Fast Stochastic Functional Path Planning in Occupancy Maps

Gilad Francis, Lionel Ott and Fabio Ramos*

***Abstract*— Path planners are generally categorised as either trajectory optimisers or sampling-based planners. The latter is the predominant planning paradigm for occupancy maps. Most trajectory optimisers require a fully defined artificial potential field for planning and cannot plan directly on a partially observed model such as an occupancy map. A stochastic trajectory optimiser capable of planning over occupancy maps was presented in [1]. However, its scalability is limited by the $\mathcal{O}(N^3)$ complexity of the Gaussian process path representation. In this work, we introduce a novel highly expressive path representation based on kernel approximations to perform trajectory optimisation over occupancy maps. This approach reduces computational complexity to $\mathcal{O}(m)$ where $m \ll N$. Moreover, to accelerate convergence we employ an adaptive sampling strategy. We present comparisons to other state-of-the-art planning methods in both simulation and with real occupancy data, which demonstrate the significant reduction in runtime resulting in performance comparable to sampling-based methods.**

I. Introduction

Occupancy maps are a probabilistic representation of the robot's environment as it is perceived from noisy sensor observations [2]. A map is a discriminative model, which returns the probability that an obstacle is present. Planning on occupancy maps is most commonly done by sampling-based planners [3]. While these methods have been proved to be probabilistically complete in finding safe paths, they do not explicitly optimise any objective function such as path length or execution time.

Trajectory optimisers offer a different take on path planning using a variational approach. This enables optimisation of any objective function, such as safety or dynamic constraints, directly in the space of trajectories. Most trajectory optimisers employ signed distance field as an obstacle cost and aside from [1], there are no implementations of trajectory optimisation in occupancy maps. The main impediment for employing a trajectory optimiser using occupancy maps lies in the optimiser's main assumption that it can draw valid samples to update the path anywhere in the map. However, such a general assumption is not applicable in occupancy maps. Built on observations, an occupancy map has unobserved areas. Samples drawn from these areas would result in uninformative path update.

In this paper, we present a novel approach for trajectory optimisation using occupancy maps. We utilise kernel approximation techniques to form an expressive and tractable non-linear path model, which can be considered as a generalisation of the motion planning in *reproducing kernel Hilbert spaces* (RKHSs) paradigm of [4]. Moreover, the approximate kernel representation is naturally updated through random samples, replacing the fixed sampling schedule of [4], [5], thus forming a hybrid method of sampling-based trajectory optimisation. Consequently, optimisation can employ *Stochastic Gradient Descent* (SGD) [6] with its convergence guarantees. Finally, with a finite path representation, updating and querying the path model has a fixed cost, which alleviates the main computational restriction of the non-parametric representation in [1]. The technical contributions of this paper are:

1) An expressive and tractable path model based on kernel matrix approximations, which can be considered a generalisation of other kernel-based trajectory optimisers such as [4], [5].
2) Computationally efficient algorithm, with constant cost per update compared to cubic in the number of samples as in [1]. Path update employs *Stochastic Gradient Descent* (SGD) [6] which ensures efficient convergence to an optimal solution under the guarantees of SGD.
3) An adaptive sampling strategy to accelerate functional update based on the effectiveness of samples. The same sampler also provides a probabilistic indicator of convergence. This sample-based indicator replaces the exhaustive convergence checks done by other FGD planners.

The remainder of this paper is organised as follows: Section II reviews the literature on path planning using occupancy maps. Section III provides background on functional gradient methods and their adaptation for path planning. Section IV provides details on the core elements of the proposed method. The results obtained in various simulation and real data scenarios are shown in Section V. Finally, Section VI draws conclusions about the proposed method.

II. Related Work

Path planning techniques can be categorically grouped into two main branches; sampling-based planning and trajectory optimisation. Sampling-based methods have been predominately used for path planning in occupancy maps with a wide range of successful algorithms such as *Rapidly exploring Random Trees* (RRT) [7], *Probabilistic Road Map* (PRM) [8] and Space skeletonisation [9], [10], [11]. A comparison of the performance of these methods for planning in occupancy maps can be found in [3].

The other approach for path planning, although not commonly used with occupancy maps, is trajectory optimisation. In this planning paradigm the resulting path is a stationary solution of an explicit optimisation problem defined by a

* Gilad Francis, Lionel Ott and Fabio Ramos are with The School of Information Technologies, University of Sydney, Australia gilad.francis@sydney.edu.au

cost function. The cost function provides a measure of path optimality which can arise from a variety of criteria, e.g. distance from obstacles or motion costs. Optimisation can take various forms. Khatib [12] introduced a path planning method based on *artificial potential fields* (APF). *Stochastic Trajectory Optimisation for Motion Planning* (STOMP) [13] performs optimisation by exploring the space of trajectories using noisy perturbations. TrajOpt [14] performs sequential convex optimisation to achieve locally optimal trajectories that comply to a set of motion constraints. Zucker et al. [15], in their work on *Covariant Hamiltonian Optimisation for Motion Planning* (CHOMP), reframed path optimisation as a variational problem directly in the space of trajectories. Similar approach using different path representations was used by [5], [4]. However, as shown in [1], all these methods fail to generate a safe path while planning in occupancy maps.

Trajectory optimisation using occupancy maps were introduced in [1]. Similar to the approach taken here, stochastic gradient samples were used to update the path. However, path variations were represented by a *Gaussian process* (GP) instead of the kernel matrix approximation used in this work. GPs offer a highly flexible path representation, albeit with a significant computational cost. Employing kernel matrix approximation, on the other hand, can ensure that the complexity is independent of the number of samples used, leading to an efficient path optimisation process.

III. Preliminaries

In the following section, we formulate *functional gradient descent* (FGD) as a general optimisation framework for motion planning.

A. FGD for Motion Planning

FGD forms a variational framework for optimisation. In the context of path planning, the optimisation objective is to produce a safe, collision-free path. A secondary objective may incorporate other costs such as smoothness of the trajectory or path length.

We begin by first introducing notation. A path, $\xi : [0,1] \rightarrow \mathcal{C} \in \mathbb{R}^D$ is a function that maps a time-like parameter $t \in [0,1]$ into configuration space $\mathcal{C}$. We define an objective functional $\mathcal{U}(\xi) : \Xi \rightarrow \mathbb{R}$ that returns a real number for each path $\xi \in \Xi$. The objective functional is used in the optimisation process to capture the path optimisation criteria such as smoothness and safety.

Path optimisation is performed by following the functional gradient. The iterative functional gradient update rule is derived from a linear approximation of the cost functional around the current trajectory, ξ_n:

$$\xi_{n+1} = \xi_n - \eta_n A^{-1} \nabla_\xi \mathcal{U}(\xi_n). \tag{1}$$

Here, n indicate iteration number, A a metric tensor and η_n is a user-defined learning rate. The form of the update rule in (1) is general, and thus, invariant to the choice of the objective function $\mathcal{U}(\xi)$ or the solution space representation. The only requirements are that A is invertible and the gradient $\nabla_\xi \mathcal{U}(\xi_n)$ exists.

B. Motion Planning Objective Functionals

The objective functional $\mathcal{U}(\xi)$ in a motion planner paradigm consists typically of a weighted sum of at least two penalties; (i) $\mathcal{U}_{obs}(\xi)$ which encodes a penalty based on proximity to obstacles; (ii) $\mathcal{U}_{dyn}(\xi)$ that regulates and constrains the shape or space-time dynamics of the trajectory. Combining both penalties using a user-defined trade-off coefficient λ we obtain the following objective function:

$$\mathcal{U}(\xi) = \mathcal{U}_{obs}(\xi) + \lambda \mathcal{U}_{dyn}(\xi). \tag{2}$$

In the following sections we define the objective functionals, $\mathcal{U}_{obs}$ and $\mathcal{U}_{dyn}$, and their corresponding functional gradients.

1) Obstacle Functional $\mathcal{U}_{obs}(\xi)$: Obstacles lie in the robot's working space $\mathcal{W} \in \mathbb{R}^3$. However, the path ξ is defined in configuration space $\mathcal{C}$. Hence, estimating the obstacle cost functional requires mapping of $\xi(t)$ from configuration space into workspace using a forward kinematic map x. To account for the size of the robot or any uncertainty in its pose, a set of body points on the robot, $\mathcal{B} \in \mathbb{R}^3$ are defined. $x\left(\xi(t), u\right)$ maps a robot configuration $\xi(t)$ and body point $u \in \mathcal{B}$ to a point in the workspace $x : \mathcal{C} \times \mathcal{B} \rightarrow \mathcal{W}$. As the obstacle functional returns a single value for each $\xi(\cdot)$, the overall cost is calculated by aggregating the workspace cost function, $c : \mathbb{R}^3 \rightarrow \mathbb{R}$, along the trajectory and robot body points using a *reduce* operator. Examples of such operators are the integral or maximum. We require that the *reduce* operator can be approximately represented by a sum over a finite set, $\mathcal{T}(\xi) = \{t, u\}_i$ of time and body points:

$$\mathcal{U}_{obs}(\xi) \approx \sum_{(t,u) \in \mathcal{T}(\xi)} c\left(x\left(\xi(t), u\right)\right). \tag{3}$$

2) Dynamics Functional $\mathcal{U}_{dyn}(\xi)$: $\mathcal{U}_{dyn}(\xi)$ acts as a penalty term based on the kinematic costs associated with ξ. A common approach is to penalise on the trajectory length by optimising the integral over the squared velocity norm [15]:

$$\mathcal{U}_{dyn}(\xi) = \frac{1}{2} \int_0^1 \left|\left| \frac{d}{dt} \xi(t) \right|\right|^2 dt. \tag{4}$$

3) Functional Gradients: Eq. 2 defines the objective functional as a weighted sum of separate penalties. Therefore, the functional gradient can be computed as the sum of the different gradient term, $\nabla_\xi \mathcal{U}_{obs}(\xi(t), u)$ and $\nabla_\xi \mathcal{U}_{dyn}(\xi(t))$.

Functional gradients are derived using the Euler-Lagrange formula [15] which when applied to the obstacle objective $\mathcal{U}_{obs}$ yields;

$$\nabla_\xi \mathcal{U}_{obs}(\xi(t), u) = \frac{\partial}{\partial \xi(t)} x(\xi(t), u) \nabla_{\mathcal{W}} c\left(x\left(\xi(t), u\right)\right), \tag{5}$$

where $\mathbf{J}(t,u) \equiv \frac{\partial}{\partial \xi(t)} x(\xi(t), u)$ is the workspace Jacobian and $\nabla_{\mathcal{W}}$ is the Euclidean gradient of the cost function c. For the planning problems described in this work, $\mathbf{J}$ is the identity matrix. With (4) as choice for the dynamic penalty $\mathcal{U}_{dyn}$, the functional gradient can be easily computed using as $\nabla_\xi \mathcal{U}_{dyn}(\xi(t)) = -\frac{d^2}{dt^2} \xi(t)$.

IV. METHOD

FGD is an efficient method for path optimisation. However, current implementations are not suitable for planning with occupancy maps. In our previous work [1], we identified two main reasons for that. First, the occupancy map's obstacle functional and its spatial gradient have a counter intuitive form, which differ significantly from the well-behaved obstacle functional based on signed distance field used by other planners (e.g. [15], [4], [5]). Second, the path is updated by a deterministic fixed sampling schedule. As this schedule is defined a-priori, samples might be from unobserved areas (invalid) or drawn far from the transition areas around obstacle boundaries (ineffective). Consequently, the solution of such planners does not have any guarantees of converging to a safe path.

The approach taken in this work replaces the spatial obstacle function c with $\mathbb{P}(\mathbf{x})$, which is the probability of occupancy for any query point $\mathbf{x} \in \mathcal{W}$. It uses the occupancy at every body point $\mathbb{P}(x\left(\xi(t), u\right))$ to decide whether to accept or reject a gradient update. Thus, there is no need to precompute any cost or gradient as the spatial gradient $\nabla_{\mathcal{W}}\mathbb{P}\left(x\left(\xi(t), u\right)\right)$ is estimated on-line where it is required.

How to compute the spatial gradient depends on the mapping method used. For occupancy grid map [2], the gradient can be approximated from one of several gradient operators used in computer vision, e.g. Sobel-Feldman operator [16]. For Gaussian Process Occupancy Maps [17], one can compute the spatial gradient in closed-form from the underlying GP. In this work, however, we opted to work with Hilbert maps [18], which provide a fast and continuous occupancy map model. We follow our previous work, presented in [1], to compute spatial occupancy gradient directly from the map model in closed form.

A. Stochastic Functional Regression

Any FGD planner optimises an objective function, such as (2). As the objective is uncountable, it is estimated via samples. Therefore, the choice of sampling schedule is paramount for a successful and efficient planner. The importance of the sampling schedule is exacerbated in occupancy maps, where not every sample can generate an informative gradient. Consequently, sampling everywhere along the curve is most desired, as this increases the chance of identifying transition areas in the map and the path expressivity. Yet, with a fixed resolution sampling scheme, defining a sufficient resolution a-priori is difficult. Hence most methods, even planners with smooth path representation [5], [4], limit the sampling resolution according to their computational resources.

Similar to the approach in [1], we sample randomly. While [1] use a costly non-parametric GP path representation, we employ a parametric path representation based on kernel approximation. This allows to update the path model using samples drawn randomly anywhere along the path while keeping a fixed computational cost.

In kernel machines, $\Upsilon(t)$ defines a mapping from $t \in [0, 1]$ into a potentially infinite-dimensional vector-valued *Reproducing Kernel Hilbert Space* (RKHS) [1] $\mathcal{H}$ [19]. The kernel function $k(t, t')$ defines the inner product $\left\langle \cdot, \cdot \right\rangle_{\mathcal{H}}$ between two points in that space. In the approximate kernel approach we denote $\hat{\Upsilon}$ as a finite set of features that approximate the RKHS inner product by a dot product;

$$k(t, t') = \left\langle \Upsilon(t), \Upsilon(t') \right\rangle_{\mathcal{H}} \approx \hat{\Upsilon}(t)^T \cdot \hat{\Upsilon}(t'). \tag{6}$$

We note that the set of features only approximates the selected kernel in expectation, hence the $\hat{}$ notation. There are several methods to generate features to approximate a kernel. For the *radial basis function* (RBF) kernel defined by $k(t, t') = \exp(-\gamma \parallel t - t' \parallel^2)$, with γ a free parameter and $\parallel \cdot \parallel$ is the Euclidean norm, we employed two different approximations:

I. *Random Fourier features* (RFF) [20]
This approximation requires m random samples of two variables; $s_i \sim \mathcal{N}(0, 2\gamma I)$ and $b_i \sim$ uniform$[-\pi, \pi]$ where $i = 1...m$.
The features vector is then given by

$$\hat{\Upsilon}^{RFF}(t) = \frac{1}{\sqrt{m}}[\cos(s_1 t + b_1), ..., \cos(s_m t + b_m)] \tag{7}$$

II. RBF features
A kernel matrix K with rank n can be approximated by projecting it into a lower rank matrix using a set of m inducing points denoted by $\hat{t}_1, ..., \hat{t}_m$ [21]. Then, $K \approx K_{n,m}\hat{K}_m^{\dagger}K_{m,n}$. $\hat{K}_m^{\dagger}$ is the pseudo inverse of $\hat{K}_m$. Using these m inducing points, the approximation features vector is given by [21]:

$$\hat{\Upsilon}^{RBF}(t) = \hat{D}^{1/2}\hat{V}^T[k(t, \hat{t}_1), ..., k(t, \hat{t}_m))]^T \tag{8}$$

Here, $\hat{D}$ is the diagonal matrix of eigenvalues of $\hat{K}_m$ and $\hat{V}$ are the corresponding eigenvectors.

We note that since the planner in [4] also uses a fixed number of support points, it implicitly employs the RBF kernel approximation. The proposed method generalises this to other kernel approximation techniques such as RFF. Moreover, by using a fixed predetermined sampling schedule, [4] restricts the expressivity of the path representation. Using random samples, our method performs path updates everywhere along the path, incorporating only the valid samples and rejecting the invalid ones.

Using a weight vector $\mathbf{w} \in \mathbb{R}^D$ we can now express the robot configuration $\xi(t)$ at t, as a function of the finite set of approximating features, $\hat{\Upsilon}(t)$:

$$\xi(t; \mathbf{w}) = \xi_o(t) + \xi_b(t) + \mathbf{w}^T\hat{\Upsilon}(t). \tag{9}$$

Here, ξ_o is an offset path, which is used to bias the solution. This permits using prior information or a rough path as the optimisation's starting point, for example, a waypoint path generated by an RRT or the current path if re-planning is needed. ξ_b is a term used to enforce boundary conditions. Both ξ_o and ξ_b are represented by an approximated kernel

[1]The path RKHS is different to the one used by the Hilbert maps.

representation with the same curve properties as ξ (continuity, differentiability, etc.), although the feature set can be different.

Once the path representation has been defined, we can treat path planning as a regression problem, i.e., optimising the weight vector $\mathbf{w}$ given a set of observations $\mathcal{T}$;

$$\mathbf{w}_{optimal} = \arg\min_{\mathbf{w}} \quad \sum_{t \in \mathcal{T}} \mathcal{U}\Big(\xi(t, \mathbf{w})\Big). \tag{10}$$

The advantage of using this approach is that the model can be learned from points sampled randomly anywhere along the path.

Eq. (9) expresses the path as a weighted sum of features, therefore the iterative update rule of (1) must be performed with respect to the weight vector $\mathbf{w}$. Following (1), we sample the functional gradient of the objective function at time t_i. We refer to these samples as stochastic, since t_i can be drawn at random from anywhere along the curve domain, $t_i \in [0, 1]$, and is not limited by a predefined sampling resolution. The sampled gradient $\nabla_\xi \mathcal{U}\big(\xi_n(t_i, \mathbf{w}_n)\big) \in \mathcal{H}$ can be viewed as a path perturbation, which is defined in $\mathcal{H}$ and thus must be projected onto the finite representation spanned by $\mathbf{w}$ using the appropriate inner product, which is approximated using (6). This approximation leads to the following iterative update rule:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta_n A^{-1} \hat{\Upsilon}(t_i)^T \hat{\Upsilon}(t_i) \nabla_\xi \mathcal{U}\big(\xi_n(t_i, \mathbf{w}_n)\big). \tag{11}$$

ξ_o and ξ_b are removed from (11) for brevity. Note that to guarantee convergence of SGD, the learning rate η_n must satisfy the Robbins-Monro conditions [22]; $\sum_{n=1} \eta_n^2 < \infty$ and $\sum_{n=1} \eta_n = \infty$.

Boundary conditions are handled in a similar fashion. We employ an additional path $\xi_b = \mathbf{w}_b^T \hat{\Upsilon}_b$ to compensate for the boundary conditions. The boundary features $\hat{\Upsilon}_b$ are not necessarily identical to $\hat{\Upsilon}$. The update rule for $\mathbf{w}_b$ is:

$$\mathbf{w}_{bn+1} = \mathbf{w}_{bn} - A^{-1} \hat{\Upsilon}_b(t_b)^T \hat{\Upsilon}_b(t_b) \Delta x_b(t_b). \tag{12}$$

Here t_b are time points where boundary conditions are defined and $\Delta x_b(t_b)$ is the corresponding difference between the current value of ξ at t_b and the desired value at the boundary. Note that this similar to (11), except η_n was omitted and the gradient was replaced by the difference to the desired boundary value.

B. Targeted Sampling

In order to speed-up convergence, we introduce a dynamic proposal distribution $\mathcal{Q}$, that draws samples based on their expected effectiveness. $\mathcal{Q}$ replaces the uniform distribution used in [1]. Such a sampling schedule increases the chance of sampling around areas where previous updates were effective while still drawing samples over the entire path domain $t \in [0, 1]$. We note that $\mathcal{Q}$ serves a similar purpose to *importance sampling* [23], however in the functional optimisation framework, $\mathcal{Q}$ must dynamically adapt to changes in the functional.

An effective $\mathcal{Q}$ needs to be proportional to the amplitude of the objective functional gradient. In addition, sampling and updating $\mathcal{Q}$ should not incur high computational costs. These two properties are achieved by representing $\mathcal{Q}$ as a mixture model:

$$\mathcal{Q}(t) = \sum_{l \in L} p(l) p(t|l), \tag{13}$$

where L is a set of intervals $l \in L$ such that $\cup_{l \in L} l = [0, 1]$. $p(t|l)$ is a uniform distribution defined over the interval l, i.e., $p(t|l) = U[l]; \forall l \in L \ . \ p(l)$ is the probability of selecting interval l, and we require $\sum_{l \in L} p(l) = 1$. The mixing probability $p(l)$ defines the importance of each interval. More importantly, $p(l)$ can be adapted during optimisation to reflect the effectiveness of samples drawn from that interval.

$\mathcal{Q}$ can also be used as an indicator of convergence. $\mathcal{Q}$ biases sampling according to the impact a sample had on the path update. Meaning, that around $\xi_{optimal}$, new samples should have little effect on the path. As a result, $\mathcal{Q}$ should return to maximum entropy sampling, i.e. $U[0, 1]$. Hence, by monitoring the entropy of $\mathcal{Q}$, one can identify whether sampling follows a uniform distribution which indicates a solution has been found.

C. Path Planning Algorithm

The pseudo-code of the stochastic approximate kernel path planner is shown in Algorithm 1. The output of this algorithm is an optimised path $\xi_{optimal}(\cdot)$, parametrised by the weight vector $\mathbf{w}_{optimal}$.

At each iteration, a mini-batch (t_s, u_s) is drawn uniformly. The occupancy of each sample $t^* \in t_s$ and $u^* \in u_s$ is assessed by querying the map model at state $\xi_n(t^*, u^*)$. If the probability of occupancy at $\xi_n(t^*)$, P_{occ}, is within the safety limits, i.e. clear of obstacles, a functional gradient update is performed. Following (11), the weight vector $\mathbf{w}$ is updated with the stochastically sampled gradient observations, leading to a new path representation $\xi_{n+1}(\cdot)$. Finally, the boundary conditions are enforced using (12).

This leads to an algorithm with low computational complexity which stems from the concise path representation and update rule. Using m approximated features, the computational cost of updating and querying the path model is $\mathcal{O}(m)$ which is constant regardless of the number of samples drawn. This is in contrast to the computational cost of the stochastic GP path planner [1], which is cubic in the number of samples, i.e. $\mathcal{O}(N^3)$. Meaning that the time per iteration increases as more samples are collected.

V. Results

In this section, we evaluate the performance of our method and compare it with other related path planning techniques on real laser data. We show that stochastic sampling is a critical aspect of path planning in occupancy maps, which is complimented by the scalable model of the approximate kernel path representation. We compare our proposed method with two other planning methods for occupancy maps; RRT* [24] and the stochastic GP path planner [1]. As both our method and RRT* optimise on path length, we used RRT* as a representative of the sampling based methods. To

Algorithm 1: Stochastic approximate kernel path planner

Input: $\mathbb{P}$: Occupancy Map.
$\xi(0), \xi(1)$: Start and Goal states.
P_{safe}: No obstacle threshold.
$\hat{\Upsilon}$: Approximated feature vector.
$\mathcal{Q}$: Proposal distribution (default: uniform).
Output: $\mathbf{w}_{optimal}, \xi_{optimal}(\cdot)$
$n = 0$
while *solution not converged* **do**
  $(t_s, u_s) \sim \mathcal{Q} \leftarrow$ Draw mini-batch from $\mathcal{Q}$
  foreach $(t^*, u^*) \in (t_s, u_s)$ **do**
    $P_{occ} \leftarrow \mathbb{P}(x(\xi_n(t^*), u^*))$ Query map
    if $P_{occ} \leq P_{Safe}$ **then**
      $\mathbf{w}_{n+1} \leftarrow$ update rule Eq. 11
    end
    Fix boundary conditions, Eq. (12)
  end
  Update $\mathcal{Q}$ using samples $n = n + 1$
end

ensure consistent runtime comparison, all methods, including Hilbert maps, are implemented in Python and tested on an Intel Core i5-4570 with 8GB RAM.

A. Real data

The map for this experiment was fitted according to laser observations from the Intel-Lab dataset (available at http://radish.sourceforge.net/). Both RFF and RBF planners used $m = 50$ features and $\gamma = 4$. The trade-off parameter $\lambda = 0.0075$ and the per iteration learning rate $\eta_n = \eta_0(n + n_0)^{-1}$, where $\eta_0 = 50$ and $n_0 = 100$. A demonstration of planning in this scenario is also available in `https://youtu.be/wKQIzCRlJXc`.

Fig. 1 and Table I show a comparison between the different methods. RRT* forms a path based on several waypoints (states) the robot should pass from start to goal. As a result, the path typically is jagged, with short jerks. In contrast, the paths generated by our method are continuous and smooth. In addition, unless using inflated obstacles, RRT* paths tend to move close to the walls or cutting corners, as indicated by the relative high, and unsafe, occupancy of RRT* in Table I. The stochastic planner follows the mid line between obstacles and performs smooth turns resulting in shorter and safer trajectories. Table I provides quantitative comparisons between these planners. All planners obtain similar path length, which corresponds to the length of the corridor. Runtime results reveal the main advantage of the approximate kernel planner. While the stochastic non-parametric GP planner of [1] requires less samples to converge, its actual runtime is almost 25 times slower. Meaning that while the non-parametric GP path representation is highly expressive, it is not scalable due the cubic computational complexity. The RRT* runtime performance is quite poor too. While RRT* have similar per iteration cost as the approximate kernel approach, its sampling efficiency in this scenario is low since most of the map is either occupied or unknown. In contrast to both methods, our method uses an approximate kernel path representation, which has a fixed linear complexity. Consequently, the properties of the path are similar to that of the non-parametric path, however adding more observations does not change the computational performance of the model. As a result, the runtime of the proposed method is much shorter compared with the GP planner and RRT*.

TABLE I: Intel dataset comparison

	Max. occupancy	Path length [m]	Samples	Runtime [s]
RBF	0.34±0.05	20.3 ± 0.2	1629 ± 287	3.2 ± 0.6
RFF	0.34±0.03	20.3 ± 0.1	1861 ± 60	5.6 ± 1.8
Gp Paths [1]	0.38±0.08	20.4 ± 0.2	398 ± 132	78.3±47.5
RRT* [24]	0.49±0.05	20.3 ± 0.4	10788 ± 1462	34.5±10.0

B. Targeted Sampling

A performance comparison over 100 simulations of planning using an adaptive distribution as opposed to a uniform scheme is shown in Table II. The adaptive sampling presents faster optimisation with smaller variance and with no degradation in performance, as length is almost unaffected by the change in sampling procedures.

$\mathcal{Q}$ is also used as a convergence indicator. Path planning with uniform sampling converges when all samples are valid, i.e. safe. An adaptive sampling method, on the other hand, can reason on other convergence conditions, such as the change in the objective functional. Fig. 2 depicts path safety and $\mathcal{Q}$'s entropy during optimisation. To emphasise this process, $\mathcal{Q}$ was set with 50 intervals. Initially, the maximum occupancy along the path, marked in red, is 1, a certain collision. As the optimisation progresses, the path clears from obstacles, indicated by a maximum occupancy below 0.5. At that point, a uniform sampling scheme would indicate converges. Although safe, this solution is not necessarily optimal with regards to the overall objective. The entropy of $\mathcal{Q}$, in blue, provides the complimentary condition for convergence.

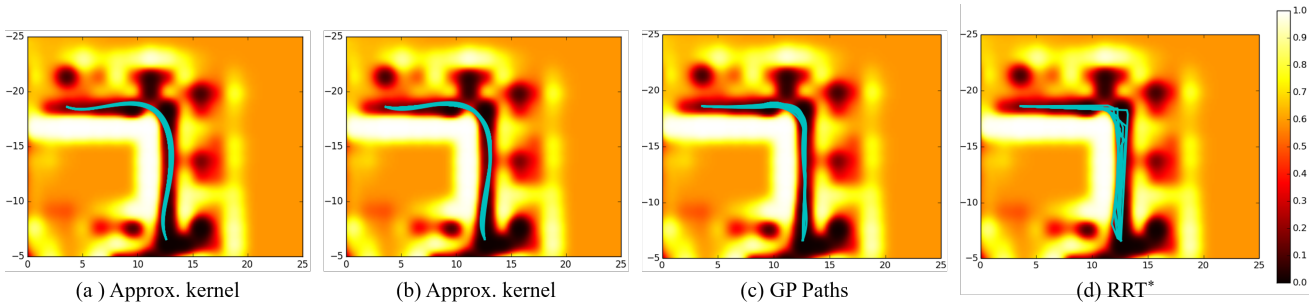
VI. Conclusions

The planning method proposed in this work employs SGD to optimise a path represented by an approximate kernel feature set. This model provides a highly expressive path with a cost effective representation. SGD combines the approximate kernel path model with a stochastic sampling schedule to form a computationally efficient optimisation process with convergence guarantees.

Using random samples across the entire path domain avoids the need to commit to an a-priori sampling resolution of the objective function. Consequently, the optimiser identifies transition areas around the borders of obstacles,

TABLE II: Adaptive sampling - Performance comparison

	Adaptive Sampling	Uniform Fixed Sampling
Iterations to converge	132 ± 43	202 ± 98
Path length [m]	21.07 ± 0.36	21.02 ± 0.35
Converged solution [%]	85	65



(a) Approx. kernel RBF (b) Approx. kernel RFF (c) GP Paths (d) RRT*

Fig. 1: Comparison of path planning methods on a continuous occupancy map of the Intel-Lab (partially shown); (a) RBF approximation, (b) RFF approximation, (c) stochastic non-parametric GP paths [1] and (d) RRT* [24]. Each image shows ten paths generated by the planning algorithm, to indicate repeated performance. The planning methods in (a), (b) and (c) produce smooth paths which follow the mid lines between walls. RRT* paths are typically not smooth, and some have small jerks. In addition, RRT* paths pass dangerously close to walls.

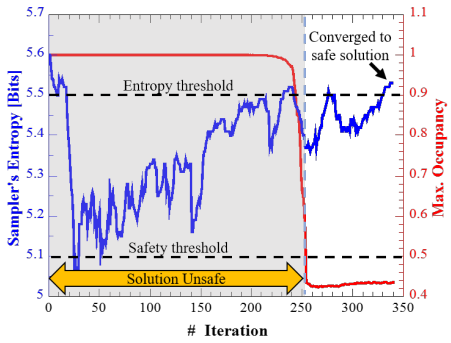


Fig. 2: Indicting convergence using the entropy of $\mathcal{Q}$. The two conditions for convergence are safety and maximum entropy. A path is considered safe when the maximum occupancy along the path is below a safety threshold. The shaded area marks iterations during the optimisation where the path was not safe. The second condition for convergence requires that the sampler's entropy would be approximately the maximum attainable entropy.

which enables the optimiser to overcome the uninformed areas formed by the obstacles.

Experimental result demonstrates the importance of random sampling for planning in occupancy maps. Combined with an approximate kernel path representation, our method offers a scalable and fast method for trajectory optimisation in occupancy maps.

References

[1] G. Francis, L. Ott, and F. Ramos, "Stochastic Functional Gradient for Motion Planning in Continuous Occupancy Maps," in *IEEE Int. Conf. on Robotics and Automation*, 2017.

[2] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, 1989.

[3] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A Review of Global Path Planning Methods for Occupancy Grid Maps Regardless of Obstacle Density," *Journal of Intelligent & Robotic Systems*, 2016.

[4] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces," in *Robotics: Science and Systems*, 2016.

[5] M. Mukadam, X. Yan, and B. Boots, "Gaussian Process Motion Planning," in *IEEE Int. Conf. on Robotics and Automation*, 2016.

[6] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Int. Conf. on Computational Statistics*, 2010.

[7] S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," tech. rep., Iowa State University., 1998.

[8] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, 1996.

[9] T. Lozano-Pérez and M. A. Wesley, "An Algorithm for Planning Collision-free Paths among Polyhedral Obstacles," *Communications of the ACM*, 1979.

[10] P. Bhattacharya and M. L. Gavrilova, "Voronoi Diagram in Optimal Path Planning," in *Int. Symp. on Voronoi Diagrams in Science and Engineering*, 2007.

[11] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching," in *IEEE/RSJ Conf. on Intelligent Robots and Systems*, 2006.

[12] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, 1986.

[13] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[14] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion Planning with Sequential Convex Optimization and Convex Collision Checking," *The International Journal of Robotics Research*, 2014.

[15] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian Optimization for Motion Planning," *The International Journal of Robotics Research*, 2013.

[16] G. Kaehler and A. Bradsk, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media Inc., 2008.

[17] S. T. O'Callaghan and F. T. Ramos, "Gaussian Process Occupancy Maps," *The International Journal of Robotics Research*, 2012.

[18] F. Ramos and L. Ott, "Hilbert Maps: Scalable Continuous Occupancy Mapping with Stochastic Gradient Descent," in *Robotics: Science and Systems*, 2015.

[19] B. Schölkopf and A. J. Smola, *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT Press, 2001.

[20] A. Rahimi and B. Recht, "Weighted Sums of Random Kitchen Sinks: Replacing Minimization with Randomization in Learning," in *Neural Information Processing Systems*, 2009.

[21] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, 1998.

[22] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, 1951.

[23] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[24] S. Karaman, "Incremental Sampling-based Algorithms for Optimal Motion Planning," *Proc. Robotics Science and Systems*, 2010.