Global Search with Bernoulli Alternation Kernel for Task-oriented Grasping Informed by Simulation

Rika Antonova[*] **Mia Kokic**[*] **Johannes A. Stork,** **Danica Kragic**
Robotics, Perception and Learning, CSC
KTH Royal Institute of Technology Sweden
{antonova, mkokic, jastork, dani}@kth.se

Abstract: We develop an approach that benefits from large simulated datasets and takes full advantage of the limited online data that is most relevant. We propose a variant of Bayesian optimization that alternates between using informed and uninformed kernels. With this Bernoulli Alternation Kernel we ensure that discrepancies between simulation and reality do not hinder adapting robot control policies online. The proposed approach is applied to a challenging real-world problem of task-oriented grasping with novel objects. Our further contribution is a neural network architecture and training pipeline that use experience from grasping objects in simulation to learn grasp stability scores. We learn task scores from a labeled dataset with a convolutional network, which is used to construct an informed kernel for our variant of Bayesian optimization. Experiments on an ABB Yumi robot with real sensor data demonstrate success of our approach, despite the challenge of fulfilling task requirements and high uncertainty over physical properties of objects.

Keywords: Bayesian optimization, Deep learning, Task-oriented grasping

1 Introduction

Recent advances in deep learning motivated using data-driven methods in robotics. However, collecting large amounts of training data is challenging, since it requires actual execution on a real system. One way to trim hours of execution time is to use simulation. Simulators make simplifying assumptions, so often there is mismatch between simulation and real world. We propose to bridge this gap by using a variant of Bayesian optimization (BO) that incorporates simulation-based information in a way that is robust to mismatch between simulation and reality. Furthermore, we apply this to the problem of task-oriented grasping. The challenge is that successful grasps are limited to specific object parts that afford the execution of a task. This presents a highly constrained problem.

Our first contribution is a variant of BO that alternates between using informed and uninformed kernels. This allows us to explore grasp adjustments online, while exploiting simulation-based information and feedback from previous grasp attempts. To inform the kernel for our BO algorithm, we propose a task-oriented grasp model. This further contribution is a deep neural network architecture that maps raw visual observation of an object to task-oriented grasp scores. The network is trained on large amounts of synthetic data. This allows us to compute task-oriented grasp candidates for previously unseen objects from online perceptual information. Simulation-based knowledge provides additional guidance for online search, making it more sample-efficient. Overall, our approach aims to emulate the human-like strategy of attempting several adjustments until a successful grasp is accomplished; Figure 1 gives a brief visual overview. We conduct experiments on an ABB Yumi robot with real sensor data, and demonstrate success of our approach to fulfill task requirements when grasping novel objects, despite high uncertainty over their physical properties.

2 Background and Related Work

2.1 Bayesian Optimization Informed By Simulation

Bayesian optimization (BO) is a data-efficient global search method (see [1] for an overview). Let $\boldsymbol{x}$ be a set of control parameters (e.g. for grasping: the approach direction and orientation of the end-effector). The problem is to find $\boldsymbol{x}$ that optimizes a given objective/cost function $f(\boldsymbol{x})$. The objective function encodes characteristics of the desired outcome: e.g. low score for failing to grasp an object, high score for grasping and completing the desired task. BO starts with a prior expressing

[*] Both of these authors contributed equally.

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

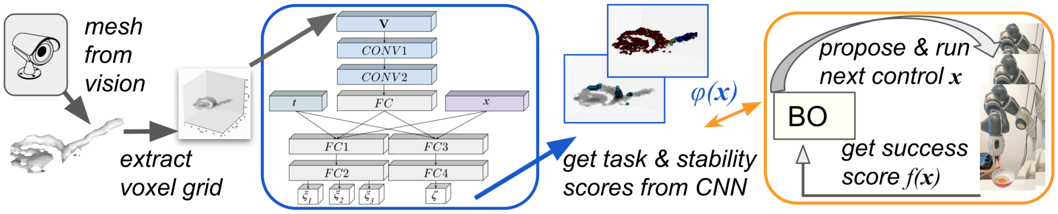


Figure 1: Overview of the overall approach. Blue highlights components trained offline; orange highlights online learning components that guide real-time decisions made by the robot. A vision system is used to first construct a mesh from raw point cloud. Then, a set of grasping points, voxelized mesh and task identity are passed through the network to obtain task suitability and grasp stability scores. These are used to construct informed kernel for BO. Then, the robot executes online search with BO to find the best task-appropriate grasp.

uncertainty over $f(\boldsymbol{x})$. After each real-world trial, BO constructs a posterior based on data obtained so far. It then uses an auxiliary function (*acquisition function*) to choose the next $\boldsymbol{x}$ to evaluate. The acquisition function selects points for which the posterior estimate of the objective f is promising. It takes into account both the posterior mean and covariance. Gaussian Process (GP) is commonly used to model the cost function: $f(\boldsymbol{x}) \sim GP(\mu(\boldsymbol{x}), k(\boldsymbol{x},\boldsymbol{x}'))$. Its kernel function captures similarity between inputs: if $k(\boldsymbol{x},\boldsymbol{x}')$ is large for $\boldsymbol{x},\boldsymbol{x}'$, then $f(\boldsymbol{x})$ has high influence on $f(\boldsymbol{x}')$. Squared Exponential (SE) kernel is frequently the default choice (its hyperparameters σ_k^2, $\boldsymbol{\ell}$ can be optimized automatically):

$$k_{SE}(\boldsymbol{x},\boldsymbol{x}') = \sigma_k^2 \exp\big(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^T \operatorname{diag}(\boldsymbol{\ell})^{-2}(\boldsymbol{x}-\boldsymbol{x}')\big), \tag{1}$$

BO with standard kernels (SE or, more generally, Matérn kernels) is effective in cases with low noise, smooth objective/cost functions or limited search space. In contrast, robotics problems exhibit high noise, sharp transitions between low- and high-cost regions, inability to limit the search space a priori without excluding optimal regions. Previous work explored using kernels informed by simulation [2, 3] and by the domain dynamics [4], in some cases allowing to learn kernel transforms automatically [5]. However, these assumed access to high-fidelity simulators/models. For many areas of robotics such simulators are not available. High anticipated simulation-reality mismatch prevents us from using simulation-based information in standard ways, like constructing the prior of the mean function for the Gaussian Process in BO. As noted in [6], in practice it can be challenging to specify prior mean effectively. Recent experiments in robotics show that performance of 'prior-based' BO can degrade in cases with significant simulation-reality mismatch [7]. Hence, embedding simulation-based information into the kernel could be a more robust alternative.

2.2 Task-oriented Grasping

Grasping simulators [8] can help generating a set of grasps labeled with quality measures, which can be used to select the best grasp for execution on a real robot. However, these quality measures often rely on access to full 3D geometry of an object, which is often not obtainable in real-world scenarios. To overcome this, recent work investigated learning to predict grasp success from partial visual inputs [9], [10], [11]. Vision alone often does not provide enough information about important object properties: it is difficult to infer inertial and, even more so, friction properties. Hence, several works proposed learning adaptively with feedback from real robot trials [12, 13, 14]. Some proposed constructing priors from simulation, but as noted in the BO background subsection, directly adding prior points or using a fixed prior mean is problematic with high simulation-reality mismatch. In the case of grasping: only medium- to low-fidelity simulators are generally available; their results only align with reality if crucial parameters, like friction, are measured and modelled meticulously. We do not assume access to estimates of friction or inertial properties of objects, and do not require modeling these in simulation either. Our main goal is to develop a method whose performance does not degrade due to incorporating data from highly inaccurate simulations. Hence, we draw inspiration from approaches that embed simulation-based information into the kernel (e.g. [15], though this prior work investigated convergence of MABs with a large number of pick-and-place simulated grasps across objects, while we emphasize finding successful task-oriented grasps with only a few BO trials on the real robot and adapt to each object separately).

Approaches from prior work described above can be successful for grasping an object in a stable manner. However, in most scenarios the ultimate goal is to allow an execution of a manipulation task, for example cutting or pouring. This problem is known as Task-oriented Grasping (TOG) [16],

[17] and is vastly more challenging than just stable grasping. The first challenge is to find grasps that are stable and also task compliant. For example, when grasping a mug most of the stable grasps for smaller parallel grippers are on the rim. However, if we want to use the mug for pouring, fingers should avoid the opening area and instead aim for the handle. This is challenging when vision is imperfect, since small task-relevant areas of the object could be obscured or distorted. The second challenge is that a robot must be able to reason about geometric properties of an object and how they relate to tasks (e.g. openings & pouring, blades & cutting). This is know as *affordance* learning and many authors have addressed this problem in the past. While some works focus solely on reasoning about object affordances, others utilize them for purpose of TOG. In [18] authors trained a CNN to predict part affordances, which are used to formulate task constraints, namely the ones on the location and orientation of the gripper. These constraints are given to optimization-based grasp planner, which executes task-oriented grasps. Similarly, [19] trained task-oriented CNNs to identify the regions a robot is allowed to contact to fulfill a task; they use a part-based approach that finds object parts with shape that is compatible with the gripper. In [20] authors proposed training TOG Network to optimize for task-oriented grasp and manipulation policy. They decomposed the problem into 1) finding task-agnostic grasps for which they use Dex-Net 2.0 [10] and 2) finding task-oriented grasps for which they train a CNN. Our work differs from this in several ways. We train a network that jointly predicts grasp stability and task suitability. Furthermore, although we could use a single score (like Dex-Net 2.0) for predicting stability, we chose to incorporate three different metrics. This provides a richer signal for BO: different metrics could be of varying importance for different tasks (e.g. one metric could score grasps based on the distance from the center of mass of the object, while another might be most sensitive to positions of the finger joints). Moreover, we train our network on a large data set of realistic 3D objects and six different tasks, while [20] use procedurally generated objects based on shape primitives and consider only two tasks (pounding and sweeping). Hence, in our case the complexity of learning object-task-grasp relationships is significantly higher.

3 Proposed Method

3.1 Bayesian Optimization with Bernoulli Alternation Kernel

Our aim is to develop an approach robust to high simulation-reality mismatch. This motivates us to look beyond solutions that put simulation-based information into GP prior or rely on simulation-based kernels alone. We first note that using a sum of kernels could be beneficial. Let $k_{sum}(\boldsymbol{x},\boldsymbol{x}') = k_{SE}(\boldsymbol{x},\boldsymbol{x}') + k_{\phi}(\boldsymbol{x},\boldsymbol{x}')$ be a kernel comprised of k_{SE} (Equation 1) and $k_{\phi} = k_{SE}(\phi(\boldsymbol{x}),\phi(\boldsymbol{x}'))$, where $\phi(\cdot)$ is akin to a warping function. Recall that in our case $\boldsymbol{x}$ is a vector of control parameters. To obtain $\phi(\boldsymbol{x})$ we could execute controls $\boldsymbol{x}$ in simulation and output relevant characteristics of the result (e.g. stability metrics for a grasp). It is useful to embed ϕ into the kernel, because we can collapse the space of unsuccessful controls. For example, ϕ could give near zero stability scores to grasps that miss or barely touch objects. This would indicate that all such failed points/controls are similar to each other, but dissimilar from successful regions of control parameters. BO can then quickly learn to neglect the non-promising regions, even though they could be far away from each other in the original space of control parameters. However, when ϕ fails to provide high-quality information, k_{sum} could be adversely impacted by the k_{ϕ} component.

To offer a more robust alternative, we propose an approach that takes contributions from both k_{SE} and k_{ϕ}, but ensures BO can not be mislead by a poor choice of ϕ. At each BO iteration/trial, we randomize the choice of whether k_{SE} or k_{ϕ} is used. For this, we define a probability distribution over kernels and draw a kernel function to be used at each BO trial independently. This defines k_{bak} as:

$$k_{bak}(\boldsymbol{x},\boldsymbol{x}') \sim \mathbb{1}_{\{\theta \le 0.5\}} k_{SE}(\boldsymbol{x},\boldsymbol{x}') + \mathbb{1}_{\{\theta > 0.5\}} k_{\phi}(\boldsymbol{x},\boldsymbol{x}'); \;\; \theta \sim Uniform(0,1) \qquad (2)$$

Note that after each iteration/trial n, the data for computing the GP posterior consists of all the points sampled so far: $\boldsymbol{D}_n = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i)) | i = 1, ..., n\}$. In BO, posterior is usually re-computed after each new sample. This aspect of BO allows us to make a choice of the kernel function for each iteration separately. So, after n iterations/trials, we first pick a kernel function k_n using Equation 2. We then compute GP posterior mean and covariance (notation from [6]):

$$mean(f_*) = \boldsymbol{k}_{*_n}^T (K_n + \sigma_{noise}^2 I)^{-1} \boldsymbol{y}_n \qquad cov(f_*) = k_n(\boldsymbol{x}_*,\boldsymbol{x}_*) - \boldsymbol{k}_{*_n}^T (K_n + \sigma_{noise}^2 I)^{-1} \boldsymbol{k}_{*_n} \qquad (3)$$

where $\boldsymbol{x}_*$ is a new point whose cost we want to predict; $f_* := f(\boldsymbol{x}_*)$; K_n is $n \times n$ matrix with $K_{ij} = k_n(\boldsymbol{x}_i,\boldsymbol{x}_j)$; $\boldsymbol{k}_{*_n} \in \mathbb{R}^n$ is a vector of covariances between $\boldsymbol{x}_*$ and each $\boldsymbol{x}_i$, i=1,...,n; $\boldsymbol{y}_n$ is a vector of evaluations for the sampled points: $[\boldsymbol{y}_n]_i = f(\boldsymbol{x}_i)$. Algorithm 1 gives a concise summary.

For intuitive insight, note that points sampled for trials when k_ϕ is used could provide fast guidance towards useful parts of the search space. These points are included in the data for subsequent posterior computations, enabling even trials in which k_{SE} is used to propose better next choices. This is important if probability of discovering a promising region is small, which is frequent in robotics. If k_ϕ is not useful, or even misleading, choices made on the trials that use k_{SE} are not impacted by poor ϕ, since with our approach BO's acquisition function makes its decisions using only one of the kernels at a time.

Algorithm 1: BO-BAK

```
sample $\boldsymbol{x}_1$ randomly, get $y_1 = f(\boldsymbol{x}_1)$ from real world
initialize: $\boldsymbol{D}_1 = \{(\boldsymbol{x}_1, y_1)\}$
for $n = 1, 2, ...$ do
    sample kernel function $k_n$ using Equation 2
    get posterior GP mean and cov using $\boldsymbol{D}_n$ & Eq. 3
    select $\boldsymbol{x}_{n+1}$ by optimizing acquisition function $\alpha$:
        $\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x}} \alpha(\boldsymbol{x}; \boldsymbol{D}_n)$
    get $y_{n+1} = f(\boldsymbol{x}_{n+1})$ from real world
    augment data $D_{n+1} = \boldsymbol{D}_n \cup \{(\boldsymbol{x}_{n+1}, y_{n+1})\}$
```

In Section 4.1 we evaluate on analytic functions that exhibit challenges similar to our target domain. We confirm that informed kernels can provide significant improvement over conventional BO. k_ϕ and k_{sum} appear somewhat sensitive to the quality of the warping function ϕ, while k_{bak} appears to be more robust empirically. As a preliminary theoretical comment, we observe that k_{bak} retains optimality guarantees of the conventional BO. Intuitively, this is because in expectation we perform N/2 trials using only SE to choose the next point. This is no worse than using conventional BO in half of the trials, and we anticipate that satisfactory consistency and regret bounds could be derived. We leave this theoretical analysis to future work. In this work, we investigate whether k_{bak} is beneficial in complex real-world robotics scenarios. These pose a further challenge, since often the assumptions made to obtain consistency and regret bounds for conventional BO do not hold in practice.

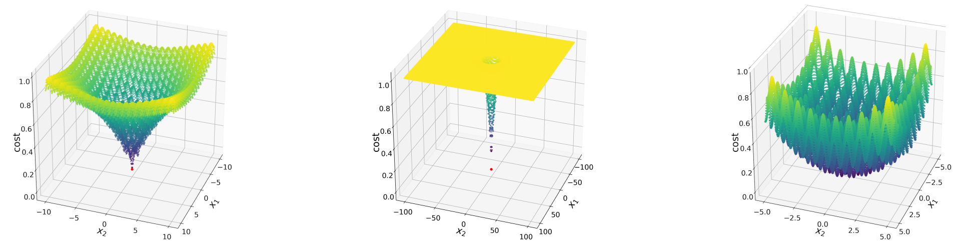
3.2 Task-oriented Grasping, Network Architecture and Training

The objective of our TOG CNN is to learn to output several grasp stability scores ξ_i and a task suitability score ζ, given as input a task t and a grasp $\boldsymbol{x}$ on an object. A grasp is parameterized by $\boldsymbol{x} := (\boldsymbol{p}, \vec{n}, \psi, d) \in \mathbb{R}^8$, where $\boldsymbol{p} \in \mathbb{R}^3$ is a point on the surface of the object and $\vec{n} \in \mathbb{R}^3$ is its corresponding unit normal, ψ is a gripper roll and d is an offset from the surface of the object. A task t is defined as a single manipulation action that starts with a grasp. To fully exploit the power of 3D representations we use volumetric architecture in convolutional layers. For this, we scale an object mesh to fit inside $50 \times 50 \times 50$ binary voxel grid (for learning, we also scale the grasping points). The actual input to the network is then: a voxel grid of an object ($\boldsymbol{V}$), a grasp ($\boldsymbol{x}$), and a task (t, encoded as 1×6 one-hot vector). For details of network architecture see Figure 1. To handle noisy data that can be encountered in the real-world, the network is trained with dropout (ratio of 0.5) in the first layer. Furthermore, for training we randomly rotate objects to account for possible orientations for tabletop grasping. The training was performed using Tensorflow on a Titan X GPU.

A single CNN is trained for all object categories and tasks (no need to classify the object explicitly). The kernel for BO is constructed "on the fly" with a forward pass on the trained CNN. To grasp a new object the steps are: 1) vision processes the object (point cloud → mesh → voxel grid); 2) voxel grid & task id are fed as input to CNN; 3) we get as output predicted stability & task scores for any sets of grasp parameters, which enables computing kernel distances quickly during BO. Once the CNN is trained, we have quick access to a kernel for any object type and task that are included in CNN training. BO is run for each object instance "from scratch", but with informed kernel incorporated into the search. In collaborative industrial settings (where the same objects could be used for many days) our method could identify optimal task-specific grasping parameters and these can be utilized for a given tool without re-optimizing. If it is unlikely that the robot will deal with the same tool/object again, BO would run each time for each new object instance. This relieves us from making restrictive assumptions about object properties that can't be inferred from vision only.

3.3 Data Generation

To obtain learning targets for our network, we simulate grasping various objects in OpenRave [8] with a parallel gripper. We provide the simulator with a "free-floating" 3D model of the gripper and objects. The objects dataset consists of 605 mesh models from ShapeNET [21] and ModelNet40 [22], containing objects from 13 different categories. For training we align each mesh with a reference frame that coincides with object's principal axes and meshes are scaled to real-world size based on object's category.



(a) Ackley function with challenging local minima.

(b) Ackley on a larger domain: only 1% of the space is < 0.9.

(c) Rastrigin function: highly multimodal, deep local minima.

Figure 4: High-dimensional analytic functions for testing optimization algorithms. Ackley and Rastrigin functions present challenges similar to those most frequent in optimization for robotics: (a) shallow local optima, (b) small low cost region & sharp cost changes, (c) deep local optima difficult to overcome. Figures show functions in 2D, with values (costs) normalized to $[0,1]$. We test on 2- to 10-dimensional versions of these.

Task	Object Class	Grasp Requirements
handover	all objects	leave handle(s) clear
screw	screwdrivers	avoid the shaft
cut	knife and scissors	avoid blade(s)
pour	bottle, can, mug, wine glass	avoid the opening area
support	pan, spatula, spoon, fork	avoid the supporting area
pound	hammer	avoid a hammer head

Figure 2: Objects, tasks and grasps requirements in our dataset.

We label the objects with target task scores by assigning positive/negative labels to parts of the object suitable/unsuitable for the task. Figure 2 shows the tasks we consider and relations to applicable objects. We execute 4500 grasps on each object with grasp parameters as follows:

- **point & normal** $(\boldsymbol{p}, \vec{n})$: randomly sample 500 points on object's surface (along with the corresponding normal directions);
- **gripper roll** ψ: try angles $\psi \in \{0, \pi/2, \pi/4\}$;
- **offset** d: try offsets $d \in \{0, 2\text{cm}, 4\text{cm}\}$.

Grasps are simulated by approaching a point on an object along the normal direction and closing the fingers. Once the fingers close, we extract scores for three of the grasp stability metrics recently analyzed by [23]:
ξ_1 – Grasp Isotropy Index [24];
ξ_2 – Distance from Center of Mass [25, 26, 27, 28];
ξ_3 – Grasp Finger Posture [29, 30, 31].

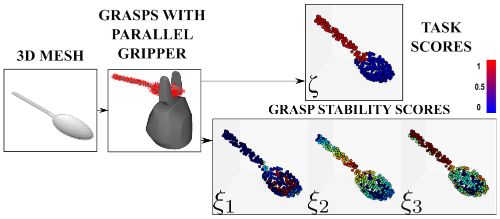


Figure 3: Data generation for *support* on spoon (with $\psi = 0$, $d = 0$). For task scores, red area denotes points suitable for grasping.

4 Experiments

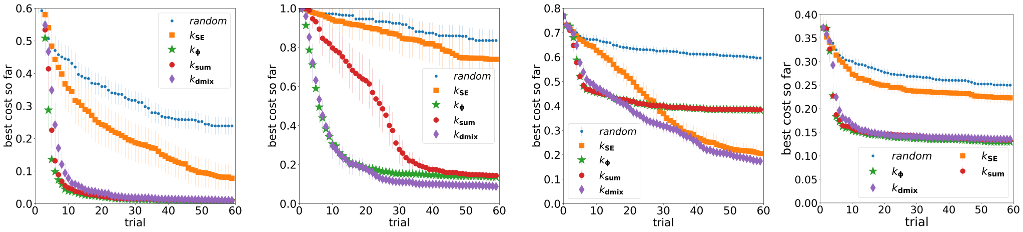
4.1 Performance of BO Variants on Synthetic Benchmarks

To anticipate the challenges of running optimization on a real-world robotics system, we first test the performance of BO variants on synthetic benchmarks. From the commonly used optimization test functions [32] we select 3 settings which exemplify the main challenges that frequently occur in robotics: numerous shallow local optima, small low-cost region and sharp cost function drops/rises, deep local optima difficult to overcome when searching for global optimum. Figure 4 visualizes the settings we consider. Figure 4a shows the Ackley function with numerous shallow local minima:

$$f_{AC}(x) = -a \cdot \exp\left(-b\sqrt{\tfrac{1}{d}\textstyle\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\tfrac{1}{d}\textstyle\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1); \;\; a=20, b=0.2, c=2\pi, x \in \mathbb{R}^d$$

For a further challenge we consider this function on a much larger domain $[-100, 100]$. This is similar to considering Easom function, which is commonly used to test robustness to steep ridges/drops in the search space. Figure 4b shows the Ackley function on this larger domain.

We are interested in emulating a setting where simulation could provide coarse guidance, but also could be limiting when searching for the global optimum. For the synthetic setting we pick ϕ with components that capture information from two first addends of the Ackley function: $[\phi(x)]_1 = -b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}$; $[\phi(x)]_2 = \frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)$. This gives a kind of 'collapsing' of the relevant features, akin to simulation that aims to capture most salient features needed to approximate real-



(a) Ackley 2D on $[-10, 10]$. (b) Ackley 2D on $[-100, 100]$. (c) Ackley 10D on $[-10, 10]$. (d) Rastrigin 10D on $[-5, 5]$.

Figure 5: BO on analytic functions. Plots of mean over 40 runs for each kernel type, 95% CIs. Comparison with random search ('random' in the legend) shows relative difficulty of each setting.

world output. Figure 5 shows comparisons of BO variants on 2D and 10D versions of the Ackley function. In 2D, using ϕ gives a significant advantage to all the informed versions of BO (with k_ϕ, k_{sum} and k_{bak} kernels). When the low-cost region is small, the gains are even more striking. In this case the performance of uninformed BO with SE kernel degrades to random search, while the informed versions get close to the optimum in less than 40 trials. However, the hint of limitations induced by using ϕ is already visible: in the case of k_ϕ and k_{sum} the improvement stagnates after 50 trials. This stagnation is even more striking when 10-dimensional version of the Ackley function is optimized. There, even the uninformed BO with SE kernel improves over k_ϕ and k_{sum} after 30 trials. In contrast, k_{bak} is able to 'recover' and outperform both informed and uninformed kernels.

To test robustness to deep local minima we use Rastrigin function: $f_{RA}(x) = \sum_i^d x_i^2 - \sum_i^d a\cos(c\pi x_i) + a \cdot d$; $a=10, c=2, x \in \mathbb{R}^d$. Figure 4c shows a 2D version of the function. As before, ϕ summarizes the first two addends: $[\phi(x)]_1 = \sum_i^d x_i^2$; $[\phi(x)]_2 = \sum_i^d a\cos(c\pi x_i)$. Figure 5d shows results for the BO variants in 10D. In this case, all informed kernels improve over uninformed SE kernel. Overall, the results on test benchmarks appear promising. Therefore we proceed with experiments on hardware, since we believe this step is crucial for validating applicability of our approach to robotics problems.

4.2 Experimental Setup for Task-oriented Grasping

To evaluate performance on a real-world setting we construct Everyday Objects Dataset (EOD), then do task-oriented grasping with an ABB Yumi robot. The objects (shown in Figure 7) are from seven categories and can be used for six different tasks. Our selection of objects is constrained by limitations of the robot: maximum payload of 500g (lower in practice), rigid plastic parallel gripper, no force-torque or tactile sensing. Once an object is placed on the table, we use Microsoft Kinect to get a dense point cloud of the scene and segment out the object. We then generate a mesh representation from the partial point cloud and attach a coordinate frame to the object[2]. Then we compute grasp points, discarding those in collision with the table. The object is voxelized and together with grasp and task representation fed through the CNN to get stability and task score estimates.

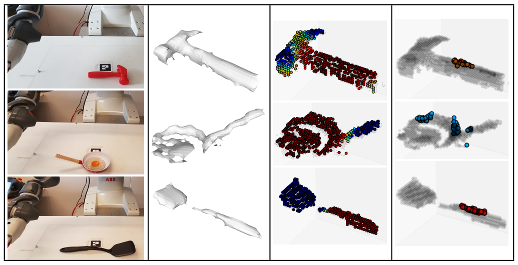


Figure 6: Three objects from our everyday objects dataset (EOD) and visualization of task & stability scores. From left to right: object in the workspace, partial mesh, ζ task scores, ξ_2 stability metric for top 100 grasps with $\zeta > 0.5$.

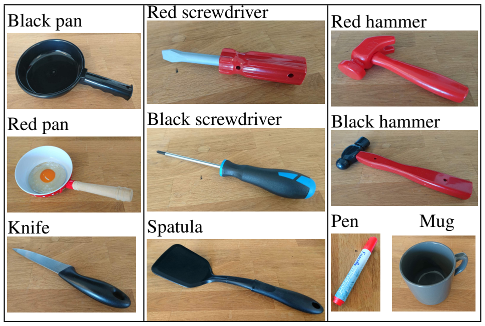


Figure 7: Ten objects from our EOD.

We execute grasps on the robot and report the results using the following scores:
[0.0-0.99] No plan: planning failed or returned partial plan;

[2] Object coordinate frame is positioned in the center of the mesh bounding box and has the same orientation in the world coordinate frame as the april tag which we use to segment the object, for details see [33].

[1.0] No lift: failed to grasp the object;
[2.0] Lift failed: the robot grasped the object but failed to lift it up (the object slipped);
[3.0] Wrong grasp: grasped the wrong part of the object (unsuitable for the task);
[4.0] Success: task-appropriate grasp was successful.
After each execution, operator re-positions the object in the workspace if needed and starts next trial.

Before starting experiments on hardware, for an initial test of our CNN we evaluated its performance on a synthetic test set. For task suitability: we report task-specific and overall MaxF1 and MAP scores in Table 1 (top). Overall, the network learns to recognize parts of objects that are suitable for grasping given a task. The most challenging task is *cut*, since blade and handle parts of a knife can have very similar appearance. For grasp stability: in Table 1 (bottom) we report RMSE for each of the stability metrics. Since we are interested in using the predicted scores for task-oriented grasping, we also report precision at top 100 for task-appropriate points ($\zeta > 0.5$).

Task	**maxF1**	**MAP**
handover	0.892	0.930
screw	0.988	0.990
cut	0.764	0.877
pour	0.874	0.963
support	0.952	0.964
pound	0.940	0.948
overall	0.909	0.943

Stability scores	ξ_1	ξ_2	ξ_3
RMSE	0.184	0.173	0.206
Precision@100	0.651	0.902	0.688

Table 1: CNN evaluation on synthetic testset.

4.3 Hardware Experiments for Top-k Grasps

To test the performance of our CNN in a real-world setting, we did task-oriented grasping for objects in our EOD. We generated 4500 grasps (as in Section 3.3), then removed those in collision with the table. For each object-task pair, 10 grasps with highest stability according to score ξ_2 (and task scores $\zeta > 0.5$) were executed on the robot. We did not perform *handover* for screwdrivers and knife, because their blades were not adequately visible to our camera. Furthermore, although some high-scoring points were found on the spatula for *handover*, they were out of reach for the robot. For all other object-task pairs multiple successful task-oriented grasps were found. Table 2 reports scores for each trial (scoring described in Section 4.2).

Our CNN mostly outputs high scores for grasp points that are on suitable parts of the object, both from task and stability perspectives. Of course not all of these grasps succeed in practice, since the CNN is trained using simulations that do not model friction or mass distribution. In reality, the most stable grasps were on spatula for *support*, white pan for both tasks and red hammer for task *pound*. The most unstable grasps where on mug *handover*, likely because of smooth surface, high mass, severely degraded partial mesh from vision.

For the above experiments, we reduced maximum gripper standoff from 4cm to 3cm. Since the tabletop was not present during training, CNN was likely to learn high scores for grasps with small standoff. These would frequently collide with the table. The alternative grasps with 4cm standoff were still successful in simulation, but slipped off the object in reality. While we could fix this issue by simply reducing the maximum standoff in this case, such simulation-reality mismatch might be harder to fix in general. This motivates experiments with adaptive search like BO, which we describe next.

Task	Scores (for each trial)
Spatula	
support	4 4 4 4 4 2 4 4 4 1
White pan	
handover	4 4 4 4 4 4 3 3 4 3
support	4 4 4 4 4 1 4 1 4 4
Black pan	
handover	1 2 2 1 4 2 4 4 4 4
support	1 1 1 1 4 4 1 2 1 4
Red hammer	
handover	1 2 4 4 2 4 2 4 4 1
pound	4 4 4 4 4 1 4 1 4 4
Black hammer	
handover	4 4 4 4 2 4 4 1 1 2
pound	4 4 4 2 4 4 1 1 1 4
Mug	
handover	2 2 4 2 4 2 2 4 2 2
pour	4 4 2 1 4 3 4 4 4 4
Red screwdriver	
screw	4 4 4 1 4 4 2 4 4 2
Black screwdriver	
screw	2 2 4 2 1 4 2 2 2 2
Knife	
cut	1 2 4 2 4 2 2 4 2 4
Pen	
handover	4 2 4 2 4 2 4 4 2 2

Table 2: Top-10 grasps on EOD.

4.4 Hardware Experiments for Bayesian Optimization

We ran experiments using the proposed BO approach from Section 3.1. k_ϕ kernel was constructed using the stability scores: $\phi(\boldsymbol{x}) = [\xi_1, \xi_2, \xi_3]$, which were obtained from the trained CNN. The vector of control parameters $\boldsymbol{x} = (\boldsymbol{p}, \vec{n}, \psi, d)$ contained: 3D coordinates for a point on the object, approach direction, gripper roll and offset (as in Section 3.2). Choice for $\boldsymbol{p}$ was constrained to the surface of the object, but other control parameters were limited only by a choice of [min,max] values. Points on the object were sampled from the output of the vision system, those with low task scores were filtered out (for a given task) using the trained CNN. Our implementation of BO was based on [34, 35]. After each BO trial, $f(\boldsymbol{x})$ evaluation (expressing the success of executing grasp with parameters $\boldsymbol{x}$) was given as described in Section 4.2.

BO usually starts with a random trial, but in our experiments we instead execute one top choice from each of the 3 stability metrics. For challenging objects this does not yield any promising points. However, from this BO can infer which regions are not promising. *Handover* task for spatula provides a clear example of this. Left side of Figure 8 shows executing top choice according to each of the 3 stability metrics (after filtering out planning failures). These choices are not successful, moreover executing top 20 choices does not yield successful grasps. Most stable grasps are not reachable due to robot's joint limits or the need to approach too close to the table. We obtain a successful handover on the 4^{th} trial (1st candidate from BO), shown in the right part of Figure 8. The grasp is just above the handle part (the handle needs to be clear for handover). It is labeled as acceptable for task completion, but is unlikely to be among top k offline choices. This is because task and stability scores are higher in other parts of the object, but those parts are inaccessible.

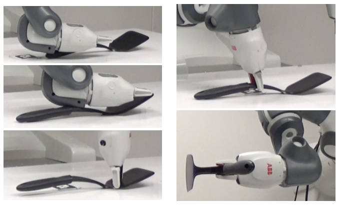


Figure 8: Left: top choices for handover task from 3 stability metrics; Right: subsequent BO trial.

Left side of Figure 9 shows success of BO on the 4^{th} trial for mug handover (1st candidate from BO). Right side shows successful pan handover on the 5^{th} trial (2nd candidate from BO). In contrast, top k grasps computed offline attempted to grasp the middle and outer part of the handle, which for this pan resulted in either slippage or tilting. The above object-task pairs presented the toughest challenges for the top-10 approach, while BO proposed successful task-oriented grasps in the first few trials.

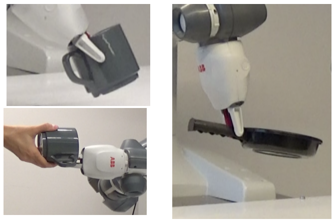


Figure 9: Left: A trial from BO for mug handover task. Right: A trial from BO for pan support task.

Overall, we did 15 runs with BO: 6 full runs with 10 trials each, 9 partial runs that we stopped early after 5 trials. We stopped a run early if the top 3 choices from CNN already suggested successful approach points, or if multiple successful grasps were executed in the first 5 trials. We obtained multiple successful task-oriented grasps for spatula, mug, hammers, pans, screwdrivers. We could not complete BO on screwdriver handover task, knife and pen, because these required getting very close to the surface of the table (within 1-3mm). BO repeatedly proposed such approaches, but the planning library rejected these as near-collisions.

For the full BO runs we focused on object-task pairs that were challenging for top-10 approach (e.g. ran spatula *handover* twice to ensure repeated success, since top-10 approach failed for this task). Figure 10 summarizes the results, showing quick significant improvement of BO over the initial top choices from the 3 stability metrics. Qualitatively, the benefits we observed from using BO were: 1) exploring various parts of the object systematically and efficiently; 2) sampling a variety of successful controllers that further improve over a merely acceptable controller.

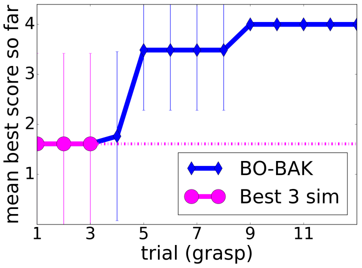


Figure 10: BO for challenging objects.

5 Conclusion and Future Work

We proposed a variant of online global search suitable for simulation-informed optimization. Our focus was on validating this approach on a challenging robotics task. We plan to extend evaluation to task-oriented grasping scenarios in clutter, external disturbances, or real-time requirements for task completion. Our data generation does not assume a tabletop scenario, since objects are simulated without a support surface. This is a strength, since there is no need to re-run offline training when the workspace properties change. However, we need to improve our planning pipeline to avoid needless planning failures when grasping very small objects from tabletop. On the theory side, it would be interesting to explore theoretical properties of the proposed informed BO, investigate which guarantees could be obtained. It would be useful to put an emphasis on retaining realistic assumptions: no bounds on simulation-reality mismatch a-priori, non-smooth objective/cost functions.

Acknowledgments

This research was supported in part by the Knut and Alice Wallenberg Foundation.

References

[1] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[2] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[3] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. G. Atkeson. Bayesian Optimization Using Domain Knowledge on the ATRIAS Biped. In *Robotics and Automation (ICRA), 2018 IEEE International Conference on*, 2018.

[4] A. Marco, P. Hennig, S. Schaal, and S. Trimpe. On the design of LQR kernels for efficient controller learning. In *56th IEEE Annual Conference on Decision and Control, CDC*, 2017.

[5] R. Antonova, A. Rai, and C. G. Atkeson. Deep kernels for optimizing locomotion controllers. In *Conference on Robot Learning*, pages 47–56, 2017.

[6] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[7] A. Rai, R. Antonova, F. Meier, and C. G. Atkeson. Using simulation to improve sample-efficiency of bayesian optimization for bipedal robots. *arXiv preprint arXiv:1805.02732*, 2018.

[8] R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 79, 2008.

[9] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[10] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[11] P. Schmidt, N. Vahrenkamp, M. Wächter, and T. Asfour. Grasping of unknown objects using deep convolutional neural networks based on depth images. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[12] O. Kroemer, R. Detry, J. Piater, and J. Peters. Combining active learning and reactive control for robot grasping. *Robotics and Autonomous systems*, 58(9):1105–1116, 2010.

[13] L. Montesano and M. Lopes. Active learning of visual descriptors for grasping using non-parametric smoothed beta distributions. *Robotics and Autonomous Systems*, 60(3):452–462, 2012.

[14] J. Oberlin and S. Tellex. Autonomously acquiring instance-based object models from experience. In *Robotics Research*, pages 73–90. Springer, 2018.

[15] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1957–1964. IEEE, 2016.

[16] D. Song, C. H. Ek, K. Huebner, and D. Kragic. Task-based robot grasp planning using probabilistic inference. *IEEE transactions on robotics*, 31(3):546–561, 2015.

[17] L. Antanas, P. Moreno, M. Neumann, R. P. de Figueiredo, K. Kersting, J. Santos-Victor, and L. De Raedt. High-level reasoning and low-level learning for grasping: A probabilistic logic pipeline. *arXiv preprint arXiv:1411.1108*, 2014.

[18] M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic. Affordance detection for task-specific grasping using deep learning. In *Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on*, pages 91–98. IEEE, 2017.

[19] R. Detry, J. Papon, and L. Matthies. Taskoriented grasping with semantic and geometric scene understanding. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[20] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *arXiv preprint arXiv:1806.09266*, 2018.

[21] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[22] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[23] C. Rubert, D. Kappler, A. Morales, S. Schaal, and J. Bohg. On the relevance of grasp metrics for predicting grasp success. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 265–272. IEEE, 2017.

[24] B.-H. Kim, S.-R. Oh, B.-J. Yi, and I. H. Suh. Optimal grasping based on non-dimensionalized performance indices. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 949–956. IEEE, 2001.

[25] E. Chinellato, A. Morales, R. B. Fisher, and A. P. Del Pobil. Visual quality measures for characterizing planar robot grasps. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(1):30–41, 2005.

[26] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on robotics and automation*, 11(6):868–881, 1995.

[27] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *The International Journal of Robotics Research*, 16(1):11–35, 1997.

[28] D. Ding, Y.-H. Lee, and S. Wang. Computation of 3-d form-closure grasps. *IEEE Transactions on Robotics and Automation*, 17(4):515–522, 2001.

[29] J. Cornella and R. Suárez. Fast and flexible determination of force-closure independent regions to grasp polygonal objects. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 766–771. IEEE, 2005.

[30] Y.-H. Liu. Computing n-finger form-closure grasps on polygonal objects. *The International journal of robotics research*, 19(2):149–158, 2000.

[31] Y. Li, Y. Yu, and S. Tsujio. An analytical grasp planning on given object with multifingered hand. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 3749–3754. IEEE, 2002.

[32] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: test functions and datasets. *Simon Fraser University, Burnaby, BC, Canada*, 13:2015, 2013.

[33] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2442–2447. IEEE, 2017.

[34] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (gpml) toolbox. *J. Mach. Learn. Res.*, 11:3011–3015, Dec. 2010.

[35] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. Cunningham. Bayesian Optimization with Inequality Constraints. In *ICML*, pages 937–945, 2014.