

Robust Task-Based Grasping as a Service

Jingyi Song¹, Ajay Tanwani¹, Jeffrey Ichnowski¹, Michael Danielczuk¹, Kate Sanders¹, Jackson Chui¹,
Juan A. Ojea², Ken Goldberg¹

Abstract—Robot grasping for automation must be robust to the inherent uncertainty in perception, control, and physical properties such as friction. Computing robust grasp points on a given object is even more challenging when there are constraints due to a task intended to be performed with the object, for example in assembly, packing, and/or tool use. To compute grasps that robustly achieve task requirements, we designed an intuitive user interface that takes an object mesh as input and displays it, allowing non-specialists to indicate “stay-out” zones by painting facets of the mesh and to indicate desired forces and torques by drawing vectors. The interface then sends this specification to our server which computes resulting grasps and send them back to the client where the resulting parallel-jaw grasp axes are displayed color-coded by robustness. We implemented this interface in the cloud-based “Dex-Net as a Service - Task (DNaaS - Task)” system that runs on any browser and reports examples. The system is available at: <https://dex-net.app>

I. INTRODUCTION

Most robot grasping algorithms aim to optimize resistance to gravity so as to optimize successful lifting, but many automation applications such as assembly, packing, and tool use require “task-based” constraints on robot grasping. These can include limits on where a given object can be touched so that delicate surfaces such as lenses and high-gloss finishes are not scratched, or requirements for the robot grasp to resist forces to be applied to the object beyond gravity, such as desired insertion forces to achieve packing or assembly, or torques needed to screw or twist an object.

As a result, grasping algorithms must also take into account the task to be performed as well as object geometry. As it can be challenging to specify the task, we present a novel and intuitive user interface that takes an object mesh as input and displays it, allowing non-specialists to indicate “stay-out” zones by painting facets of the mesh and to indicate desired forces and torques to apply by drawing vectors. The system then computes and displays associated parallel-jaw grasp axes color-coded by robustness. We report on an implemented version of this interface and examples.

This paper makes three contributions:

- 1) An intuitive “task-based” grasping user interface that takes as input a 3D object mesh and allows non-specialists to indicate task constraints with “stay-out” zones and desired wrenches.
- 2) A modification of the Dex-Net 1.0 algorithm to compute robust grasps that meet these task-based constraints with wrench resistance metric.

¹University of California, Berkeley. {jingyi-song, ajay.tanwani}@berkeley.edu

²Siemens Corporation. juan.aparicio@siemens.com

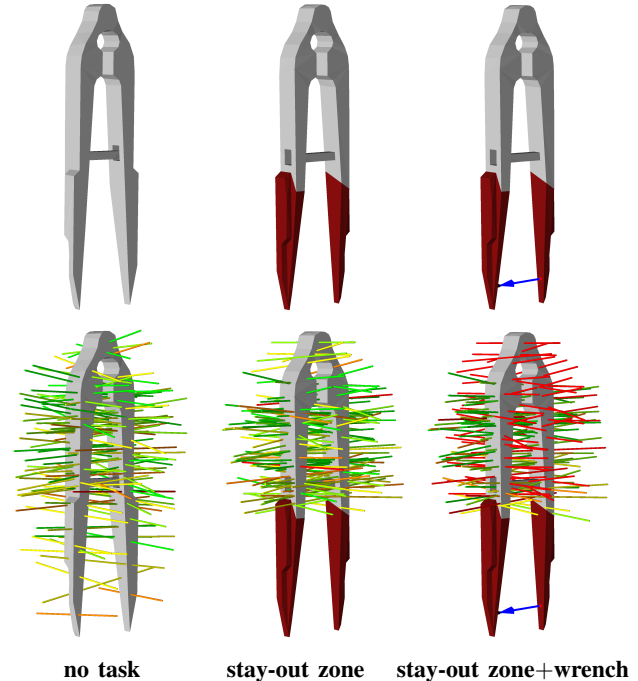


Fig. 1: **Task-directed grasping results for a tweezer object.** The top row shows the object mesh, without constraints (*left*), with stay-out zones in red (*middle*), and with a desired applied force in blue (*right*). The bottom row shows computed parallel-jaw grasps for each case; each grasp axis is displayed as a color-coded “whisker” corresponding to its robustness to the constraints and perturbations.

- 3) A web-based implementation of robust task-based grasping as a service.

II. RELATED WORK

Research into task-based grasping has explored various formulations of the problem and approaches. In this section, we present this prior research categorized into task-directed grasping, followed by data-driven strategies to learn the task-based grasping strategies.

A. Task-Directed Grasping

Grasping diversely shaped and sized novel objects has a wide range of applications in industrial and consumer markets. Grasping is often subdivided across grasp synthesis and grasp quality evaluation. Grasp synthesis generates grasp candidates from contact locations, while grasp optimization evaluates the quality of candidate grasps subject to criteria such as force closure or wrench resistance [14, 20]. As force closure is a more conservative metric and

guarantees resistance to wrench in any direction, we use wrench resistance to quantify the ability of a grasp to resist disturbances along certain directions that specify the task. Li and Sastry [9], Prats *et al.* [18] and Haschke *et al.* [4] also investigate task-oriented grasping strategies using an external task wrench. Ortenzi *et al.* [15] advocate the need of a task oriented metric for goal-directed robot manipulation in addition to stability and mean picks per hour.

In addition to grasp analysis, recent work has also focused on using information about the planned trajectory to plan grasps. Mavrakis *et al.* [12] use reasoning about robot kinematics after grasping to select grasp contact points and later minimize the work of the resulting trajectory by choosing from a set of possible grasps [13]. Pardi *et al.* [16] focus on choosing grasps that will lead to collision-free trajectories during the proceeding manipulation task. Similarly, Zimmermann *et al.* [24] simultaneously optimize grasp and motion planning to perform pick-and-place and handover tasks as part of an assembly pipeline. However, each of these papers only considers a task where the robot must execute a trajectory after grasping, and do not consider external wrenches that must be resisted by the grasp as part of the task (see [20] for more details). Holladay *et al.* [5] present a planner for robot tasks that require motions and forces with tool-use as an example application.

B. Learning Grasping Strategies

Song *et al.* [21] presented a task-directed grasping model taking into account task, action and object information in the Bayesian setting. Kokic *et al.* [7] use CNN's for learning object affordances, class and object orientation to specify grasp constraints for task-based grasping. Recently, the authors proposed to predict a suitable task-specific grasping region by taking an object point cloud as input, where the hand-object pose labels are learned from human-activity datasets [6]. Fang *et al.* [3] propose Task-Oriented Grasping Network (TOG-Net) that uses self-supervision to jointly optimize both the task-based grasp and the manipulation policy for a tool. Qin *et al.* [19] propose learning keypoints of tools, such as grasp point and effect point, through self-supervision of tasks and using point-cloud observations. Pas *et al.* [17] detect graspable object parts from 3D point clouds. Zhirong Wu *et al.* [23] use a volumetric representation to study the 3D representation of objects. Xu *et al.* [22] propose a learning-based approach to separately assess the stability of grasps for several sub-tasks.

In this work, we synthesize grasp locations based on the stay-out zones, and evaluate the grasp quality with respect to a given direction in which task wrench is applied to satisfy the task. We extend the work in [8] to provide robust task-based grasping as a service for public use.

III. PROBLEM FORMULATION

Given a 3D mesh, represented by a set of faces and vertices \mathcal{M} , and a task that requires the application of a wrench τ , we define a semantic task-based grasping model m as a combination of τ and the stay-out zone $\mathcal{M}_{\text{out}} \subset \mathcal{M}$. Then,

given a model m , we identify a set of grasp candidates \mathcal{G} . Each grasp candidate $g \in \mathcal{G}$, defined by a center and grasp axis, has contact points entirely contained in $\mathcal{M}_{\text{out}}^C$, the complement of \mathcal{M}_{out} , and has an associated quality $r \in [0, 1]$ that measures the ability of g to robustly resist the task wrench τ under perturbations. Here, r reflects the relative ability of g to complete the task for the given object. Higher values of r reflect robustness to perturbations in the task wrench.

A. System Design Considerations

In this section, we review design considerations for a robust task-based grasping model, analysis, system, and user interface.

Objects and tools used in everyday life are well understood by their users, and often by the general population. Studies [1, 2] have shown that humans are capable of identifying the suitable grasps for specific tasks. We propose the following design considerations:

a) Objects may be used for more than one task: As such the best grasp for an object will be dependent on the task the robot will perform.

b) Availability of meshes: Meshes for many objects are readily available in online repositories such as Thingiverse [11]. Manufactures of newly designed objects are also likely to have meshes from the design and manufacturing process.

c) Leverage human knowledge: Objects designed for use by humans have a wealth of history and knowledge about the wrenches and surfaces needed for a task.

d) Model usable by algorithms: The semantic task-based model should be both intuitive to humans and usable by grasp-analysis algorithms.

IV. TASK-BASED GRASPING SYSTEM

This section describes a proposed robust-task based grasping system consisting of three parts: 1) stay-out zones and desired forces and torques that define the semantics of a task-based grasp, 2) an informed sampling-based algorithm that uses the grasp zones to perform grasp analysis and generate a ranked set of candidate grasps, and 3) a cloud-based web interface for defining task-specific grasping models for objects.

A. Semantic Task-Based Grasp Model

To represent a semantic task-based grasp model we use a mesh augmented with a stay-out zone and an external wrench. The mesh, as a representation of the surface, facilitates grasp analysis based on sampled points and area-contact models. Stay-out zones prevent contacts that hinder or prevent the task. The wrench specification defines wrench resistance to perform the task. An example of this model is shown in Figure 1 with stay-out zones painted in red and task wrench (force vector) with a blue arrow.

Algorithm 1 Task-Based Grasp Analysis

Require: A mesh \mathcal{M} , A stay-out region $\mathcal{M}_{\text{out}} \subset \mathcal{M}$, and a task wrench τ

```

1:  $\mathcal{G}_{\text{samples}} \leftarrow \emptyset$ 
2: while  $|\mathcal{G}_{\text{samples}}| < n_{\text{samples}}$  and not max_iterations do
3:    $p_0 \leftarrow$  randomly sample point on  $\mathcal{M}$ 
4:    $p_1 \leftarrow$  shoot ray from  $p_0$  within friction cone until intersection with  $\mathcal{M}$ 
5:    $g \leftarrow (\frac{1}{2}(p_0 + p_1), (p_1 - p_0)/\|p_1 - p_0\|)$ 
6:   if gripper at  $g$  not in collision with  $\mathcal{M}$ 
     and gripper wide enough for  $g$ 
     and  $\{p_0, p_1\} \cup \mathcal{M}_{\text{out}} = \emptyset$  then
7:      $\mathcal{G}_{\text{samples}} \leftarrow \mathcal{G}_{\text{samples}} \cup \{g\}$ 
8:    $\mathcal{G}_{\text{analysis}} \leftarrow \emptyset$ 
9:   for all  $g \in \mathcal{G}_{\text{samples}}$  do {Compute grasp quality}
10:     $r \leftarrow 0$ 
11:    for  $i \leftarrow 1, \dots, n_p$  do
12:       $g_i \leftarrow$  random perturbation of  $g$ 
13:      if  $g_i$  not in collision and resists  $\tau$  then
14:         $r \leftarrow r + 1/n_p$ 
15:       $\mathcal{G}_{\text{analysis}} \leftarrow \mathcal{G}_{\text{analysis}} \cup \{(g, r)\}$ 
16: return  $\mathcal{G}_{\text{analysis}}$ 

```

B. Task-Based Grasp Analysis Algorithm

To perform a task-based grasp analysis on an object and identify candidate grasps for a task, we propose a modification to the Dex-Net 1.0 grasp planning algorithm in Mahler *et al.* [10] for computing analytical grasps on meshes. An overview of the algorithm is shown in Algorithm 1. The algorithm takes the stay-out zones and external wrench on an object mesh as input, and generates a set of grasps along with associated quality scores. The grasps in the output avoid the stay-out zones. The associated quality score reflects the robustness of the grasp in the range $[0, 1]$ indicating how robustly the gripper is robust to perturbations while resisting the external task's wrench, with 0 indicating the grasp will fail to complete the task, 1 indicating the grasp is highly likely to complete the task, and values between proportionally representing the likelihood of successful completion.

The algorithm operates by sampling candidate contact points on the object mesh. Points falling within the stay-out zone are discarded. We shoot rays from the sampled candidate contact points in random directions within the friction cone until they intersect with the object mesh to create antipodal contact point pairs. From the points, a grasp candidate g is constructed by taking the center of these points and computing the grasp axis as the vector between the points. We then prune these grasp candidates by: 1) the maximum width of the parallel-jaw gripper, and 2) collisions with the gripper. We continue to iteratively sample grasp candidates until we obtain the desired number of grasp candidates or we reach a fixed maximum number of sampling iterations.

The algorithm subsequently evaluates grasp candidates based on their robustness to perturbations in resisting the

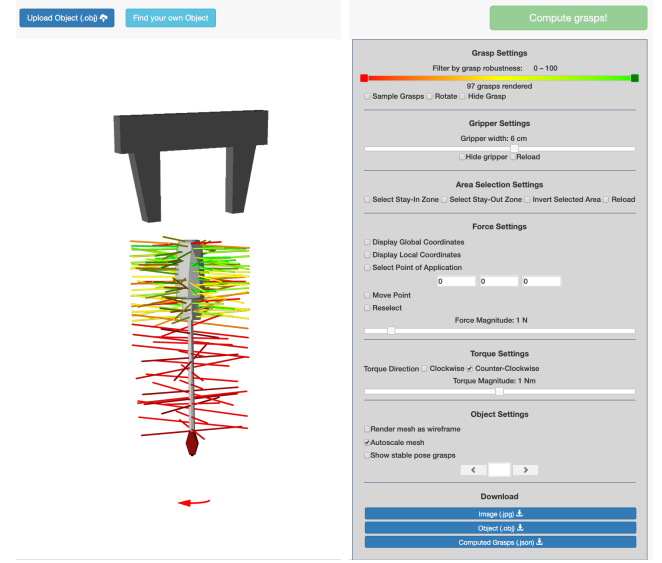


Fig. 2: Dex-Net as a Service - Task (DNaaS-Task) online user interface. The column on the right includes numerical inputs, including settings on grasp filter, gripper width, stay-out zone selection, point of application and magnitude for force, orientation and magnitude for torque. On the left is the object mesh where stay-out zones can be painted and wrenches indicated. Also, this is where resulting grasps are displayed after computation. The figure shows the most robust grasps (in green) for the screw driver for the applied screw driving task, where the tip of the screw driver is masked as the stay out zone and a 0.5 Nm clockwise torque is desired. The online interface is available at: <https://dex-net.app>

specified external wrench. For each grasp candidate, we generate n_p perturbations of the grasp parameters by adding Gaussian noise to the translation and rotation of the grasp center point and axis, respectively. We then analyze each of the grasp candidate perturbations to determine if it can resist the external task wrench. Mathematically,

$$\begin{aligned}
 r &= \frac{1}{n_p} \sum_{i=0}^{n_p} \mathbb{I}(\min_{\xi_i} \|G_i \xi_i - \tau_o\|_2 < \epsilon) \\
 \text{s.t.} \quad & \mathbf{A} \xi_i \leq \mathbf{h} \\
 & g_i = g + \mathcal{N}(\mathbf{0}, \Sigma) \\
 & g_i \notin \mathcal{M}_{\text{out}}
 \end{aligned}$$

where $G_i \in \mathbb{R}^{6 \times n}$ is the grasp matrix that transforms the contact wrenches $\xi_i \in \mathbb{R}^n$ corresponding to the perturbed grasp g_i into object frame, $\tau_o \in \mathbb{R}^6$ is the external wrench transformed into the object frame, $\epsilon > 0$ is a small positive number, $\mathcal{N}(\mathbf{0}, \Sigma)$ denotes the multivariate distribution from where the perturbation parameters are sampled, $\mathbf{A} \in \mathbb{R}^{p \times n}$ and $\mathbf{h} \in \mathbb{R}^p$ define linear constraints on the contact wrenches, and \mathcal{M}_{out} is the stay-out zone. We average r over the n_p perturbed grasps, as outlined in Algorithm 1.

C. Task-Based Grasping Web Interface

We build on Dex-Net as a Service (DNaaS) [8] to develop a cloud-based public API for computing robust task-based grasps with parallel-jaw grippers on user specified meshes. The API takes as input an object specified as a 3D triangular








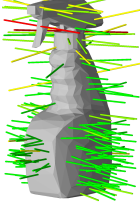
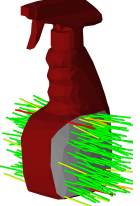





	No Task specified	Lift	Squeeze Trigger	Pack in Box	Open Nozzle	Open Bottle	Place on Shelf
Task Model							
Grasps							

Fig. 3: **Task-directed grasping for spray bottle.** The stay-out zones of the spray bottle object displayed on the second row model the tasks described on the first row, resulting filtered grasps showed on the third row.





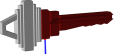
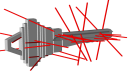
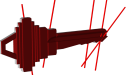
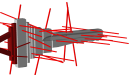

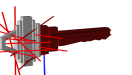
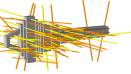
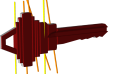
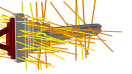
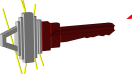
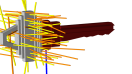
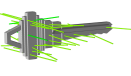
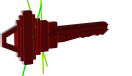
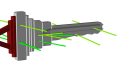
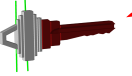
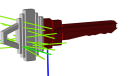
	No Task specified	Inspect top or bottom surface	Hang on hook	Turn in lock	Polish key teeth
Task Model					
$r \in [0.0, 0.25]$					
$r \in [0.25, 0.75]$					
$r \in (0.75, 1.0]$					

Fig. 4: **Standard door key with different task models and the resulting grasps.** The same tool mesh for a key with different task models produces different results in the grasp analysis. The labels on top row describe the task that the second row models. In the second row, the purple region marks the stay-out zone, the blue arrow shows the direction of force, and the red arrow shows the torque. The third row shows the sampled grasps with colors corresponding to the quality of grasps, and the next 3 rows show the same grasps split into 3 groups based on quality scores.

mesh, and user specified task-based parameters defined by graspable zone and external wrench and outputs a set of collision-free parallel-jaw grasps filtered by graspable zones, ranked by their robustness to perturbations in object pose, gripper pose, the Coulomb friction coefficient and resistance to the external wrench. The robust quasi-static grasp analysis engine of the API assumes quasi-static physics, a rigid object with uniform mass density, and a known friction coefficient. We assume that the input mesh has triangular faces and fewer than 70k total faces to ensure grasp computation latency remains under two minutes.

In this user interface, shown in Figure 2, the object’s mesh is shown in the center window along with a representation of a parallel-jaw gripper for testing grasp analysis. In this window, the user can hover over the mesh surface to mark stay-out zones, and generate the grasp analysis to preview the effect of the model.

The external wrench is further specified by clicking to select the point of application, force settings and torque settings. The force setting allows the user to indicate a desired force by directly drawing a force vector on the 3D mesh object. The torque settings allows the user to create a

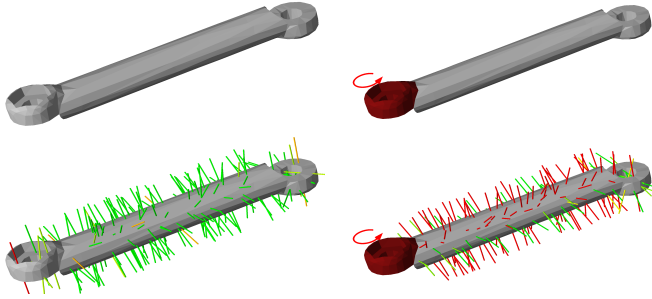


Fig. 5: **Task-directed grasping for wrench** (left) mesh representation on top and nominal grasps on bottom, (right) task representation with stay-out zone and external torque for rotating the wrench on top and resulting grasps on bottom. Note that the grasps along the direction and farther from the torque application point are preferred.

torque constraint on the object with specified rotation axis, direction, and magnitude.

D. System Architecture

The back-end for task-directed grasp evaluation comprises of three distinct layers of abstraction. The front-end of the system is a web-based graphical user interface based on jQuery that parses user mesh models and grasp computation requests from a web browser. The frontend uploads mesh models and makes requests for grasp computations via public grasping API [8]. Requests are forwarded to the robust grasp-analysis backend using a Python-based Flask API. The backend spawns worker processes which analyze the input mesh model using the robust grasp analysis engine. Each worker process returns a set of parallel-jaw grasps with robustness metrics. The grasps are retrieved from the worker by a monitor process on the API server, which relays the JSON encoded grasps to the frontend via HTTP. Finally, the frontend renders the grasps on the 3D object model in the browser.

The client-side user interface and server-side Flask API run on a quad-core Intel(R) Xeon(R) CPU E3-1220 v3 with a clockrate of 3.10GHz and 16GB of RAM. The website is written using HTML, JavaScript, and CSS served statically by an Apache web server. We use `three.js` to render a 360° 3D scene in the browser where candidate grasps are superimposed on the target object mesh. The page is designed using a flexible box layout for easy accessibility across modern web-browsers (Chrome, Safari, Firefox) and on mobile devices. The website uses the latest version of jQuery for DOM manipulation, event handling, and Promise-based asynchronous HTTP requests. The graphical user interface combines elements from jQuery UI, Bootstrap, and custom CSS.

V. EXPERIMENTS

A. Effect of Stay-Out Zone on Grasp Filtering

Figure 3 shows a number of examples that can be routinely performed with the spray bottle including lifting up, packing

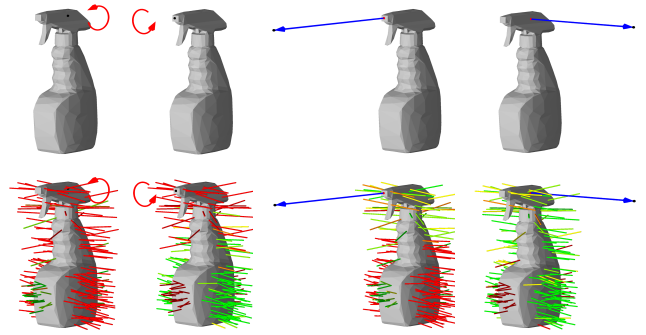


Fig. 6: **Effect of external wrench on task-directed grasping** Grasps along the direction of rotation and farther from the point of rotation are preferred on (left) and (left-middle); grasps along the direction of force and close to the point of application are preferred on (right-middle) and (right).



Fig. 7: **Multi-part assembly task.** In this assembly, parts need to be sequentially grasped and fit together.

into a box, nozzle opening, liquid re-filling, placing on shelf and so on to capture the effect of stay-out zones on filtering grasps.

B. Effect of External Wrench on Grasp Quality

We examine the effects of a task wrench with the spray bottle in Figure 6. We make two observations: 1) grasps along the direction of torque are preferred with farther grasps providing a higher moment arm to resist the wrench, 2) grasps along the direction of force are preferred with grasps close to the point of application providing a better support than farther away grasps.

C. Combined Effect of Stay-Out Zone and External Wrench

While the stay-out zones capture the form of the task, the task wrench is useful to capture the functionality associated with the task. In Figure 1 and Figure 5, we demonstrate the combined effect of stay-out zone and external force and torque respectively while restricting the grasps to lie within semantic task-specific regions.

D. Household Task

Figure 4 demonstrates a set of common tasks that can be performed with a key in household environments. Note that grasps that align with the force direction are ranked as robust grasps, but grasps that can not resist the task wrench

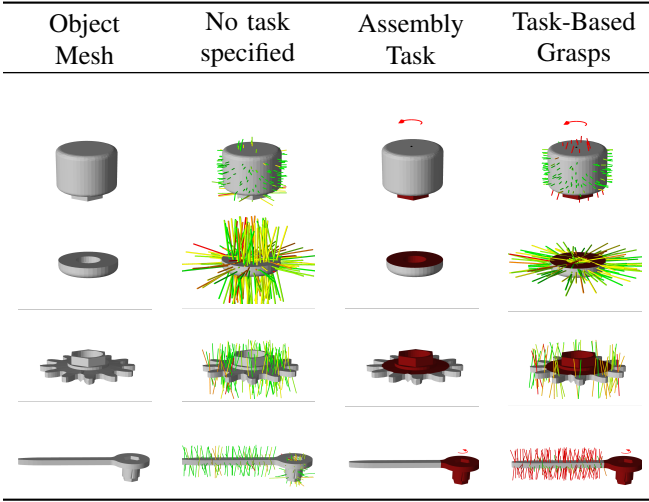


Fig. 8: **Assembly with different task models and the resulting grasp analysis.** Each row of this above figure show a single object needed for the multi-part assembly task. The first column shows the object mesh without the task-based grasp model. The second column shows the grasp analysis without the task-based grasp model. The third column shows the stay-out zones and wrenches needed for the assembly. The fourth column shows the grasp-analysis based on the model in the third column. With the task-based model, the robot is able to compute grasps that facilitate assembly.

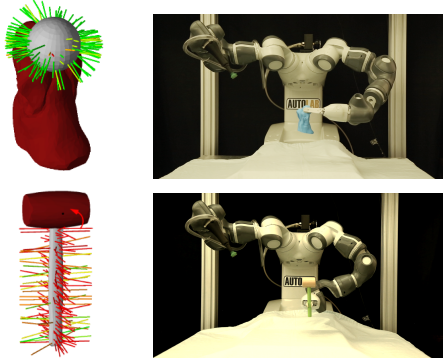


Fig. 9: Task-based grasping set-up with the yumi robot: (left) task-directed grasps on the pawn object (top) with stay-out zone, (right) pick-and-place trajectory following task with the yumi robot on the pawn object. (left) task-directed grasps on the mallet object (bottom) with torque constraints, (right) hammering task snapshot on the mallet object.

are ranked as less robust. Moreover, the grasps that can rotate the object in the specific direction are ranked as robust.

Figure 9 shows preliminary results of transferring the grasps on the yumi robot, where the robot grasps the pawn object from the top for following a pick-and-place task, and the mallet for hammering task with external torque constraints.

E. Assembly Task

Figure 7 shows a multi-part assembly and Figure 8 shows the grasps generated by DNaaS-Task for 4 objects in the assembly. The stay-out zones prevent the robot from interfering with assembly contacts and the wrenches define the assembly forces and torques, signifying its appeal to handle

Object	Spray Bottle	Tweezers	Wrench	Screwdriver	key	Overall
Number of faces	812	270	576	292	500	390
Average time(s)	7.41	6.74	7.73	7.03	7.25	7.232

Spray Bottle	Lift	Squeeze Trigger	Pack in Box	Place on Shelf	Overall
Number of faces in stay-out zone	640	774	770	747	732
Average time(s)	6.91	3.3	4.09	6.80	5.25

Fig. 10: **Computation time measurement.** The top table provides statistics for average computation time of five objects: a spray bottle, a tweezers, a wrench, a screwdriver and a key. The bottom table provides statistics for average computation time of spray bottle under various task constrains.

a wide variety of task constraints.

F. Computation Times

The top table in Figure 10 shows grasp computation times for 5 objects: spray bottle, a tweezers, a wrench, a screwdriver and a key. Computation time grows with object complexity (measured by the number of triangular faces in a mesh).

The bottom table in Figure 10 shows grasp computation time for a spray bottle under various task constrains presented in Figure 3 specifically represented by stay-out zone. There is an inverse relationship between computation time and the size of the stay-out zone (measured by the number of stay-out zone faces in a mesh).

VI. CONCLUSIONS

In this paper we present an intuitive task-based grasping interface and modification to the Dex-Net 1.0 grasp planning algorithm to compute robust grasps consistent with task constraints. The system computes robust task-based grasps for a given specification of a stay-out zone and/or an external wrench. To demonstrate the interface, analysis, and back-end, we present experimental results for a variety of objects and tasks in household and industrial environments. In future work, we would like to extend DNaaS-Task to suction-based and multi-finger grippers, and to multilateral manipulation with two or more grippers.

ACKNOWLEDGEMENTS

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, Berkeley Deep Drive (BDD), the Swarm Lab, the Real-Time Intelligent Secure Execution (RISE) Lab, and the CITRIS “People and Robots” (CPAR) Initiative. The work was supported in part by donations from Siemens. The authors would like to thank Pusong Li and Sophie Huang for their helpful contributions.

REFERENCES

- [1] M. J. Aein, E. E. Aksoy, and F. Wörgötter, “Library of actions: Implementing a generic robot execution framework by using manipulation action semantics,” *The International Journal of Robotics Research*, vol. 38, no. 8, pp. 910–934, 2019.
- [2] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, “Physical human interactive guidance: Identifying grasping principles from human-planned grasps,” *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910, 2012.

- [3] 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," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 202–216, 2020.
- [4] R. Haschke, J. J. Steil, I. Steuwer, and H. Ritter, "Task-oriented quality measures for dextrous grasping," in *2005 International Symposium on Computational Intelligence in Robotics and Automation*, 2005, pp. 689–694.
- [5] R. Holladay, T. Lozano-Pérez, and A. Rodriguez, "Force-and-motion constrained planning for tool use," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7409–7416.
- [6] M. Kokic, D. Kragic, and J. Bohg, "Learning task-oriented grasping from human activity datasets," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3352–3359, 2020.
- [7] M. Kokic, J. A. Stork, J. A. Hausstein, and D. Kragic, "Affordance detection for task-specific grasping using deep learning," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov. 2017, pp. 91–98.
- [8] P. Li, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Goldberg, "Dex-net as a service (dnaas): A cloud-based robust robot grasp planning system," in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, IEEE, 2018, pp. 1420–1427.
- [9] Z. Li and S. S. Sastry, "Task-oriented optimal grasping by multifingered robot hands," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 32–44, 1988.
- [10] 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 *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, 2016.
- [11] MakerBot Industries, LLC, *Thingiverse - Digital Designs for Physical Objects*, <https://www.thingiverse.com/>, 2020.
- [12] N. Mavrakis, M. Kopicki, R. Stolkin, A. Leonardis, et al., "Task-relevant grasp selection: A joint solution to planning grasps and manipulative motion trajectories," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 907–914.
- [13] N. Mavrakis, R. Stolkin, L. Baronti, M. Kopicki, M. Castellani, et al., "Analysis of the inertia and dynamics of grasped objects, for choosing optimal grasps to enable torque-efficient post-grasp manipulations," in *Int. Conf. on Humanoid Robots (Humanoids)*, IEEE, 2016, pp. 171–178.
- [14] V.-D. Nguyen, "Constructing force-closure grasps," *Int. Journal of Robotics Research (IJRR)*, vol. 7, no. 3, pp. 3–16, 1988.
- [15] V. Ortenzi, M. Controzzi, F. Cini, J. Leitner, M. Bianchi, M. A. Roa, and P. Corke, "Robotic manipulation and the role of the task in the metric of success," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 340–346, Aug. 2019.
- [16] T. Pardi, R. Stolkin, et al., "Choosing grasps to enable collision-free post-grasp manipulations," in *Int. Conf. on Humanoid Robots (Humanoids)*, IEEE, 2018, pp. 299–305.
- [17] A. ten Pas and R. P. Jr., "Localizing antipodal grasps in point clouds," *CoRR*, vol. abs/1501.03100, 2015. arXiv: 1501.03100.
- [18] M. Prats, P. J. Sanz, and A. P. del Pobil, "Task-oriented grasping using hand preshapes and task frames," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1794–1799.
- [19] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese, *Keto: Learning keypoint representations for tool manipulation*, 2019. arXiv: 1910.11977 [cs.LG].
- [20] E. Rimon and J. Burdick, *The Mechanics of Robot Grasping*. Cambridge University Press, 2019.
- [21] D. Song, C. H. Ek, K. Huebner, and D. Kragic, "Task-based robot grasp planning using probabilistic inference," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 546–561, 2015.
- [22] J. Xu, A. Bhardwaj, G. Sun, T. Aykut, N. Alt, M. Karimi, and E. Steinbach, "Learning-based modular task-oriented grasp stability assessment," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3468–3475.
- [23] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.
- [24] S. Zimmermann, G. Hakimifard, M. Zamora, R. Poranne, and S. Coros, "A multi-level optimization framework for simultaneous grasping and motion planning," *IEEE Robotics & Automation Letters*, vol. 5, no. 2, pp. 2966–2972, 2020.