over the negative gradient direction, where A dictates the shape of the manifold over trajectories. In practice, this algorithm commits to a waypoint parametrization of the trajectory, *i.e.* trajectories are finite vectors of discretized waypoints [2, 13, 14]. CHOMP requires a commitment to a parametrization in order to instantiate the gradient computation. In the following section, we derive a directly instantiable gradient update for trajectories parametrized as part of an RKHS, which generalizes the waypoint parametrization.

3 Motion Planning in an RKHS

3.1 Trajectories in an RKHS

An RKHS of trajectories $\mathcal{H}$ is much like the Hilbert space of trajectories with elements $\xi : [0,1] \to \mathcal{C}$, mapping time to robot configurations, but with additional structure [7, 15, 16]. Trajectories $\xi \in \mathcal{H}$ in an RKHS can be represented as a sum of point evaluation functions $k(t_i, \cdot) \in \mathcal{H}$ [9], where $k : [0,1] \times [0,1] \to \mathbb{R}$ is the kernel associated with the RKHS. Here $\xi \equiv \xi(\cdot)$ denotes a trajectory as a function in the Hilbert space of trajectories. $\xi(t) \in \mathcal{C}$ corresponds to a robot configuration evaluated at time t, which in a coordinate independent notation is equivalent to a value in $\mathbb{R}$. A function evaluation and the inner product in $\mathcal{H}$ of two functions $\xi_1(\cdot) = \sum_i a_i k(t_i, \cdot)$ and $\xi_2(\cdot) = \sum_j b_j k(t_j, \cdot)$ are defined as:

$$\xi(\cdot) = \sum_i a_i k(t_i, \cdot) \quad , \qquad\qquad \langle \xi_1, \xi_2 \rangle_{\mathcal{H}} = \sum_{i,j} a_i b_j k(t_i, t_j) \tag{1}$$

In this paper we represent trajectories as real valued functions, describing a single robot degree of freedom separately (a function per DOF): Ξ is the product of d RKHSs of 1-dimensional functions, so that the full kernel function is a sum over each coordinate i, $k(t, t') = \sum_i k_i(t, t')$. This implies that the norm in this space cannot have interactions between the different joints. We could extend this derivation to RKHSs of vector-valued functions [17, 18], we leave that as possible future work.

Cost Functional. We present a cost functional over the space of robot configurations $\mathcal{U} : \Xi \to \mathbb{R}$ which maps each trajectory to a scalar cost. $\mathcal{U}$ trades off between a regularization term that measures the shape of the trajectory, and an obstacle term that measures its proximity to obstacles:

$$\mathcal{U}[\xi] = \mathcal{U}_{obs}[\xi] + \beta \|\xi\|_{\mathcal{H}} \tag{2}$$

A key component of our work is formulating trajectory optimization as the minimization of a *regularized cost functional* (4). This enables us to take advantage of the Representer Theorem [8, 12], which *guarantees that the optimal trajectory will be described in terms of finite point evaluations* $\mathcal{T} = \{t_j\} : j \in [N]$:

$$\xi^*(\cdot) = \sum_j \alpha_j k(t_j, \cdot), \quad \alpha_j \in \mathbb{R} \tag{3}$$

We introduce a cost functional trading off between an obstacle term and a norm regularization (2):

$$\mathcal{U}[\xi] = \sum_j c\left(\, x(\xi(t_j), u_j) \, \right) + \frac{\beta}{2} \|\xi\|^2_{\mathcal{H}} \tag{4}$$

While we ensure smoothness by constraining the norm of the trajectory in the RKHS, we define the obstacle term to become a functional over a *finite* set of time points that is able to avoid obstacles. The obstacle cost functional (2) is defined on trajectories, but obstacles live in the robot's workspace $\mathcal{W} \equiv \mathbb{R}^3$. In (4) we use a cost field in the workspace, $c : \mathcal{W} \to \mathbb{R}$, computed based on a signed distance field which measures how far the closest obstacle boundary is: positive outside of obstacles, and negative inside. This field is usually computed offline for efficiency [2]. x represents the forward kinematics function that maps a configuration in $\mathcal{C}$ and a body point on the robot in $\mathcal{B}$ to the location of that body point in $\mathcal{W}$, and access its cost in c. Previous work defines a similar cost functional in terms of the arc-length integral of the trajectory [6], but this functional could not, as easily, be described in terms of point evaluations. Here we choose instead a cost functional that is given in terms of point evaluations in the RKHS and still achieves a collision free trajectory.

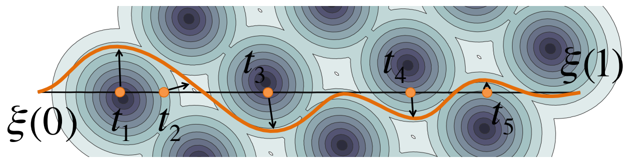


Figure 1: At every iteration, the optimizer takes the current trajectory (gray) and identifies the points of maximum obstacle cost $\{t_j\}$(orange points). It then updates the trajectory by a sum of point evaluation functions centered around the set $\{t_j\}$.

Our obstacle term penalizes points in the trajectory close to maximum cost regions in workspace (regions inside/near obstacles). This set of points, $\mathcal{T} = \{t_j\} : j \in [N]$, are local maxima of the workspace cost function c in some neighborhood b_j centered at the points t_j . The size of this neighborhood b_j is arbitrary, and defines the granularity of the search for max points. We can pick many points as local maxima with small neighborhoods vs. a few points over larger segments of the trajectory. The latter will require fewer parameters and will be computationally faster, however it can take more iterations to achieve a fully collision free trajectory. This characteristic is inherent to the choice of cost functional we adopted and does not apply to other costs. In this paper we picked maximum points over a ball of fixed radius, smaller than the size of our obstacles, which approximately results in a point per obstacle:

$$(t_j, u_j) = \arg\max_{t \in b_j} \max_{u \in \mathcal{B}} c(\ x(\xi(t), u)\ ) \tag{5}$$

Therefore, instead of integrating the workspace cost c along the trajectory, the new obstacle functional evaluates c at only those few key time points—the points of maximum obstacle penetration (see Fig.1).

$$\mathcal{U}_{obs}[\xi] = \sum_j c(\ x(\xi(t_j), u_j)\ ), \qquad \nabla_\xi \mathcal{U}_{obs}[\xi] = \sum_j \nabla c(\ x(\xi(t_j), u_j)\ ) J^T k(t_j, \cdot) \tag{6}$$

The workspace function c is differentiable and its gradient can be computed by finite differencing and stored offline for efficiency. The obstacle functional admits a weak derivative in terms of ξ, because in general the workspace cost function is not convex for an arbitrary trajectory ξ. We thus consider the gradient approximation to be the sum of the workspace gradient at the maximum violating points [19]. In (6) J is a line of the full Jacobian matrix corresponding to changes of the respective degree of freedom $J = \frac{\partial q}{\partial x}$ in workspace. The smoothness term $\mathcal{U}_{smooth}[\xi] = \frac{1}{2}||\xi||^2_{\mathcal{H}}$ has a gradient given by $\nabla_\xi \mathcal{U}_{smooth}[\xi] = \xi$.

Optimization. We perform functional gradient descent on the cost functional $\mathcal{U}$. At every iteration, we linearize the functional about the current trajectory ξ^i, using $\mathcal{U}[\xi] \approx \mathcal{U}[\xi^i] + \langle \xi - \xi^i, \nabla_\xi \mathcal{U}[\xi^i] \rangle_{\mathcal{H}}$.

We minimize this functional in a ball around the current trajectory shaped by the norm in the RKHS, or, equivalently, subject to a regularization around the linearized point that constraints the trajectory to be "close" to the previous one (analogous to [2]):

$$\xi^{i+1} = \arg\min_{\xi} \ \langle \xi - \xi^i, \nabla \mathcal{U}[\xi^i] \rangle_{\mathcal{H}} + \lambda \|\xi - \xi^i\|^2_{\mathcal{H}} \tag{7}$$

Considering the end-point constraints, for fixed start and goal configurations we impose: $\xi(0) = \xi^i(0)$, and $\xi(1) = \xi^i(1)$. We solve the constrained optimization problem in closed form by setting the derivative of (7) to 0:

$$\xi^*(\cdot) = \left(1 - \frac{\beta}{\lambda}\right) \xi^i(\cdot) - \frac{1}{\lambda} \sum_{t_j \in \mathcal{T}} \nabla c[\xi^i, t_j] J^T k(t_j, \cdot) \tag{8}$$

Adding the fixed endpoint constraints with the respective Lagrange multipliers, γ_0, γ_1, for the fixed start and end points respectively, yields $\xi(0) = \xi^i(0) \rightarrow \gamma_0 \langle \xi - \xi^i, k(0, \cdot) \rangle = 0$, $\xi(1) = \xi^i(1) \rightarrow \gamma_1 \langle \xi - \xi^i, k(1, \cdot) \rangle = 0$. Applying the KKT conditions defines the Lagrange multipliers $\gamma_0 = \frac{a - \gamma_1 k(1,0)}{k(0,0)}$ and $\gamma_1 = \frac{b\ k(0,0) - a\ k(0,1)}{k(1,1)k(0,0) - k(0,1)k(0,1)}$ for $a = \gamma_0 k(0,0) + \gamma_1 k(1,0)$, and $b = \gamma_0 k(0,1) + \gamma_1 k(1,1)$. Finally the fixed start/end-points solution can be obtained in closed form as well:

$$\xi^{i+1}(\cdot) = \left(1 - \frac{\beta}{\lambda}\right) \xi^i(\cdot) - \frac{1}{\lambda} \left[\sum_{t_j \in \mathcal{T}} \nabla c[\xi^i, t_j] J^T k(t_j, \cdot) - \gamma_0 k(0, \cdot) - \gamma_1 k(1, \cdot) \right] \tag{9}$$

This solution is a generic form of linearized functional gradient optimization in a *directly instantiable* obstacle gradient under the norm in the RKHS. The optimal solution is searched over the full function space of trajectories with desired norm properties, without committing to a discretization, offering a more expressive form of representation for fewer number of parameters. This update rule holds for any arbitrary kernel. In our experimental section, we use an RBF kernel for our main comparison to the waypoint parametrization. We also show experiments with an RKHS over different kernels. Next, we show that the waypoint parametrization is also a special case of the RKHS.

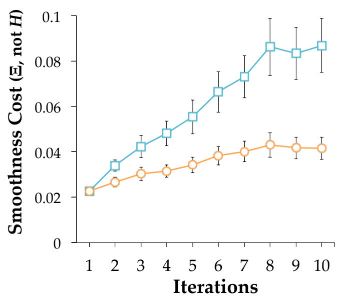
Waypoint Parametrization as an Instance of RKHS. In previous work, the Hilbert space of trajectories, Ξ, is equipped with an analogous inner product $\langle \xi_1, \xi_2 \rangle_\Xi = \langle \xi_1, A\xi_2 \rangle$ [2]. In the waypoint representation, A is typically the Hessian matrix over points in the trajectory, which makes the norm in Ξ penalize unsmooth and inefficient trajectories. The minimization under this norm performs a line search over the negative gradient direction, where A dictates the shape of the manifold over trajectories. This paper generalizes the waypoint parametrization, we can represent waypoints by representing the trajectory in terms of delta Dirac basis functions $\langle \xi, \delta(t, \cdot) \rangle = \xi(t)$ with an additional smoothness metric A. In the limit as $\sigma \to 0$ the RBF representation becomes a waypoint representation, where each individual point is allowed to change without affecting points in the vicinity. With only δ as the kernel, each individual point is allowed to changed without affecting points in its vicinity. CHOMP overcomes this caveat by introducing a new metric that propagates changes of a single waypoint to all the other waypoints, hence kernel evaluation is defined in terms of $k(t_i, \cdot) = A^{-1}\delta(t_i, \cdot)$, where $\xi(t) = \sum_i a_i A^{-1}\delta(t_i, \cdot)$. The inner product of two function in this representation becomes: $\langle \xi_1, \xi_2 \rangle_A = \sum_{i,j} a_i b_j A^{-1}\delta(t_i, t_j)$.

4 Experiments

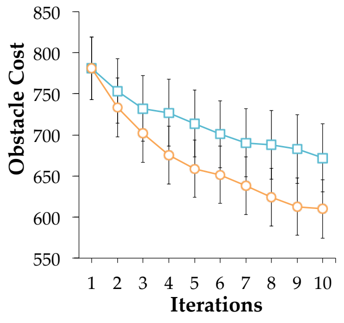
In what follows, we describe our main experiment (Section 4.1), which compares the waypoint and RKHS parametrizations on a set of motion planning problems in a 2D world as in Fig.1. We then introduce a series of smaller experiments that dive deeper into why RKHSs improve optimization (Section 4.2), and how different kernels affect the performance of the algorithm (Section 4.3).

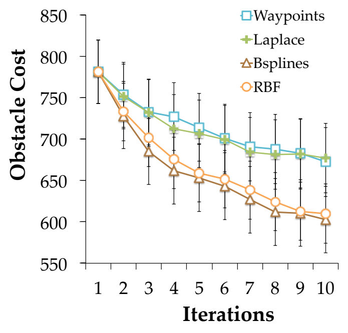
4.1 Main Experiment: RKHS with Radial Basis Functions vs. Waypoints

For our main experiment, we systematically evaluate the two parametrizations across a series of planning problems.
Manipulated Factors: We manipulate the parametrization (waypoints vs Gaussian RBFs) as well as the number of iterations (which we use as a covariate in the analysis). To control for the cost functional as a confound, we use the max formulation for both parametrizations. We use iterations as a factor because they are a natural unit in optimization, and because the amount of time per iteration is similar: the computational bottleneck is computing the maximum penetration points. To control for meta-parameters as a confound, we optimize each optimizer's meta-parameters separately on a training set of start-goal pairs.
Dependent Measures: We measure the obstacle and smoothness cost of the resulting trajectories. For the smoothness cost, we use the norm in the waypoint parametrization as opposed to the norm in the RKHS as the common metric.
Hypothesis: *The RKHS parametrization will result in significantly lower obstacle and smoothness cost for the same number of iterations.* We expect this to be true because with the RBF RKHS parametrization can take larger steps without breaking smoothness. The next section, Section 4.2 explicitly supports this argument.
Analysis: Fig.2 shows the smoothness and obstacle cost respectively. We use 100 different random obstacle placements and keep the start and goal configurations fixed as our experimental setup. The trajectory is represented with 4 maximum violation points over time and robot body points. In the analysis we performed a t-test using the last iteration samples, and showed that the the RBF RKHS representation resulted in significantly lower obstacle cost ($t(99) = -2.63$, $p < .01$) and smoothness cost ($t(99) = -3.53$, $p < .001$), supporting our hypothesis.



(a) Waypoints vs. RBF cost



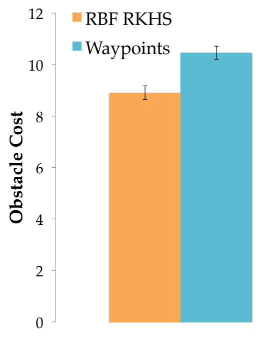


(c) Obstacle cost vs. kernel choice

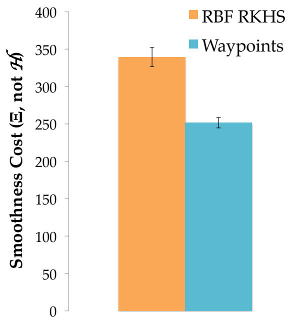
Figure 2: Cost over iterations for a 3DoF robot in 2D. Error bars show the standard error over 100 samples.

4.2 RKHSs Allow Larger Steps than Waypoints

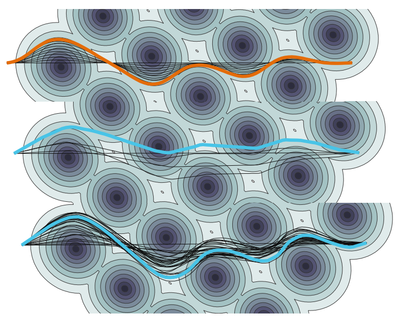
The main practical advantage of using an RBF RKHS instead of the waypoint parametrization is the ability to take large steps during the optimization. Fig.4.2 compares the two while taking large steps: it takes 5 RBF iterations to solve the problem, but would take 28 iterations with smaller steps for the waypoint parametrization – otherwise, large steps cause oscillation and break smoothness. Fig.3 (a and b) show the same comparison, averaged over multiple planning problems. The resulting obstacle cost is always lower with RBFs ($t(99) = 5.32$, $p < .0001$). The smoothness cost is higher ($t(99) = 8.86$, $p =< .0001$), as we saw in the previous experiment as well– qualitatively, however, as Fig.3(c) shows, the RBF trajectories appear smoother as they do not break differential continuity. So far, we used 100 waypoints to represent the trajectory, and only 5 kernel evaluation points for the RKHS. We did also test the waypoint parametrization when the number of waypoints is 5 in order to have an equivalently low dimensional representation, which resulted in much poorer behavior with regards to differential continuity. In order to take gradient steps in the RKHS, we adopt a maximum cost version of the cost functional, (5). We show that our new formulation does not hinder the optimization – that it leads to practically equivalent results as an integral over time and body points [2]. To do so, we manipulate the cost functional formulation, and measure the resulting trajectories' cost in terms of the integral formulation. We observed the integral cost increased by only 5% when optimizing for the max.



(a) $\mathcal{U}_{obs}$, large steps



(b) $||\xi||_{\Xi}$, large steps



(c) top: RBF large steps; middle: waypoints large steps; bottom: waypoints smaller steps

Figure 3: a) b) The costs after 5 large steps (a and b), and the comparison between optimizing using our obstacle cost formulation vs. the integral formulation. c) A comparison between RBFs and Waypoints for 5 large steps (a and b), along with a Waypoint trajectory after over 5 times as many smaller steps.

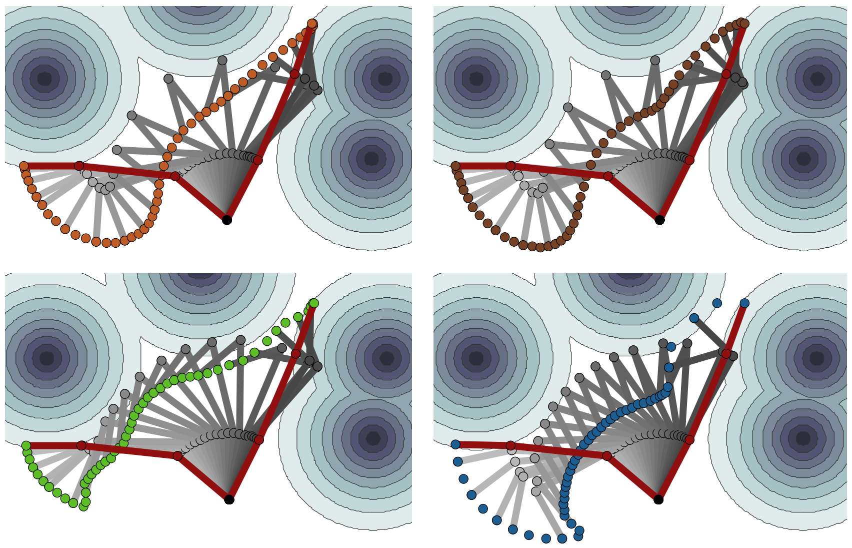


Figure 4: Robot 3DoF in C-space. Trajectory after 10 iterations: top-left: RBF kernel, top-right: Bsplines kernel, bottom-left: Laplace kernel, bottom-right: Waypoints.

4.3 Other Kernel Functions

Although RBFs are a default choice of kernel in many kernel methods, RKHSs can also easily represent other types of kernel functions, *e.g.* B-splines, a popular parametrization of smooth functions [20, 22, 23] used. Although B-splines are finite dimensional kernels they are able to express smooth trajectories while avoiding obstacles. Laplace Kernels yield similar results as the waypoint parametrization since it is less affected by the choice of the width of the kernel. Figure Fig.4 provides a qualitative evaluation of the effect of different kernel choices. We compare the effectiveness of obstacle avoidance over 10 iterations in 100 trials of 12 randomly placed obstacles in a 2D environment. We observe that Waypoints and Laplace kernels with large widths have similar behavior, while RBF and Bsplines kernel provide a naturally smoother parametrization that allows the algorithm to take larger steps at each iteration. These kernels provide the additional benefit of controlling the motion amplitude, being the most suitable in the implementation of an adaptive motion planner. Other kernel functions could be easily considered under the optimization framework we presented in this paper. Their choice should be application driven, in this work we presented a ubiquitous, and expressive kernel approach to trajectory representation.

5 Discussion and Future Work

We introduced a trajectory optimization method in RKHSs, that can represent smooth trajectories with only a few adaptive parameters. Our results suggest that optimization in this spaces can take large steps, leading to a smooth and collision-free trajectory faster. Our work is only the first step in exploring RKHSs for motion planning. Overall, we are excited to contribute the flexibility of RKHSs to trajectory optimization for motion planning, and we look forward to future steps in this direction, including the ability to plan using learned kernels—for example, we could learn a kernel that penalizes unpredictable or non-human-like motion. First, a low-dimensional trajectory parametrization enable us to more easily generate a diverse set of initial trajectories for the optimizer, which aids techniques that learn how to score initial trajectories for a new motion planning problem based on data from old problems [24]. Second, RKHSs enable us to plan using kernels learned from user demonstrations, leading to spaces in which more predictable motions have lower norm, and ultimate fostering better human-robot interaction [25].

References

[1] Khatib O. Quinlan, S. Elastic bands: Connecting path planning and control. In *ICRA*, 1993.

[2] et. al Zucker, M. CHOMP: Covariant hamiltonian optimization for motion planning. *IJRR*, 2013.

[3] et. al Schulman, J. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *RSS*, 2013.

[4] Li W. Todorov, E. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *acc*, 2005.

[5] Abbeel P. Goldberg K. van den Berg, J. Lqg-mp: Optimized path planning. *IJRR*, 2011.

[6] N. Ratliff, M. Zucker, J.A. Bagnell, and S.S. Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *ICRA*, 2009.

[7] B. Scholkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[8] G. S. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. 1971.

[9] N. Aronszajn. Theory of reproducing kernels. In *Transactions of the American Mathematical Society*, 1950.

[10] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 2008.

[11] Cai T. Yuan, M. A reproducing kernel hilbert space approach to functional linear regression. *Annals of Statististics*, 2010.

[12] Schölkopf B. Smola, A.J. and K.R. Müller. The connection between regularization operators and support vector kernels. *Neural Network.*, 1998.

[13] Pan J. Park, C. and D. Manocha. Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In *ICAPS*, 2012.

[14] M. Kalakrishnan and Theodorou E. Pastor P. Schaal S. Chitta, S. STOMP: Stochastic Trajectory Optimization for Motion Planning. In *ICRA*, 2011.

[15] G. Wahba. *Advances in Kernel Methods*. MIT Press, 1999.

[16] N.D. Ratliff and J. A. Bagnell. Kernel conjugate gradient for fast kernel machines. In *IJCAI*, 2007.

[17] Sindhwani V. Minh, H. Q. Vector-valued manifold regularization. In *ICML*, 2011.

[18] L. Murino V. Minh H., Bazzani. A unifying framework for vector-valued manifold regularization and multi-view learning. In *ICML*, 2013.

[19] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[20] A. Blake and M. Isard. *Active Contours*. Springer-Verlag New York, Inc., 1998.

[21] Y.C. Chen. Solving robot trajectory planning problems with uniform cubic b-splines. In *Optimal Control Applications and Methods*, 1991.

[22] J. Zhang and A. Knoll. An enhanced optimization approach for generating smooth robot trajectories in the presence of obstacles. In *ECAC*, 1995.

[23] Zhang L. Manocha D. Pan, J. Collision-free and smooth trajectory computation in cluttered environments. In *IJRR*, 1995.

[24] Liu T. Y. Hebert M. Bagnell J.A. Dey, D. Contextual sequence prediction with application to control library optimization. In *RSS*, 2012.

[25] Srinivasa S. Dragan, A. Familiarization to robot motion. *International Conference on Human-Robot Interaction (HRI)*, 2014.