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

Jaeyong Sung, Seok Hyun Jin, and Ashutosh Saxena

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.

***Keywords*— Robotics and Learning**, Crowd-sourcing, Manipulation

1 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 their manipulation ability to novel 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 an appropriate manipulation trajectory given a point-cloud and natural language instructions.

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

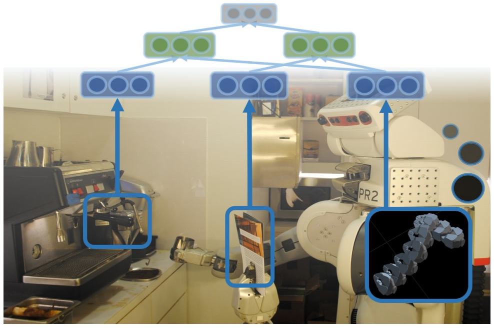


Fig. 1: **First encounter of an espresso machine** by our PR2 robot. Without ever having seen the machine before, given the language instructions and a point-cloud from Kinect sensor, our robot is capable of finding appropriate manipulation trajectories from prior experience using our deep learning model.

The key 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 that allows 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. Classification 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 [7, 33, 17], we directly use 3D point-cloud for manipulation planning using machine learning algorithms.

Such machine learning algorithms require a large dataset for training. However, collecting such large dataset of expert demonstrations is very expensive as it requires joint physical presence of the robot, an expert, and the object to be manipulated. In this work, we show that we can crowd-source the collection of manipulation demonstrations to the public over the web through our Robobarista platform and still outperform the model trained with expert demonstrations.

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 significant amount of noise in crowd-sourced manipulation demonstrations. Deep learning has made impact in related application areas (e.g., vision [29, 5], natural language processing [47]). 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. Furthermore, in contrast to previous approaches based on learning from demonstration (LfD) that learn a mapping from a state to an action [4], our work 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*

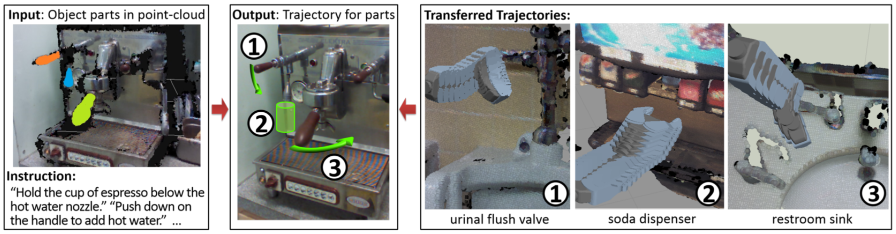

Input: Object parts in point-cloud
Instruction:
“Hold the cup of espresso below the hot water nozzle.” “Push down on the handle to add hot water.” ...
Output: Trajectory for parts
1
2
3
Transferred Trajectories:
1
urinal flush valve
2
soda dispenser
3
restroom sink

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).

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 that allows manipulation of novel 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.

2 Related Work

Scene Understanding. There has been great advancement in scene understanding [33, 28, 63], in human activity detection [52, 21], and in features for RGB-D images and point-clouds [48, 31]. And, similar to our idea of using part-based transfers, the deformable part model [17] was effective in object detection. However, classification of objects, object parts, or human activities alone does not provide enough information for a robot to reliably plan manipulation. Even a simple category such as kitchen sinks has so much variation in its instances, each differing in how it is operated: pulling the handle upwards, pushing upwards, pushing sideways, and so on. On the other hand, direct perception approach skips the intermediate object labels and directly perceives affordance based on the shape of the object [16, 30]. It focuses on detecting the part known to afford certain action such as ‘pour’ given the object, while we focus on predicting the correct motion given the object part.

Manipulation Strategy. For highly specific tasks, many works manually sequence different controllers to accomplish complicated tasks such as baking cookies [8] and folding the laundry [35], or focus on learning specific motions such as grasping [26] and opening doors [13]. Others focus on learning to sequence different movements [53, 36] but assume that there exist perfect controllers such as *grasp* and *pour*.

For a more general task of manipulating new instances of objects, previous approaches rely on finding articulation [51, 41] or using interaction [25], but they are limited by tracking performance of a vision algorithm. Many objects that humans operate daily have parts such as “knob” that are small, which leads to significant occlusion as manipulation is demonstrated. Another approach using part-based transfer between objects has been shown to be successful for grasping [10, 12]. We extend this approach and introduce a deep learning model that enables part-based transfer of *trajectories* by automatically learning relevant features. Our focus is on the generalization of manipulation trajectory via part-based transfer using point-clouds without knowing objects a priori and without assuming any of the sub-steps (‘approach’, ‘grasping’, and ‘manipulation’).

Learning from Demonstration (LfD). The most successful approach for teaching robots tasks, such as helicopter maneuvers [1] or table tennis [37], has been based on LfD [4]. Although LfD allows end users to demonstrate the task by simply taking the robot arms, it focuses on learning individual actions and separately relies on high level task composition [34, 11] or is often limited to previously seen objects [40, 39]. We believe that learning a single model for an action like “turning on” is impossible because human environment has many variations.

Unlike learning a model from demonstration, instance-based learning [2, 15] replicates one of the demonstrations. Similarly, we directly transfer one of the demonstrations but focus on generalizing manipulation planning to completely new objects, enabling robots to manipulate objects they have never seen before.

Deep Learning. There has been great success with deep learning, especially in the domains of vision and natural language processing (e.g. [29, 47]). In robotics, deep learning has previously been successfully used for detecting grasps on multi-channel input of RGB-D images [32] and for classifying terrain from long-range vision [18].

Deep learning can also solve multi-modal problems [38, 32] and structured problems [46]. Our work builds on prior works and extends neural network to handle three modalities which are of completely different data type (point-cloud, language, and trajectory) while handling lots of label-noise originating from crowd-sourcing.

Crowd-sourcing. Teaching robots how to manipulate different objects has often relied on experts [4, 1]. Among previous efforts to scale teaching to the crowd [9, 54, 23], Forbes et al. [15] employs a similar approach towards crowd-sourcing but collects multiple instances of similar table-top manipulation with same object, and others build web-based platform for crowd-sourcing manipulation [56, 57]. These approaches either depend on the presence of an expert (due to a required special software) or require a real robot at a remote location. Our Robobarista platform borrows some components of [3], but works on any standard web browser with OpenGL support and incorporates real point-clouds of various scenes.

3 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 handles three modalities of data and deals with noise in crowd-sourced data. Then, we introduce the crowd-sourcing platform Robobarista to easily scale the collection of manipulation demonstrations to non-experts on the web.

3.1 Problem Formulation

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

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

Point-cloud Representation. Each instance of point-cloud $p \in \mathscr{P}$ 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 for each instance. These points are often obtained by stitching together a sequence of sensor data from an RGBD sensor [22].

Trajectory Representation. Each trajectory $\tau \in \mathscr{T}$ 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 [43].

Smooth Trajectory. To acquire a smooth trajectory from a waypoint-based trajectory τ, we interpolate intermediate waypoints. Translation is linearly interpolated and the quaternion is interpolated using spherical linear interpolation (Slerp) [45].

3.2 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, based on point-cloud information, that allows a *direct transfer of a trajectory without any modification*.

Challenges. Making a trajectory compatible when transferred to a different object or to a different instance of the same object without modification can be challenging depending on the representation of trajectories and the variations in the location of the object, given in point-clouds.

For robots with high degrees of freedom arms such as PR2 or Baxter robots, trajectories are commonly represented as a sequence of joint angles (in configuration space) [55]. With such representation, the robot needs to modify the trajectory for an object with forward and inverse kinematics even for a small change in the object's position and orientation. Thus, trajectories in the configuration space are prone to errors as they are realigned with the object. They can be executed without modification only when the robot is in the exact same position and orientation with respect to the object.

One approach that allows execution without modification is representing trajectories with respect to the object by aligning via point-cloud registration (e.g. [15]). However, if the object is large (e.g. a stove) and has many parts (e.g. knobs and handles), then object-based representation is prone to errors when individual parts have different translation and rotation. This limits the transfers to be between different instances of the same object that is small or has a simple structure.

Lastly, it is even more challenging if two objects require similar trajectories, but have slightly different shapes. And this is made more difficult by limitations of the point-cloud data. As shown in left of Fig. 2, the point-cloud data, even when stitched from multiple angles, are very noisy compared to the RGB images.

Our Solution. 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 of the object *part* rather than the robot or the object.

Trajectories can be represented in the task space by recording only the position and orientation of the end-effector. By doing so, we can focus on the actual interaction between the robot and the environment rather than the movement of the arm. It is very rare that the arm configuration affects the completion of the task as long as there is no collision. With the trajectory represented as a sequence of gripper position and orientation, the robot can find its arm configuration that is collision free with the environment using inverse kinematics.

However, representing the trajectory in task space is not enough to make transfers compatible. It has to be in a common coordinate frame regardless of object's orientation and shape. Thus, we align the negative z-axis along gravity and align the x-axis along the principal axis of the object *part* using PCA [20]. With this representation, even when the object part's position and orientation changes, the trajectory does not need to change. The underlying assumption is that similarly operated object parts share similar shapes leading to a similar direction in their principal axes.

4 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 [5] 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 ($f : \mathscr{P} \times \mathscr{L} \to \mathscr{T}$) is converted to a binary classification problem ($f : (\mathscr{P} \times \mathscr{L}) \times \mathscr{T} \to \{0,1\}$). 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 $y = 0$).

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 $(= h^0)$ [38]. The next layer learns the relations between the input (p,l) and the output τ of 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 [65] as the activation function:

$$h^i = a(W^i h^{i-1} + b^i) \text{ where } a(\cdot) = max(0, \cdot)$$

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) due to crowd-sourcing, not all crowd-sourced trajectories should be trusted as equally appropriate as will be shown in Sec. 7.

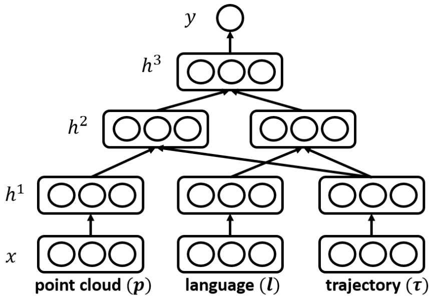


Fig. 3: **Our deep learning model** for transferring manipulation trajectory. Our model takes the input x of three different modalities (point-cloud, language, and trajectory) and outputs y, whether it is a good match or bad match. It first learns features separately (h^1) for each modality and then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

For every pair of input $(p,l)_i$, we have $\mathscr{T}_i = \{\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,n_i}\}$, a set of trajectories submitted by the crowd for $(p,l)_i$. First, the best candidate label $\tau_i^* \in \mathscr{T}_i$ for $(p,l)_i$ is selected as one of the labels with the smallest average trajectory distance (Sec. 5) to other labels:

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

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.

Once 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 $\tau' \in \mathscr{T}$ 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 $\tau' \in \mathscr{T}$ 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 [61, 65]. 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 [65] with max-norm constraint [49]:

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

$$\text{where} \quad \hat{h}^{i-1} = f(W^i h^i + b^i) \qquad h^i = f(W^{i^T} \tilde{h}^{i-1} + b^i) \qquad \tilde{h}^{i-1} = h^{i-1} X$$

$$\|W^i\|_2 \leq c \qquad X \sim B(1,p)$$

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 [19], 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 $\mathscr{T}$:

$$\underset{\tau' \in \mathscr{T}}{\arg\max}\, P(Y=1|x=((p,l),\tau');W,b)$$

Since the space of trajectory $\mathscr{T}$ 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. 3.1, each of the modalities (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 language instruction is represented as a fixed-size bag-of-words representation with stop words removed. Finally, for each trajectory $\tau \in \mathscr{T}$, we first compute its smooth interpolated trajectory $\tau_s \in \mathscr{T}_s$ (Sec. 3.1), and then normalize all trajectories $\mathscr{T}_s$ to the same length while preserving the sequence of gripper status.

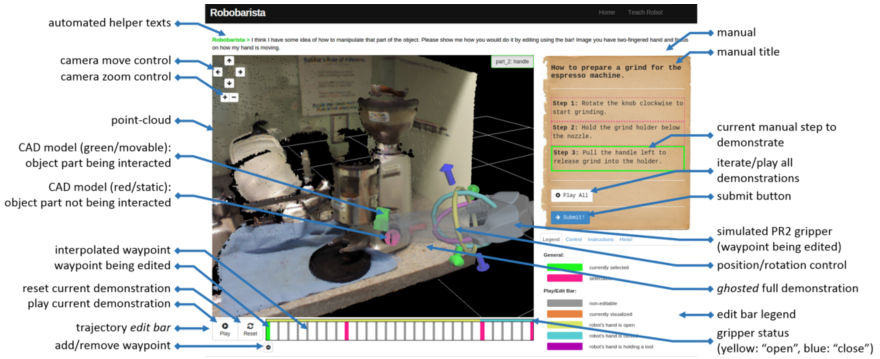
5 Loss Function for Manipulation Trajectory

Prior metrics for trajectories consider only their translations (e.g. [27]) and not their rotations *and* gripper status. We propose a new measure, which uses dynamic time warping, for evaluating manipulation trajectories. This measure non-linearly warps two trajectories of arbitrary lengths to produce a matching, and cumulative distance is computed as the sum of cost of all matched waypoints. The strength of this measure is that weak ordering is maintained among matched waypoints and that every waypoint contributes to the cumulative distance.

For two trajectories of arbitrary lengths, $\tau_A = \{\tau_A^{(i)}\}_{i=1}^{m_A}$ and $\tau_B = \{\tau_B^{(i)}\}_{i=1}^{m_B}$, we define matrix $D \in \mathbb{R}^{m_A \times m_B}$, where $D(i,j)$ is the cumulative distance of an optimally-warped matching between trajectories up to index i and j, respectively, of each trajectory. The first column and the first row of D is initialized as $D(i,1) = \sum_{k=1}^{i} c(\tau_A^{(k)},\tau_B^{(1)}) \, \forall i \in [1,m_A]$ and $D(1,j) = \sum_{k=1}^{j} c(\tau_A^{(1)},\tau_B^{(k)}) \, \forall j \in [1,m_B]$, where c is a local cost function between two waypoints (discussed later). The rest of D is completed using dynamic programming: $D(i,j) = c(\tau_A^{(i)},\tau_B^{(j)}) + \min\{D(i-1,j-1), D(i-1,j), D(i,j-1)\}$

Given the constraint that $\tau_A^{(1)}$ is matched to $\tau_B^{(1)}$, the formulation ensures that every waypoint contributes to the final cumulative distance $D(m_A,m_B)$. Also, given a matched pair $(\tau_A^{(i)},\tau_B^{(j)})$, no waypoint preceding $\tau_A^{(i)}$ is matched to a waypoint succeeding $\tau_B^{(j)}$, encoding weak ordering.

The pairwise cost function c between matched waypoints $\tau_A^{(i)}$ and $\tau_B^{(j)}$ is defined:



Robobarista > I think I have some idea of how to manipulate that part of the object. Please show me how you would do it by editing using the bar! Image you have two-fingered hand and focus on how my hand is moving.

How to prepare a grind for the espresso machine.

Step 1: Rotate the knob clockwise to start grinding.

Step 2: Hold the grind holder below the nozzle.

Step 3: Pull the handle left to release grind into the holder.

Fig. 4: **Screen-shot of Robobarista,** the crowd-sourcing platform running on Chrome browser. We have built Robobarista platform for collecting a large number of crowd demonstrations for teaching the robot.

$$c(\tau_A^{(i)},\tau_B^{(j)};\alpha_T,\alpha_R,\beta,\gamma)=w(\tau_A^{(i)};\gamma)w(\tau_B^{(j)};\gamma)\left(\frac{d_T(\tau_A^{(i)},\tau_B^{(j)})}{\alpha_T}+\frac{d_R(\tau_A^{(i)},\tau_B^{(j)})}{\alpha_R}\right)\left(1+\beta d_G(\tau_A^{(i)},\tau_B^{(j)})\right)$$

$$\begin{aligned}\text{where}\qquad d_T(\tau_A^{(i)},\tau_B^{(j)}) &= ||(t_x,t_y,t_z)_A^{(i)}-(t_x,t_y,t_z)_B^{(j)}||_2\\ d_R(\tau_A^{(i)},\tau_B^{(j)}) &= \text{angle difference between } \tau_A^{(i)} \text{ and } \tau_B^{(j)}\\ d_G(\tau_A^{(i)},\tau_B^{(j)}) &= \mathbb{1}(g_A^{(i)}=g_B^{(j)})\\ w(\tau^{(i)};\gamma) &= exp(-\gamma\cdot||\tau^{(i)}||_2)\end{aligned}$$

The parameters α,β are for scaling translation and rotation errors, and gripper status errors, respectively. γ weighs the importance of a waypoint based on its distance to the object part. Finally, as trajectories vary in length, we normalize $D(m_A,m_B)$ by the number of waypoint pairs that contribute to the cumulative sum, $|D(m_A,m_B)|_{path^*}$ (i.e. the length of the optimal warping path), giving the final form:

$$distance(\tau_A,\tau_B)=\frac{D(m_A,m_B)}{|D(m_A,m_B)|_{path^*}}$$

This distance function is used for noise-handling in our model and as the final evaluation metric.

6 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. 4). 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

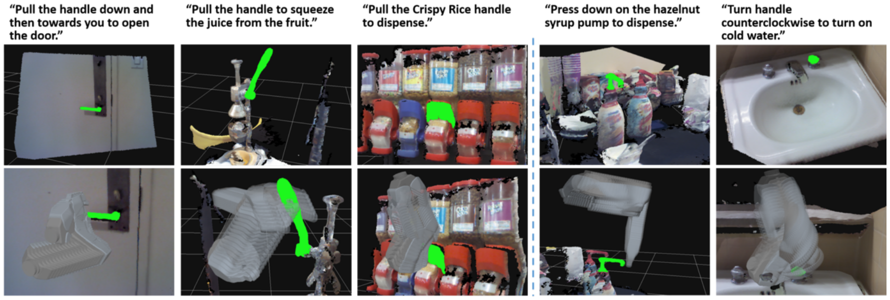

"Pull the handle down and then towards you to open the door."
"Pull the handle to squeeze the juice from the fruit."
"Pull the Crispy Rice handle to dispense."
"Press down on the hazelnut syrup pump to dispense."
"Turn handle counterclockwise to turn on cold water."

Fig. 5: **Examples from our dataset,** each of which consists 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).

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.

The user starts by selecting one of the instructions on the right to demonstrate (Fig. 4). Once selected, the target object part is highlighted and the trajectory *edit bar* appears below the 3-D viewer. Using the *edit bar*, which works like a video editor, the user can playback and edit the demonstration. Trajectory representation as a set of waypoints (Sec. 3.1) is directly shown on the *edit bar*. The bar shows not only the set of waypoints (red/green) but also the interpolated waypoints (gray). The user can click the 'play' button or hover the cursor over the edit bar to examine the current demonstration. The blurred trail of the current trajectory (*ghosted*) demonstration is also shown in the 3-D viewer to show its full expected path.

Generating a full trajectory from scratch can be difficult for non-experts. Thus, similar to Forbes et al. [15], we provide a trajectory that the system has already seen for another object as the initial starting trajectory to edit.[1]

In order to simulate a realistic experience of manipulation, instead of simply showing a static point-cloud, we have overlaid CAD models for parts such as 'handle' so that functional parts actually move as the user tries to manipulate the object.

A demonstration can be edited by: 1) modifying the position/orientation of a waypoint, 2) adding/removing a waypoint, and 3) opening/closing the gripper. Once a waypoint is selected, the PR2 gripper is shown with six directional arrows and three rings. Arrows are used to modify position while rings are used to modify the orientation. To add extra waypoints, the user can hover the cursor over an interpolated (gray) waypoint on the *edit bar* and click the plus(+) button. To remove an existing waypoint, the user can hover over it on the *edit bar* and click minus(-) to remove. As modification occurs, the edit bar and ghosted demonstration are updated with a new interpolation. Finally, for editing the status (open/close) of the gripper, the user can simply click on the gripper.

For broader accessibility, all functionality of Robobarista, including 3-D viewer, is built using Javascript and WebGL.

[1] We have made sure that it does not initialize with trajectories from other folds to keep *5-fold cross-validation* in experiment section valid.

7 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 5). 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 [22] 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`

Baselines. We compared our model against several baselines:
1) *Random Transfers (chance)*: Trajectories are selected at random from the set of trajectories in the training set.
2) *Object Part Classifier*: To test our hypothesis that intermediate step of classifying object part does not guarantee successful transfers, we built an object part classifier using multiclass SVM [58] on point-cloud features including local shape features [28], histogram of curvatures [42], and distribution of points. Once classified, the nearest neighbor among the same object part class is selected for transfer.
3) *Structured support vector machine (SSVM)*: It is a standard practice to hand-code features for SSVM [59], which is solved with the cutting plane method [24]. We used our loss function (Sec. 5) to train and experimented with many state-of-the-art features.
4) *Latent Structured SVM (LSSVM) + kinematic structure*: The way an object is manipulated depends on its internal structure, whether it has a revolute, prismatic, or fixed joint. Borrowing from Sturm et al. [51], we encode joint type, center of the joint, and axis of the joint as the latent variable $h \in \mathscr{H}$ in Latent SSVM [64].
5) *Task-Similarity Transfers + random*: It finds the most similar training task using (p,l) and transfer any one of the trajectories from the most similar task. The pair-wise similarities between the test case and every task of the training examples are computed by average mutual point-wise distance of two point-clouds after ICP [6] and similarity in bag-of-words representations of language.
6) *Task-similarity Transfers + weighting*: The previous method is problematic when non-expert demonstrations for the same task have varying qualities. Forbes et al. [15] introduces a score function for weighting demonstrations based on weighted distance to the "seed" (expert) demonstration. Adapting to our scenario of not having any expert demonstration, we select the τ that has the lowest average distance from all other demonstrations for the same task (similar to noise handling of Sec. 4).

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

7) *Our model without Multi-modal Layer*: This deep learning model concatenates all the input of three modalities and learns three hidden layers before the final layer.
8) *Our model without Noise Handling*: Our model is trained without noise handling. All of the trajectory collected from the crowd was trusted as a ground-truth label.
9) *Our model with Experts*: Our model is trained using trajectory demonstrations from an expert which were collected for evaluation purpose.

7.1 Results and Discussions

Table 1: **Results 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.

	per manual	**per instruction**	
Models	**DTW-MT**	**DTW-MT**	**Accuracy (%)**
chance	28.0 (±0.8)	27.8 (±0.6)	11.2 (±1.0)
object part classifier	-	22.9 (±2.2)	23.3 (±5.1)
Structured SVM	21.0 (±1.6)	21.4 (±1.6)	26.9 (±2.6)
LSSVM + kinematic [51]	17.4 (±0.9)	17.5 (±1.6)	40.8 (±2.5)
similarity + random	14.4 (±1.5)	13.5 (±1.4)	49.4 (±3.9)
similarity + weights [15]	13.3 (±1.2)	12.5 (±1.2)	53.7 (±5.8)
Ours w/o Multi-modal	13.7 (±1.6)	13.3 (±1.6)	51.9 (±7.9)
Ours w/o Noise-handling	14.0 (±2.3)	13.7 (±2.1)	49.7 (±10.0)
Ours with Experts	12.5 (±1.5)	12.1 (±1.6)	53.1 (±7.6)
Our Model	13.0 (±1.3)	12.2 (±1.1)	**60.0** (±5.1)

We evaluated all models on our dataset using *5-fold cross-validation* and the results are in Table 1. 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) from Sec. 5. 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 transferred 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.

Can manipulation trajectory be transferred from completely different objects? Our full model performed 60.0% in accuracy (Table 1), outperforming the chance as well as other baseline algorithms we tested on our dataset.

Fig. 6 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. 3.2), even though cereal and coffee dispenser handles are located and oriented differently, the transfer is a success. The second example shows a successful transfer from a DC power supply to a slow cooker, which have "knobs" of similar shape. The transfer was successful despite the difference in instructions ("Turn the switch.." and "Rotate the knob..") and object type.

The last example of Fig. 6 shows an unsuccessful transfer. Despite the similarity in two instructions, transfer was unsuccessful because the grinder's knob was facing towards the front and the speaker's knob was facing upwards. We fixed the z-axis along gravity because point-clouds are noisy and gravity can affect some manipulation tasks, but a more reliable method for finding the object coordinate frame and a better 3-D sensor should allow for more accurate transfers.

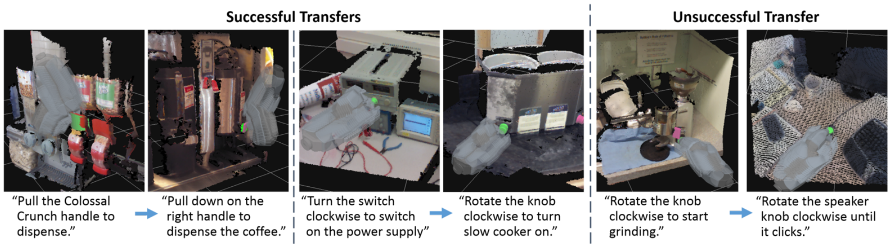

Successful Transfers
Unsuccessful Transfer
"Pull the Colossal Crunch handle to dispense."
"Pull down on the right handle to dispense the coffee."
"Turn the switch clockwise to switch on the power supply"
"Rotate the knob clockwise to turn slow cooker on."
"Rotate the knob clockwise to start grinding."
"Rotate the speaker knob clockwise until it clicks."

Fig. 6: **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.

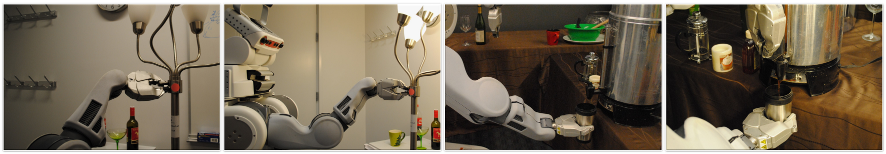


Fig. 7: **Examples of transferred trajectories** being executed on PR2. On the left, PR2 is able to rotate the 'knob' to turn the lamp on. On the right, using two transferred trajectories, PR2 is able to hold the cup below the 'nozzle' and press the 'lever' of 'coffee dispenser'.

Does it ensure that the object is actually correctly manipulated? 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. The contribution of this work is in the novel approach to manipulation planning which enables robots to manipulate objects they have never seen before. For some of the objects, correctly executing a transferred manipulation trajectory may require incorporating visual and force feedbacks [62, 60] in order for the execution to adapt exactly to the object as well as find a collision-free path [50].

Can we crowd-source the teaching of manipulation trajectories? When we trained our full model with expert demonstrations, which were collected for evaluation purposes, it performed at 53.1% compared to 60.0% by our model trained with crowd-sourced data. Even with the significant noise in the label as shown in last two examples of Fig. 5, we believe that our model with crowd demonstrations performed better because our model can handle noise and because deep learning benefits from having a larger amount of data. Also note that all of our crowd users are real non-expert users from Amazon Mechanical Turk.

Is segmentation required for the system? In vision community, even with the state-of-the-art techniques [14, 29], detection of 'manipulatable' object parts such as 'handle' and 'lever' in point-cloud is by itself a challenging problem [31]. Thus, we rely on human expert to pre-label parts of object to be manipulated. The point-cloud of the scene is over-segmented into thousands of supervoxels, from which the expert chooses the part of the object to be manipulated. Even with the input of the expert, segmented point-clouds are still extremely noisy because of the poor performance of the sensor on object parts with glossy surfaces.

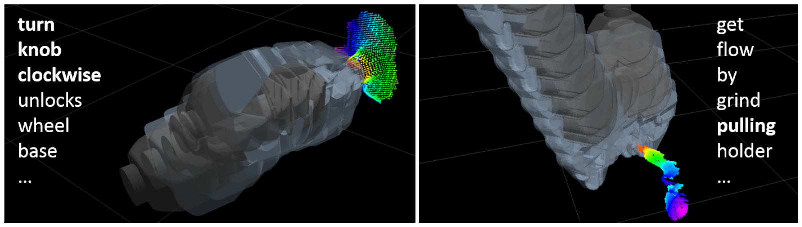


Fig. 8: **Visualization** of a sample of learned high-level feature (two nodes) at last hidden layer h^3. The point-cloud in the picture is given arbitrary axis-based color for visualization purpose. The left shows a node #1 at layer h^3 that learned to ("turn", "knob", "clockwise") along with relevant 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, and a trajectory that maximize the activation at each layer and passing the highest activated set of inputs to higher level.

Is intermediate object part labeling necessary? The *Object Part Classifier* performed at 23.3%, even though the multiclass SVM for finding object part label achieved over 70% accuracy in five major classes of object parts ('button', 'knob', 'handle', 'nozzle', 'lever') among 13 classes. Finding the part label is not sufficient for finding a good manipulation trajectory because of large variations. Thus, our model which does not need part labels outperforms the *Object Part Classifier*.

Can features be hand-coded? What kinds of features did the network learn? For both SSVM and LSSVM models, we experimented with several state-of-the-art features for many months, and they gave 40.8%. The *task similarity* method gave a better 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 at test time and requires a large storage as the size of training data increases.

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 the top layer h^3 such as those shown in Fig. 8. The left shows a node that correctly associated point-cloud (axis-based coloring), trajectory, and language for the motion of turning a knob clockwise. The right shows a node that correctly associated for the motion of pulling the handle.

Also, as shown for two other baselines using deep learning, when modalities were simply concatenated, it gave 51.9%, and when noisy labels were not handled, it gave only 49.7%. Both results show that our model can handle noise from crowd-sourcing while learning relations between three modalities.

7.2 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 trajectories to be transferred were found for the given point-clouds and languages. Given the trajectories which are defined as set of waypoints, the robot followed the trajectory by impedance controller (`ee_cart_imped`) [8]. Some of the examples of successful execution on PR2 robot are shown in Figure 7 and in video at the project website: `http://robobarista.cs.cornell.edu`

8 Conclusion

In this work, we introduced a novel approach to predicting manipulation trajectories via part based transfer, which allowed robots to successfully manipulate objects it has never seen before. We formulated it as a structured-output problem and presented a deep learning model capable of handling three completely different modalities of point-cloud, language, and trajectory while dealing with large noise in the manipulation demonstrations. We also designed a crowd-sourcing platform Robobarista that allowed non-expert users to easily give manipulation demonstration over the web. Our deep learning model was evaluated against many baselines on a large dataset of 249 object parts with 1225 crowd-sourced demonstrations. In future work, we plan to share the learned model using the knowledge-engine, RoboBrain [44].

Acknowledgments

We thank Joshua Reichler for building the initial prototype of the crowd-sourcing platform. We thank Ian Lenz and Ross Knepper for useful discussions. This research was funded in part by Microsoft Faculty Fellowship (to Saxena), NSF Career award (to Saxena) and Army Research Office.

[1] P. Abbeel, A. Coates, and A. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 2010.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine learning*, 1991.

[3] B. Alexander, K. Hsiao, C. Jenkins, B. Suay, and R. Toris. Robot web tools [ros topics]. *Robotics & Automation Magazine, IEEE*, 19(4):20–23, 2012.

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

[5] 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.

[6] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

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

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

[9] C. Crick, S. Osentoski, G. Jay, and O. C. Jenkins. Human and robot perception in large-scale learning from demonstration. In *HRI*. ACM, 2011.

[10] H. Dang and P. K. Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *IROS*, 2012.

[11] C. Daniel, G. Neumann, and J. Peters. Learning concurrent motor skills in versatile solution spaces. In *IROS*. IEEE, 2012.

[12] R. Detry, C. H. Ek, M. Madry, and D. Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *ICRA*, 2013.

[13] F. Endres, J. Trinkle, and W. Burgard. Learning the dynamics of doors for robotic manipulation. In *IROS*, 2013.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.

[15] 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.

[16] J. J. Gibson. *The ecological approach to visual perception*. Psychology Press, 1986.

[17] R. Girshick, P. Felzenszwalb, and D. McAllester. Object detection with grammar models. In *NIPS*, 2011.

[18] R. Hadsell, A. Erkan, P. Sermanet, M. Scoffier, U. Muller, and Y. LeCun. Deep belief net learning in a long-range vision system for autonomous off-road driving. In *IROS*, pages 628–633. IEEE, 2008.

[19] 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.

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

[21] N. Hu, Z. Lou, G. Englebienne, and B. Krse. Learning to recognize human activities from soft labeled data. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

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

[23] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning preferences for manipulation tasks from online coactive feedback. In *International Journal of Robotics Research (IJRR)*, 2015.

[24] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[25] D. Katz, M. Kazemi, J. Andrew Bagnell, and A. Stentz. Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects. In *ICRA*, pages 5003–5010. IEEE, 2013.

[26] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg. Cloud-based robot grasping with the google object recognition engine. In *ICRA*, 2013.

[27] H. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.

[28] H. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. *NIPS*, 2011.

[29] 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.

[30] O. Kroemer, E. Ugur, E. Oztop, and J. Peters. A kernel-based approach to direct action perception. In *ICRA*, 2012.

[31] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *ICRA*, 2014.

[32] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *RSS*, 2013.

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

[34] O. Mangin, P.-Y. Oudeyer, et al. Unsupervised learning of simultaneous motor primitives through imitation. In *IEEE ICDL-EPIROB*, 2011.

[35] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *IJRR*, 2012.

[36] D. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *RSS*, 2014.

[37] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *IJRR*, 32(3):263–279, 2013.

[38] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *ICML*, 2011.

[39] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009.

[40] M. Phillips, V. Hwang, S. Chitta, and M. Likhachev. Learning to plan for constrained manipulation from demonstrations. In *RSS*, 2013.

[41] S. Pillai, M. Walter, and S. Teller. Learning articulated motions from visual demonstration. In *RSS*, 2014.

[42] R. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, 2011.

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

[44] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula. Robo brain: Large-scale knowledge engine for robots. *Tech Report*, Aug 2014.

[45] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH*, 19(3):245–254, 1985.

[46] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 2011.

[47] R. Socher, J. Pennington, E. Huang, A. Ng, and C. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 2011.

[48] R. Socher, B. Huval, B. Bhat, C. Manning, and A. Ng. Convolutional-recursive deep learning for 3d object classification. In *NIPS*, 2012.

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

[50] M. Stilman. Task constrained motion planning in robot joint space. In *IROS*, 2007.

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

[52] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from rgbd images. In *ICRA*, 2012.

[53] J. Sung, B. Selman, and A. Saxena. Synthesizing manipulation sequences for under-specified tasks using unrolled markov random fields. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[54] S. Tellex, R. Knepper, A. Li, T. Howard, D. Rus, and N. Roy. Asking for help using inverse semantics. *RSS*, 2014.

[55] S. Thrun, W. Burgard, D. Fox, et al. *Probabilistic robotics*. MIT press Cambridge, 2005.

[56] R. Toris and S. Chernova. Robotsfor. me and robots for you. In *Proceedings of the Interactive Machine Learning Workshop, Intelligent User Interfaces Conference*, pages 10–12, 2013.

[57] R. Toris, D. Kent, and S. Chernova. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction*, 3(2):25–49, 2014.

[58] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*. ACM, 2004.

[59] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, and Y. Singer. Large margin methods for structured and interdependent output variables. *JMLR*, 6(9), 2005.

[60] F. Vina, Y. Bekiroglu, C. Smith, Y. Karayiannidis, and D. Kragic. Predicting slippage and learning manipulation affordances through gaussian process regression. In *Humanoids*, 2013.

[61] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[62] S. Wieland, D. Gonzalez-Aguirre, N. Vahrenkamp, T. Asfour, and R. Dillmann. Combining force and visual feedback for physical interaction tasks in humanoid robots. In *Humanoid Robots*, 2009.

[63] C. Wu, I. Lenz, and A. Saxena. Hierarchical semantic labeling for task-relevant rgb-d perception. In *RSS*, 2014.

[64] C.-N. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*, 2009.

[65] M. D. Zeiler, M. Ranzato, R. Monga, et al. On rectified linear units for speech processing. In *ICASSP*, 2013.