Manipulation Trajectory Optimization with Online Grasp Synthesis and Selection

Lirui Wang[1], Yu Xiang[2], Dieter Fox[1,2]
[1]University of Washington, [2]NVIDIA
liruiw@cs.washington.edu, yux@nvidia.com, fox@cs.washington.edu

Abstract—In robot manipulation, planning the motion of a robot manipulator to grasp an object is a fundamental problem. A manipulation planner needs to generate a trajectory of the manipulator to avoid obstacles in the environment and plan an end-effector pose for grasping. While trajectory planning and grasp planning are often tackled separately, how to efficiently integrate the two planning problems remains a challenge. In this work, we present a novel method for joint motion and grasp planning. Our method integrates manipulation trajectory optimization with online grasp synthesis and selection, where we apply online learning techniques to select goal configurations for grasping, and introduce a new grasp synthesis algorithm to generate grasps online. We evaluate our planning approach and demonstrate that our method generates robust and efficient motion plans for grasping objects in cluttered scenes.

I. INTRODUCTION

In order to manipulate objects in a cluttered scene, a robot needs to plan its motion to reach and grasp a target object in the scene. Due to the complexity of this manipulation planning problem, it is usually decomposed into two sub-problems and tackled separately: arm motion planning and grasp planning. While arm motion planning aims at reaching the target goal and avoiding obstacles in the scene, grasp planning generates feasible grasps of the target object based on its properties such as its 3D shape and material. However, separating motion planning and grasp planning requires some post-processing steps to connect the two plans. For instance, a common strategy is to loop over each planned grasp and check if there exists a trajectory to reach the grasp. The post-processing usually involves heuristics and leads to sub-optimal solutions. How to efficiently integrate motion planning and grasp planning remains a challenge.

Solutions to this problem are previously introduced both in the traditional motion planning literature [10, 41] and the recent Task And Motion Planning (TAMP) literature [26, 40]. According to when grasps are generated during planning, these methods can be classified into two categories. Methods in the first category generate a fixed set of grasps offline, and formulate the problem as a grasp selection problem [2, 10, 11]. Generating grasps offline can take advantages of well-designed grasp planning algorithms such as [29]. However, a fixed set of grasps limits the number of grasp options during motion planning. Methods in the second category formulate and solve a joint optimization problem of trajectories and grasps, which generate grasps online during trajectory planning [41, 15, 19]. Online grasp planning explores more grasp options. However,
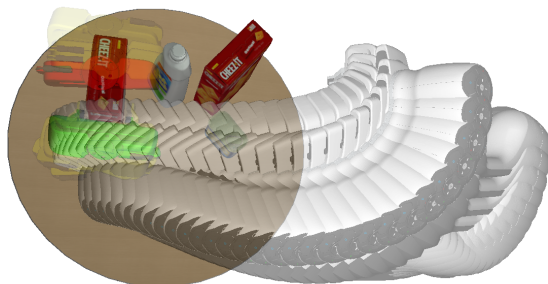

Fig. 1: A planning scene example. The goal set, initial grasp and selected grasp are yellow, red and green, respectively. Our algorithm synthesizes and selects grasps online and generates motion trajectories to reach the selected grasp.

since the space of possible grasps is large and highly non-convex for various objects, approximations are often made to simplify the grasp planning problem. For instance, [41, 40] use spheres to approximate the object shape in optimizing grasps, which limits the quality of the generated grasps.

In this work, to overcome the limitations of existing integrated motion and grasp planning methods [41, 15, 19], we introduce a novel integrated Optimization-based Motion and Grasp Planner (OMG Planner). Our planner combines trajectory optimization with online grasp synthesis and selection. In this context, the planning phase is offline, and we do not consider execution or re-planning. "Online" mainly attributes to the fact that the final grasp is not fixed and can be refined during trajectory optimization. Grasp selection enables us to leverage dedicated grasp planning methods [29, 12] to generate an initial grasp set, which guarantees the quality of the grasps and handles objects with arbitrary shapes. By performing online grasp synthesis, our method augments the initial grasp set and provides more grasp candidates during motion planning. More importantly, the grasp selection problem is formulated as an online learning problem, where a probability distribution of the grasp set is estimated and updated during trajectory optimization, and the best grasp is selected accordingly.

Specifically, our planner optimizes for motion trajectories and grasps jointly. An iterative update algorithm is introduced that alternatively updates trajectories and grasps. In each iteration, given a grasp goal configuration, we utilize the CHOMP trajectory optimization method [33] with a goal set constraint [11] to update the trajectory. To update the grasp,

we first assume the availability of a set of grasp candidates that can be obtained from any grasp planning method. Our online grasp selection algorithm then leverages the interplay between the grasp set and the current trajectory to update the probability distribution of the grasp set, from which the optimal goal in the grasp set can be selected. In order to augment the grasp set, we propose a grasp synthesis procedure named Configuration Space Iterative Surface Fitting (C-Space ISF) that refines the grasp configuration of the selected grasp by matching the 3D shapes of the robot gripper and the object. The overall framework leverages the grasp set structure to improve motion and grasp generation. Fig. 1 illustrates a planning scene example from our method.

We conduct qualitative and quantitative evaluation to demonstrate the performance of our method for manipulation planning. We also conduct ablation studies on each component of the method. Overall, our contributions are:

1) We introduce an integrated planner that models joint motion and grasp planning as an optimization problem. The computation of feasible and reachable grasps and the search for collision-free trajectories are connected.
2) We propose an online learning algorithm and a grasp synthesis approach to select grasps and generate grasps online, which eliminates the need for a perfect grasp set and grasp selection heuristics.
3) We showcase our algorithm for manipulation planning tasks in cluttered scenes. By comparing the proposed approach with existing methods, we demonstrate that our integrated planner improves performance.

II. RELATED WORK

A. Motion Planning

1) Sampling-based Methods: Sampling-based motion planners such as Rapidly exploring Random Trees (RRTs) [28, 25, 23] and Fast Marching Tree (FMT) [21] find trajectories by incrementally building space filling trees through directed sampling. RRTs offer probabilistic completeness. If there exists a solution to the problem, given sufficient time, a feasible trajectory can be found. However, sampling-based algorithms can be difficult to use in some applications due to the computational challenges and requirements for post-processing steps such as smoothing.

2) Trajectory Optimization: Trajectory optimization starts with a possibly infeasible trajectory and then optimizes the trajectory by minimizing a cost function. CHOMP [33] and related methods [11, 10] optimize a cost functional using covariant gradient descent, while STOMP [22] uses stochastic sampling of noisy trajectories to optimize non-differentiable costs. More recently, TrajOpt [36] solves a sequential quadratic program and performs convex continuous-time collision checking. GPMP2 [31] formulates the problem as inference on a factor graph and finds the maximum a posteriori trajectory by solving a nonlinear least squares problem. Trajectory optimization methods are fast, but can only find locally optimal solutions. The local nature of previous methods can often be improved by multiple initialization and goals [36, 10].

Our method is based on CHOMP [33] for trajectory optimization. Instead of optimizing trajectories on fixed endpoints, we also consider goal selection and synthesis in our planner. While [10] model the goal selection problem as a regression on trajectory attributes through offline training, our method selects the target goal online during planning.

B. Grasp Planning

1) Analytic Methods: Analytic methods [32, 14, 6] often require full object information to define grasp qualities such as maximal disturbance resistance, force closure and antipodal grasps. For instance, [7] strive to reduce the dimensionality of the search space for robust grasps and use eigengrasps in the Graspit! simulator [29] to plan with the simulated annealing algorithm. Recently, [13] propose to find robust grasp poses by iterative surface fitting that optimizes error metrics of normal alignments and distances as a least squares problem.

2) Data-driven Methods: Compared to analytic methods, data-driven approaches [18, 3] have gained popularity since they can work with partial information of objects. For determining object grasps from images, [34] assume objects are put on a table top, and a parallel gripper grasps objects from top-down directions. Due to the multi-modality of grasping tasks, [38] sample 6 DOF grasps and train a neural network to predict a quality score. [30] also focus on the diversity of grasp synthesis, which allows and benefits motion planning with multiple goals. Compared to these grasp planning methods, our method jointly plans arm motion and grasps to reach objects.

C. Integrated Motion and Grasp Planning

The advantage of an integrated planner is that motion planning and grasp planning impose constraints on each other. The closest works to ours are the integrated motion and grasp planners [41, 15, 19]. These methods are built upon sampling-based motion and grasp planners. They bias a motion planner to sample nodes that lead to better grasps while a grasp sampling planner synthesizes grasps online. Different from these methods, we focus on studying the grasp selection problem in a trajectory optimization framework. Therefore, our method can utilize high-quality grasps generated from well-designed grasp planners, such as grasps synthesized from physics simulation [29, 12].

D. Task and Motion Planning

TAMP methods focus on how to combine high-level task planning with low-level motion planning [26, 37, 16, 40], where the tasks usually involve hybrid states and require multiple steps of actions such as picking, placing and pushing. For example, [26] leverage geometric constraints from motion planning to reduce the search space of possible actions. [16] build factored transition systems with sampling-based algorithms for robotic manipulation problems. [40] formulate a constrained path optimization problem for sequential manipulation that involves physical interactions by logic geometric programming [39]. Our approach is related to TAMP for
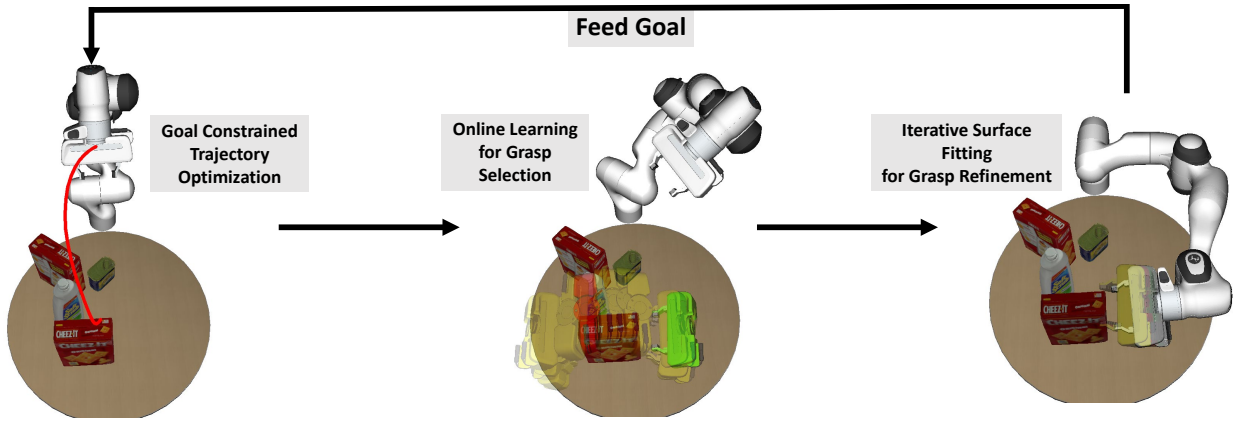
Fig. 2: Our Optimization-based Motion and Grasp Planner (OMG Planner) consists of three components. The motion planner component optimizes the current trajectory denoted by the red line for a fixed target goal. The online learning component considers the goal set (yellow), and update the initial grasp (red) with a new grasp (green). The grasp refinement component iteratively improves the grasp. The refined grasp is used as the target goal for the next iteration.

manipulation planning. However, we neither perform any high-level reasoning nor model sequences of actions. We focus on obstacle avoidance with kinematics constraints for grasping. While TAMP methods usually simplify or approximate the geometry of the manipulated objects, we do not make such approximations. Our method can benefit TAMP by providing a joint motion and grasp planner for task-level reasoning.

E. Online Learning

Compared with statistical learning in which data is assumed to come from a distribution and the goal is to minimize an excess risk, online learning instead considers a setting where a learner makes a decision that triggers a loss function online at every time step and the learner needs to dynamically improves its decision making by minimizing the accumulated losses [4, 20, 27]. In the regret-based framework, the learner tries to minimize the regret on the loss sequence. Online interactive learning has been successfully applied to tackle different problems in robotics such as imitation learning [35], manipulation [24], navigation [17], and model predictive control [42]. Here, we formulate the grasp selection problem as an adversarial linear bandit [27] in the online learning framework, where our learner (the planner) needs to choose the grasp goal online and adapt to some measured costs during every iteration of the trajectory optimization.

III. METHODOLOGY

The key idea of our method is to simultaneously optimize the arm motion trajectory and the end-effector configuration for grasping. We formulate the problem as a constrained trajectory optimization problem, where we require the trajectory to avoid obstacles and to be smooth. Meanwhile, the last configuration of the trajectory should afford grasping. To achieve this, our method synthesizes and selects grasps online during trajectory optimization. Fig. 2 presents an overview of our optimization-based motion and grasp planner.

Formally, we define a trajectory $\xi : [0, 1] \mapsto \mathcal{Q}$ to be a function that maps time $[0, 1]$ to the robot configuration space

$\mathcal{Q}$, where $\mathcal{Q} \subset \mathbb{R}^d$ and d is the degree of freedom of the robot manipulator. Given a scene of multiple objects, we define the feasible grasp set of a target object as $\mathcal{G} \subset \mathcal{Q}$. Then we solve the following optimization problem to find the optimal trajectory of the robot manipulator for grasping:

$$\xi^* = \underset{\xi}{\arg\min} \, f_{\text{motion}}(\xi)$$
$$\text{s.t. } \xi(1) \in \mathcal{G}, \tag{1}$$

where f_{motion} is the objective function of the trajectory, and $\xi(1)$ indicates the last configuration of the trajectory. The constraint in Eq. (1) requires the last configuration at time 1 to be a feasible grasping configuration.

A. Trajectory Objective Functional

Similar to CHOMP [33], we model the objective functional

$$f_{\text{motion}}(\xi) = f_{\text{obstacle}}(\xi) + \lambda f_{\text{prior}}(\xi), \tag{2}$$

where the obstacle term f_{obstacle} bends the trajectory away from obstacles by penalizing parts of the robot that are close to or already in collision with objects in the scene, the prior term f_{prior} measures the dynamical quantities across the trajectory such as velocities and accelerations, and λ is a weight to balance the two terms. In this work, the prior term is defined as the integral over squared velocity norms:

$$f_{\text{prior}}(\xi) = \frac{1}{2} \int_0^1 \|\xi'(t)\|^2 \, dt, \tag{3}$$

where $\xi'(t)$ indicates the velocity of the trajectory at time t.

To define the obstacle term, let $\mathcal{B} \subset \mathbb{R}^3$ be a set of body points on the robot and $x(q, u) : \mathcal{Q} \times \mathcal{B} \mapsto \mathbb{R}^3$ the forward kinematics mapping a body point $u \in \mathcal{B}$ with configuration $q \in \mathcal{Q}$ to the workspace. Furthermore, let $c_{\text{obstacle}} : \mathbb{R}^3 \mapsto \mathbb{R}$ be a workspace cost function that penalizes points in the workspace inside and around the obstacles using the Signed Distance Field (SDF). Since we want to drive the body points away from collision, the obstacle term in Eq. (2) is an integral

that collects the cost of body points on the robot in the workspace along the trajectory:

$$f_{\text{obstacle}}(\xi) = \int_0^1 \int_{\mathcal{B}} c_{\text{obstacle}}\big(x(\xi(t), u)\big) \left\| \frac{d}{dt} x(\xi(t), u) \right\| du\, dt. \tag{4}$$

Please refer to [33] for the definition of c_{obstacle} and the derivation of the gradients of f_{prior} and f_{obstacle}.

B. Iterative Update Rule

In practice, we use a discretization of the trajectory function over time: $\xi \approx (q_1^\top, q_2^\top, ..., q_n^\top)^\top \in \mathbb{R}^{n \times d}$ for resolution n. Under this parametrization, we can write the prior term as:

$$f_{\text{prior}}(\xi) = \frac{1}{2} \| K\xi + e \|^2 = \frac{1}{2} \xi^\top A \xi + \xi^\top a + c, \tag{5}$$

where K is a finite differencing matrix, e is a vector that accounts for the contributions of the configurations that remain constant in the trajectory, i.e., the start configuration and the goal configuration, and $A = K^\top K$ is the dynamic matrix. The covariant idea of CHOMP comes from the inverse of the dynamic matrix. A^{-1} acts as a smoothing operator that propagates the Euclidean gradient of the objective along the trajectory and seeks to make small changes in the average acceleration. Given this parametrization, CHOMP is a variant of gradient descent that minimizes the linear approximation of f_{motion} about ξ within an ellipsoid trust region. The distance metric that shapes this ellipsoid is defined by A, and we obtain the update rule in gradient descent for the ith iteration:

$$\xi_{i+1} = \xi_i - \frac{1}{\eta_{\text{motion}}} A^{-1} v_i, \tag{6}$$

where $v_i = \nabla f_{\text{motion}}(\xi_i)$ is the discretized functional gradient and η_{motion} is the step size.

C. Goal Set Constraint

The end-point of ξ_i can be varied during planning for a set of feasible goal configurations $\mathcal{G}$. Following [11], this is achieved by introducing a constraint on the trajectory. Given a general constraint $h(\xi) = 0$, we can linearize h around ξ_i: $h(\xi) \approx h(\xi_i) + \frac{\partial}{\partial \xi} h(\xi_i)(\xi - \xi_i) = C(\xi - \xi_i) + b$, where C is the Jacobian of h evaluated at ξ_i and $b = h(\xi_i)$. We use Lagrangian to solve the linearized constraint and yield the projected gradient update rule:

$$\begin{aligned}
\xi_{i+1} = \xi_i &- \frac{1}{\eta_{\text{motion}}} A^{-1} v_i \\
&+ \frac{1}{\eta_{\text{motion}}} A^{-1} C^\top (C A^{-1} C^\top)^{-1} C A^{-1} v_i \\
&- A^{-1} C^\top (C A^{-1} C^\top)^{-1} b.
\end{aligned} \tag{7}$$

Assume our goal configuration is g at the ith iteration. We simply model the goal set constraint as

$$h(\xi_i) = \xi_i(1) - g = 0, \tag{8}$$

and denote the update rule in Eq. (7) by *CHOMP-Proj*(ξ, g). With this update rule, we denote $f_g(\xi_i)$ as the objective cost if we use g as the goal from the ith iteration. As we move the
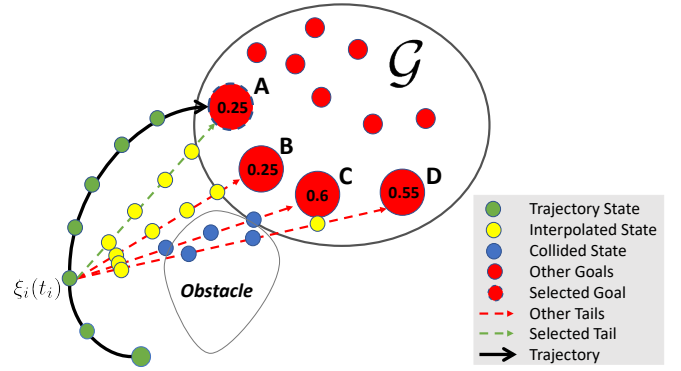

Fig. 3: Online goal selection algorithm at the ith iteration. The goal costs $c(g)$ are denoted inside red circles. In this example, the online learning algorithm is about to shift goal from A to B, which yields a collision-free yet shorter trajectory.

end-point, CHOMP smoothly updates the remaining trajectory. This formulation is different from the vanilla CHOMP in that the goal is not fixed and can change in different iterations.

D. Online Learning for Grasp Selection

While [11] address the problem of CHOMP planning with a goal set constraint, modeling goal selection as projection by Eq. (8) can often be simplistic. [10] propose a few trajectory attributes such as goal radius, elbow room, distance, etc., for a regression formulation. Indeed, the optimality of a goal should be induced by the objective function and thus related to the success of motion planning. Given that we can optimize $f_g(\xi_i)$ with a target goal, we propose to choose the goal that maximizes our motion generation success under an online linear bandit scheme [4, 27]. An online linear optimization framework is suitable because at every iteration, we can model the objective costs of the goal set as our suffered losses and relax our goal selection as probability distributions.

Assuming the goals in $\mathcal{G}$ are intricately correlated, our strategy should dynamically choose the goal for the next iteration until it finds the best goal. In CHOMP projection, we suffer from switching to goals that can heavily perturb our current trajectory into collision. Therefore, to allow for principled behavior that selects the next goal g_{i+1}, we need a strategy that balances exploitation and exploration of the costs.

In the ith iteration, denote the goal distribution on a discrete goal set $\mathcal{G}$ by p_i. We apply online learning to solve for p_{i+1}. To use optimization for goal selection, we need to define the cost of each goal in the goal set, denoted by a cost vector c_i. Ideally, we would like to have $c_i(g) = f_g(\xi_i)$, then the objective cost directly tells us which target goal to use. However, since $f_g(\xi_i)$ is not available without running CHOMP for each goal, we generate surrogates to our objective cost $f(\widehat{\xi}_i^g) \approx f_g(\xi_i)$ by estimating the potentially optimized trajectory $\widehat{\xi}_i^g$ as if the trajectory ξ_i ends at $g \in \mathcal{G}$.

We propose that to compute $f(\widehat{\xi}_i^g)$, it is more natural to look at the tail of trajectory denoted by $\widehat{\xi}_i^g$ starting at $t_i = \frac{i}{N}$, where N is our optimization horizon. To motivate this, recall that in each *CHOMP-Proj* step, we project onto the goal constraint

and propagate the changes smoothly. Therefore, as the time step i approaches N, the changes to the trajectory impact more on the tail than the head of the trajectory. Our estimated costs would be more accurate if we use the tail $\widetilde{\xi}_i^g$ instead of the whole trajectory. Moreover, the goal cost vector needs to converge for stability of the algorithm and it should also be dynamic from the learning perspective. Another way to think about this is that we are "simulating" the optimal control execution of the trajectory during planning and estimating the running cost for dynamic goal selection. As shown in Fig. 3, since the prior of CHOMP is the minimum smoothness trajectory, one can approximate the trajectory tail $\widetilde{\xi}_i^g$ as the interpolation between $\xi_i(t_i)$ and the goal g with constant velocity. The associated cost for the next iteration is defined as $c_{i+1}(g) = f(\widetilde{\xi}_i^g)$ and normalized to have unit norm.

Given our estimated objective cost $c_{i+1} = \{f(\widetilde{\xi}_i^g) \mid g \in \mathcal{G}\}$ and previous costs $c_1, ..., c_i$ for the goal set $\mathcal{G}$, we apply online learning algorithms to update the goal distribution p_{i+1} as described later. Then the expected loss at the $(i+1)$th iteration under this distribution is $\langle c_{i+1}, p_{i+1} \rangle$ and g_{i+1} is defined as the mode of the distribution p_{i+1}, which turns out to be more stable than a sample. Overall, online learning for goal selection aims to minimize the regret

$$\mathcal{R}_N = \sum_{i=1}^{N} \langle c_i, p_i \rangle - \min_{g \in \mathcal{G}} \sum_{i=1}^{N} c_i(g), \qquad (9)$$

which compares the loss of our strategy p_i up to time N with the best cost of any fixed goal in hindsight. In the following, a few commonly used optimization methods for online learning problems [4] are presented. These methods can be used to solve p_{i+1} at the ith iteration.

1) Follow the Cheapest (FTC): FTC greedily moves the distribution p_{i+1} entirely to the point that has the minimum cost $c_{i+1}(g)$. *CHOMP-Proj* turns out to be a FTC algorithm with the costs defined by the distances between the end-point of the current trajectory and the goals.

2) Follow the Leader (FTL): FTL computes the point $g \in \mathcal{G}$ that causes the lowest total cost $\sum_{j=1}^{i+1} c_j(g)$ and then makes the distribution p_{i+1} concentrated entirely on g.

3) Exponential Weighting (EXP): EXP begins with an uniform distribution over goals: $p_1(g) = \frac{1}{|\mathcal{G}|}, \forall g \in \mathcal{G}$. At time i, it first multiplies the previous probabilities with the corresponding exponential costs: $\widehat{p}_{i+1}(g) = \exp(-\eta_{\text{ol}} c_{i+1}(g)) p_i(g), \forall g \in \mathcal{G}$, where η_{ol} is the learning rate. Then it normalizes the vector to make it a probability distribution: $p_{i+1}(g) = \widehat{p}_{i+1}(g)/\sum_{g \in \mathcal{G}} \widehat{p}_{i+1}(g)$.

4) Mirror Descent (MD): We refer the readers to [1, 27] for this more general form of regularized optimization in a bandit setting. A natural way to regularize probability distributions is to associate the Bregman divergence with entropy, which becomes the KL divergence. At time step i, the MD update rule is $p_{i+1} = \operatorname*{argmin}_{p \in \Delta} \eta_{\text{ol}} \langle c_{i+1}, p \rangle + D_{KL}(p||p_i)$, where Δ is the probability simplex and η_{ol} is the learning rate. The argmin of this update rule can be solved by Lagrangian.
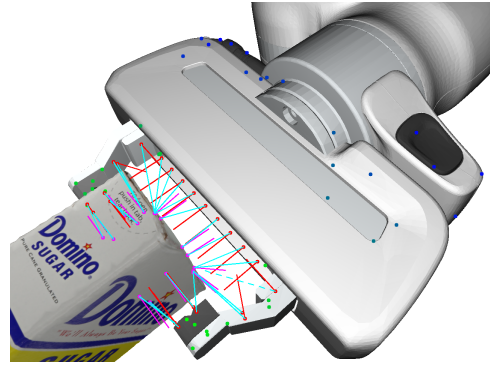

Fig. 4: Red points denote the sampled contact points u_h with normals u_h^n and purple points denote the corresponding object points u_o with normals u_o^n. The correspondence lines are colored in cyan. Body points u range from blue to green, denoting from far to close with respect to objects.

E. Online Grasp Synthesis

So far we have assumed that the goal set is given and fixed. A fixed set of grasps limits the number of grasp possibilities. Even with a set of perfect grasps in the workspace, it is unclear that solving inverse kinematics would give a sufficient goal set in the configuration space. A fixed set also has low adaptability to the environment since it fails to explore new grasps.

Starting from an initial finite goal set $\mathcal{G}$, we propose to synthesize more candidate grasp configurations online. Suppose after the online learning process for goal selection, the configuration end goal g_i is selected at iteration i. For simplicity, we drop the iteration index i in the followings. Our proposed grasp synthesis algorithm aims to improve the grasp affordance by minimizing the objective function:

$$f_{\text{grasp}}(g) = f_{\text{isf}}(g) + \gamma f_{\text{collision}}(g), \qquad (10)$$

where $f_{\text{isf}}(g)$ measures the grasp quality using the Iterative Surface Fitting (ISF) algorithm, $f_{\text{collision}}(g)$ measures collisions in the scene, and γ is a constant to balance the two terms. We use $f_{\text{isf}}(g)$ to refine the end configuration g such that the end effector transformation $T \in \mathbb{SE}(3)$ makes the surface of the robot gripper $\mathcal{S}$ match against the surface of the target object $\mathcal{O}$. We softly penalize hand and arm collisions and use gradient descent to optimize the objective. This enables efficient and stable optimization in the configuration space.

Specifically, we generalize the forward kinematics mapping $x(g, u)$ of a body position u in Sec. III-A to include directions: $x(g, u^n)$ maps a body direction vector u^n and a robot configuration g to a direction vector in the workspace. Observe that the end effector transformation T in the workspace is the hand coordinate transformed by g. We can write all workspace costs on T as the configuration space costs on g. We define a set of hand contact points $\{h_j\}$ and a set of contact normals $\{h_j^n\}$ on the robot hand and fingers, indexed by j (see Fig. 4). Denote $u_h = \{x(g, h_j)\}$ and $u_h^n = \{x(g, h_j^n)\}$ as the two sets in the workspace transformed by forward kinematics. We then search for the nearest neighbor point u_o and normal u_o^n on object $\mathcal{O}$ associated with u_h, u_h^n. For m pairs of points and

normals, we define the point matching loss as

$$f_{\text{pml}}(g) = \sum_{j=1}^{m} \langle u_{h,j} - u_{o,j}, u_{o,j}^n \rangle^2, \tag{11}$$

and the normal alignment loss as

$$f_{\text{align}}(g) = \sum_{j=1}^{m} (\langle u_{h,j}^n, u_{o,j}^n \rangle + 1)^2. \tag{12}$$

The point matching loss closely attaches the hand to the object, and the normal alignment loss aligns their surface normals. Overall, the ISF loss is defined as

$$f_{\text{isf}}(g) = f_{\text{pml}}(g) + \alpha f_{\text{align}}(g), \tag{13}$$

where α is a weight coefficient.

Since the hand has to remain collision-free with respect to the object, we evaluate the object points in the SDF of the robot hand and penalize any hand collision $f_{\text{hand\_obstacle}}(g)$. This is different from the obstacle cost in CHOMP, where robot body points are evaluated in the SDFs of objects. Note that the gradient for object points in the SDF of the hand can be treated as hand points pushing in opposite gradient directions. We also add the CHOMP obstacle cost evaluated at a single configuration g, $f_{\text{obstacle}}(g)$, to allow local arm obstacle avoidance. The overall collision cost for grasp synthesis is

$$f_{\text{collision}}(g) = f_{\text{hand\_obstacle}}(g) + \beta f_{\text{obstacle}}(g), \tag{14}$$

where β is a balancing weight.

Algorithm 1 presents our C-Space ISF algorithm for grasp refinement. The workspace cost f_{grasp} in Eq. (10) and its gradient ∇f_{grasp} are computed by the $Error$ function. The gradient ∇f_{grasp} is updated via the exponential map exp from $se(3)$ to $\mathbb{SE}(3)$ and pulled back to the configuration space ∇g through the Jacobian transpose $J^\top$ with a step size η_{grasp}. We then update the grasp g in $\mathcal{G}$ and return back to motion planning step. In practice, one can run the C-Space ISF algorithm 1 multiple times for each motion planning iteration.

Algorithm 1: C-Space ISF $(g_i, u_h, u_h^n, \mathcal{O}, \mathcal{S})$

---
1 $u_o, u_o^n = NearestNeighbor(u_h, u_h^n, \partial \mathcal{O})$
2 $f_{\text{grasp}}, \nabla f_{\text{grasp}} = Error(u_h, u_h^n, u_o, u_o^n, \mathcal{O}, \mathcal{S})$
3 $\nabla g_i = J^\top exp(\nabla f_{\text{grasp}})$
4 $g_i = g_i - \eta_{\text{grasp}} \nabla g_i$

F. Joint Motion and Grasp Planning

We have defined our iterative update rule for the ith iteration, involving optimizations of smoothness, collision, goal selection, and grasp quality. Our integrated planner simply runs this iterative update rule for N times. A pre-termination criterion can be set with the threshold on grasp quality, trajectory smoothness, and collision-free requirement. Our joint planning algorithm is shown in Algorithm 2. *CHOMP-Proj* provides a way to smoothly update the trajectory to avoid obstacles given a goal. The *ONLINE-LEARNING* as described
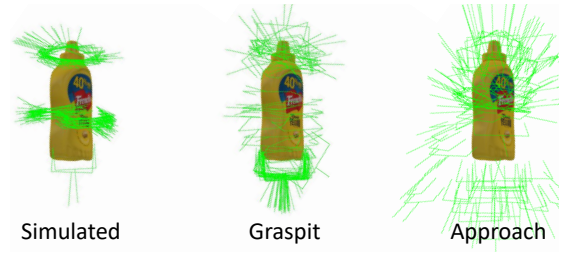

Fig. 5: Three initial grasp sets of the mustard bottle.

in Sec. III-D considers all grasps in the goal set and updates our goal distribution for the next iteration. The *C-Space ISF* applies to the selected goal to refine its quality.

It is worth to mention that we tried to formulate a joint optimization problem by combining the two objective functions in Eq. (1) and Eq. (10). However, we found that the ISF algorithm heavily depends on the initialization and is often stuck in local optima with bad initializations. Our current formulation provides good initialization to the ISF algorithm using the goal set, which makes the optimization more stable.

Algorithm 2: OMG Planner

---
1 initial trajectory $\xi_0, \mathcal{O}, \mathcal{G}, \mathcal{S}, p_0, g_0$
2 **for** $i = 0, ..., N$ **do**
3   $\quad \xi_{i+1} = CHOMP\text{-}Proj(\xi_i, g_i)$
4   $\quad p_{i+1} = ONLINE\text{-}LEARNING(\xi_{i+1}, \mathcal{G})$
5   $\quad g_{i+1} = \text{argmax}(p_{i+1})$
6   $\quad g_{i+1} = C\text{-}Space\ ISF(g_{i+1}, \mathcal{O}, \mathcal{S})$
7 **end**

IV. EXPERIMENTS

We conduct experiments with the Panda Franka arm, a 7-DOF arm with a parallel gripper (Fig. 2). We generate cluttered scenes by dropping sampled YCB objects [5] with random upright poses on a table in the Pybullet Simulator [8]. Each scene samples 3 to 7 objects. The objects used are master chef can, cracker box, sugar box, tomato soup can, mustard bottle, potted meat can, pitcher base, bleach cleanser, bowl, and mug.

We experiment with three different sets of initial grasps as shown in Fig. 5. The "Simulated" grasps are sampled from a physics-based simulator [12]. The second set is sampled from the Graspit! planner [7]. The "Approach" grasps are collision-free grasps naively sampled by random approach directions. The initial start configuration is fixed, and goal configurations are generated from inverse kinematics and checked collision-free. A trajectory is then initialized as a cubic spline connecting the start configuration and the initial goal.

A. Evaluation Metrics

Motion planning methods are usually evaluated by the feasibility of the generated paths, i.e., collision-free paths. In addition, we measure several attributes of the paths that are desirable in our grasping tasks.

100 scenes, 30 grasps each scene, 10 runs						
Algorithm	Fixed	Proj	FTC	FTL	EXP	MD
Execution	74%	75%	**81**%	79%	80%	**81**%
Planning	78%	79%	91%	89%	90%	**92**%
Smoothness	18.87	**11.24**	15.78	15.86	15.71	15.54
Collision	5.97	12.06	4.68	4.10	4.95	**3.87**
Time	0.18	**0.15**	0.20	0.22	0.23	0.30

TABLE I: Comparison between two baselines and four online learning algorithms for grasp selection.

100 scenes, 60 grasps, 30 iteration refinements, 10 runs						
Object Name	S	S Refined	G	G Refined	A	A Refined
Cracker Box	2.47	**2.32**	4.27	3.31	6.58	4.36
Mug	5.11	**3.93**	7.24	5.55	9.05	5.18
Tomato Soup Can	2.38	**2.27**	3.15	2.83	7.06	6.27
Mustard Bottle	4.40	**3.19**	4.91	3.79	8.15	5.73
Potted Meat Can	2.96	**1.95**	3.08	2.54	7.47	4.91
Pitcher Base	4.41	**2.97**	5.68	3.76	8.69	6.14
Mean	3.62	**2.77**	4.72	3.63	7.83	5.43
Planning	92%	**93%**	84%	84%	71%	73%
Execution	80%	**82%**	56%	59%	26%	33%

TABLE II: Grasp costs for six YCB objects before and after refinement. "S", "G" and "A" denotes the simulated set, the graspit set, and the approach set, respectively.
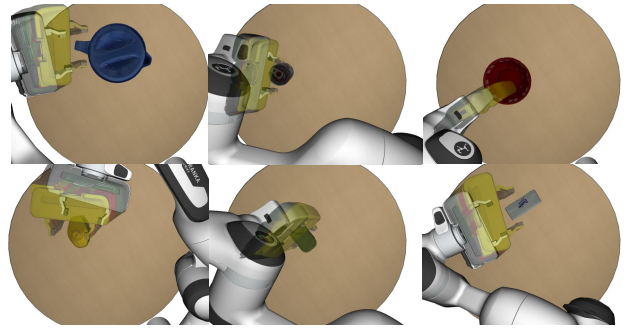

Fig. 6: Examples of grasps before and after refinement.

Trajectories close to obstacles should be avoided in practice. We measure "collision" as the sum of the obstacle costs c_{obstacle} of the sampled points inside the minimum padding distance (5cm) by densely interpolating each trajectory to contain 200 time steps. The smoothness of a trajectory is another desired property. We measure "smoothness" by the squared velocity norm as in f_{prior}. To measure the planning performance, we define the "planning" success as a collision-free path with its smoothness below a threshold of 30.

After planning, it is important to see how the plan is executed for grasping. The success of grasp execution involves system dynamics, feedback controllers and grasp robustness, in addition to trajectory collision and smoothness. We execute a plan in the Pybullet simulator and define the "execution" success if the target object is lifted. The execution success rate takes failed plans into account as well. We also compute the "grasp" cost $f_{\text{grasp}}(\xi(1))$ from our C-Space ISF method to measure the grasp quality. Finally, planning "time" is measured in seconds for different algorithms.

B. Implementation Details

1) CHOMP: We set the smooth weight $\lambda = 0.1$, discretization resolution $n = 30$, and learning rate $\eta_{\text{motion}} = 0.01$. The obstacle padding is 0.2m within which the cost increases quadratically [33]. A collision-free path is assumed to have a minimum distance of 5cm from obstacles. Instead of modeling the arm with swept spheres as in the original CHOMP, we uniformly sample body points from the vertices of the robot 3D mesh, and select 500 points in the trajectory with the largest obstacle costs to update the obstacle gradient (see Fig. 4). These modifications make the integral over $\mathcal{B}$ more general and precise as well as stabilize the optimization. We sample 20 points from each body part in our implementation. Since grasping tasks require collision-free gripper poses, we also include sampled points from hand and fingers.

2) Online Learning: The EXP algorithm uses the learning rate $\eta_{\text{ol}} = \sqrt{\log(|\mathcal{G}|)/N}$. The MD algorithm takes the learning rate over an ensemble of experts with $\eta_{\text{ol}} = 2^k \log(N)$ and k is selected from $\{-2, -1, 0, 2, 4\}$.

3) C-Space ISF: We set the normal penalty weight $\alpha = 0.01$, the arm collision coefficient $\beta = 0.001$, the hand collision constant $\gamma = 0.5$, and the learning rate $\eta_{\text{grasp}} = 0.05$. The nearest neighbors are found with a KD-Tree on 1000 sampled points for each object, where repeated neighbors are filtered out to increase robustness.

C. Grasp Selection

We first evaluate different algorithms for grasp selection in our trajectory optimization framework. In this experiment, the "Simulated" grasp set is used and no grasp refinement is performed. Therefore, the dominating factor in planning performance is the goal selection strategy. Table I presents the results for 100 scenes, where we compare the four online learning algorithms with two baselines. 1) "Fixed" selects the best goal based on the initial objective estimate and fixes the goal during planning. 2) "Proj" [11] uses the *CHOMP-Proj* update rule which always projects to the closest goal during trajectory optimization.

Our results in Table I indicate that the online learning methods improve the success of planning and execution. From the improvement over "Fixed", it is clear that planning toward a fixed goal does not provide the same performance as online algorithms that adaptively select goals, since the initial goal may be sub-optimal. Comparing the four online learning algorithms with "Proj", we can see that the cost estimate based on our objective function performs better than just using the distance between the trajectory end-point and grasps in the goal set. In addition, modeling the probability distribution of the goal set for grasp selection is beneficial. Our empirical analysis shows that these online learning methods provide a scheduling behavior for a planner to adaptively switch goals while keeping optimization stability. While the grasp set size presents a trade-off between runtime and performance, we find 30 grasps is a good balancing point. Finally, MD performs slightly better than the others with a trade-off in running time.
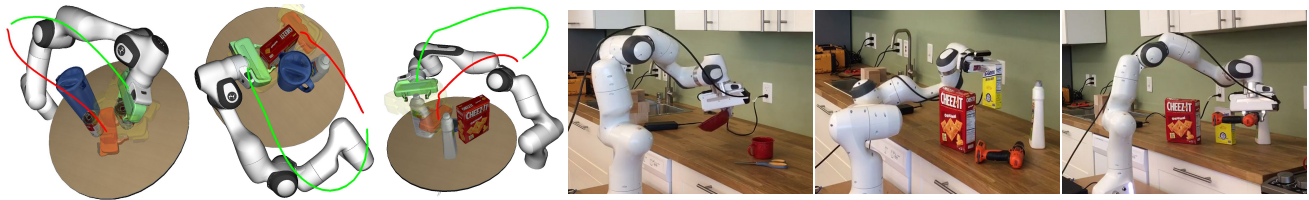
Fig. 7: Left) Planning scene examples with CHOMP trajectories in red and our OMG-Planner trajectories in green. Right) Successful real world grasping examples by executing trajectories from our planner using estimated 6D object poses from [9].

100 scenes, 30 grasps each scene, 10 runs					
Algorithm	CHOMP [33]	CHOMP-C [11]	FMT [21]	RRT-C [25]	OMG
Execution	77%	75%	66%	64%	**84%**
Planning	82%	79%	80%	78%	**93%**
Smoothness	19.12	**11.24**	27.21	26.45	15.43
Collision	4.78	12.06	6.37	6.52	**3.92**
Grasp	3.43	3.35	3.49	3.48	**3.04**
Time	0.25	**0.15**	0.67	0.22	0.42

TABLE III: Comparison on planning methods for joint motion and grasp planning.

Fig. 8: Ablation study of n and λ. The metrics (except planning) are scaled between 0 and 1.

D. Grasp Refinement

We investigate the performance of the C-Space ISF algorithm for grasp refinement. In this experiment, we apply the MD algorithm for grasp selection and test the C-Space ISF algorithm on the three different grasp sets (Fig. 5). Table II shows the detailed grasp costs and the planning and execution success before and after grasp refinement.

Overall, we observe a consistent improvement on different initial grasps from grasp refinement. The "Simulated" set achieves the minimum grasp cost and the highest success rate. Because these grasps are sampled and verified in simulation by grasping and shaking an object to see if the object falls or not [12]. The drawback is that it is time-consuming to generate these high-quality grasps. Our refinement algorithm can further improve the quality of these simulated grasps. We can also see that the C-Space ISF algorithm heavily depends on the initial grasps. Therefore, the improvement on the "Graspit" set and the "Apporach" set is limited. This is also the reason why we combine grasp selection with grasp refinement, instead of optimizing grasps from random initial grasps. Fig. 6 shows some qualitative examples of our grasp refinement method.

E. Planner Performance

We compare our OMG planner with several other motion planners in Table III, where the "Simulated" grasp set is used. The OMG planner uses the MD algorithm for grasp selection and the C-Space ISF algorithm for grasp refinement. Since grasp planning is not covered by CHOMP [33], FMT [21] and RRT-Connect [25], we adopt the common routine used in practice to generate manipulation trajectories for them: 1) ranking the goals in the grasp set based on heuristics such as our objective estimates; 2) planning through the goal set until a feasible trajectory is found. We perform planning experiments for 100 scenes with 30 initial grasps each scene. The numbers in Table III are averaged over 10 runs for each scene.
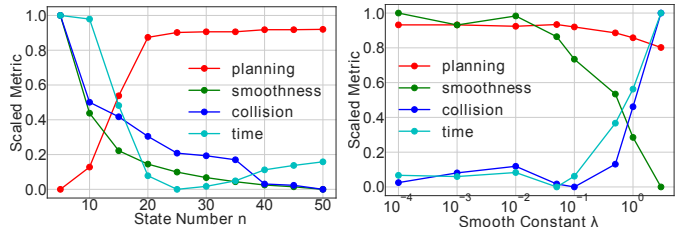
Comparing OMG with other planners, we can see the improvement on planning and execution thanks to our grasp selection and refinement process. By modeling a probability distribution on the grasp set, our method globally selects the optimal grasp for planning. In our metrics, planning success acts as a feasibility and optimality of the trajectory but execution involves control and system noises. By terminating at the first feasible trajectory, other methods may generate sub-optimal plans to execute. In contrast, with the option to select goals in the goal set, our trajectory optimization has fundamentally more solutions and more likely to yield robust results. Some qualitative planning results are shown in Fig. 7.

We also conduct ablation study on the influences of two important design choices in our OMG planner: the discretization n defined in Sec. III-B and the smoothness constant λ defined in Eq. (2). The results are presented in Fig. 8. By increasing n, a fine-grained trajectory better approximates a continuous curve and therefore induces better smoothness and planning success. The smoothness weight λ around 0.1 acts as a good balance in terms of collision and planning success.

V. CONCLUSION

We have presented a joint motion and grasp planning approach based on first-order trajectory optimization to generate grasping trajectories. Our OMG planner does not require a perfect goal configuration, since the candidate goals are determined during the planning process. A goal set is explicitly modeled as a probability distribution and the best goal is selected online. Moreover, our method synthesizes new grasps online to augment the goal set. Our experiments demonstrate that the OMG planner can plan grasping trajectories for cluttered scenes efficiently and robustly. While our work is based on CHOMP, we believe that the online goal selection and refinement procedure can also benefit other planning methods and tasks that involve a set of goals.

REFERENCES

[1] Amir Beck. *First-order methods in optimization*, volume 25. SIAM-Society for Industrial and Applied Mathematics, 2017.

[2] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research (IJRR)*, 30(12):1435–1460, 2011.

[3] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[4] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[5] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron Dollar. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *IEEE Robotics and Automation Magazine*, 22(3):36–52, 2015.

[6] I-Ming Chen and Joel Burdick. Finding antipodal point grasps on irregularly shaped objects. *IEEE transactions on Robotics and Automation*, 9(4):507–512, 1993.

[7] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *Robotics: Science and Systems Manipulation Workshop-Sensing and Adapting to the Real World (RSS)*, 2007.

[8] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[9] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6D object pose estimation for robot manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[10] Anca Dragan, Geoffrey Gordon, and Siddhartha Srinivasa. Learning from experience in manipulation planning: Setting the right goals. In *International Symposium on Robotics Research (ISRR)*, 2011.

[11] Anca Dragan, Nathan Ratliff, and Siddhartha Srinivasa. Manipulation planning with goal sets using constrained trajectory optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4582–4588, 2011.

[12] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. A billion ways to grasp: An evaluation of grasp sampling schemes on a dense, physics-based grasp data set. In *International Symposium on Robotics Research (ISRR)*, 2019.

[13] Yongxiang Fan, Hsien-Chung Lin, Te Tang, and Masayoshi Tomizuka. Grasp planning for customized grippers by iterative surface fitting. In *IEEE International Conference on Automation Science and Engineering (CASE)*, pages 28–34, 2018.

[14] Carlo Ferrari and John Canny. Planning optimal grasps. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2290–2295, 1992.

[15] Joan Fontanals, Bao-Anh Dang-Vu, Oliver Porges, Jan Rosell, and Máximo Roa. Integrated grasp and motion planning using independent contact regions. In *IEEE-RAS International Conference on Humanoid Robots*, pages 887–893, 2014.

[16] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research (IJRR)*, 37(13-14): 1796–1825, 2018.

[17] Julio Godoy, Ioannis Karamouzas, Stephen Guy, and Maria Gini. Adaptive learning for multi-agent navigation. In *The International Conference on Autonomous Agents and Multiagent Systems*, pages 1577–1585, 2015.

[18] Corey Goldfeder and Peter Allen. Data-driven grasping. *Autonomous Robots*, 31(1):1–20, 2011.

[19] Joshua Haustein, Kaiyu Hang, and Danica Kragic. Integrating motion and hierarchical fingertip grasp planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3439–3446, 2017.

[20] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[21] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research (IJRR)*, 34(7):883–921, 2015.

[22] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.

[23] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research (IJRR)*, 30(7):846–894, 2011.

[24] Oliver Kroemer, Renaud Detry, Justus Piater, and Jan Peters. Combining active learning and reactive control for robot grasping. *Robotics and Autonomous systems*, 58(9):1105–1116, 2010.

[25] James Kuffner and Steven LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001, 2000.

[26] Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research (IJRR)*, 33(14):1726–1747, 2014.

[27] Tor Lattimore and Csaba Szepesvári. Bandit algorithms.

preprint, 2018.

[28] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Technical Report. Computer Science Department, Iowa State University*, 1998.

[29] Andrew Miller and Peter Allen. Graspit! A versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[30] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DOF GraspNet: Variational grasp generation for object manipulation. In *International Conference on Computer Vision (ICCV)*, pages 2901–2910, 2019.

[31] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research (IJRR)*, 37 (11):1319–1340, 2018.

[32] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research (IJRR)*, 7(3): 3–16, 1988.

[33] Nathan Ratliff, Matt Zucker, Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[34] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322, 2015.

[35] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[36] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research (IJRR)*, 33(9):1251–1270, 2014.

[37] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, 2014.

[38] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research (IJRR)*, 36 (13-14):1455–1473, 2017.

[39] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[40] Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems (RSS)*, 2018.

[41] Nikolaus Vahrenkamp, Martin Do, Tamim Asfour, and Rüdiger Dillmann. Integrated grasp and motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2883–2888, 2010.

[42] Nolan Wagener, Ching-An Cheng, Jacob Sacks, and Byron Boots. An online learning approach to model predictive control. In *Robotics: Science and Systems (RSS)*, 2019.