

Stochastic Functional Gradient for Motion Planning in Continuous Occupancy Maps

Gilad Francis, Lionel Ott and Fabio Ramos*

Abstract—Safe path planning is a crucial component in autonomous robotics. The many approaches to find a collision free path can be categorically divided into trajectory optimisers and sampling-based methods. When planning using occupancy maps, the sampling-based approach is the prevalent method. The main drawback of such techniques is that the reasoning about the expected cost of a plan is limited to the search heuristic used by each method. We introduce a novel planning method based on trajectory optimisation to plan safe and efficient paths in continuous occupancy maps. We extend the expressiveness of the state-of-the-art functional gradient optimisation methods by devising a stochastic gradient update rule to optimise a path represented as a Gaussian process. This approach avoids the need to commit to a specific resolution of the path representation, whether spatial or parametric. We utilise a continuous occupancy map representation in order to define our optimisation objective, which enables fast computation of occupancy gradients. We show that this approach is essential in order to ensure convergence to the optimal path, and present results and comparisons to other planning methods in both simulation and with real laser data. The experiments demonstrate the benefits of using this technique when planning for safe and efficient paths in continuous occupancy maps.

I. INTRODUCTION

Motion planning is a basic building block in autonomous robotics. Essentially, it is a decision making process that ensures safe travel from the robot’s current configuration to its goal. As safety is the primary objective, the planned trajectory must avoid collision with obstacles. It is a prolific branch of robotics that has been studied for decades, producing a wide range of planning methods which can be categorically grouped into two main branches; sampling-based planning and trajectory optimisation.

Planning a safe path using a *Occupancy Grid Map* (OGM) is typically done by sampling-based planners [1]. Most planners break the planning process into two phases. First, the planner finds a feasible, collision-free, crude path. Then, the following step improves the resulting path by applying certain heuristics.

Trajectory optimisers optimise an objective function such as control cost or execution. However, there are no trajectory optimiser implementations for path planning using occupancy maps. The main challenge lies in the optimiser’s need for contextual information anywhere along the path. Gaps or non-informative gradients will cause the optimiser to converge into a non-optimal and unsafe solution.

In this paper, we present a new planning paradigm using occupancy maps. We utilise the recently introduced Hilbert

maps [2] instead of OGMs. Hilbert maps provide a fast and continuous linear discriminative model for occupancy mapping. We take advantage of the fact that spatial gradients of the occupancy can be calculated in closed form, and use them in the functional gradient motion planner update cycle. We present a novel path planner based on a *Gaussian Process* (GP) path representation. Unlike other functional gradient path planning techniques (e.g. [3],[4]), the proposed planner does not commit to a predetermined resolution, whether spatial or parametric. It replaces the regularisation of the step size used in the functional gradient method with a stochastic gradient approach. This is a key element in the planner’s optimisation strategy as it allows a resolution-free gradient update, which is required to ensure convergence.

The technical contributions of this paper are:

- 1) A novel path optimisation approach for continuous occupancy maps. Effectively, this method extends previous work done on discrete cost maps to a continuous environment representation.
- 2) A stochastic functional gradient motion planner based on GP path representation. The stochastic samples allow flexible support for the path, instead of an a-priori fixed set used in previous work.

The remainder of this paper is organised as follows. Section II surveys the work related to path planning using occupancy maps. Section III reasons on the need for a specific planner for occupancy maps and provides details on the proposed method. Experimental results and analysis for various simulation and real data scenarios are shown in IV. Finally, Section V draws conclusions on the proposed method.

II. RELATED WORK

Path planning using occupancy maps is commonly approached by sampling-based methods with several very successful algorithmic families such as: *Rapidly exploring Random Trees* (RRT), *Probabilistic RoadMap* (PRM), *Visibility Graphs* (VG) and *Space Skeletonisation* [1]. Sampling-based methods typically work by first building a graph representation of the configuration space, where edges represent valid connections. After the graph is built a valid path is obtained using a search algorithm on the graph structure. The visibility graphs method builds a graph where the nodes are the vertexes of the obstacles [5]. Space skeletonisation uses *Generalised Voronoi Diagram* (GVD) to compute safe paths [6], [7]. PRM randomly samples the configuration space for free space configuration and then uses a local planner to find edges to connect these configurations to existing nodes

* Gilad Francis, Lionel Ott and Fabio Ramos are with The School of Information Technologies, University of Sydney, Australia gfra8070@uni.sydney.edu.au

[8]. Next a tree search method is used to determine the path. Another successful and prolific method is RRT, which randomly grows a tree rooted at the start configuration [9]. The main drawback of sampling based methods is that while they are very successful in finding safe paths, there is no explicit optimisation of an objective function, such as length or smoothness.

Optimisation is a widely used approach for finding feasible paths, where the planned path is the local extrema of a pre-defined arbitrary cost function. Loosely speaking, the cost function captures the costs and penalties associated with a configuration-space state, e.g. distance from obstacles. Khatib pioneered the use of artificial potential field for collision avoidance [10]. *Covariant Hamiltonian Optimisation for Motion Planning* (CHOMP) utilises covariate gradients from a precomputed obstacle cost to minimise the trajectory's obstacle and smoothness functionals [4]. The *Stochastic Trajectory Optimisation for Motion Planning* (STOMP) planner uses noisy perturbations to perform optimisation under constraints where the cost functional is non-differentiable [11]. Both CHOMP and STOMP commit to a waypoint representation which require to trade-off expressiveness with computational costs. Mukadam et al. proposed the Gaussian process motion planner which uses a Gaussian process generated by linear time varying stochastic differential equations for path representation [12]. Marinho et al. perform trajectory optimisation in a *Reproducing Kernel Hilbert Space* (RKHS) [3]. However, all these methods fall short when planning using occupancy maps as discussed in section III-C.

Hilbert maps are a scalable, fast approach for continuous occupancy mapping that was recently presented in [2]. The model works by a non-linear mapping of observations into a high-dimensional feature space represented by a RKHS. To ensure a linear incremental update the parameters of the model are trained by optimising a convex objective function, which ensures convergence to the global optimum.

Traditional occupancy grid maps discretise the map into a fixed grid in order to estimate the occupancy posterior [13]. In order to make computations tractable each cell is considered as an independent random variable. The computational gains are substantial since the posterior calculation can be done separately for each cell. The drawback is the loss of spatial relationship between neighbouring cells. To alleviate this problem, a non-parametric approach based on Gaussian Processes (GPs) was proposed in [14]. The *Gaussian processes occupancy map* (GPOM) produces probabilistic occupancy posteriors based on sensor observations. Using a parameterised covariance function, GPOM captures spatial relationships, which enables continuous inference. The computational complexity is its main limitation, as it scales cubically with the number of observations.

Hilbert maps take the advantages of both OGM and GPOM. It maps observations to a high-dimensional feature vector, whose dot product approximate the *radial basis function* (RBF) kernel. Similar to GPOM, the use of kernels keeps spatial relationship which enables continuous inference. The computational complexity, on the other hand, de-

pends linearly on the number of features. The only downside, compared with GPOM, is that posterior is a point value and not probabilistic.

III. PATH PLANNING USING HILBERT MAPS

In this section we describe the proposed method, functional gradient motion planning using Hilbert maps. First, we describe the building blocks of our planner, Hilbert maps and functional gradient motion planning. Then, we present our algorithm which combines these components in a single planner.

A. Hilbert Maps

In this section we briefly review the basics of Hilbert maps, which we use as our continuous occupancy representation. Building an occupancy map requires sensor (e.g. laser range finder or sonar) inputs. The dataset, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_1^N$, contains N observations, captured by the robot while moving through the environment, where $\mathbf{x}_i \in \mathbb{R}^D$ is a $2D$ or $3D$ position and $y_i \in \{-1, +1\}$ represents observed occupancy at \mathbf{x}_i .

The discriminative model that predicts the occupancy at a new query point, \mathbf{x}^* , is based on the *logistic regression classifier* (LR) model. Given a vector of parameters \mathbf{w} the probability of LR occupancy is given by:

$$p(y^* = +1 | \mathbf{x}^*, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}^*)}. \quad (1)$$

As occupancy is a binary random variable, the probability of non-occupancy is given by $p(y^* = -1 | \mathbf{x}^*, \mathbf{w}) = 1 - p(y^* = +1 | \mathbf{x}^*, \mathbf{w})$. We note that (1) is the *logit* sigmoid function, $\sigma(\mathbf{z}_L)$ applied on the linear projection $\mathbf{z}_L = \mathbf{w}^T \mathbf{x}$.

The linear projection of the basic LR classifier cannot capture the complexity of a real environment. To support a richer family of functions, Hilbert maps employ nonlinear classification using approximate kernels. The kernel, $k(\cdot, \cdot)$, defines a nonlinear, and potentially infinite dimensional, mapping, $\Phi(\mathbf{x}, \cdot)$ that projects the input into a high dimensional RKHS. The inner product between two features is then $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}, \cdot), \Phi(\mathbf{x}', \cdot) \rangle$. To reduce model training time, Hilbert maps replace the kernel with an approximation, $\hat{\Phi}(\cdot)$ [15]. $\hat{\Phi}(\cdot)$ defines a finite feature vector where the dot product of these features can, in expectation, approximate the selected kernel; $k(\mathbf{x}, \mathbf{x}') \approx \hat{\Phi}(\mathbf{x})^T \hat{\Phi}(\mathbf{x}')$. Under these assumptions, the predictive occupancy posterior becomes:

$$p(y^* = +1 | \mathbf{x}^*, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \hat{\Phi}(\mathbf{x}^*))} = \sigma(\mathbf{z}_{NL}), \quad (2)$$

where \mathbf{z}_{NL} denotes the nonlinear mapping achieved by the feature vector.

There are several methods to generate features to approximate a kernel [2]. For the RBF kernel defined by;

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2) \quad (3)$$

there are three different approximations; Random Fourier features [16], the Nyström method [17] and the sparse random features [18].

After the selection of the approximation method, parameters w need to be calculated. The objective function used to estimate w is a regularised *negative log-likelihood* (NLL) [2].

B. Functional Gradient Motion Planning

In this section, we describe functional gradient based optimisation methods used in path planning. We first introduce the notation. A path, $\xi : [0, 1] \rightarrow \mathcal{C} \in \mathbb{R}^D$, is a function that maps time, $t \in [0, 1]$, into configuration space \mathcal{C} . We define an objective functional, $\mathcal{U}(\xi) : \Xi \rightarrow \mathbb{R}$, that return a real number for each path $\xi \in \Xi$. The objective functional captures the path optimisation criteria, such as the path safety and kinematic costs.

The objective functional, $\mathcal{U}(\xi)$, varies between the different planning methods. However, it is typically a weighted sum of two penalties; (i) $\mathcal{U}_{obs}(\xi)$ which penalises proximity to obstacles; (ii) $\mathcal{U}_{dyn}(\xi)$ that regulates either the curve shape or motion dynamics:

$$\mathcal{U}(\xi) = \mathcal{U}_{obs}(\xi) + \lambda \mathcal{U}_{dyn}(\xi). \quad (4)$$

As obstacles are defined in the robot's working space $\mathcal{W} \in \mathbb{R}^3$, estimating the obstacle cost functional is done by mapping a path from configuration space into workspace using a forward kinematic map, x . Given $\mathcal{B} \in \mathbb{R}^3$, a set of points on the robot, $x(\xi(t), u)$ maps a robot configuration, $\xi(t)$ and body point $u \in \mathcal{B}$ to a point in the workspace $x : \mathcal{C} \times \mathcal{B} \rightarrow \mathcal{W}$. Then, the obstacle cost functional is estimated by aggregating the workspace cost function, $c : \mathbb{R}^3 \rightarrow \mathbb{R}$, along the trajectory and robot body points using a *reduce* operator such as an integral or a maximum. The only requirement is that the *reduce* operator can be approximately represented by a sum over a finite set, $\mathcal{T}(\xi) = \{t, u\}_i$ of time t_i and body points u_i :

$$\mathcal{U}_{obs}(\xi) \approx \sum_{(t, u) \in \mathcal{T}(\xi)} c(x(\xi(t), u)). \quad (5)$$

$\mathcal{U}_{dyn}(\xi)$ is a secondary objective functional, which typically penalises based on kinematic costs or curve properties. The exact choice depends on the implementation and path representation used. In most cases the penalty deals with the derivatives of the path, for example in [4] the squared velocity norm was used as the dynamic penalty:

$$\mathcal{U}_{dyn}(\xi) = \frac{1}{2} \int_0^1 \left\| \frac{d}{dt} \xi(t) \right\|^2 dt. \quad (6)$$

In [3], the optimisation regulariser was the L_2 norm of ξ which implicitly assumes the zero-line, connecting starting point to the goal point, as the preferable solution. We will show in section IV that such a regulariser is not suitable when planning using occupancy maps.

Given the cost functional in (4), optimisation of ξ can be performed by an iterative approach following the functional gradient. The functional gradient update at each iteration is derived from a linear approximation of the cost functional around the current trajectory, ξ_n :

$$\mathcal{U}(\xi) \approx \mathcal{U}(\xi_n) + \nabla_{\xi} \mathcal{U}(\xi_n)(\xi - \xi_n) + \mathcal{O}(\|\xi - \xi_n\|^2). \quad (7)$$

Following [4], the optimisation update rule becomes

$$\xi_{n+1} = \arg \min_{\xi} \mathcal{U}(\xi_n) + (\xi - \xi_n)^T \nabla_{\xi} \mathcal{U}(\xi_n) + \frac{\beta}{2} \|\xi - \xi_n\|_M^2. \quad (8)$$

where the term $\|\xi - \xi_n\|_M^2 = (\xi - \xi_n)^T M (\xi - \xi_n)$ is the squared norm with respect to a metric tensor M and β is a regularisation factor. A closed form solution of (8) is obtained by differentiating the right hand side of (8) with respect to ξ and setting to zero. The update rule then becomes:

$$\xi_{n+1}(\cdot) = \xi_n(\cdot) - \frac{1}{\beta} M^{-1} \nabla_{\xi} \mathcal{U}(\xi_n)(\cdot). \quad (9)$$

Given (4) the update rule can also be expressed as:

$$\xi_{n+1}(\cdot) = \xi_n(\cdot) - \frac{1}{\beta} M^{-1} [\nabla_{\xi} \mathcal{U}_{obs}(\xi_n(\cdot)) + \lambda \nabla_{\xi} \mathcal{U}_{dyn}(\xi_n(\cdot))]. \quad (10)$$

C. GP Paths using Hilbert maps

In this section we discuss the shortcoming of the current functional gradient methods when planning paths in occupancy maps. We then propose a method that utilises a continuous GP path representation combined with stochastic sampling to solve this problem using Hilbert maps.

Occupancy maps create several challenges for gradient based path planners. Most importantly, gradient information in an occupancy map is not necessarily useful. The obstacle cost functional defined in (5) requires a workspace cost function, $c(x(\xi(t), u))$. Most trajectory optimisers work with a precomputed cost map, which produce noiseless and informative gradients anywhere in the map. Estimation of the obstacle cost in an occupancy map is more challenging. While the occupancy map (or blurred map if using OGM) can act as $c(x(\xi(t), u))$, the obstacle cost is only defined well in observed areas of the map. Occluded or unreachable regions of the map are labelled as unknown with a probability of occupancy of 0.5. While the spatial gradient of occupancy "pushes" trajectory away from obstacles, the direction might be wrong as it might push the trajectory into unobserved, unsafe regions, as shown schematically in Fig. 1.

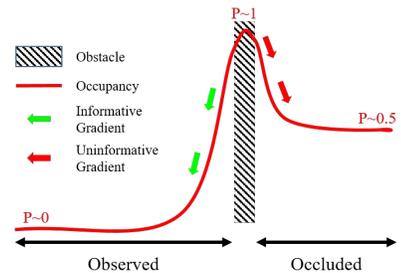


Fig. 1. The occupancy gradient is not necessarily useful for trajectory optimisers. The occupancy, depicted in red, drops with the distance from the obstacle. However, following the occupancy gradient (green and red arrows) does not guarantee safety as it might pull the planner into unobserved, unsafe regions of the map.

The other challenge arising from planning with trajectory optimisers using occupancy maps is that all planners commit to a specific path parametrisation to solve (9). Waypoint parametrisations, as used by [4], [11], [19], have to find a balance between expressiveness and computational complexity. The more recent work in functional gradient motion planning in RKHS [3] and the Gaussian process motion planner [12], [20] define a parametric support with finite resolution for their path representation. These methods produce highly expressive trajectories. However, given the finite resolution of the support, as the optimisation process deforms the trajectory, the spatial density of the support points changes. As a result some areas in the workspace have low support density, which result in a low update rate. This problem is exacerbated when using occupancy maps as some regions in the map have no informative gradients. To prevent corrupted optimisation process, such uninformative updates are rejected, reducing even further the effective density of the support.

Our approach uses Hilbert maps to produce the obstacle functional gradient. We combine a flexible stochastic gradient approach to generate support to form an expressive path based on an iterative GP representation.

1) *Hilbert maps as cost functional*: Most functional gradient motion planners perform optimisation using a well-defined, and usually precomputed, cost map based on the distance to obstacle edges. In our approach, the spatial cost function $c(x(\xi(t), u))$ is the Hilbert occupancy map, which is estimated by (2) along the trajectory and robot body points. Although Hilbert maps do not form a tangible grid as OGM, querying the spatial occupancy gradient is as straightforward as querying an occupancy grid. By applying the chain rule, the Euclidean space gradient of (2) around a query point, \mathbf{x}^* becomes:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}^*} p(y^* = +1 | \mathbf{x}^*, \mathbf{w}) &= \frac{\partial \sigma(\mathbf{z}_{NL})}{\partial \mathbf{z}_{NL}} \frac{\partial \mathbf{z}_{NL}}{\partial \mathbf{x}^*} \\ &\approx \sigma(\mathbf{z}_{NL})(1 - \sigma(\mathbf{z}_{NL})) \mathbf{w}^T \frac{\partial}{\partial \mathbf{x}^*} \hat{\Phi}(\mathbf{x}^*). \end{aligned} \quad (11)$$

Here we used the fact the the derivative of LR is given by $\frac{\partial}{\partial \mathbf{x}} \sigma(\mathbf{x}) = \sigma(\mathbf{x})(1 - \sigma(\mathbf{x}))$.

2) *GP Path*: GPs provide a principled way to represent smooth trajectories. The GP model requires a small set of support points, which define waypoints in configuration space the path should follow. GP regression provides us with a complete solution that allows querying the model at any given time and handles boundary conditions we wish to impose. Unlike other GP-based motion planners [12], [20], the method presented here does not require a predefined support, but rather learns and builds the trajectory support while optimising the path.

A GP path is defined as a vector-valued (multiple output) GP [21]:

$$\xi(t) \sim \mathcal{GP}(\mu(t), K(t, t')), \quad t, t' \in [0, 1]. \quad (12)$$

Here, $\mu(t) \in \mathbb{R}^D$ is the vector-valued mean function of t and $K(t, t') \in \mathbb{R}^D \times \mathbb{R}^D$ is a positive matrix-valued kernel

between $\xi(t)$ and $\xi(t')$ with a corresponding kernel matrix for two different time instances, $\mathbf{K}(t, t')$:

$$\mathbf{K}(t, t') = \begin{bmatrix} k_{1,1}(t, t') & k_{1,2}(t, t') & \dots & k_{1,D}(t, t') \\ \vdots & \ddots & & \vdots \\ k_{D,1}(t, t') & k_{D,2}(t, t') & \dots & k_{D,D}(t, t') \end{bmatrix}. \quad (13)$$

Each element in \mathbf{K} , $k_{d,d'}(t, t')$ represents the effect joint $[\xi t]_d$ at time t has on joint $[\xi t']_{d'}$ at t' .

Updating the model requires conditioning the GP model with waypoint observations. The term observations here is used loosely, and means the states, $\xi^o(t^o)$ at time t^o that the trajectory must pass through. By conditioning the GP with these observations we can compute the *maximum a posteriori* (MAP) path at any query time, t^* :

$$\bar{\xi}(t^*) = \mu(t^*) + \mathbf{K}(t^*, t^o) \mathbf{K}(t^o, t^o)^{-1} (\xi^o(t^o) - \mu(t^o)). \quad (14)$$

The choice of t^o differentiates this work from other GP path representations, such as [12], [20]. While in previous work t^o was fixed a-priori, in this work t^o is learned online.

There are several advantages of using GPs to represent the path. First, we do not need to discretise the path. Instead a finite set of N points, $\{t_i^o, \xi_i^o(t_i^o)\}_{i=1}^N$, serve as the curve support, which can be queried for its MAP value at any given time t^* using (14). Second, the mean function μ provides an explicit prior, which can be exploited by initialising the optimisation with a rough path from a fast path planning method. Finally, boundary conditions can be imposed by treating them explicitly as observations. Besides the obvious boundary conditions at the start and goal points one can define must-visit waypoints along the trajectory or define the robot's direction by including derivative observations [22].

3) *Stochastic Gradient*: A drawback of other functional gradient motion planners is that they either utilise a spatial parameterisation or commit to a finite parametric resolution to represent and update the path. In both cases, this may lead to gaps in the sampling of the objective functional. To overcome this we adopt a resolution-free sampling method. Since the functional objective of (4) can be approximated by a sum of individual points along the path, optimisation of the objective can be performed using *stochastic gradient descent* (SGD) [23]. From (4)-(6), we define an empirical objective functional that approximates the real objective:

$$\hat{\mathcal{U}}(\xi) = \sum_{t,u} \mathcal{U}_{obs}(\xi(t, u)) + \lambda \mathcal{U}_{dyn}(\xi(t, u)) \xrightarrow{n \rightarrow \infty} \mathcal{U}(\xi). \quad (15)$$

A consequence of (15) is that minimising the objective requires reasoning over many points all along the curve. Such a process, which effectively resembles batch optimisation, is computationally infeasible. The approach taken by other trajectory optimisation methods (e.g. [3], [12]) is to estimate $\hat{\mathcal{U}}(\xi)$ with a finite resolution support. However, it is clear from (15) that while this is computationally attractive, such an approach cannot guarantee convergence to the optimum.

The stochastic functional gradient path planner utilises SGD to ensure convergence [24]. SGD randomly selects a

mini-batch from the dataset and then updates the solution in small steps, based on the gradient computed from that mini-batch. In a similar fashion, our method uses random samples $[t^*, u^* \xi_n(t^*)]$ as stochastic training points. The update rule of 9 is then replaced by:

$$\xi_{n+1}(\cdot) = \xi_n(\cdot) - \eta_n A^{-1} \nabla_{\xi} \mathcal{U}(\xi_n(t^*), u^*) \quad (16)$$

where $\eta_n > 0$ is the step size parameter and can be either constant or asymptotically decaying. Matrix A can be seen as a preconditioner that may help accelerate convergence rate, and in many cases is set to the identity matrix [24]. Note the difference between (9) and (16) where the constant regulariser β is replaced by $\frac{1}{\eta_n}$.

Using SGD ensures the convergence, in expectation, of the empirical objective in (15) to the optimal objective, while keeping computational cost per iteration low [24].

4) *Planning on Hilbert Maps Algorithm:* To finalise the stochastic functional gradient path planning algorithm, we need to define the functional gradient of \mathcal{U}_{obs} and \mathcal{U}_{dyn} . For a general functional of the form $\mathcal{F}(\xi) = \int_a^b v(t, \xi, \xi') dt$, the gradient is given by:

$$\nabla \mathcal{F}_{\xi}(\xi) = \frac{\partial v}{\partial \xi} - \frac{d}{dt} \frac{\partial v}{\partial \xi'}. \quad (17)$$

Applying (17) to \mathcal{U}_{obs} at a sampled time t^* and for robot body point u^* yields;

$$\nabla \mathcal{U}_{obs}(\xi(t^*), u^*) = \frac{\partial}{\partial \xi(t)} x(\xi(t^*), u^*) \nabla_x c(x(\xi(t^*), u^*)). \quad (18)$$

Here, $\mathbf{J}(t^*, u^*) \equiv \frac{\partial}{\partial \xi(t)} x(\xi(t^*), u^*)$ is the workspace Jacobian. ∇_x emphasises that this is a Euclidean gradient of the cost function c .

We opted to use the squared velocity norm integral, as shown in (6), as the dynamic penalty \mathcal{U}_{dyn} . Using the L_2 norm such as in [3] is less attractive as it implicitly defines a favourable simple mean solution which requires tuning of regularisation coefficients for different scenarios. With (6), the functional gradient can be easily computed as:

$$\nabla \mathcal{U}_{dyn}(\xi(t^*)) = -\frac{d^2}{dt^2} \xi(t^*). \quad (19)$$

Again, as $\xi(t)$ is represented by a GP, computing the derivative is straight-forward. The update rule at time t^* and robot body point u^* can now be summarised from (16), (18), and (19) as:

$$\begin{aligned} \xi_{n+1}(t^*) &= \xi_n(t^*) - \\ \eta_n A^{-1} &\left(\mathbf{J}(t^*, u^*)^T \nabla_x c(x(\xi(t^*), u^*)) + \lambda \frac{d^2}{dt^2} \xi(t^*) \right). \end{aligned} \quad (20)$$

The algorithm for path planning using Hilbert maps is shown in Algorithm (1). The algorithm accepts an optional initial solution to start optimisation with, otherwise a straight line trajectory is used initially. This initial path is then used as the mean function of the GP path.

Algorithm 1: Functional gradient path planning using Hilbert maps

Input: \mathcal{H} : Hilbert Occupancy Map.
 $\xi(0), \xi(1)$: Start and Goal states.
 P_{safe} : No obstacle Threshold.
 $\xi_{initial}(t)$: Initial solution (optional).
 K : covariance function for GP path.

Output: $\xi_{min}(t)$
// Use prior guess/solution if available

if $\xi_{initial}$ **then**
| $\mu_0 \leftarrow \xi_{initial}$
else
| $\mu_0 \leftarrow (\xi(1) - \xi(0))t + \xi(0)$
end
 $\xi_0 \leftarrow \mathcal{GP}_0 \sim \mathcal{GP}(\mu_0, K)$
 $n = 0$
while ξ **not converged** **do**
| // Stochastic sampling
| $(t_{sup}, u_{sup}) \leftarrow$ Draw mini-batch randomly
| **foreach** $(t^*, u^*) \in (t_{sup}, u_{sup})$ **do**
| | $P_{occ} \leftarrow \mathcal{H}(x(\xi_n(t^*), u^*))$ Eq. (2)
| | **if** $P_{occ} \leq P_{Safe}$ **then**
| | | $\xi_{n+1}(t^*) \leftarrow$ update rule Eq. 20
| | | $\xi_n(t) \leftarrow$ update GP with $(t^*, \xi_{n+1}(t^*))$
| | **end**
| **end**
| // Update boundary conditions
| $\xi_n(t) \leftarrow$ **update:** $(0, \xi(0)), (1, \xi(1))$
| $\xi_{n+1}(t) \leftarrow \mathcal{GP}(\xi_n(t), K)$
| $n = n + 1$
end

At each iteration, a mini-batch (t_{sup}, u_{sup}) is drawn randomly. For each point $t^* \in t_{sup}$ and $u^* \in u_{sup}$ the corresponding state $\xi_n(t^*)$ is computed from the active GP path model, $\xi_n(t)$. To perform the functional update, the Hilbert map is queried at $x(\xi(t^*), u^*)$, and the probability of occupancy P_{occ} is obtained. Functional updates may only happen if the occupancy is within safe limits, i.e. free from obstacles. Using (20) new states $\xi_{n+1}(t^*)$ are computed and the path model $\xi_n(t)$ is updated with the new path observations. To enforce a valid path the boundary conditions are then incorporated into the GP model as additional observations. Finally, a new path model is initialised with the previous model as its mean function $\xi_{n+1}(t) \sim \mathcal{GP}(\xi_n(t), K)$.

IV. RESULTS

In this section, we evaluate the performance of the stochastic functional gradient path planner in simulation and with real data and provide comparisons to other methods.

A. Simulations

In this section we compare the proposed method with the functional gradient motion planner in RKHS [3], as both methods are related. We show that while [3] provides a

flexible path representation, changes are needed in order to perform optimisation in occupancy maps.

Most trajectory optimisers assume full knowledge about the location of obstacles and precompute offline a cost field, $c(\mathbf{x})$ in workspace \mathcal{W} which penalises the proximity to obstacles, for example [25]:

$$c(\mathbf{x}) = \begin{cases} -d(\mathbf{x}) + \frac{1}{2}\epsilon & d(\mathbf{x}) < 0 \\ \frac{1}{2\epsilon}(d(\mathbf{x}) - \epsilon)^2 & 0 \leq d(\mathbf{x}) \leq \epsilon, \\ 0 & \text{otherwise} \end{cases}, \quad (21)$$

where $d(\mathbf{x})$ is the distance of \mathbf{x} to the boundary of the nearest obstacle and ϵ is a minimal safety buffer from obstacles. An example of path planning with a precomputed cost field based on [3] is shown in Fig. 2: 2a depicts the iterative optimisation process and 2b shows the optimal path. These noiseless obstacles result in a cost gradient that is well-defined anywhere in the workspace, including inside obstacles. As the gradient is precomputed, evaluating the update rule is also computationally efficient, leading to fast convergence at a local minima. However, planning using occupancy maps adds several challenges to this optimisation process. The cost field, and more importantly its spatial gradients, are not necessarily informative. In addition, as the map is generated by laser observations, all predictions are noisy. As a result, the assumptions at the core of the functional gradient-based planner are no longer valid and a change to the planning process is required.

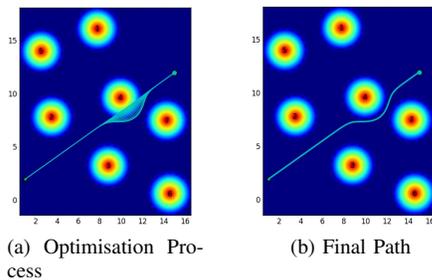


Fig. 2. RKHS motion planning [3] in a precomputed cost field calculated according to (21). Obstacle potential costs and their spatial gradients are well-defined anywhere in the workspace. Dashed lines illustrate the obstacles boundaries. Star and pentagon mark the start and goal points, respectively. (a) paths generated during the RKHS optimisation process. (b) shows the optimal path.

Constructing a Hilbert map for a simulated environment of randomly placed obstacles is achieved by generating an occupancy dataset. The dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_1^N$ is created by randomly placing occupied observations, $y_i = +1$, on the boundaries of all obstacles and free observations $y_i = -1$ outside obstacles. There are no laser points inside obstacles. After the dataset is created, the map model is fitted. Fig. 3 depicts a Hilbert continuous occupancy map generated for the environment shown in Fig 2. Using the continuous map representation, the occupancy and its gradient can be queried at any location.

Fig. 4 shows an attempt to plan a path in an occupancy map using the functional gradient method described in [3]

with various support size ($N = 5, 50, 100, 200$). The cyan line depicts the optimal path after the algorithm has converged. Clearly, the resulting path is unsafe regardless of the size of support, N , used in the optimisation process. As expected, increasing the size of support lead to a more expressive path. Yet, even with $N = 200$ the resulting path was not safe. The reason lies in the lack of informative gradient in the occupancy map and the finite parametric resolution of the path representation. As the optimisation process deforms the curve, the spatial density of the support changes too. Consequently, there is less support around critical areas of the map such as the boundaries of obstacles. Increasing resolution even further will not alleviate this problem since we have no *a-priori* knowledge of the number of required support points. Inadvertently, increasing the number of support points create unnecessary jerks in the curve, as a response to noisy occupancy gradient, as shown in Fig 4.

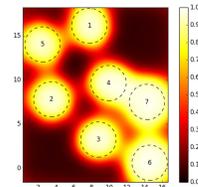


Fig. 3. Continuous occupancy Hilbert map for the environment shown in Fig 2. The map shows the probability of occupancy, $p(y^* = +1|\mathbf{x}^*, \mathbf{w})$ as in (2). Note that the optimal path of in Fig. 2 passes through the gap between obstacles 4 and 7. In the Hilbert map, such a path is considered invalid as it passes occupied or unsafe area.

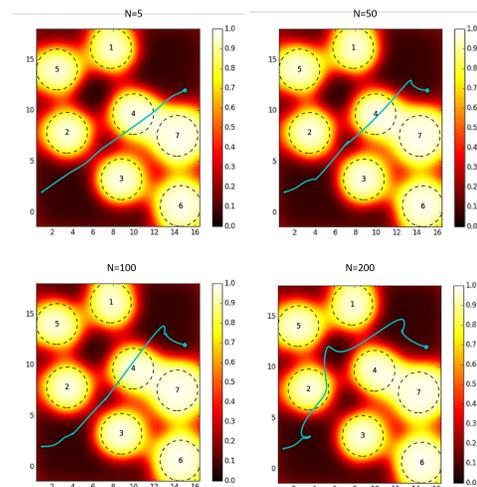


Fig. 4. RKHS motion planning [3] fails to plan using Hilbert maps. The individual plots compare the effect the size of the path support N has. The final path is depicted in cyan, while the cyan star and pentagon mark the start and goal points, respectively. The RKHS motion planner does not find a valid solution since it represents the path with a parametric resolution, which leads to gaps in the sampling of the objective functional.

Our stochastic functional gradient method does not commit to a specific parametric resolution, rather it randomly selects points along the curve during the optimisation. Fig. 5a shows in cyan the average path planned using the occupancy

map of Fig 3. Since the optimisation objective balances an obstacle cost with a penalty on the trajectory shape, the optimal path is collision-free and smooth. Finding a path relies on generating enough samples to instantiate a gradient update, especially in areas of high importance such as obstacles boundaries. Fig. 5b shows the convergence of the optimisation process to the minima of the objective, by plotting the maximum occupancy along the trajectory at each iteration. The maximum occupancy along the path drops steadily as the optimisation progresses. However, it cannot drop further than 0.4, since the predictive occupancy between obstacles 2 and 4 is approximately that number. Yet, Fig. 5b provides empirical evidence for the expected optimality of the stochastic process.

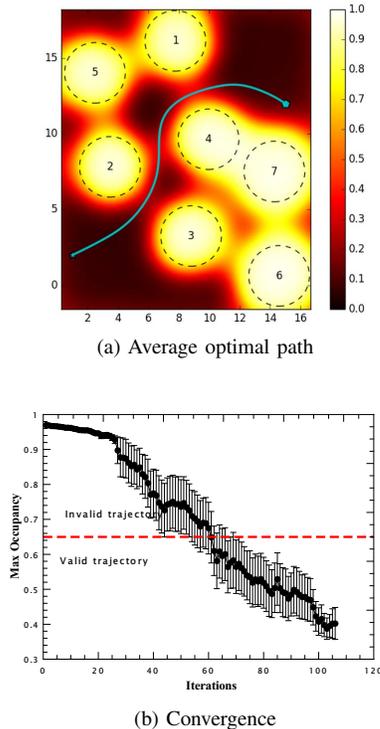


Fig. 5. Stochastic functional gradient motion planner results. (a) Average path over 10 repetitions. The trajectory is depicted in cyan while the star and pentagon marker indicate the start and goal points, respectively. The average path follows the mid line between obstacles, which reduces the obstacle cost. (b) Shows the convergence of the stochastic functional gradient motion planner. The maximum occupancy along the trajectory is plotted as a function of the iterations. Data shown is the average over 10 repetitions. The $P = 0.5$ dashed red line marks the threshold for a valid trajectory. Note that the maximum occupancy does not reduce to zero, as the continuous occupancy map predictions for the gap between obstacles 1,2 and 4 are approximately 0.4.

B. Real data

The map for this experiment was generated using the Intel-Lab dataset (available at <http://radish.sourceforge.net/>). This map contains many rooms and dead-ends that might challenge the optimiser. We compared the optimal trajectory of our proposed method with two other standard planning methods; RRT* [26] and PRM* [27] using implementations from the *Open Motion Planning Library* (OMPL) [28].

Fig. 6 shows a comparison between the different methods. Both RRT* and PRM* generate a path from start to goal. The path consists of waypoints (states) the robot should pass through. The list of waypoints provides a very sparse representation of the path that requires additional resources in order to transform into robot actions. In contrast, the proposed stochastic functional motion planner provides a detailed and smooth path represented by a function of t . Furthermore, the paths generated by RRT* and PRM* might follow close to walls or overshoot the corner leading to a higher risk of collision, and longer paths. The paths of our stochastic planner tend to follow the mid line between obstacles and perform smooth turns resulting in shorter and safer trajectories.

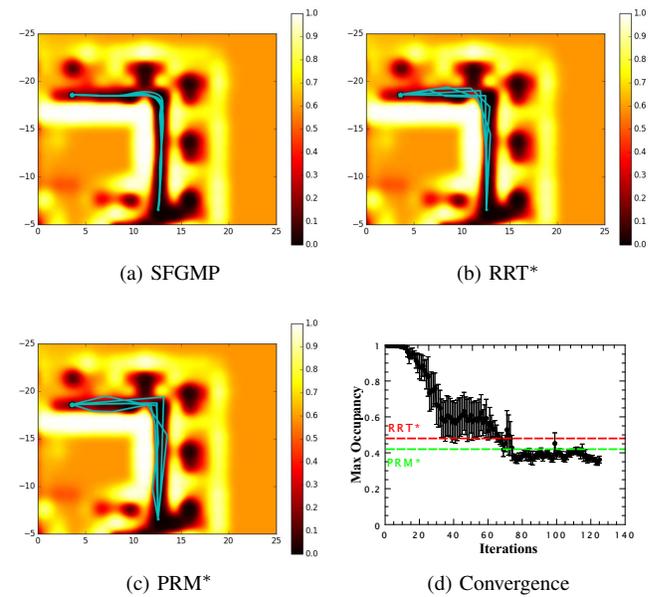


Fig. 6. Comparison of path planning methods on a continuous occupancy map of Intel-Lab; (a) stochastic functional gradient motion planner (SFGMP), (b) RRT* and (c) PRM*. Each image shows five repetitions of path planning with each method. SFGMP paths are smooth and follow the mid lines between walls. RRT* and PRM* produce paths that move the robot dangerously close to walls at times. (d) shows convergence of the stochastic functional gradient motion planner. The maximum occupancy along the trajectory is plotted as a function of the iteration. Data shown is the average over 5 repetitions. Red and green dashed lines mark the average performance of RRT* and PRM* respectively. Our proposed stochastic planner significantly outperforms the other methods. Note that the maximum occupancy does not reduce to zero, as the continuous occupancy map predictions for the end point is approximately 0.35.

Table I and Fig. 6d provide a quantitative comparison between our stochastic functional gradient motion planner (SFGMP), RRT* and PRM*. The objective of the optimisation is to minimise the obstacle cost. Fig. 6d shows the reduction of the maximum occupancy along trajectory as the optimisation progresses. After converging to the optimal solution, the safety of the path of the proposed method is significantly better than that of the other two methods. The results in Table I summarises the expected performance. The maximum occupancy along the path, which indicates the path safety, is 0.36 for the proposed method and 0.42 and 0.48

for PRM* and RRT*, respectively. Furthermore, the penalty term, \mathcal{U}_{dyn} , in the objective in (15) leads optimisation to prefer shorter paths. Consequently, the path generated by our method outperforms the other sampling-based methods.

TABLE I
PERFORMANCE COMPARISON

	SFGMP	RRT* [26]	PRM* [27]
Maximum occupancy	0.36 ± 0.02	0.44 ± 0.03	0.46 ± 0.03
Path length [m]	20.90 ± 0.10	21.50 ± 0.10	22.50 ± 0.50

V. CONCLUSIONS

This paper introduced a novel method for path optimisation using occupancy maps. Sampling-based techniques are the prevalent method for path planning using occupancy maps. Although these techniques are flexible and have a high success rate in finding safe paths, optimising additional properties of the path such as length and execution time are not part of their reasoning. Trajectory optimisers, on the other hand, are designed for that purpose. Yet, the current implementations of trajectory optimisers require a finite resolution in the trajectory support and rely on having access to a well defined cost potential field. However, neither of these requirements are met by occupancy maps. Gradients obtained from the map's occupancy are noisy and not necessarily informative, which limits the choice of trajectory support, especially when the resolution is finite.

The planning method used in this paper employs stochastic optimisation to enhance the expressiveness of the basic functional gradient motion planner. It removes the need to commit to an a-priori parametric resolution, which allows our planner to better handle obstacles. The GP paths used in the planner provide a structured and flexible representation that can easily incorporate prior knowledge or initial solutions, such as coarse paths generated by a sampling based method.

Future areas of work include improving convergence rates which could be approached by targeting under-sampled areas of the curve by biasing the stochastic sampling. Using the variance prediction provided by the GP path, one can employ a Bayesian optimiser to direct the sampling toward unexplored regions. Another avenue for improvement is to take advantage of modern multi-core systems by parallelising the optimisation.

REFERENCES

- [1] E. G. Tsardoulis, 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*, pp. 1–30, 2016.
- [2] F. Ramos and L. Ott, "Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent," in *Proc. Robotics: Science and Systems (RSS)*, 2015.
- [3] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces," in *Proc. Robotics: Science and Systems (RSS)*, 2016.
- [4] 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 (IJRR)*, 2013.

- [5] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, 1979.
- [6] P. Bhattacharya and M. L. Gavrilo, "Voronoi diagram in optimal path planning," in *Proc. International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, 2007.
- [7] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path planning for mobile robot navigation using voronoi diagram and fast marching," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.
- [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] S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep., 1998.
- [10] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research (IJRR)*, 1986.
- [11] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
- [12] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *Proc. IEEE Conference on Robotics and Automation (ICRA)*, 2016.
- [13] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, 1989.
- [14] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research (IJRR)*, 2012.
- [15] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Mathematical Programming*, 2011.
- [16] A. Rahimi and B. Recht, "Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning," in *Proc. Neural Information Processing Systems (NIPS)*, 2009.
- [17] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Neural Information Processing Systems (NIPS)*, 2001.
- [18] A. Melkumyan and F. Ramos, "A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.
- [19] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental Trajectory Optimization for Real-Time Replanning in Dynamic Environments," in *Proc. International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.
- [20] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs," in *Proc. Robotics: Science and Systems (RSS)*, 2016.
- [21] M. A. Alvarez, L. Rosasco, and N. D. Lawrence, "Kernels for Vector-Valued Functions: a Review," *arXiv preprint arXiv:1106.6251*, jun 2011.
- [22] E. Solak, R. Murray-Smith, W. Leithead, C. Rasmussen, and D. Leith, "Derivative observations in Gaussian process models of dynamic systems," in *Proc. Neural Information Processing Systems (NIPS)*, 2002.
- [23] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of COMPSTAT*, 2010.
- [24] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," *arXiv preprint arXiv:1606.04838*, 2016.
- [25] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. International Conference on Robotics and Automation (ICRA)*, 2009.
- [26] S. Karaman, "Incremental sampling-based algorithms for optimal motion planning," *Proc. Robotics Science and Systems (RSS)*, vol. 104.
- [27] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research (IJRR)*, 2011.
- [28] I. A. Sucas, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *Robotics & Automation Magazine*, 2012.