CHiMP: A Contact based Hilbert Map Planner

Constantin Uhde[1], Emmanuel Dean-Leon[2] and Gordon Cheng[3]

***Abstract*— This work presents a new contact-based 3D path planning approach for manipulators using robot skin. We make use of the *Stochastic Functional Gradient Path Planner*, extending it to the 3D case, and assess its usefulness in combination with multi-modal robot skin. Our proposed algorithm is verified on a 6 DOF robot arm that has been covered with multi-modal robot skin. The experimental platform is combined with a skin based compliant controller, making the robot inherently reactive. We implement different state-of-the-art planners within our contact-based robot system to compare their performance under the same conditions. In this way, all the planners use the same skin compliant control during evaluation. Furthermore, we extend the stochastic planner with tactile-based explorative behavior to improve its performance, especially for unknown environments. We show that CHiMP is able to outperform state of the art algorithms when working with skin-based sparse contact data.**

I. Introduction

When children explore their surroundings, they use their tactile senses of touch in conjunction with their sight. In fact, the interdependence of these two senses, particularly during the learning phase, is fundamental [1]. This interdependence helps humans to navigate in unknown environments, with both vision and tactile exploration, in order to maximize the knowledge of their surroundings. Tactile feedback is used, especially if vision is temporarily impaired, for example, if one tries to find the light switch in a dark room.

Planning algorithms are used in robotics to achieve tasks such as grasping and motion planning. The environment of the robot is often recorded with the help of laser scanners, cameras, and depth sensors. This leaves out tactile information (e.g., robot skin), which can provide additional knowledge especially when these optical sensors suffer from occlusions.

Existing state-of-the-art planning algorithms based on sampling [2], [3], or optimization [4], [5], have yet to be optimized for planning with robot skin.

The concept of *intentional contact* was introduced with the work of Guadarrama et al. [6], with a focus on control, similar to the work of Jain et al. [7]. The goal of this work is to integrate a feasible *planning*-based approach with robot skin to enhance exploration and general planning behavior in unknown environments.

To achieve this, we selected the *Stochastic Functional Gradient Path Planning in Occupancy Maps* [8] as planning approach since it is able to robustly incorporate un-occupied and occupied data samples into the environment reconstruction.

Institute for Cognitive Systems, Technical University of Munich
[1]c.uhde@tum.de, [2]dean@tum.de, [3]gordon@tum.de

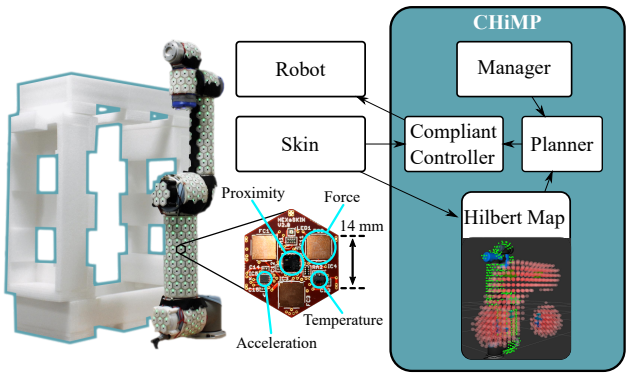

Fig. 1: Overview of the system. The planner interacts with the compliant skin controller to provide reactive planning in the presence of complex obstacles. The robot skin [9] is a multi-modal sensor providing proximity, acceleration, force, and temperature information on each skin cell.

A. Structure of this Work

The rest of this paper is structured as follows: Section II covers the related work, Section III covers the chosen approach and Section IV proposes a flexible exploration versus exploitation extension. Results on a real robot and a comparison of different planners are presented in Section V. The paper is concluded with Section VI.

II. Related Work

A. Hilbert Maps

This work utilizes the Octomap implementation for data preparation and sensor-fusion [10]. Octomaps are a very efficient and scalable 3D environment representation that handles robust sensor fusion. It includes descriptions for both the occupied and unoccupied regions. The Octomap is used as an input for the Hilbert map occupancy classifier. Hilbert maps [11] are a solution to generating a robust probabilistic occupancy representation. The big advantage over occupancy distance fields is the fact that Hilbert maps can be generated and updated online [4], [5]. The map uses observations to generate a continuous representation of the world without a priori discretization into grid cells. It also captures statistical relations between observations intuitively.

This occupancy inference is important for working with sparse occupancy data, for example, contact information produced by the robot skin. Hilbert maps are built by first transforming observations via a kernel approximation into a high dimensional Reproducing Kernel Hilbert Space, where a linear classifier learns the map representation. We use the Nystroem approximation [12] since it is simple to implement while providing good generalization performance by correctly labeling undiscovered areas. The objective function

is convex in all parameters, which ensures optimality. The model is trained via Stochastic Gradient Descent which decouples computational cost from the number of observations.

B. Planning

The continuous occupancy reconstruction from sparse data allows employing an optimization-based planning approach that is able to maximize the trajectory distance to the obstacle by design. This helps with reducing the number of required contact points during the exploration phase. Following the principles behind the Hilbert Map, the selected planner [13], [8] utilizes functional gradient descent and doesn't commit to a fixed trajectory resolution. This makes the planner a very elegant and conceptually compact solution to the problem of planning in maps.

III. Approach

A. Querying the Hilbert Map Classifier

Ramos et al. [11] describe the Hilbert map classifier. The occupancy of a new position x_* is queried with the following sigmoid logit function:

$$\mathcal{H}(x_*) = P(y_* = -1|x_*, w) = \frac{1}{1 + \exp(w^T x_*)} \quad (1)$$

where $y_* = -1$ is the value for non-occupancy and w is the vector of model parameters. The key to correctly modeling a complex environment lies in not directly applying the highly nonlinear position values of x but a high-dimensional feature projection $\hat{\Phi}(x)$. This effectively results in a kernel projection in the logit function.

$$\hat{\Phi}(w)^T \hat{\Phi}(x_*) \approx k(w, x_*) \quad (2)$$

This *kernel trick* can be used, to train the linear logistic regression classifier on highly nonlinear data by transforming it into high-dimensional space, in which a suitable linear hyperplane may exist.

B. Nystroem features

Belongie et al. provide the steps to calculate the Nystroem approximation [12] in a concise fashion [14]. The idea is that a subset of the training data is sufficient for the classifier to work. The subset is called *inducing points* from here on. The standard kernel matrix can then be displayed as

$$K = \begin{bmatrix} K_{11} & K_{21}^T \\ K_{21} & K_{22} \end{bmatrix} \quad (3)$$

$K_{11} \in \mathbb{R}^{m \times m}$ is full rank and needs full evaluation only once, since it is comprised of the static *inducing points*-set. $K_{21} \in \mathbb{R}^{(n-m) \times m}$ is calculated for all points of interest, and $K_{22} \in \mathbb{R}^{(n-m) \times (n-m)}$ can be inferred from the other two. This reduces the number of required computations greatly. Using the eigendecomposition of K gives

$$K = U \Lambda U^T = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \Lambda \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^T = \begin{bmatrix} U_1 \Lambda U_1^T & U_1 \Lambda U_2^T \\ U_2 \Lambda U_1^T & U_2 \Lambda U_2^T \end{bmatrix} \quad (4)$$

Where U_1 and Λ can be found by eigendecomposition of K_{11} which is comprised of the inducing points. $K_{21} = U_2 \Lambda U_1^T$ can be rewritten as

$$U_2 = K_{21} U_1 \Lambda^{-1} \quad (5)$$

Which leaves us with $K_{22} = U_2 \Lambda U_2^T$. Substituting U_2 with equation 5 results in

$$K_{22} = \left(K_{21} K_{11}^{-\frac{1}{2}} \right) \left(K_{21} K_{11}^{-\frac{1}{2}} \right)^T \quad (6)$$

Note here that this equation contains the feature transform for the non-inducing points in K_{22} which will be important later on. K can then be reconstructed by writing

$$K = \hat{\Phi} \hat{\Phi}^T = \begin{bmatrix} K_{11}^{\frac{1}{2}} \\ K_{21} K_{11}^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} K_{11}^{\frac{1}{2}} \\ K_{21} K_{11}^{-\frac{1}{2}} \end{bmatrix}^T \quad (7)$$

$K_{11}^{-\frac{1}{2}}$ itself is easy to calculate by utilizing the eigendecomposition and applying the exponent to the elements of the diagonal matrix Λ. The feature vector is then calculated as follows:

$$\hat{\Phi}(x) = (k(x, \hat{x}_1), ..., k(x, \hat{x}_m)) \hat{\Lambda}^{-\frac{1}{2}} \hat{U}_{11}^T \quad (8)$$

Here, for new samples, only the $K_{21} = (k(x, \hat{x}_1), ..., k(x, \hat{x}_m))$ matrix is calculated with the kernel function $k(x, x_*)$.
The Nystroem kernel approximation speeds up the Hilbert map and trajectory training and querying to allow near-real-time performance of the planning algorithm.

C. Hilbert Map Update rule

According to Ramos et al [11], gradient descent is employed to train the map model, which minimizes the following objective function of the weight vector:

$$NLL(w) = \sum_{i=1}^{N} \log \left(1 + \exp(-y_i w^T \cdot \hat{\Phi}(x_i)) \right) + R(w) \quad (9)$$

where $R(w) \in \mathbb{R}$ is a regularizer term (L1, L2, or the linear combination of L1 and L2 called elasticnet) and x_i, y_i are the data points and their occupancy label, respectively. The gradient is calculated as:

$$\nabla NLL(w) = \sum_{i=1}^{N} -y_i \hat{\Phi}(x_i)(1 + \exp(y_i w^T \cdot \hat{\Phi}(x_i)))^{-1} + \frac{\partial R(w)}{\partial w} \quad (10)$$

This gradient function is then used in Stochastic Gradient Descent, which randomly selects one sample from the training data and updates the weights in an iterative fashion with the following update rule:

$$w_t = w_{t-1} - \eta_t A_t^{-1} \frac{\partial}{\partial w} NLL(w) \quad (11)$$

where $\eta > 0$ is the learning rate and A is a preconditioning matrix, which can be set to the identity matrix in the naive case. This update rule can be directly used in online learning, feeding new data points into the model as they arrive. Figures 3 a) and b) depict a basic Hilbert Map generated from binary occupancy samples in a two-dimensional scenario.

D. Defining the Planner Cost-Function

Similar to points in the the Hilbert map, one trajectory is a point in its own high dimensional Hilbert Space. Francis et al. [13] [8] describe the steps.

$$\xi : [0, 1] \rightarrow \mathcal{Q} \in \mathbb{R}^D \quad (12)$$

A path ξ which maps time $t \in [0,1]$ to the configuration space $\mathcal{Q}$ (in our case the joint space). The Hilbert Space properties allow for homologous transformation between trajectories. The gradient information for direction can be obtained by constructing an objective functional which assigns a scalar real-valued cost to a trajectory: $\mathcal{U}(\xi) : \Xi \to \mathbb{R}$

The functional is typically comprised of two penalties: a obstacle proximity and a shape or dynamics part with a multiplier λ to balance the functionals impact.

$$\mathcal{U}(\xi) = \mathcal{U}_{obs}(\xi) + \lambda \mathcal{U}_{dyn}(\xi) \tag{13}$$

$$\mathcal{U}_{obs}(\xi) \approx \sum_{(t,u)\in\mathcal{T}(\xi)} c(x(\xi(t),u)) \tag{14}$$

$$\mathcal{U}_{dyn}(\xi) = \frac{1}{2}\int_0^1 \left\| \frac{d}{dt}\xi(t) \right\|^2 dt \tag{15}$$

Here, $\mathcal{T}(\xi) = t, u_i$ is a finite set of time t_i and body points u_i. c is the workspace cost function.

The obstacle functional $\mathcal{U}_{obs}$ is defined in Configuration Space. Obstacles are represented by the Hilbert map in Cartesian Space. To calculate the cost in Configuration Space, a kinematic mapping is required which maps a $\mathcal{Q}$-space configuration $\xi(t)$ together with a point on the robot $u \in B$ to a point in Workspace $x(\xi(t), u)$. The obstacle cost functional is a reduction via a workspace cost function $c : \mathbb{R}^3 \to \mathbb{R}$ over all trajectory and robot points. In reality, the *reduce* operator needs to be representable by a sum over a finite set of trajectory and robot points to be practical.

The dynamics functional $\mathcal{U}_{dyn}$ is a secondary constraint which smooths the results of the obstacle functional. Therefore, it is important to balance the functional multiplier λ correctly to not conflict with the actual obstacle avoidance provided by $\mathcal{U}_{obs}$, while providing sufficient trajectory smoothing. $\mathcal{U}_{dyn}$ can be implemented in various ways, with one of the more popular approaches being a penalty on first and second order derivatives of the trajectory since this is independent of time parametrization and works with already given information. The downside of this variant of the regularizer is the lack of direct control over real-world dynamics.

The cost function $\mathcal{U}$ is used to train a linear regression model in combination with the *kernel trick*, which we utilize for generating trajectory candidates.

E. Functional gradient descent

The cost functional is not used directly, but its gradient, similar to the approach for Hilbert maps. The obstacle and dynamics costs in Eq. 14 and 15 are both of the form $\int_a^b v(t, \xi, \xi')$ which allows to formulate the gradient according to Zucker et al. [15] $\nabla\mathcal{F}_\xi(\xi) = \frac{\partial v}{\partial \xi} - \frac{d}{dt}\frac{\partial v}{\partial \xi'}$. This results in the following cost function gradient.

$$\nabla_\xi \mathcal{U}(\xi) = \frac{\partial}{\partial \xi(t)} x(\xi(t),u) \nabla_x c(x(\xi(t),u)) - \frac{d^2}{dt^2}\xi(t) \tag{16}$$

where ∇_x is the Euclidean gradient and $\frac{\partial}{\partial \xi(t)} x(\xi(t), u)$ is the workspace Jacobian.

Algorithm 1: Stochastic Gradient Path Planner

Require: $\mathcal{H}$: Occupancy Map Eq. 1
$\xi(0), \xi(1)$: Start and Goal states
P_{safe}: Safety threshold
$\hat{\Phi}(t', \cdot)$: Inducing Feature vector Eq. 8
Ensure: w_{min} Optimized Trajectory Weights
$w \leftarrow$ initialize with lin. interpolation from $\xi(0)$ to $\xi(1)$
$n \leftarrow 0$
while not converged and $n < n_{max}$ **do**
  Stochastic sampling:
  (t_s, u_s) $U[0,1] \leftarrow$ Draw sample uniformly
  $P_{occ} \leftarrow \mathcal{H}(x(\xi_n(t^*), u^*))$
  if $P_{occ} \leq P_{safe}$ **then**
    $w_{n+1} \leftarrow$ update Eq. 21
  end if
  Fix boundary conditions Eq. 22
end while
$n \leftarrow n + 1$

F. Planner Update rule

Francis et al. [13] [8] describe the update rule, which is derived as linear approximation in the region of the current trajectory ξ_n.

$$\mathcal{U}(\xi) \approx \mathcal{U}(\xi_n + \nabla_\xi \mathcal{U}(\xi_n)(\xi - \xi_n) + \mathcal{O}((\xi - \xi_n)^2) \tag{17}$$

where $\mathcal{O}((\xi - \xi_n)^2)$ is a regularizer based on the norm of the update $\frac{1}{2\eta_n}\|\xi - \xi_n\|_M^2$ which is the squared Mahalonobis distance and a learning rate η_n.

$$\xi_{n+1} = \arg\min_\xi \mathcal{U}(\xi_n) + (\xi - \xi_n)^T \nabla_\xi \mathcal{U}(\xi_n) + \frac{1}{2\eta_n}\|\xi - \xi_n\|_M^2 \tag{18}$$

By differentiating eq. 18 with respect to ξ we get the general update rule, which is valid for various cost functions $\mathcal{U}$:

$$\xi_{n+1} = \xi_n - \eta_n M^{-1} \nabla_\xi \mathcal{U}(\xi_n) \tag{19}$$

The actual trajectory is sampled from the weight matrix w at time t:

$$\xi(t) = \xi_0(t) + w^T \hat{\Phi}(t')^T \hat{\Phi}(t) \tag{20}$$

Where ξ_0 is the initialization of the trajectory, which can be a simple interpolation between desired start and goal configuration. This can be left out, if the weights are an interpolation themselves, as w has dimensionality $\mathbb{R}^{c\times m}$ where c is the configuration space dimensionality and m the number of inducing points t' for the Nystroem Kernel. $\hat{\Phi}(t')\hat{\Phi}(t)^T$ is a new instance of the Nystroem kernel approximation discussed in Section III-B. This changes the optimization goal to $w_{optimal} = \arg\min_w \mathcal{U}(w)$ and allows to formulate the weight update rule:

$$w_{n+1} \approx w_n - \eta_n M^{-1} \hat{\Phi}(t')^T \hat{\Phi}(t_i) \nabla_\xi \mathcal{U}(\xi_n)(t_i) \tag{21}$$

The boundary conditions are handled by a similar equation to 21. The difference lies in the used time-stamps $t_b \in 0, 1$ and cost function $\Delta x_b(t_b)$, which is the deviation from the start and goal states:

$$w_{n+1} = w_n - M^{-1} \hat{\Phi}(t')^T \hat{\Phi}(t_b) \Delta x_b(t_b) \tag{22}$$

The resulting weight update algorithm can be seen in Algorithm 1. Note that the difference to the employed algorithm in [8] lies in not using batch processing and utilizing the same

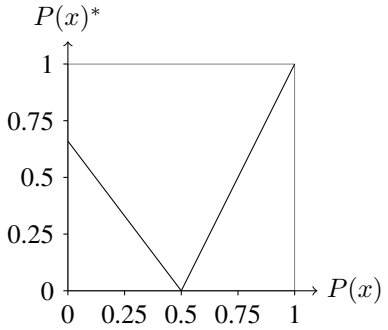

Fig. 2: Simple occupancy probability remapping from regressor output $P(x)$ to simple remapped output $P(x)^*$ with a set point $s = 0.5$ for pure explorative behaviour. Where $P(x)$ represents the *occupancy probability* of position x.

weight matrix for initial trajectory and bounds correction. Both changes are simplifications to the algorithm by giving up part of its generality. The removed batch processing also means that typical iteration counts are visually higher, while actually performing the same amount of calculations.

IV. Exploration vs. Exploitation

The presented approach provides a sample efficient framework for planning in known environments. With CHiMP, we offer a very elegant and simple contribution for seamless transition between exploration and exploitation to facilitate working with incomplete information about the surroundings. The optimization based algorithm, as discussed in Section III, tries to fit a trajectory into the areas of the Hilbert Map that are least probable to contain obstacles. This probability is normalized in the [0,1] range with 0 being zero probability of occupancy and 1 describing certain occupancy.

This map results in very conservative behavior as the trajectory is drawn to known and occupancy-free areas of the environment. To prioritize undiscovered regions, we propose to change the occupancy cost functional to facilitate exploration. This can be achieved by remapping the probability values in a way that minimizes probabilities between 0 and 1, thus the robot is drawn towards areas of medium probability. With roughly equal amounts of occupied and non-occupied training samples, the *occupancy probability* for position x, $P(x) = 0.5$ represents two phenomena: Since the map is continuous, the value of 0.5 occurs at the transition between free and occupied areas, for example right in front of a known wall. Additionally, the same value may arise in unknown areas where the classifier, using the Nystroem Kernel approximation, is not certain and thus defaults to the inherent probability value ($P(x) = 0.5$).

A naive remapping approach looks similar to an inverted triangular pulse function:

$$P(x)^* = \begin{cases} (s - P(x))/(\rho s) & \text{for } 0 \leq P(x) < s \\ (P(x) - s)/s & \text{for } s \leq P(x) \leq 1 \end{cases} \quad (23)$$

Where s is the target probability to be explored. The mapping is visualized in Fig. 2. In this paper, we use $s = 0.5$ as an example for purely explorative behaviour. $\rho > 1$ guarantees

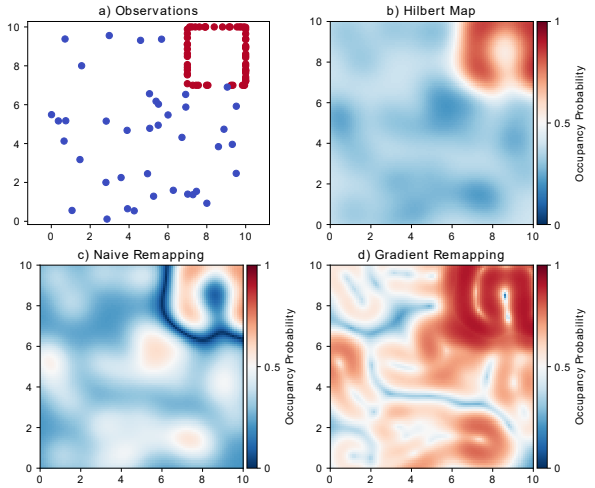

Fig. 3: Implementation of two dimensional Hilbert Map remapping variants. a) The observation data used for training, where red indicates occupied (1) and blue free (0) regions. The goal is to emphasize the undiscovered areas (neither blue nor red points). b) The standard continuous Hilbert map output for exploitative planning. c) The naive remapping applied to b), incorrectly highlighting the transitional regions around obstacles (dark blue region). In this case, the robot would fit the trajectory close to obstacles. d) The gradient-based remapping applied to b), favouring the undiscovered regions correctly, while excluding the transitional areas.

that the remapped values at $P(x) = 0$ are lower than $P(x) = 1$, in this way, the system prefers *free* over *occupied* areas in densely sampled and thus already known regions. We heuristically defined the value of $\rho = 1.5$. Applying the naive remapping to a Hilbert Map regressor results in a map as depicted in Figure 3 c). The mapping is continuous, but not differentiable at the set point s, see Fig. 2. For the presented work, this is not relevant as we use numeric derivations up to the second order, which are robust and produces non-diverging output. If one wants to analytically derive the probability function, a mapping variant based on polynomial functions can be used instead. Looking at the results in Figure 3 c), this naive approach emphasizes not only areas with low data coverage but also the transition zones between free and occupied areas. This may be useful in some scenarios where planning close to known walls is of interest but not in the case of explorative intent. Therefore, additional information about the environment is required to filter out these transitional zones. A good indication is given by the gradient information of these remapped zones. In the original version of the map, the gradient at the points of interest is small in contrast to the undesired transitional zones. An extended version of the remapping function is thus given below:

$$P(x)^*_{grad} = P(x)^* \frac{(1 - \arctan(\|\nabla P(x)\|_2)}{0.5\pi} \quad (24)$$

This results in Figure 3 d). As can be seen, including gradient information drastically enhances the output. The rim around the object is correctly classified as occupied. Furthermore, all regions of low data density are highlighted as free and thus facilitate exploration correctly. Only the region inside

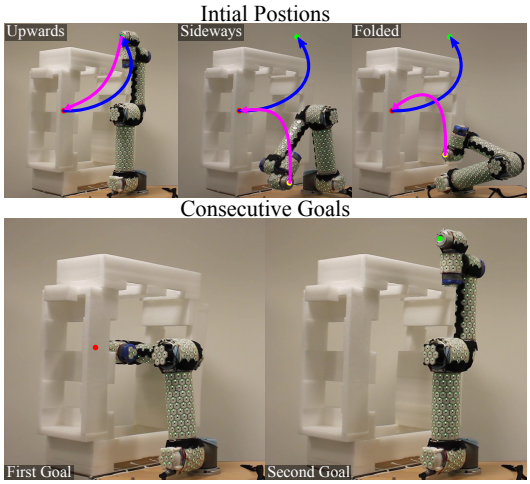

Fig. 4: Planner comparison experimental setup. a) The consecutive first and b) second goal, c) upright, d) sideways and e) folded starting conditions.

of objects is problematic since the gradient between the known border and unknown inside of objects has a similar gradient to the outside of the object. However, these artifacts are completely enclosed by high occupancy regions and can therefore be ignored since the trajectory update only occurs below a user-defined occupancy threshold.

V. Experiments

All experiments are included in the accompanying video, with the exception of the scalability test V-A which was done in simulation. We reduced the execution speed of all planned trajectories in this experiment, to accommodate for the slow dynamics, caused by conservative compliant controller parameters. A longer version is available at: `https://github.com/cuhde/chimp`.

A. Configuration space scalability

As a first test, the CHiMP planner was set up in simulation and validated with three different kinematic configurations. A 3DOF, 5DOF, and 7DOF robot, with identical simulation environments. Average planning times indicate that the algorithm scales well, as can be seen in Figure 6.
The data show, that the algorithm [8] employed in CHiMP is able to not only scale from 2D to 3D space but can also be applied to high DOF systems.

B. Compliant exploration planner comparison

Our approach was validated in comparison to four other planners. Two sampling-based algorithms: i) rapidly-exploring random trees (RRT) [2]; and ii) probabilistic roadmaps (PRM) [16], and two more recent optimization based algorithms: i) Covariant Hamiltonian optimization for motion planning (CHOMP) [15] [4]; and ii) stochastic trajectory optimization for motion planning (STOMP) [5]. The former two planners utilize the sparse contact samples retrieved from the skin directly through the Octomap, while the latter two generate a distance field in addition.

For the comparison test, the UR5 robot covered with multi-modal skin is required to plan inside a loosely stacked packing-foam loop and back out, without toppling the construction (see Figure 4). The environment is not known in advance and has to be detected solely by the proximity sensors of the robot skin. The goal of this experiment is to evaluate which planner is able to effectively navigate unknown environments using the data that the robot skin provides.
All planners are given 10 consecutive planning tries per goal, each starting from the resulting position of the preceding one. The idea is to produce more natural behaviors, where in case of obstacle detection during exploration, the system continues from where it encountered something unexpected. This is more natural and human-like. We chose to evaluate all approaches with these same constraints. Three starting positions are tested. The results can be seen in Table I. Even though it can get stuck in local minima (this impact can be reduced by random restarts), the CHiMP planner is able to navigate most runs with a minimal number of contacts in practice. The *Hilbert Map* is able to infer object shapes from only a few data samples. Due to the contact-based nature of the approach, the provided data is heavily skewed towards free samples. Utilizing this information is crucial for efficient planning with robot skin as we want to keep the number of contacts – and thus occupied samples – low. In contrast to pure collision detection or distance field calculation, which is used with CHOMP and STOMP, this approach utilizes this non-occupied data to shape the Gaussian approximation based probability representation. This can be seen in samples being off-center to the red blobs in Figure 5.

TABLE I: Comparative test results with three different starting poses. Contact numbers in brackets indicate that the corresponding run failed due to not finding a valid path after the first contact. In our tests, CHOMP was generally not able to find a valid path with very few Octomap samples as provided by the skin, but succeeded with dense obstacle data.

Upwards	Average plan. time [s]	Execution time [s]	Contacts	Successful
CHiMP	7.98	68	1	yes
RRT	3.19	62	(1)	no
CHOMP	1.02	29	(1)	no
STOMP	0.96	76	2	yes

Sideways	Average plan. time [s]	Execution time [s]	Contacts	Successful
CHiMP	7.54	95	3	yes
RRT	0.08	73	6	no
PRM	5.03	320	9	no
STOMP	2.92	126	4	yes

Folded	Average plan. time [s]	Execution time [s]	Contacts	Successful
CHiMP	6.80	84	1	yes
PRM	5.01	95	2	yes
CHOMP	0.81	39	(1)	no
STOMP	4.15	160	5	no

C. Exploration with Hilbert map remapping

Inciting explorative behavior has been tested with CHiMP and the same setup as in Experiment V-B. The remapping function causes the system to prefer undiscovered regions, as expected. Due to requiring gradient information for all Hilbert map calls, the planning time is nearly tripled. Narrow

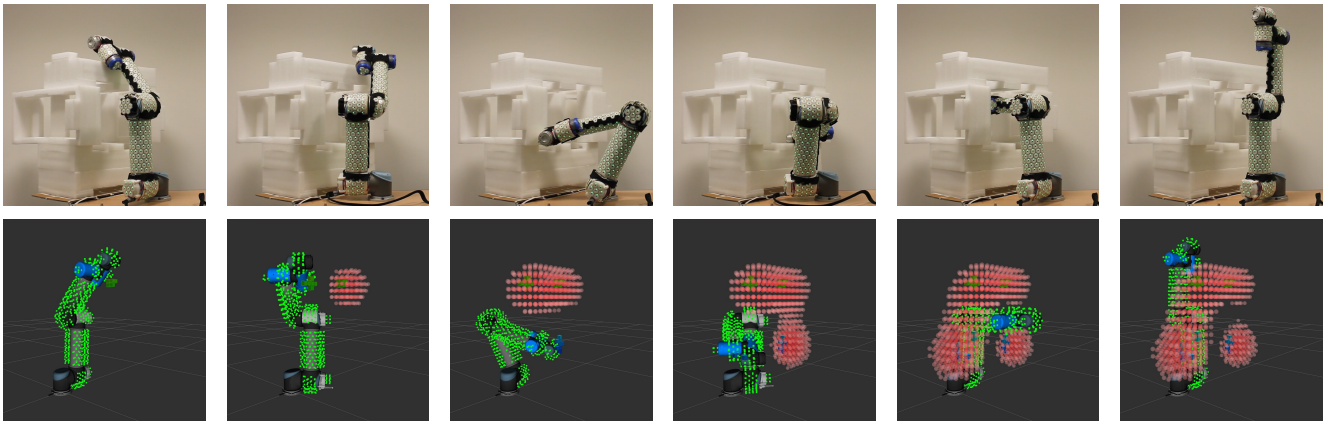

Fig. 5: Narrow goal with CHiMP planner setup. The planner is able to avoid the obstacle with only four contact points. The first four columns show the contacts, the last two depict the two successfully reached goals. The reconstruction in the Hilbert map can be seen in the last two shots. Notice the difference between the Octomap and the Hilbert map, where the former shows a discrete non-gradient representation, while the latter provides gradient information useful for the planner. For visibility purposes, we only show the occupancy probabilities $P(x) > 0.7$.

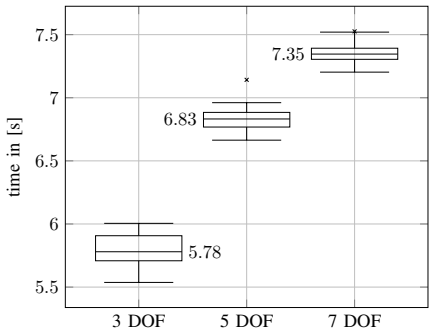

Fig. 6: Comparison of CHiMP planning time in seconds for three diferent joint configurations over 20 runs each. The algorithms scales well with the number of joints.

goals cannot be reached, due to the degradation of gradient information caused by halving the probability range. The remapping set point (here $s = 0.5$) can be moved between planning attempts to first prefer explorative behavior, before exploiting the collected data with regular planning. The experiment can be seen in the accompanying video.

The presented numeric implementation for Hilbert map derivation is comparatively slow. This is particularly apparent in the remapping case as it requires more sample points for the gradient-based filtering. The number of calls can be drastically reduced by implementing analytic partial derivations with respect to the workspace dimensions. This requires the somewhat lengthy derivation of the regression model in conjunction with the feature space transformation performed by the kernel approximation.

D. Planning with new obstacles

The online nature of Hilbert maps allows planning with newly appearing obstacles in mind. This experiment tested if the system using the CHiMP planner is able to react to new obstacles in already discovered regions. The CHiMP planner was able to detect and avoid the obstacle with a big margin. The experiment can be seen in the accompanying video.

E. Planning towards narrow goals

For a final setup, the CHiMP planner was tested with a narrow version of the test setup in experiment V-B. The planner successfully approximated the obstacle with only four contact points and reached both goals. The contacts and goal sequence can be seen in Figure 5, and the accompanying video.

VI. Conclusion

This work has shown that planning with multi-modal skin is feasible, given enough robot surface skin coverage. The difference in data acquisition, compared to global camera systems requires novel approaches for planning. In this work we extended the *Stochastic Functional Gradient Path Planning* algorithm to 3D space, verified that it works with high dimensional configuration spaces, integrated the approach into a *intentional contact*-based system, added seamlessly controllable explorative behavior and showed that the CHiMP planner is able to utilize the skin-based sparse contact data effectively by approximating the obstacle shape, where available state-of-the-art planners struggle. As depicted in Table I, even though the proposed planner reached all the goals successfully, it doesn't match the convergence time of state-of-the-art planners. This performance optimization is considered as future work.

Acknowledgment

The research reported in this paper has been (partially) supported by the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 "EASE - Everyday Activity Science and Engineering", University of Bremen (http://www.ease-crc.org/). The research was conducted in subproject R1: NEEM-based embodied knowledge system.

REFERENCES

[1] M. Gori, M. D. Viva, G. Sandini, and D. C. Burr, "Young Children Do Not Integrate Visual and Haptic Form Information," *Current Biology*, vol. 18, pp. 694–698, 2008.

[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 489–494.

[5] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4569–4574.

[6] J. R. Guadarrama-Olvera, E. Dean, and G. Cheng, "Using intentional contact to achieve tasks in tight environments," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1000–1005.

[7] A. Jain, M. D. Killpack, A. Edsinger, and C. C. Kemp, "Reaching in clutter with whole-arm tactile sensing," *The International Journal of Robotics Research*, vol. 32, no. 4, pp. 458–482, 2013.

[8] G. Francis, L. Ott, and F. T. Ramos, "Stochastic Functional Gradient Path Planning in Occupancy Maps," *CoRR*, vol. abs/1705.05987, 2017.

[9] P. Mittendorfer and G. Cheng, "Humanoid Multimodal Tactile-Sensing Modules," *IEEE Transactions on Robotics*, vol. 27, pp. 401–410, 2011.

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, pp. 189–206, 2013.

[11] F. T. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *I. J. Robotics Res.*, vol. 35, pp. 1717–1730, 2015.

[12] C. K. I. Williams and M. W. Seeger, "Using the Nyström Method to Speed Up Kernel Machines," in *NIPS*, 2000.

[13] G. Francis, L. Ott, and F. T. Ramos, "Stochastic functional gradient for motion planning in continuous occupancy maps," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3778–3785, 2017.

[14] S. Belongie, C. Fowlkes, F. Chung, and J. Malik, "Spectral partitioning with indefinite kernels using the nyström extension," in *European conference on computer vision*. Springer, 2002, pp. 531–542.

[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*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[16] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.