Where2Act: From Pixels to Actions for Articulated 3D Objects

Kaichun Mo*[1] Leonidas Guibas[1] Mustafa Mukadam[2] Abhinav Gupta[2] Shubham Tulsiani[2]
[1]Stanford University [2]Facebook AI Research
https://cs.stanford.edu/~kaichun/where2act

Abstract

One of the fundamental goals of visual perception is to allow agents to meaningfully interact with their environment. In this paper, we take a step towards that long-term goal – we extract highly localized actionable information related to elementary actions such as pushing or pulling for articulated objects with movable parts. For example, given a drawer, our network predicts that applying a pulling force on the handle opens the drawer. We propose, discuss, and evaluate novel network architectures that given image and depth data, predict the set of actions possible at each pixel, and the regions over articulated parts that are likely to move under the force. We propose a learning-from-interaction framework with an online data sampling strategy that allows us to train the network in simulation (SAPIEN) and generalizes across categories. Check the website for code and data release.

1. Introduction

We humans interact with a plethora of objects around us in our daily lives. What makes this possible is our effortless understanding of *what* can be done with each object, *where* this interaction may occur, and precisely *how* our we must move to accomplish it – we can pull on a handle to open a drawer, push anywhere on a door to close it, flip a switch to turn a light on, or push a button to start the microwave. Not only do we understand what actions will be successful, we also intuitively know which ones will not *e.g.* pulling out a remote's button is probably not a good idea! In this work, our goal is to build a perception system which also has a similar understanding of general objects *i.e.* given a novel object, we want a system that can infer the myriad possible interactions[1] that one can perform with it.

The task of predicting possible interactions with objects is one of central importance in both, the robotics and the computer vision communities. In robotics, the ability to predict feasible and desirable actions (*e.g.* a drawer can be pulled out) can help in motion planning, efficient exploration and interactive learning (sampling successful trials faster). On the other hand, the computer vision community has largely focused on inferring semantic labels (*e.g.* part segmentation, keypoint estimation) from visual input, but such passively learned representations provide limited understanding. More specifically, passive learning falls short on the ability of agents to perform actions, learn prediction models (forward dynamics) or even semantics in many cases (categories are more than often defined on affordances themselves!). Our paper takes a step forward in building a common perception system across diverse objects, while creating its own supervision about what actions maybe successful by actively interacting with the objects.

The first question we must tackle is how one can parametrize the predicted action space. We note that any

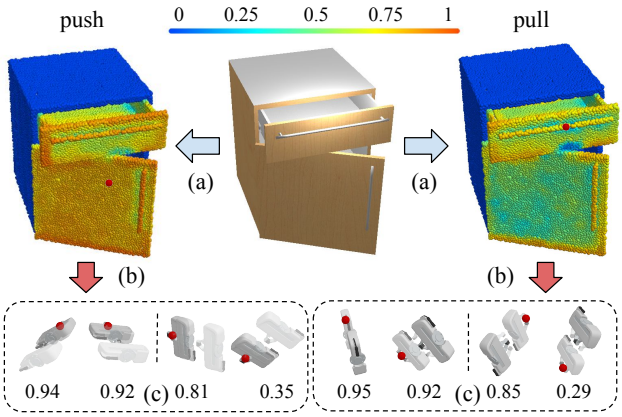


Figure 1. **The Proposed Where2Act Task.** Given as input an articulated 3D object, we learn to propose the actionable information for different robotic manipulation primitives (*e.g. pushing*, *pulling*): (a) the predicted actionability scores over pixels; (b) the proposed interaction trajectories, along with (c) their success likelihoods, for a selected pixel highlighted in red. We show two high-rated proposals (left) and two with lower scores (right) due to interaction orientations and potential robot-object collisions.

*The majority of the work was done while Kaichun Mo was a research intern at Facebook AI Research.

[1]Gibson proposed the idea of affordances – opportunities of interaction. Classical notion of object affordance involves consideration of agent's morphology. Our interactions are more low-level actions.

long-term interaction with an object can be considered as a sequence of short-term 'atomic' interactions like pushing and pulling. We therefore limit our work to considering the plausible short-term interactions that an agent can perform given the current state of the object. Each such atomic interaction can further be decomposed into *where* and *how* *e.g.* where on the cabinet should the robot pull (*e.g.* drawer handle or drawer surface) and how should the motion be executed (*e.g.* pull parallel or perpendicular to handle). This observation allows us to formulate our task as one of dense visual prediction. Given a depth or color image of an object, we learn to infer for each pixel/point, whether a certain primitive action can be performed at that location, and if so, how it should be executed.

Concretely, as we illustrate in Figure 1 (a), we learn a prediction network that given an atomic action type, can predict for each pixel: a) an 'actionability' score, b) action proposals, and c) success likelihoods. Our approach allows an agent to learn these by simply interacting with various objects, and recording the outcomes of its actions – labeling ones that cause a desirable state change as successful. While randomly interacting can eventually allow an agent to learn, we observe that it is not a very efficient exploration strategy. We therefore propose an on-policy data sampling strategy to alleviate this issue – by biasing the sampling towards actions the agents thinks are likely to succeed.

We use the SAPIEN [45] simulator for learning and testing our approach for six types of primitive interaction, covering 972 shapes over 15 commonly seen indoor object categories. We empirically show that our method successfully learns to predict possible actions for novel objects, and does so even for previously unseen categories.

In summary, our contributions are:

- we formulate the task of inferring affordances for manipulating 3D articulated objects by predicting per-pixel action likelihoods and proposals;
- we propose an approach that can learn from interactions while using adaptive sampling to obtain more informative samples;
- we create benchmarking environments in SAPIEN, and show that our network learns actionable visual representations that generalize to novel shapes and even unseen object categories.

2. Related Works

Predicting Semantic Representations. To successfully interact with a 3D object, an agent must be able to 'understand' it given some perceptual input. Several previous works in the computer vision community have pursued such an understanding in the form of myriad semantic labels. For example, predicting category labels [44, 3], or more fine-grained output such as semantic keypoints [6, 52] or part segmentations [51, 24] can arguably yield more actionable representations *e.g.* allowing one to infer where 'handles', or 'buttons' *etc.* are. However, merely obtaining such semantic labels is clearly not sufficient on its own – an agent must also understand *what* needs to be done (*e.g.* an handle can be 'pulled' to open a door), and *how* that action should be accomplished *i.e.* what precise movements are required to 'pull open' the specific object considered.

Inferring Geometric and Physical Properties. Towards obtaining information more directly useful for *how* to act, some methods aim for representations that can be leveraged by classical robotics techniques. In particular, given geometric representations such as the shape [3, 22, 45, 47], alongwith the rigid object pose [46, 40, 39, 41, 4], articulated part pose [12, 50, 42, 49, 18, 45, 15] pose, or shape functional semantics [17, 13, 14], one can leverage off-the-shelf planners [23] or prediction systems [21] developed in the robotics community to obtain action trajectories. Additionally, the ability to infer physical properties *e.g.* material [36, 19], mass [36, 37] *etc.* can further make this process accurate. However, this two-stage procedure for acting, involving a perception system that predicts the object 'state', is not robust to prediction errors and makes the perception system produce richer output than possibly needed *e.g.* we don't need the full object state to pull out a drawer. Moreover, while this approach allows an agent to precisely execute an action, it sidesteps the issue of what action needs to/can be performed in the first place *e.g.* how does the agent understand a button can be pushed?

Learning Affordances from Passive Observations. One interesting approach to allow agents to learn what actions can be taken in a given context is to leverage (passive) observations – one can watch videos of other agents interacting with an object/scene and learn what is possible to do. This technique has been successfully used to learn scene affordances (sitting/standing) [8], possible contact locations [2], interaction hotspots [26], or even grasp patterns [10]. However, learning from passive observations is challenging due to several reasons *e.g.* the learning agent may differ in anatomy thereby requiring appropriate retargeting of demonstrations. An even more fundamental concern is the distribution shift common in imitation learning – while the agent may see examples of what can be done, it may not have seen sufficient negative examples or even sufficiently varied positive ones.

Learning Perception by Interaction. Most closely related to our approach is the line of work where an agent learns to predict affordances by generating its own training data – by interacting with the world and trying out possible actions. One important task where this approach has led to impressive results is that of planar grasping [30, 16], where the agent can learn which grasp actions would be successful. While subsequent approaches have attempted to apply these

ideas to other tasks like object segmentation [29, 20], planar pushing [31, 53], or non-planar grasps [25], these systems are limited in the complexity of the actions they model. In parallel, while some methods have striven for learning more complex affordances, they do so without modeling for the low-level actions required and instead frame the task as classification with oracle manipulators [27]. In our work, driven by availability of scalable simulation with diverse objects, we tackle the task of predicting affordances for richer interactions while also learning the low-level actions that induce the desired change.

3. Problem Statement

We formulate a new challenging problem **Where2Act** – inferring per-pixel 'actionable information' for manipulating 3D articulated objects. As illustrated in Fig. 1, given a 3D shape S with articulated parts (*e.g.* the drawer and door on the cabinet), we perform per-pixel predictions for (a) *where* to interact, (b) *how* to interact, and (c) the interaction outcomes, under different action primitives.

In our framework, the input shape can be represented as a 2D RGB image or a 3D partial point cloud scan. We parametrize six types of short-term primitive actions (*e.g.* pushing, pulling) by the robot grippper pose in the $SE(3)$ space and consider an interaction successful if it interacts with the intended contact point on object validly and causes part motion to a considerable amount.

With respect to every action primitive, we predict for each pixel/point p over the visible articulated parts of a 3D shape S the following: (a) an actionability score a_p measuring how likely the pixel p is actionable; (b) a set of interaction proposals $\{R_{z|p} \in SO(3)\}_z$ to interact with the point p, where z is randomly drawn from a uniform Gaussian distribution; (c) one success likelihood score $s_{R|p}$ for each action proposal R.

4. Method

We propose a learning-from-interaction approach to tackle this task. Taking as input a single RGB image or a partial 3D point cloud, we employ an encoder-decoder backbone to extract per-pixel features and design three decoding branches to predict the 'actionable information'.

4.1. Network Modules

Fig. 2 presents an overview of the proposed method. Our pipeline has four main components: a backbone feature extractor, an actionability scoring module, an action proposal module, and an action scoring module. We train an individual network for each primitive action.

Backbone Feature Extractor. We extract dense per-pixel features $\{f_p\}_p$ over the articulated parts. In the real-world robotic manipulation both RGB cameras or RGB-D scanners are used. Therefore, we evaluate both settings. For the 2D case, we use the UNet architecture [35] and implementation [48] with a ResNet-18 [11] encoder, pretrained on ImageNet [5], and a symmetric decoder, trained from scratch, equipped with dense skip links between the encoder and decoder. For the 3D experiments, we use PointNet++ segmentation network [32] and implementation [43] with 4 set abstraction layers with single-scale grouping for the encoder and 4 feature propagation layers for the decoder. In both cases, we finally produce per-pixel feature $f_p \in \mathbb{R}^{128}$.

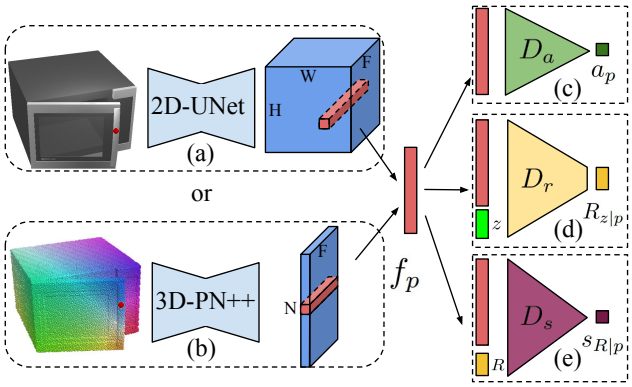


Figure 2. **Network Architecture.** Our network takes an 2D image or a 3D partial scan as input and extract per-pixel feature f_p using (a) Unet [35] for 2D images and (b) PointNet++ [32] for 3D point clouds. To decode the per-pixel actionable information, we propose three decoding heads: (c) an actionability scoring module D_a that predicts a score $a_p \in [0,1]$; (d) an action proposal module D_r that proposes multiple gripper orientations $R_{z|p} \in SO(3)$ sampled from a uniform Gaussian random noise z; (e) an action scoring module D_s that rates the confidence $s_{R|p} \in [0,1]$ for each proposal.

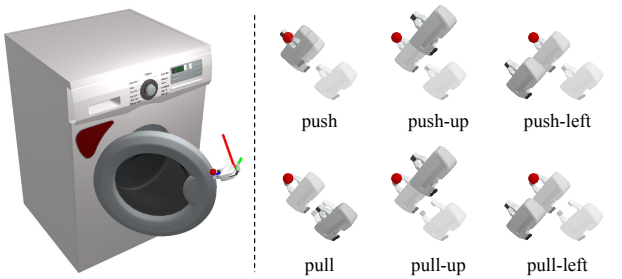
Actionability Scoring Module. For each pixel p, we predict an actionability score $a_p \in [0,1]$ indicating how likely the pixel is actionable. We employ a Multilayer Perceptron (MLP) D_a with one hidden layer of size 128 to implement this module. The network outputs one scalar a_p after applying the Sigmoid function, where a higher score indicates a higher chance for successful interaction. Namely,

$$a_p = D_a(f_p) \tag{1}$$

Action Proposal Module. For each pixel p, we employ an action proposal module that is essentially formulated as a conditional generative model to propose high-recall interaction parameters $\{R_{z|p}\}_z$. We employ another MLP D_r with one hidden layer of size 128 to implement this module. Taking as input the current pixel feature f_p and a randomly sampled Gaussian noise vector $z \in \mathbb{R}^{10}$, the network D_p predicts a gripper end-effector 3-DoF orientation $R_{z|p}$ in the $SO(3)$ space

$$R_{z|p} = D_r(f_p, z). \tag{2}$$

We represent the 3-DoF gripper orientation by the first two orthonormal axes in the 3×3 rotation matrix, following the proposed 6D-rotation representation in [54].



(a) (b)

Figure 3. (a) Our interactive simulation environment: we show the local gripper frame by the red, green and blue axes, which corresponds to the leftward, upward and forward directions respectively; (b) Six types of action primitives parametrized in the $SE(3)$ space: we visualize each pre-programmed motion trajectory by showing the three key frames, where the time steps go from the transparent grippers to the solid ones, with $3\times$ exaggerated motion ranges.

Action Scoring Module. For an action proposal R at pixel p, we finally estimate a likelihood $s_{R|p} \in [0,1]$ for the success of the interaction parametrized by tuple $(p,R) \in SE(3)$. One can use the predicted action scores to filter out low-rated proposals, or sort all the candidates according to the predicted scores, analogous to predicting confident scores for bounding box proposals in the object detection literature.

This network module D_s is also parametrized by an MLP with one hidden layer of size 128. Given an input tuple (f_p,R), we produce a scalar $s_{R|p} \in [0,1]$,

$$s_{R|p} = D_s(f_p,R), \tag{3}$$

where $s_{R|p} > 0.5$ indicates a positive action proposal R during the testing time.

4.2. Collecting Training Data

It is extremely difficult to collect human annotations for the predictions that we are pursuing. Instead, we propose to let the agent learn by interacting with objects in simulation. As illustrated in Fig. 3 (a), we create an interactive environment using SAPIEN [45] where a random 3D articulated object is selected and placed at the center of the scene. A flying robot gripper can then interact with the object by specifying a position $p \in \mathbb{R}^3$ over the shape geometry surface with an end-effector orientation $R \in SO(3)$. We consider six types of action primitives (Fig. 3 (b)) with pre-programmed interaction trajectories, each of which is parameterized by the gripper pose $(p,R) \in SE(3)$ at the beginning.

We employ a hybrid data sampling strategy where we first sample large amount of offline random interaction trajectories to bootstrap the learning and then adaptively sample online interaction data points based on the network predictions for more efficient learning.

Offline Random Data Sampling. We sample most of the training data in an offline fashion as we can efficiently sample several data points by parallelizing simulation environments across multiple CPUs. For each data point, we first randomly sample a position p over the ground-truth articulated parts to interact with. Then, we randomly sample an interaction orientation $R \in SO(3)$ from the hemisphere above the tangent plane around p, oriented consistently to the positive normal direction, and try to query the outcome of the interaction parametrized by (p,R). We mark orientation Rs from the other hemisphere as negative without trials since the gripper cannot be put inside the object volume.

In our experiments, for each primitive action type, we sample enough offline data points that give roughly $10,000$ positive trajectories to bootstrap the training. Though parallelization allows large scale offline data collection, such random data sampling strategy is highly inefficient in querying the interesting interaction regions to obtain positive data points. Statistics show that only 1% data samples are positive for the *pulling* primitive. This renders a big data imbalance challenge in training the network and also hints that the most likely pullable regions occupy really small regions, which is practically very reasonable since we most likely pull out doors/drawers by their handles.

Online Adaptive Data Sampling. To address the sampling-inefficiency of offline random data sampling, we propose to conduct online adaptive data sampling that samples more over the subregions that the network that we are learning predicts to be highly possible to be successful.

In our implementation, during training the network for the action scoring module D_s with data sample (p,R), we infer the action score predictions $\{s_{R|p_i}\}_i$ over all pixels $\{p_i\}_i$ on articulated parts. Then, we sample one position p_* to conduct an additional interaction trial (p_*,R) according to the SoftMax normalized probability distribution over all possible interaction positions. By performing such online adaptive data sampling, we witness an increasingly growing positive data sample rate since the network is actively choosing to sample more around the likely successful subregions. Also, we observe that sampling more data around the interesting regions helps network learn better features at distinguishing the geometric subtleties around the small but crucial interactive parts, such as handles, buttons and knobs.

While this online data sampling is beneficial, it may lead to insufficient exploration of novel regions. Thus, in our final online data sampling procedure, we sample 50% of data trajectories from the random data sampling and sample the other 50% from prediction-biased adaptive data sampling.

4.3. Training and Losses

We empirically find it beneficial to first train the action scoring module D_s and then train the three decoders jointly. We maintain separate data queues for feeding same amount of positive and negative interaction data in each training batch to address the data imbalance issue. We also balance sampling shapes from different object categories equally.

Action Scoring Loss. Given a batch of B interaction data points $\{(S_i, p_i, R_i, r_i)\}_i$ where $r_i = 1$ (positive) and $r_i = 0$ (negative) denote the ground-truth interaction outcome, we train the action scoring module D_s with the standard binary cross entropy loss

$$\mathscr{L}_s = -\frac{1}{B}\sum_i r_i \log\left(D_s(f_{p_i|S_i}, R_i)\right) + (1-r_i)\log\left(1 - D_s(f_{p_i|S_i}, R_i)\right). \quad (4)$$

Action Proposal Loss. We leverage the Min-of-N strategy [7] to train the action proposal module D_r, which is essentially a conditional generative model that maps a pixel p to a distribution of possible interaction proposals $R_{z|p}$'s. For each positive interaction data, we train D_r to be able to propose one candidate that matches the ground-truth interaction orientation. Concretely, for a batch of B interaction data points $\{(S_i, p_i, R_i, r_i)\}_i$ where $r_i = 1$, the Min-of-N loss is defined as

$$\mathscr{L}_r = \frac{1}{B}\sum_i \min_{j=1,\cdots,100} dist\left(\left(D_r\left(f_{p_i|S_i}; z_j\right)\right), R_i\right), \quad (5)$$

where z_j is *i.i.d* randomly sampled Gaussian vectors and *dist* denotes a distance function between two 6D-rotation representations, as defined in [54].

Actionability Scoring Loss. We define the 'actionability' score corresponding to a pixel as the expected success rate when executing a random proposal generated by our proposal generation module D_r. While one could estimate this by actually executing these proposals, we note that our learned action scoring module D_s allows us to directly evaluate this. We train our 'actionability' scoring module to learn this expected score across proposals from D_r, namely,

$$\hat{a}_{p_i|S_i} = \frac{1}{100}\sum_{j=1,\cdots,100} D_s\left(f_{p_i|S_i}, D_r\left(f_{p_i|S_i}, z_j\right)\right); \quad \mathscr{L}_a = \frac{1}{B}\sum_i \left(D_a(f_{p_i|S_i}) - \hat{a}_{p_i|S_i}\right)^2. \quad (6)$$

This strategy is computationally efficient since we are reusing the 100 proposals computed in Eq. 5. Also, since the action proposal network D_r is optimized to cover all successful interaction orientations, the estimation $\hat{a}_{p_i|S_i}$ is expected to be approaching 1 when most of the proposals are successful and 0 when the position p is not actionable (*i.e.* all proposals are rated with low success likelihood scores).

Final Loss. After adjusting the relative loss scales to the same level, we obtain the final objective function

$$\mathscr{L} = \mathscr{L}_s + \mathscr{L}_r + 100 \times \mathscr{L}_a. \quad (7)$$

5. Experiments

We set up an interactive simulation environment in SAPIEN [45] and benchmark performance of the proposed method both qualititively and quantitatively. Results also show that the networks learn representations that can generalize to novel unseen object categories and real-world data.

5.1. Framework and Settings

We describe our simulation environment, simulation assets and action primitive settings in details below.

Environment. Equipped with a large-scale PartNet-Mobility dataset, SAPIEN [45] provides a physics-rich simulation environment that supports robot actuators interacting with 2,346 3D CAD models from 46 object categories. Every articulated 3D object is annotated with articulated parts of interests (*e.g.* doors, handles, buttons) and their part motion information (*i.e.* motion types, motion axes and motion ranges). SAPIEN integrates one of the state-of-the-art physical simulation engines NVIDIA PhysX [28] to simulate physics-rich interaction details.

We adapt SAPIEN to set up our interactive environment for our task. For each interaction simulation, we first randomly select one articulated 3D object, which is zero-centered and normalized within a unit-sphere, and place it in the scene. We initialize the starting pose for each articulated part, with a 50% chance at its rest state (*e.g.* a fully closed drawer) and 50% chance with a random pose (*e.g.* a half-opened drawer). Then, we use a Franka Panda Flying gripper with 2 fingers as the robot actuator, which has 8 degree-of-freedom (DoF) in total, including the 3 DoF position, 3 DoF orientation and 2 DoF for the 2 fingers. The flying gripper can be initialized at any position and orientation with a closed or open gripper. We observe the object in the scene from an RGB-D camera with known intrinsics that is mounted 5-unit far from the object, facing the object center, located at the upper hemisphere of the object with a random azimuth $[0°, 360°)$ and a random altitude $[30°, 60°]$. Fig. 3 (a) visualizes one example of our simulation environment.

Simulation Assets. We conduct our experiments using 15 selected object categories in the PartNet-Mobility dataset, after removing the objects that are either too small (*e.g.* pens, USB drives), requiring multi-gripper collaboration (*e.g.* pliers, scissors), or not making sense for robot to manipulate (*e.g.* keyboards, fans, clocks). We use 10 categories for training and reserve the rest 5 categories only for testing, in order to analyze if the learned representations can generalize to novel unseen categories. In total, there are 773 objects in the training categories and 199 objects in the testing ones. We further divide the training split into 586 training shapes and 187 testing shapes, and only use the training shapes from the training categories to train our networks. Table 1 summarizes the detailed statistics of the final data splits.

Train-Cats	All	Box	Door	Faucet	Kettle	Microwave
Train-Data	586	20	23	65	22	9
Test-Data	187	8	12	19	7	3

	Fridge	Cabinet	Switch	TrashCan	Window
	32	270	53	52	40
	11	75	17	17	18

Test-Cats	All	Bucket	Pot	Safe	Table	Washing
Test-Data	199	36	23	29	95	16

Table 1. We summarize the shape counts in our dataset. Here, *pot* and *washing* are short for kitchen pot and washing machine.

Action Settings. We consider six types of primitive actions: *pushing*, *pushing-up*, *pushing-left*, *pulling*, *pulling-up*, *pulling-left*. All action primitives are pre-programmed with hard-coded motion trajectories and parameterized by the gripper starting pose $R \in SE(3)$ in the camera space. At the beginning of each interaction simulation, we initialize the robot gripper slightly above a surface position p of interest approaching from orientation R.

We visualize the action primitives in Fig. 3 (b). For *pushing*, a closed gripper first touches the surface and then pushes 0.05 unit-length forward. For *pushing-up* and *pushing-left*, the closed gripper moves forward by 0.04 unit-length to contact the surface and scratches the surface to the up or left direction for 0.05 unit-length. For *pulling*, an open gripper approaches the surface by moving forward for 0.04 unit-length, performs grasping by closing the gripper, and pulls backward for 0.05 unit-length. For *pulling-up* and *pulling-left*, after the attempted grasping, the gripper moves along the up or left direction for 0.05 unit-length. Notice that the *pulling* actions may degrade to the *pushing* ones if the gripper grasps nothing but just touches/scratches the surface.

We define one interaction trial successful if the part that we are interacting with exhibits a considerable part motion along the intended direction. The intended direction is the forward or backward direction for *pushing* and *pulling*, and is the up or left direction for the rest four directional action types. We measure the contact point motion direction and validate it if the angle between the intended direction and the actual motion direction is smaller than 60°. For thresholding the part motion magnitude, we measure the gap between the starting and end part 1-DoF pose and claim it successful if the gap is greater than 0.01 unit-length or 0.5 relative to the total motion range of the articulated part.

5.2. Metrics and Baselines

We propose two quantitative metrics for evaluating performance of our proposed method, compared with three baseline methods and one ablated version of our method.

Evaluation Metrics. A natural set of metrics is to evaluate the binary classification accuracy of the action scoring network D_s. We conduct random interaction simulation trials in the SAPIEN environment over testing shapes with random camera viewpoints, interaction positions and orientations. With random interactions, there are many more failed interaction trials than the successful ones. Thus, we report the F-score balancing precision and recall for the positive class.

To evaluate the action proposal quality, we introduce a sample-success-rate metric *ssr* that measures what fraction of interaction trials proposed by the networks are successful. This metric jointly evaluates all the three network modules and mimics the final use case of proposing meaningful actions when a robot actuator wants to operate the object. Given an input image or partial point cloud, we first use the actionability scoring module D_a to sample a pixel to interact, then apply the action proposal module D_r to generate several interaction proposals, and finally sample one interaction orientation according to the ratings from the action scoring module D_s . For both sampling operations, we normalize the predicted scores over all pixels or all action proposals as a probabilistic distribution and sample among the ones with absolute probability greater than 0.5. For the proposal generation step, we sample 100 action proposals per pixel by randomly sampling the inputs to D_r from a uniform Gaussian distribution. For each sampled interaction proposal, we apply it in the simulator and observe the ground-truth outcome. We define the final measure as below.

$$ssr = \frac{\text{\# successful proposals}}{\text{\# total proposals}} \tag{8}$$

Baselines and Ablation Study. Since we are the first to propose and formulate the task, there is no previous work for us to compare with. To validate the effectiveness of the proposed method and provide benchmarks for the proposed task, we compare to three baseline methods and one ablated version of our method:

- **B-Random**: a random agent that always gives a random proposal or scoring;
- **B-Normal**: a method that replaces the feature f_p in our method with the 3-dimensional ground-truth normal, with the same decoding heads, losses and training scheme as our proposed method;
- **B-PCPNet**: a method that replaces the feature f_p in our method with predicted normals and curvatures, which are estimated using PCPNet [9] on 3D partial point cloud inputs, with the same decoding heads, losses and training scheme as our proposed method;
- **Ours w/o OS**: an ablated version of our method that removes the online adaptive data sampling strategy and only samples online data with random exploration. We make sure that the total number of interaction queries is the same as our method for a fair comparison.

Among baseline methods, **B-Random** presents lower bound references for the proposed metrics, while **B-Normal** is designed to validate that our network learns localized but interaction-oriented features, rather than simple geometric features such as normal directions. **B-PCPNet** further

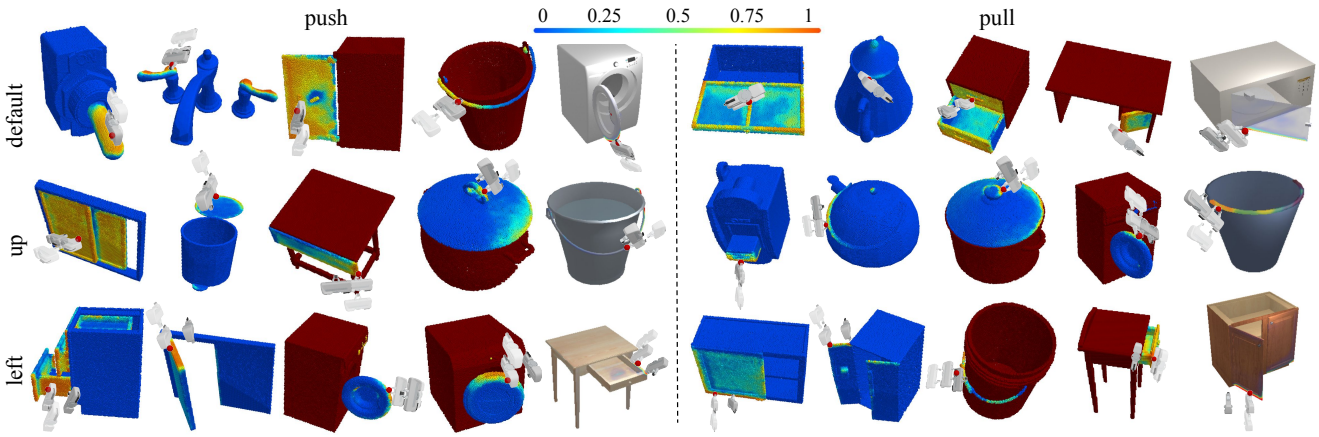


Figure 4. We visualize the per-pixel action scoring predictions over the articulated parts given certain gripper orientations for interaction. In each set of results, the left two shapes shown in blue are testing shapes from training categories, while the middle two shapes highlighted in dark red are shapes from testing categories. The rightmost columns show the results for the 2D experiments.

		F-score (%)	Sample-Succ (%)
pushing	B-Random	12.02 / 7.40	6.80 / 3.79
	B-Normal	31.94 / 17.39	21.72 / 11.57
	B-PCPNet	32.01 / 18.21	18.04 / 9.15
	2D-ours	34.21 / 22.68	21.36 / 10.58
	3D-ours	**43.76 / 26.61**	**28.54 / 14.74**
pushing-up	B-Random	4.92 / 3.31	2.70 / 1.62
	B-Normal	13.37 / 7.56	8.93 / 4.81
	B-PCPNet	15.08 / 7.50	8.09 / 4.86
	2D-ours	15.35 / 8.69	8.70 / 5.76
	3D-ours	**21.64 / 11.20**	**12.06 / 6.56**
pushing-left	B-Random	6.18 / 4.05	3.08 / 2.26
	B-Normal	18.52 / 10.72	11.59 / 5.72
	B-PCPNet	18.66 / 10.81	9.69 / 4.43
	2D-ours	18.93 / 12.04	11.68 / 7.22
	3D-ours	**26.04 / 16.06**	**15.95 / 9.31**
pulling	B-Random	2.26 / 3.19	1.07 / 1.55
	B-Normal	6.20 / 8.02	3.79 / 4.18
	B-PCPNet	7.19 / 8.57	4.15 / 3.71
	2D-ours	7.04 / 8.98	4.07 / 4.70
	3D-ours	**10.95 / 12.88**	**7.51 / 7.85**
pulling-up	B-Random	5.01 / 4.13	2.22 / 2.41
	B-Normal	13.64 / 9.40	8.67 / 6.08
	B-PCPNet	14.73 / 10.98	8.37 / 6.19
	2D-ours	15.74 / 12.88	9.71 / 7.10
	3D-ours	**22.24 / 16.28**	**13.53 / 9.28**
pulling-left	B-Random	5.83 / 4.16	3.06 / 2.31
	B-Normal	17.52 / 10.51	11.14 / 5.82
	B-PCPNet	18.89 / 11.00	9.12 / 5.19
	2D-ours	16.20 / 10.16	10.15 / 6.05
	3D-ours	**25.22 / 14.49**	**14.25 / 7.10**

Table 2. **Quantitative Evaluations and Comparisons.** We compare our method to three baseline methods (*i.e.* **B-Random**, **B-Normal** and **B-PCPNet**). In each entry, we report the numbers evaluated over the testing shapes from training categories (before slash) and the shapes from the test categories (after slash). We use **3D-** and **2D-** to indicate the data input modalities. The baseline methods are not sensitive to the input kinds. We observe that **3D-ours** achieves the best performance.

validates that our network learns geometric features more than local normals and curvatures. An ablated version **Ours w/o OS** further proves the improvement provided by the proposed *online adaptive data sampling* (OS) strategy.

		F-score (%)	Sample-Succ (%)
pushing	Ours w/o OS	40.54 / 25.66	25.18 / 11.76
	Ours	**43.76 / 26.61**	**28.54 / 14.74**
pushing-up	Ours w/o OS	21.03 / **11.57**	**12.88** / 6.43
	Ours	**21.64** / 11.20	12.06 / **6.56**
pushing-left	Ours w/o OS	24.71 / 14.91	14.12 / 7.02
	Ours	**26.04 / 16.06**	**15.95 / 9.31**
pulling	Ours w/o OS	10.28 / 12.09	5.62 / 5.87
	Ours	**10.95 / 12.88**	**7.51 / 7.85**
pulling-up	Ours w/o OS	20.51 / 13.70	12.18 / 7.96
	Ours	**22.24 / 16.28**	**13.53 / 9.28**
pulling-left	Ours w/o OS	23.41 / **15.07**	14.23 / 6.81
	Ours	**25.22** / 14.49	**14.25 / 7.10**

Table 3. **Ablation Study.** We compare our method to an ablated version, where we remove the online adaptive sampling. It is clear to see that using *online data sampling* (OS) helps in most cases.

5.3. Results and Analysis

Table 2 presents quantitative comparisons of our method to the three baselines, where we observe that **3D-Ours** performs the best. Our network learns localized but interaction-oriented geometric features, performing better than **B-Normal** and **B-PCPNet** which only use normals and curvatures as features. Though lacking of explicit 3D information and thus performing worse than the 3D networks, we observe competitive results from the 2D-version **2D-Ours**. Our networks also learn representations that generalize successfully to unseen novel object categories. The ablation study shown in Table 3 further validates that the *online data sampling* (OS) strategy helps boost the performance.

We visualize the predicted action scores in Fig. 4, where we clearly see that given different primitive action types and gripper orientations, our network learns to extract geometric features that are action-specific and gripper-aware. For example, for *pulling*, we predict higher scores over high-curvature regions such as part boundaries and handles, while for *pushing*, almost all flat surface pixels belonging to a pushable part

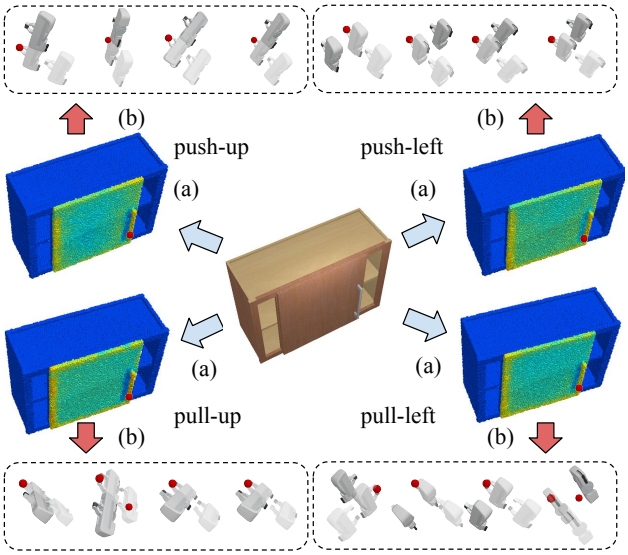


Figure 5. We visualize (a) the actionability scoring and (b) the action proposal predictions on an example cabinet with a door that can be slipped to open. We show the top-4 rated proposals.

are equally highlighted and the pixels around handles are reasonably predicted to be not pushable due to object-gripper collisions. For the directional interaction types, it is obvious to see that the action direction is of important consideration to the predictions. For instance, the *pushing-left* agent learns to scratch the side surface pixels of the cabinet drawers to close them (third-row, the leftmost column) and the *pulling-up* one learns to lift up the handle of a bucket by grasping it and pulling up (second-row, the rightmost column).

We illustrate the estimated actionability scores over the articulated part for the six action primitives in Fig. 1 and Fig. 5. We obverse that the door/drawer handles and part boundaries are highlighted, especially for *pulling* and *pulling-up*, where reasonable interaction proposals are produced. Fig. 1 clearly shows the different actionability predictions over the door pixels, where the door surface pixels are in general pushable, while only the handle part is pullable. Fig. 5 presents comparisons among the four directional interaction types. We observe similar actionability predictions for *pushing-up* and *pushing-left* but different orientation proposals for interacting with the same pixel. Interestingly, comparing *pulling-up* and *pulling-left*, we see that the operation of grasping is in function for *pulling-up*, making it more actionable than *pulling-left* when attempting to slide open the cabinet door. We present more results in the supplementary.

5.4. Real-world Data

We directly applied our networks trained on synthetic data to real-world data. Fig. 6 presents our predictions of the action scoring module on real 3D scans and 2D images, which shows promising results that our networks transfer the learned actionable information to real-world data.

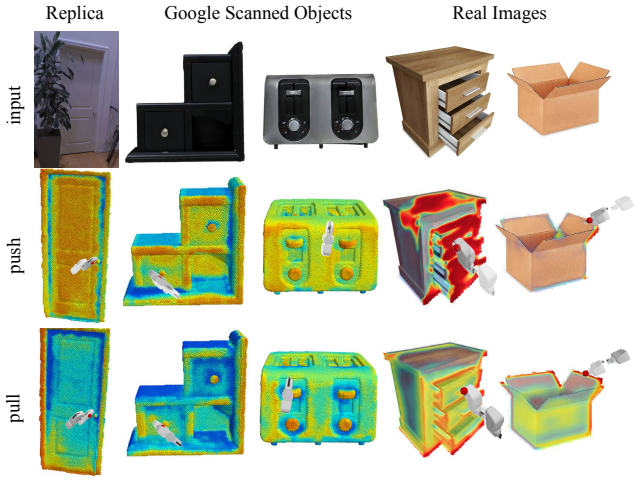


Figure 6. We visualize our action scoring predictions given certain gripper orientations over three real-world 3D scans from the Replica dataset [38] and Google Scanned Objects [34, 33], as well as on two 2D real images [1]. Results are shown over all pixels because of no access to the articulated part masks. Though there is no guarantee for the predictions over pixels outside the articulated parts, the results make sense if we allow motion for the entire objects.

6. Conclusion

We formulate a new challenging task to predict per-pixel actionable information for manipulating articulated 3D objects. Using an interactive environment built upon SAPIEN and the PartNet-Mobility dataset, we train neural networks that map pixels to actions: for each pixel on a articulated part of an object, we predict the actionability of the pixel related to six primitive actions and propose candidate interaction parameters. We present extensive quantitative evaluations and qualitative analysis of the proposed method. Results show that the learned knowledges are highly localized and thus generalizable to novel unseen object categories.

Limitations and Future Works. We see many possibilities for future extensions. First, our network takes single frame visual input, which naturally introduces ambiguities for the solution spaces if the articulated part mobility information cannot be fully determined from a single snapshot. Second, we limit our experiments to six types of action primitives with hard-coded motion trajectories. One future extension is to generalize the framework to free-form interactions. Finally, our method does not explicitly model the part segmentation and part motion axis, which may be incorporated in the future works to further improve the performance.

Acknowledgements

This work was supported primarily by Facebook during Kaichun's internship, while also by NSF grant IIS-1763268, a Vannevar Bush faculty fellowship, and an Amazon AWS ML award. We thank Yuzhe Qin and Fanbo Xiang for providing helps on setting up the SAPIEN environment.

References

[1] Two real images. https://www.uline.com/Product/Detail/S-4080/Corrugated-Boxes-200-Test/8-x-6-x-4-Corrugated-Boxes, https://donbaraton.es/p/mesa-de-noche-3-cajones-acabado-efecto-madera-alabama. 8

[2] Samarth Brahmbhatt, Cusuh Ham, Charles C Kemp, and James Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8709–8719, 2019. 2

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[4] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11973–11982, 2020. 2

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[6] Helin Dutagaci, Chun Pan Cheung, and Afzal Godil. Evaluation of 3d interest point detection techniques via human-generated ground truth. *The Visual Computer*, 28(9):901–917, 2012. 2

[7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 5

[8] David F. Fouhey, Vincent Delaitre, Abhinav Gupta, Alexei A. Efros, Ivan Laptev, and Josef Sivic. People watching: Human actions as a cue for single-view geometry. In *ECCV*, 2012. 2

[9] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018. 6

[10] Henning Hamer, Juergen Gall, Thibaut Weise, and Luc Van Gool. An object-dependent hand pose prior from sparse training data. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 671–678. IEEE, 2010. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[12] Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. Learning to predict part mobility from a single static snapshot. *ACM Transactions on Graphics (TOG)*, 36(6):1–13, 2017. 2

[13] Ruizhen Hu, Manolis Savva, and Oliver van Kaick. Functionality representations and applications for shape analysis. In *Computer Graphics Forum*, volume 37, pages 603–624. Wiley Online Library, 2018. 2

[14] Ruizhen Hu, Zihao Yan, Jingwen Zhang, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. Predictive and generative neural networks for object functionality. *arXiv preprint arXiv:2006.15520*, 2020. 2

[15] Ajinkya Jain, Rudolf Lioutikov, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. *arXiv preprint arXiv:2008.10518*, 2020. 2

[16] Mohi Khansari, Daniel Kappler, Jianlan Luo, Jeff Bingham, and Mrinal Kalakrishnan. Action image representation: Learning scalable deep grasping policies with zero real world data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3597–3603. IEEE, 2020. 2

[17] Vladimir G Kim, Siddhartha Chaudhuri, Leonidas Guibas, and Thomas Funkhouser. Shape2pose: Human-centric shape analysis. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014. 2

[18] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3706–3715, 2020. 2

[19] Hubert Lin, Melinos Averkiou, Evangelos Kalogerakis, Balazs Kovacs, Siddhant Ranade, Vladimir Kim, Siddhartha Chaudhuri, and Kavita Bala. Learning material-aware local descriptors for 3d shapes. In *2018 International Conference on 3D Vision (3DV)*, pages 150–159. IEEE, 2018. 2

[20] Martin Lohmann, Jordi Salvador, Aniruddha Kembhavi, and Roozbeh Mottaghi. Learning about objects by learning to interact with them. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[21] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016. 2

[22] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. The rbo dataset of articulated objects and interactions. *The International Journal of Robotics Research*, 38(9):1013–1019, 2019. 2

[23] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1824–1829. IEEE, 2003. 2

[24] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019. 2

[25] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-dof grasping for target-driven object manipulation in clutter. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6232–6238. IEEE, 2020. 3

[26] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8688–8697, 2019. 2

[27] Tushar Nagarajan and Kristen Grauman. Learning affordance landscapes for interaction exploration in 3d environments. In *NeurIPS*, 2020. 3

[28] Nvidia. PhysX physics engine. https://www.geforce.com/hardware/technology/physx. 5

[29] Deepak Pathak, Yide Shentu, Dian Chen, Pulkit Agrawal, Trevor Darrell, Sergey Levine, and Jitendra Malik. Learning instance segmentation by interaction. In *CVPR Workshop on Benchmarks for Deep Learning in Robotic Vision*, 2018. 3

[30] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016. 2

[31] Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2161–2168. IEEE, 2017. 3

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 3

[33] Google Research. Black decker stainless steel toaster 4 slice. https://fuel.ignitionrobotics.org/1.0/GoogleResearch/models/Black_Decker_Stainless_Steel_Toaster_4_Slice, 2020. 8

[34] Google Research. Victor reversible bookend. https://app.ignitionrobotics.org/GoogleResearch/fuel/models/Victor_Reversible_Bookend, 2020. 8

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[36] Manolis Savva, Angel X Chang, and Pat Hanrahan. Semantically-enriched 3d models for common-sense knowledge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 24–31, 2015. 2

[37] Lin Shao, Angel X Chang, Hao Su, Manolis Savva, and Leonidas Guibas. Cross-modal attribute transfer for rescaling 3d models. In *2017 International Conference on 3D Vision (3DV)*, pages 640–648. IEEE, 2017. 2

[38] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 8

[39] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018. 2

[40] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018. 2

[41] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 2

[42] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019. 2

[43] Erik Wijmans. Pointnet++ pytorch. https://github.com/erikwijmans/Pointnet2_PyTorch, 2018. 3

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong 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. 2

[45] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020. 2, 4, 5

[46] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 2

[47] Xianghao Xu, David Charatan, Sonia Raychaudhuri, Hanxiao Jiang, Mae Heitmann, Vladimir Kim, Siddhartha Chaudhuri, Manolis Savva, Angel Chang, and Daniel Ritchie. Motion annotation programs: A scalable approach to annotating kinematic articulations in large 3d shape collections. *3DV*, 2020. 2

[48] Pavel Yakubovskiy. Segmentation models pytorch. https://github.com/qubvel/segmentation_models.pytorch, 2020. 3

[49] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Van Kaick, Hao Zhang, and Hui Huang. Rpm-net: Recurrent prediction of motion and parts from point cloud. *ACM Transactions on Graphics (TOG)*, 38(6):240, 2019. 2

[50] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *arXiv preprint arXiv:1809.07417*, 2018. 2

[51] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 2

[52] Yang You, Yujing Lou, Chengkun Li, Zhoujun Cheng, Liangwei Li, Lizhuang Ma, Cewu Lu, and Weiming Wang. Keypointnet: A large-scale 3d keypoint dataset aggregated

from numerous human annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[53] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. 2018. 3

[54] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 3, 5