Robobarista: Object Part based Transfer of Manipulation Trajectories from Crowd-sourcing in 3D Pointclouds

Jaeyong Sung, Seok Hyun Jin, and Ashutosh Saxena
Department of Computer Science, Cornell University, USA.
Email: jysung@cs.cornell.edu, sj372@cornell.edu, asaxena@cs.cornell.edu

***Abstract*—There is a large variety of objects and appliances in human environments, such as stoves, coffee dispensers, juice extractors, and so on. It is challenging for a roboticist to program a robot for each of these object types and for each of their instantiations. In this work, we present a novel approach to manipulation planning based on the idea that many household objects share similarly-operated object parts. We formulate the manipulation planning as a structured prediction problem and design a deep learning model that can handle large noise in the manipulation demonstrations and learns features from three different modalities: point-clouds, language and trajectory. In order to collect a large number of manipulation demonstrations for different objects, we developed a new crowd-sourcing platform called Robobarista. We test our model on our dataset consisting of 116 objects with 249 parts along with 250 language instructions, for which there are 1225 crowd-sourced manipulation demonstrations. We further show that our robot can even manipulate objects it has never seen before.** [1]

I. Introduction

Consider the espresso machine in Figure 1 — even without having seen the machine before, a person can prepare a cup of latte by visually observing the machine and by reading a natural language instruction manual. This is possible because humans have vast prior experience of manipulating differently-shaped objects that share common parts such as 'handles' and 'knobs'. In this work, our goal is to enable robots to generalize to different objects and tasks (e.g. toaster, sink, water fountain, toilet, soda dispenser). Using a large knowledge base of manipulation demonstrations, we build an algorithm that infers a manipulation trajectory given a point-cloud and natural language instructions.

The key idea in our work is that many objects designed for humans share many similarly-operated *object parts* such as 'handles', 'levers', 'triggers', and 'buttons'; and manipulation motions can be transferred even among completely different objects if we represent motions with respect to *object parts*. For example, even if the robot has never seen the 'espresso machine' before, the robot should be able to manipulate it if it has previously seen similarly-operated parts in other objects such as 'urinal', 'soda dispenser', and 'restroom sink' as illustrated in Figure 2. Object parts that are operated in similar fashion may not carry the same part name (e.g., 'handle') but would rather have some similarity in their shapes allowing the motion to be transferred between completely different objects.

If the sole task for the robot is to manipulate one specific espresso machine or just a few types of 'handles', a roboticist could manually program the exact sequence to be executed. However, in human environments, there is a large variety in the types of object and their instances. Detection of objects or object parts (e.g. 'handle') alone does not provide enough information for robots to actually manipulate them. Thus, rather than relying on scene understanding techniques [3, 11, 6], we directly use 3D point-cloud for manipulation planning using machine learning algorithms.

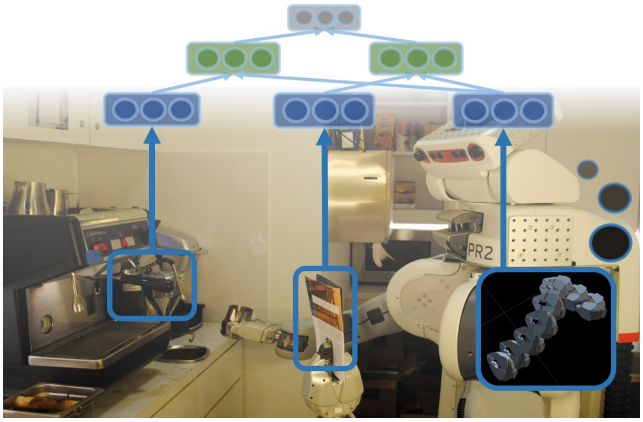

Fig. 1. **First encounter of an espresso machine** by our PR2 robot. Without ever having seen the machine before, given the language instructions and a point-cloud from Kinect sensor, our robot is capable of finding appropriate manipulation trajectories from prior experience using our deep learning model.

The key challenges in our problem are in designing features and a learning model that integrates three completely different modalities of data (point-cloud, language and trajectory), and in handling the significant noise in manipulation demonstrations from crowd-sourcing. Deep learning has made impact in related application areas (e.g., vision [10, 2], natural language processing [14]). In this work, we present a deep learning model that can handle large noise in labels, with a new architecture that learns relations between the three different modalities.

Such learning algorithms require a large dataset for training. However, collecting such large dataset of expert demonstrations is very expensive as it requires the presence of the robot, an expert, and the object to be manipulated. In this work, we show that we can crowd-source the demonstrations of manipulation to the public over the web through our Robobarista platform and outperform the model trained with expert demonstrations. Furthermore, in contrast to previous approaches based on learning from demonstration (LfD) that learn a mapping from a state to an action [1], our work

[1]Full technical report is available at http://robobarista.cs.cornell.edu

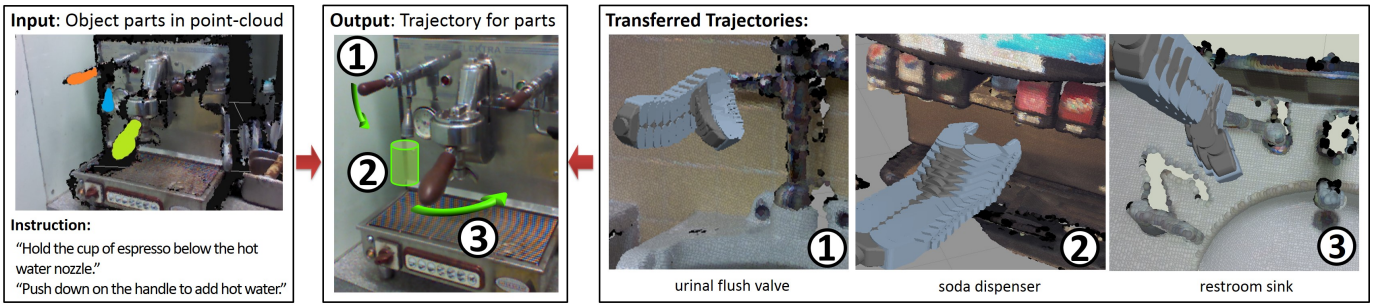

Fig. 2. **Object part and natural language instructions input to manipulation trajectory as output.** Objects such as the espresso machine consist of distinct object parts, each of which requires a distinct manipulation trajectory for manipulation. For each part of the machine, we can re-use a manipulation trajectory that was used for some other object with similar parts. So, for an object part in a point-cloud (each object part colored on left), we can find a trajectory used to manipulate some other object (labeled on the right) that can be *transferred* (labeled in the center). With this approach, a robot can operate a new and previously unobserved object such as the 'espresso machine', by successfully transferring trajectories from other completely different but previously observed objects. Note that the input point-cloud is very noisy and incomplete (black represents missing points).

complements LfD as we focus on the entire manipulation motion (as opposed to a sequential state-action mapping).

In order to validate our approach, we have collected a large dataset of *116 objects* with *250 natural language instructions* for which there are *1225 crowd-sourced manipulation trajectories* from 71 non-expert users via our Robobarista web platform (http://robobarista.cs.cornell.edu). We also present experiments on our robot using our approach. In summary, the key contributions of this work are:

- a novel approach to manipulation planning via *part-based transfer* between different objects,
- incorporation of *crowd-sourcing* to manipulation planning,
- introduction of *deep learning model* that handles three modalities with noisy labels from crowd-sourcing, and
- contribution of the first large manipulation dataset and experimental evaluation on this dataset.

II. Our Approach

The intuition for our approach is that many differently-shaped objects share similarly-operated object parts; thus, the manipulation trajectory of an object can be transferred to a completely different object if they share similarly-operated parts. We formulate this problem as a structured prediction problem and introduce a deep learning model that can handle three modalities of the data while dealing with noisy labels in crowd-sourced data. Then, we introduce the crowd-sourcing platform Robobarista for easily scaling the collection of manipulation demonstrations to non-expert public on the web.

A. Problem Formulation

The goal is to learn a function f that maps a given pair of point-cloud $p \in \mathcal{P}$ of object part and language $l \in \mathcal{L}$ to a trajectory $\tau \in \mathcal{T}$ that can manipulate the object part as described by l:

$$f : \mathcal{P} \times \mathcal{L} \rightarrow \mathcal{T}$$

Point-cloud Representation. Each instance of point-cloud $p \in \mathcal{P}$ is represented as a set of n points in three-dimensional Euclidean space where each point (x, y, z) is represented with its RGB color (r, g, b): $p = \{p^{(i)}\}_{i=1}^{n} = \{(x, y, z, r, g, b)^{(i)}\}_{i=1}^{n}$. The size of the set vary by object parts. These points are often obtained by stitching together a sequence of sensor data from an RGBD sensor [9].

Trajectory Representation Each trajectory $\tau \in \mathcal{T}$ is represented as a sequence of m *waypoints*, where each waypoint consists of gripper status g, translation (t_x, t_y, t_z), and rotation (r_x, r_y, r_z, r_w) with respect to the origin: $\tau = \{\tau^{(i)}\}_{i=1}^{m} = \{(g, t_x, t_y, t_z, r_x, r_y, r_z, r_w)^{(i)}\}_{i=1}^{m}$ where $g \in \{\text{"open"}, \text{"closed"}, \text{"holding"}\}$. g depends on the type of the end-effector, which we have assumed to be a two-fingered gripper like that of PR2 or Baxter. The rotation is represented as quaternions (r_x, r_y, r_z, r_w) instead of the more compact Euler angles to prevent problems such as the gimbal lock [13].

B. Can transferred trajectories adapt without modification?

Even if we have a trajectory to transfer, a conceptually transferable trajectory is not necessarily directly compatible if it is represented with respect to an inconsistent reference point. To make a trajectory compatible with a new situation without modifying the trajectory, we need a representation method for trajectories that allows a *direct transfer of a trajectory without any modification* relying on point-cloud information.

Transferred trajectories become compatible across different objects when trajectories are represented 1) in the task space rather than the configuration space, and 2) in the principal-axis based coordinate frame [8] of the object *part* rather than the robot or the object.

III. Deep Learning for Manipulation Trajectory Transfer

We use deep learning to find the most appropriate trajectory for the given point-cloud and natural language. Deep learning is mostly used for binary or multi-class classification or regression problem [2] with a uni-modal input. We introduce a deep learning model that can handle three completely different modalities of point-cloud, language, and trajectory and solve a structural problem with lots of label noise.

The original structured prediction problem ($f : \mathcal{P} \times \mathcal{L} \rightarrow \mathcal{T}$) is converted to a binary classification problem ($f : (\mathcal{P} \times \mathcal{L}) \times \mathcal{T} \rightarrow \{0, 1\}$). Intuitively, the model takes the input of point-cloud, language, and trajectory and outputs whether it is a good match (label $y = 1$) or a bad match (label $y = 0$).

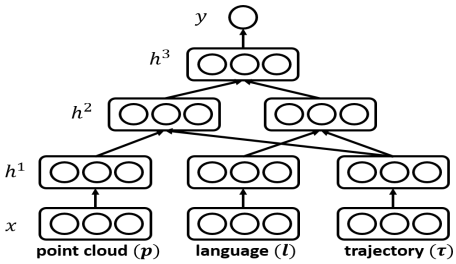

Fig. 3. **Our deep learning model** for transferring manipulation trajectory. Our model takes the input x of three different modalities (point-cloud, language, and trajectory) and outputs y whether it is a good match or bad match, It first learns features separately (h^1) for each modality and then learn relation (h^2) between input and output of original structured problem. Finally, last hidden layer h^3 learns relations of all these modalities.

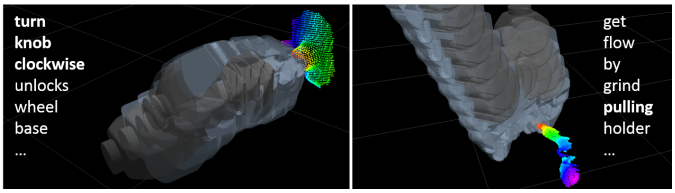

Fig. 4. **Visualization** of a sample of learned high-level feature (two nodes) at last hidden layer h^3. The point-cloud in the picture is given arbitrary axis-based color for visualization purpose. The left shows a node #1 at layer h^3 that learned to ("turn", "knob", "clockwise") along with relevant point-cloud and trajectory. The right shows a node #51 at layer h^3 that learned to "pull" handle. The visualization is created by selecting a set of words, a point-cloud, a trajectory that maximizes the activation at each layer and passing the highest activated set of inputs to higher level.

Model. Given an input of point-cloud, language, and trajectory, $x = ((p, l), \tau)$, as shown at the bottom of Figure 3, the goal is to classify as either $y = 0$ or 1 at the top. The first h^1 layer learns a separate layer of features for each modality of x $(= h^0)$ [12]. The next layer learns the relations between the input (p, l) and the output τ for the original structured problem, combining two modalities at a time. The left combines point-cloud and trajectory and the right combines language and trajectory. The third layer h^3 learns the relation between these two combinations of modalities and the final layer y represents the binary label.

Every layer h^i uses the rectified linear unit [18] as the activation function: $h^i = a(W^i h^{i-1} + b^i)$ where $a(\cdot) = max(0, \cdot)$ with weights to be learned $W^i \in \mathbb{R}^{M \times N}$, where M and N represent the number of nodes in $(i-1)$-th and i-th layer respectively. The logistic regression is used in last layer for predicting the final label y. The probability that $x = ((p, l), \tau)$ is a "good match" is computed as: $P(Y = 1|x, W, b) = 1/(1 + e^{-(Wx+b)})$

Label noise. When data contains lots of noisy label (noisy trajectory τ) from the crowd-sourced data and lacks a ground-truth label (an expert demonstration), the crowd-sourced data should not be treated equally as will be shown in Sec. V.

For every pair of input $(p, l)_i$, we have $\mathcal{T}_i = \{\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,n_i}\}$, a set of trajectories submitted by the crowd for $(p, l)_i$. First, the best candidate label $\tau_i^* \in \mathcal{T}_i$ for $(p, l)_i$ is selected as one of the labels with the smallest average trajectory distance (DTW-MP in Sec. V) to other labels:

$$\tau_i^* = \underset{\tau \in \mathcal{T}_i}{\operatorname{argmin}} \frac{1}{n_i} \sum_{j=1}^{n_i} \Delta(\tau, \tau_{i,j})$$

We assume that at least half of the crowd tried to give a reasonable demonstration. Thus a demonstration with the smallest average distance to all other demonstrations must be a good demonstration.

Since we have found the most likely label τ_i^* for $(p, l)_i$, we give the label 1 ("good match") to $((p, l)_i, \tau_i^*)$, making it the first positive example for the binary classification problem. Then we find more positive examples by finding other trajectories $\tau' \in \mathcal{T}$ such that $\Delta(\tau_i^*, \tau') < t_g$ where t_g is a threshold determined by the expert. Similarly, negative examples are generated by finding trajectories $\tau' \in \mathcal{T}$ such that it is above some threshold $\Delta(\tau_i^*, \tau') > t_w$, where t_w is determined by expert, and they are given label 0 ("bad match").

Pre-training. We use the stacked sparse de-noising auto-encoder (SSDA) to train weights W^i and bias b^i for each layer [17, 18]. Training occurs layer by layer from bottom to top trying to reconstruct the previous layer using SSDA. To learn parameters for layer i, we build an auto-encoder which takes the corrupted output $\tilde{h}^{i-1}$ (binomial noise with corruption level p) of previous layer as input and minimizes the loss function [18] with max-norm constraint [15]:

$$W^* = \underset{W}{\operatorname{argmin}} \|\hat{h}^{i-1} - h^{i-1}\|_2^2 + \lambda \|h^i\|_1$$

$$\text{where} \quad \hat{h}^{i-1} = f(W^i h^i + b^i) \qquad h^i = f(W^{i^T} \tilde{h}^{i-1} + b^i)$$
$$\tilde{h}^{i-1} = h^{i-1} X \qquad \|W^i\|_2 \leq c$$
$$X \sim B(1, p)$$

Fine-tuning. The pre-trained neural network can be fine-tuned by minimizing the negative log-likelihood with the stochastic gradient method with mini-batches: $NLL = -\sum_{i=0}^{|D|} log(P(Y = y^i|x^i, W, b))$ To prevent over-fitting to the training data, we used dropout [7], which randomly drops a specified percentage of the output of every layer.

Inference. Given the trained neural network, inference step finds the trajectory τ that maximizes the output through sampling in the space of trajectory $\mathcal{T}$: $\operatorname{argmax}_{\tau' \in \mathcal{T}} P(Y = 1|x = ((p, l), \tau'), W, b)$. Since the space of trajectory $\mathcal{T}$ is infinitely large, based on our idea that we can transfer trajectories across objects, we only search trajectories that the model has seen in training phase.

Data pre-processing. As seen in Sec. II-A, each of the modality (p, l, τ) can have any length. Thus, we pre-process to make each fixed in length.

We represent point-cloud p of any arbitrary length as an occupancy grid where each cell indicates whether any point lives in the space it represents. Because point-cloud p consists of only the part of an object which is limited in size, we can represent p using two occupancy grids of size $10 \times 10 \times 10$ with different scales: one with each cell representing $1 \times 1 \times 1(cm)$ and the other with each cell representing $2.5 \times 2.5 \times 2.5(cm)$.

Each natural language instruction is represented as a fixed-size bag-of-words representation with stop words removed. Finally, for each trajectory $\tau \in \mathcal{T}$, we first compute its smooth interpolated trajectory $\tau_s \in \mathcal{T}_s$, and then normalize all trajectories $\mathcal{T}_s$ to the same length while preserving the sequence of gripper status.

IV. ROBOBARISTA: CROWD-SOURCING PLATFORM

In order to collect a large number of manipulation demonstrations from the crowd, we built a crowd-sourcing web platform that we call Robobarista (see Fig. 5). It provides a virtual environment where non-expert users can teach robots via a web browser, without expert guidance or physical presence with a robot and a target object.

The system simulates a situation where the user encounters a previously unseen target object and a natural language instruction manual for its manipulation. Within the web browser, users are shown a point-cloud in the 3-D viewer on the left and a *manual* on the right. A manual may involve several instructions, such as "Push down and pull the handle to open the door". The user's goal is to demonstrate how to manipulate the object in the scene for each instruction.

V. EXPERIMENTS

Data. In order to test our model, we have collected a dataset of 116 point-clouds of objects with 249 object parts (examples shown in Figure 6). There are also a total of 250 natural language instructions (in 155 manuals).[2] Using the crowd-sourcing platform Robobarista, we collected 1225 trajectories for these objects from 71 non-expert users on the Amazon Mechanical Turk. After a user is shown a 20-second instructional video, the user first completes a 2-minute tutorial task. At each session, the user was asked to complete 10 assignments where each consists of an object and a manual to be followed.

For each object, we took raw RGB-D images with the Microsoft Kinect sensor and stitched them using Kinect Fusion [9] to form a denser point-cloud in order to incorporate different viewpoints of objects. Objects range from kitchen appliances such as 'stove', 'toaster', and 'rice cooker' to 'urinal', 'soap dispenser', and 'sink' in restrooms. The dataset will be made available at http://robobarista.cs.cornell.edu

Metric. We propose a new measure DTW-MT for evaluating manipulation trajectories that include translation, rotation and gripper status. DTW-MT uses dynamic time warping and non-linearly warps two trajectories of arbitrary lengths to produce a matching between waypoints, and a cumulative distance is the sum of pairwise distances between matched waypoints.

A. Results and Discussions

We evaluated all models on our dataset using *5-fold cross-validation* and the result is in Table I. Rows list the models we tested including our model and baselines. Each column shows one of three evaluations. First two use dynamic time warping for manipulation trajectory (DTW-MT). The first column shows averaged DTW-MT for each instruction manual consisting of one or more language instructions. The second column shows averaged DTW-MT for every test pair (p, l).

As DTW-MT values are not intuitive, we added the extra column "accuracy", which shows the percentage of trajectories with DTW-MT value less than 10. Through expert surveys, we found that when DTW-MT of manipulation trajectory is less than 10, the robot came up with a reasonable trajectory and will very likely be able to accomplish the given task.

[2]Although not necessary for training our model, we also collected trajectories from the expert for evaluation purposes.

TABLE I

RESULT ON OUR DATASET WITH *5-fold cross-validation*. ROWS LIST MODELS WE TESTED INCLUDING OUR MODEL AND BASELINES. AND EACH COLUMN SHOWS A DIFFERENT METRIC USED TO EVALUATE THE MODELS. FOR DETAILS OF EACH BASELINE, PLEASE REFER TO FULL TECH REPORT AVAILABLE AT PROJECT WEBSITE.

	per manual	**per instruction**	
Models	**DTW-MT**	**DTW-MT**	**Accuracy (%)**
chance	28.0 ($\pm$0.8)	27.8 ($\pm$0.6)	11.2 ($\pm$1.0)
object part classifier	-	22.9 ($\pm$2.2)	23.3 ($\pm$5.1)
Structured SVM	21.0 ($\pm$1.6)	21.4 ($\pm$1.6)	26.9 ($\pm$2.6)
LSSVM + kinematic [16]	17.4 ($\pm$0.9)	17.5 ($\pm$1.6)	40.8 ($\pm$2.5)
similarity + random	14.4 ($\pm$1.5)	13.5 ($\pm$1.4)	49.4 ($\pm$3.9)
similarity + weights [5]	13.3 ($\pm$1.2)	12.5 ($\pm$1.2)	53.7 ($\pm$5.8)
Ours w/o Multi-modal	13.7 ($\pm$1.6)	13.3 ($\pm$1.6)	51.9 ($\pm$7.9)
Ours w/o Noise-handling	14.0 ($\pm$2.3)	13.7 ($\pm$2.1)	49.7 ($\pm$10.0)
Ours with Experts	12.5 ($\pm$1.5)	12.1 ($\pm$1.6)	53.1 ($\pm$7.6)
Our Model	13.0 ($\pm$1.3)	12.2 ($\pm$1.1)	**60.0** ($\pm$5.1)

Our full model performed 60.0% in accuracy (Table I), outperforming the chance as well as other baseline algorithms we tested on our dataset. Fig. 7 shows two examples of successful transfers and one unsuccessful transfer by our model. In the first example, the trajectory for pulling down on a cereal dispenser is transferred to a coffee dispenser. Because our approach to trajectory representation is based on the principal axis (Sec. II-B), even though cereal and coffee dispenser handles are located and oriented differently, the transfer is a success.

We do not claim that our model can find and execute manipulation trajectories for all objects. However, for a large fraction of objects which the robot has never seen before, our model outperforms other models in finding correct manipulation trajectories. And, when we trained our full model with expert demonstrations, which were collected for evaluation purposes, it performed 53.1%.

The *task similarity* method, which searches for most similar training task, gave a result of 53.7% but it requires access to all of the raw training data (all point-clouds and language) at test time, which leads to heavy computation and large storage.

For more traditional approach of using hand-coded features with Structural SVM and Latent SSVM models, we experimented with many state-of-the-art features for many months, and it gave 40.8%. While it is extremely difficult to find a good set of features for three modalities, our deep learning model which does not require hand-designing of features learned features at top layer h^3 such as ones shown in Fig. 4.

B. Robotic Experiments.

As the PR2 robot stands in front of the object, the robot is given a natural language instruction and segmented point-cloud. Using our algorithm, manipulation trajectory to be transferred were found for the given point-cloud and language. Given the trajectories which are defined as set of waypoints, the robot followed the trajectory by impedance controller (`ee_cart_imped`) [4]. Our PR2 robot, which has never seen the espresso machine before, was even able to make a cup of latte with very little help from the expert. Several examples of successful execution on PR2 robot are shown in Figure 8 and in video at the project website: http://robobarista.cs.cornell.edu

REFERENCES

[1] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *RAS*, 2009.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[3] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*. 2008.

[4] M. Bollini, J. Barry, and D. Rus. Bakebot: Baking cookies with the pr2. In *IROS PR2 Workshop*, 2011.

[5] M. Forbes, M. J.-Y. Chung, M. Cakmak, and R. P. Rao. Robot programming by demonstration with crowdsourced action fixes. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[6] R. B. Girshick, P. F. Felzenszwalb, and D. A. McAllester. Object detection with grammar models. In *NIPS*, volume 5, page 6, 2011.

[7] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[8] K. Hsiao, S. Chitta, M. Ciocarlie, and E. Jones. Contact-reactive grasping of objects with partial shape information. In *IROS*, 2010.

[9] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *ACM Symposium on UIST*, 2011.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *CVPR*. IEEE, 2009.

[12] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 689–696, 2011.

[13] A. Saxena, J. Driemeyer, and A. Ng. Learning 3-d object orientation from images. In *ICRA*, 2009.

[14] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[15] N. Srivastava. *Improving neural networks with dropout*. PhD thesis, University of Toronto, 2013.

[16] J. Sturm, C. Stachniss, and W. Burgard. A probabilistic framework for learning kinematic models of articulated objects. *JAIR*, 41(2):477–526, 2011.

[17] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[18] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, et al. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013.

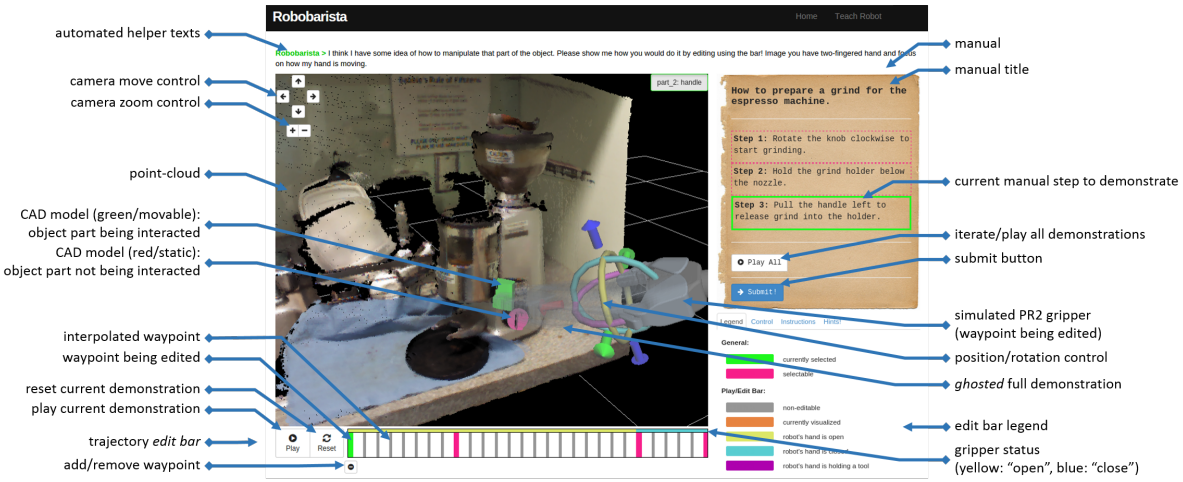

Fig. 5. **Screen-shot of Robobarista,** the crowd-sourcing platform running on Chrome browser. We have built Robobarista platform for collecting a large number of crowd demonstration for teaching the robot.

Fig. 6. **Examples from our dataset** which consist of a natural language instruction (top), an object part in point-cloud representation (highlighted), and a manipulation trajectory (below) collected via Robobarista. Objects range from kitchen appliances such as stove and rice cooker to urinals and sinks in restrooms. As our trajectories are collected from non-experts, they vary in quality from being likely to complete the manipulation task successfully (left of dashed line) to being unlikely to do so successfully (right of dashed line).

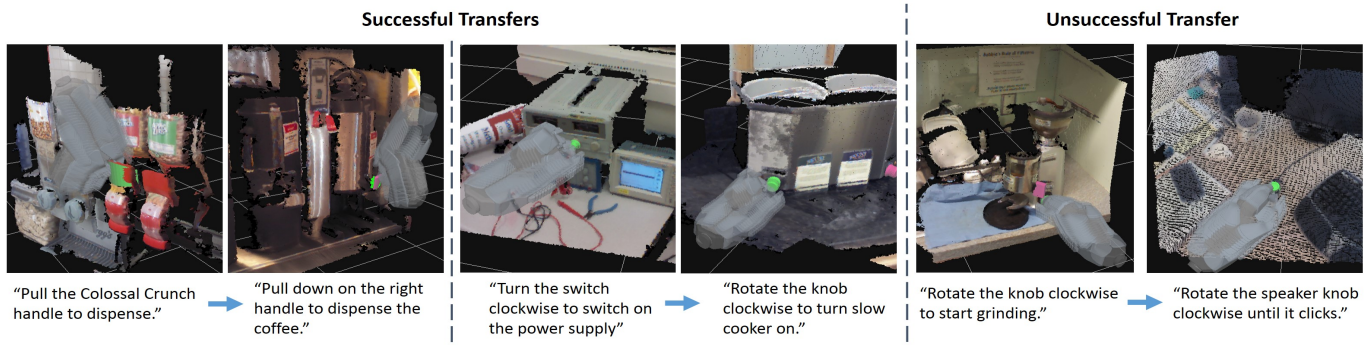

Fig. 7. **Examples of successful and unsuccessful transfers** of manipulation trajectory from left to right using our model. In first two examples, though the robot has never seen the 'coffee dispenser' and 'slow cooker' before, the robot has correctly identified that the trajectories of 'cereal dispenser' and 'DC power supply', respectively, can be used to manipulate them.

Fig. 8. **Examples of transferred trajectories** being executed on PR2. On the left, PR2 is able to rotate the 'knob' to turn the lamp on. On the right, using two transferred trajectories, PR2 is able to hold the cup below the 'nozzle' and press the 'lever' of 'coffee dispenser'.