Scene-level Programming by Demonstration

Zhen Zeng  Zheming Zhou  Zhiqiang Sui  Odest Chadwicke Jenkins

Abstract—Scene-level Programming by Demonstration (PbD) is faced with an important challenge - perceptual uncertainty. Addressing this problem, we present a scene-level PbD paradigm that programs robots to perform goal-directed manipulation in unstructured environments with grounded perception. Scene estimation is enabled by our discriminatively-informed generative scene estimation method (DIGEST). Given scene observations, DIGEST utilizes candidates from discriminative object detectors to generate and evaluate hypothesized scenes of object poses. Scene graphs are generated from the estimated object poses, which in turn is used in the PbD system for high-level task planning. We demonstrate that DIGEST performs better than existing method and is robust to false positive detections. Building a PbD system on DIGEST, we show experiments of programming a Fetch robot to set up a tray for delivery with various objects through demonstration of goal scenes.

I. INTRODUCTION

We aim to provide a system for a user to effectively program a robot to complete manipulation tasks. We focus on understanding the goal of manipulation tasks. The user only needs to demonstrate once to the robot a desired scene state. As the scene changes, the robot should be able to manipulate the objects in the scene to restore the desired scene state. In tasks such as organizing objects in a household environment, users can program the robot to keep the living room at a tidy state.

The most related field to this objective is Programming by Demonstration (PbD), and similar notions of learning by imitation, which provide a natural way for naive users to convey skills and tasks to robots. To effectively program a robot in a general manner, the robot should be able to understand the goal of the task, i.e., what to imitate. With an understood goal, a robot can reason and adapt its actions to reach this goal in various scenarios and changing environments.

While there has been considerable advances in robot learning, scene perception remains a challenge in general goal-directed manipulations. Traditional PbD works using kinesthetic teaching [11], [19], [4], [1], [21] focused on robot learning of a configuration-space control policy for a particular task or skill. Scene perception in workspace is important in that a robot should be able to adapt a learned skill to current scene. The perceptual uncertainty of scene perception has limited robot task-level reasoning to limited domains due to issues of computational tractability.

In this paper, we describe our approach to goal-directed PbD over scenes. Our work is motivated by a service robot scenario, as illustrated in Figure 1. We argue that scene perception is a critical missing piece for general goal-directed PbD. With proposed scene estimation method DIGEST, we demonstrate how goal-directed PbD can be performed at the level of scenes in different scenarios of setting up a tray. A critical distinction of our work is that we are estimating goals as workspace scene graphs, represented by object relations and poses, instead of the configuration space of the robot. Thus we focus on estimating the goal as the desired scene state, s.t. any capable robot can perform the task.

Our contribution is two-fold:

- We propose a discriminatively-informed generative method (DIGEST) to estimate object poses in cluttered scenes for goal-directed manipulation
- We present a goal-directed PbD system at the level of scene graphs generated from the estimated object poses, where the robot is an operator in the scene

Our experiments involve 1) evaluating the goal-directed PbD system in tray setup tasks with a Fetch Robotics robot, 2) evaluating DIGEST in cluttered scene. And DIGEST outperforms the D2P [20] on the household occlusion dataset [2].

II. RELATED WORK

A. Scene Perception for Manipulation

Feature-based methods and analysis-by-synthesis are two traditional approaches for model-based object pose estimation. Feature-based methods try to match hand designed or learned features between models and observations. Spin
images [12] and FPFH [22] are two local feature based methods, where the features are extracted and matched between models and observation for pose estimation. OUR-CVFH [3] and VFH [23] are examples of global feature based methods that are more robust to object occlusions. Different features are learned with respect to the object pose. However, the performance of feature-based methods will degrade when the environment becomes more cluttered and key features are occluded.

Analysis-by-synthesis, also referred as rendering and verification method, is a generative approach that renders multiple hypotheses, and finds the hypothesis that best explains the observation. The early works by Stevens et al. [24] proposed an iterative hypothesize, render and verify process to estimate 3D pose of an object. Our previous work APF [26] [27] use MCMC to search for the scene hypothesis that best explains the observation. Similarly, D2P [20] uses A* as the search method to estimate 3 Dof object poses. Both APF and D2P assume object identities are known a priori. However, there does not exist an ideal recognition system that can identify all the objects correctly in a cluttered environment yet. DIGEST aims to avoid such a strong assumption.

B. Goal-Directed PbD

There have been some works in PbD focusing on learning low level control. Grollman and Jenkins [9] simultaneously segment a demonstrated task and learn the policy for each individual subtask. Chernova and Veloso [6] present an interactive algorithm for policy learning from demonstration. Ijspeert et al. [11] and Nakanishi et al. [19] teach robot biped locomotion patterns through demonstrations. In an incremental learning system by Calinon and Billard [4], a HOAP-3 humanoid learns different gestures via generalizing over demonstrated joint angle trajectories. Akgun et al. [1] learns generalized trajectory by identifying a set of keyframes. Nieken et al. [21] presents an approach to discover finite-state representations of multi-step tasks. For domains such as locomotion and gesture control, it is sufficient to generalize over the robot configuration space and proprioceptive contacts. While in the domain of robot manipulations, in contrast, it is important to generalize over new scenes in the workspace.

Goal-directed PbD is complementary to aforementioned works, in addition to learning generalized trajectories over control from demonstration, perceiving and inferring goal from demonstrations are also important. In goal-directed PbD, understanding the goal of a demonstrated scene is faced with challenges such as perceptual uncertainty, especially in cluttered scenes involving occlusions and object interactions. Mohan et al. [18] and Kirk et al. [15] present a cognitive system that learns task formulations, and perform goal-directed control with grounded perception. However, they assume that all the objects are solid-colored, so that the visual system can infer the scene state reliably. Similarly, Chao et al. [5] limit the demonstrations to planar objects with distinguished colors. Unlike these works, our perception system does not rely on assumptions of object colors or planar objects.

To deal with scene perception uncertainty, the robot can make a decision and act based on a belief space of the scene, which can be formulated by POMDP [13], but the problem becomes intractable as the state space increases. Kaelbling and Lozano-Perez [14] build their work on blending probabilistic and task inference in a logical representation of belief space. We argue that if viable scene perception can be realized, then decision making and probabilistic inference of the scene can be decoupled as in the domain of autonomous navigation. In our work, we maintain a distribution over the scene state, i.e., the inter-object relations as well as the object poses. We use the maximal likely scene state as the scene estimate, and perform goal-directed reasoning based on the scene estimate of the demonstrated and start scene.

III. Problem Statement

Assume a robot R as a physically capable agent in the scene, a set of geometries V = {v1, . . . , vK} of known objects in the scene, and a set of manipulation actions A = {a1, . . . , aN} with known pre- and post- conditions. Our objective is to infer goal scene state sG and current scene state st at time t from the observations of user’s desired scene αG and current scene αt, respectively, and plan a sequence of actions {ai, . . . , aj} for R to carry out to reorganize the scene such that the scene state transits from st to sG.

We use a list of axiomatic assertions to describe a scene. The scene state at time t is expressed as a scene graph st = {h′(x)}t≥0, where h′t ∈ {exist, clear, on, in} is an axiomatic assertion parametrized by x′t = {w′t(q′t, v′t)}t≥0, with w′t(q′t, v′t) representing that at time t object w′t has pose q′t and geometry v′t, Nt being the number of objects, and M is the total number of axiomatic assertions. In our work, the assertions are limited to spatial relations that can be tested geometrically or physically.

The 6 Dof pose q′t = [x′t, y′t, z′t, φ′t, ψ′t, θ′t] of each object is estimated, which consists of a 3D position (x′t, y′t, z′t) and
orientation ($\phi_i^j, \psi_i^j, \theta_i^j$). The number of objects $N_c$ in the scene is known in advance, but the object categories are not known. Scene graph can be inferred from estimated object poses, as explained later in section IV-B.

### IV. Methods

Perception is the core of our approach to PbD. To address the perceptual uncertainty, we aim to decouple probabilistic scene state estimation and action planning s.t. the robot takes actions based on the current estimate of the scene. Our pipeline consists of perception, plan and execution stages, as shown in Figure 2. Given observations of a cluttered scene, the generative process of scene estimation is informed by a pre-trained object detector. A scene graph that represents the inter-object relations are geometrically inferred from the estimated object poses.

#### A. DIGEST: Cluttered Scene Estimation

Given observation $o_t$ as the depth image of a cluttered scene at time $t$, the objective is to estimate the object poses $q_i^j, j = 1, \cdots, N_c$. We utilize the discriminative power of a pre-trained object detector and classifier to first obtain a set of bounding boxes and object labels. These bounding boxes are used to guide the generative process of sampling scene hypothesis. The overview of the cluttered scene estimation is as illustrated in Figure 3.

1) Object Detection and Hypothesized Scene Generation: Any object detection method that can produce candidates including bounding boxes and object labels from the observation suits our method. Assume there are $m$ bounding boxes detected from the object detector. Let $B_i$ ($0 \leq i \leq m$) denote each bounding box, $L_j$ ($1 \leq j \leq k, k$ is the number of object class) denote each object label in the bounding box $B_i$, and $v(L_j|B_i)$ is the confidence of object label $L_j$ in the bounding box $B_i$. For each $B_i$, we generate a candidate $C_i$,

$$C_i = \{\arg \max_{L_j} v(L_j|B_i), B_i\}$$

which is a set including the object label with highest confidence score and the current bounding box.

For $m$ generated candidates, the number of hypothesized scene $h$ is defined as:

$$h = \begin{cases} 
  mC_{N_c}, & \text{if } N_c \leq m \\
  1, & \text{otherwise} 
\end{cases}$$

So if there are more or equal candidates than objects in the scene, each hypothesized scene $H_i$ contains a combination of $N_c$ candidates selected from $m$ candidates. If there are fewer candidates than $N_c$, just one hypothesized scene with $m$ candidates will be generated.

2) Particle Filtering for Pose Estimation: Each hypothesized scene $H_i$ is modeled as a random state variable $x_i$, comprised of a set of real-valued object poses. We model the inference of the state $x_i$ from a history of robot observations $z_{1:t}$ as a sequential Bayesian filter,

$$p(x_i|z_{1:t}) \propto \int p(x_i|x_{i-1}, u_{i-1})p(x_{i-1}|z_{1:i-1})dx_{i-1}$$

The posterior belief is formed at time $t$ by updating a prior belief from time $t-1$ with a dynamic resampling and likelihood evaluation step. The sequential Bayesian filter in Eq. 3 is often approximated by a collection of $N$ weighted

---

**Fig. 3: DIGEST pipeline.** First, the test RGB image is passed through an R-CNN object detector trained on our dataset and a set of bounding boxes with object labels is generated. Then possible combinations of hypothesized scenes are enumerated and evaluated against the pre-processed observed point cloud. The particle filtering pose estimation method evaluated each scene and produced weight. At last, the result comes from the scene with the max weight.
particles, \( \{x^{(j)}_t, w^{(j)}_t\}_{j=1}^N \) with weight \( w^{(j)}_t \) for particle \( x^{(j)}_t \), expressed as:

\[
p(x_t|z_{1:t}) \propto \sum_j w^{(j)}_{t-1} p(x_t|x^{(j)}_{t-1})
\]

as described by [7]. Inference is then performed by computing the likelihood of each hypothesis, normalizing the weights to one, and drawing \( N \) scene hypotheses by importance sampling iteratively.

\[
x^{(j)}_t \sim \sum_j w^{(j)}_{t-1} p(x_t|x^{(j)}_{t-1})
\]

\[
w^{(j)}_t = w^{(j)}_{t-1} / \sum_k w^{(k)}_{t-1}
\]

Through the z-buffer of a 3D graphics engine, each particle \( x^{(j)}_t \) is rendered into a depth image, represented as \( z^{(j)}_t \). We project back the rendered depth image into a point cloud \( \hat{P}^{(j)}_t \) in camera frame given intrinsic parameters. Then, we compute the likelihood for each particle with the observation \( z^*_t \):

\[
p(z^*_t|x^{(j)}_t) = e^{-\lambda_d d(R, O)}
\]

where \( \lambda_d \) is a constant scaling factor and \( d(R, O) \) is the Euclidean distance between the rendered point cloud and the observation point cloud:

\[
d(R, O) = \sum_{(a,b) \in I} \| (R(a,b) - O(a,b)) \|
\]

where \( a \) and \( b \) are indices in rendered point cloud \( R \) and observed point cloud \( O \), and \( || \cdot || \) is the norm of the norm of a 3D vector.

We apply Iterated Likelihood Weighting [17] to bootstrap the scene estimation process, where the state transition in the action model is represented by a zero-mean Gaussian noise. Once the bootstrap filter converges, the scene \( H_t \) from the most likely particle \( \hat{x}_t \) is taken as the scene estimate:

\[
H_t = \arg \max \ p(x^{(j)}_t|z_{1:t})
\]

and the unnormalized \( w^{(j)}_t \) weight for the mostly likely scene is taken as the weight for ranking the all scenes.

3) Final Scene Ranking: After evaluating all hypothesized scenes, we rank them based on the weight computed from our pose estimation method. The scene hypothesis with highest weight is taken as the maximal likely scene estimate.

B. Scene Graph Structure

The objects pose estimation of a cluttered scene can be turned into a scene graph composed by a set of axiomatic assertions that describes the scene. We use \( \text{exist}(w^i(q^i, v^i)) \) to assert that object \( w^i \) exists in the scene with pose \( q^i \) and geometry \( v^i \). \( \text{clear}(w^i) \) for assertion that the top of object \( w^i \) is clear and no other objects are stacked on it. \( \text{on}(w^j, w^i) \) for assertion that object \( w^j \) is stacked on object \( w^i \). An example of a scene graph is given in Figure 4.

We extend the tree-structured scene graph representation used in APF [26] by introducing a scene element (virtual object). With a virtual object \( w^v(q^v, v^v) \), a scene graph can express proximity relations between objects. To assert the proximity relations between two objects \( w^j, w^i \), we add a virtual object \( w^v(q^v, v^v) \) into the scene graph, with \( v^v \) being an arbitrary shape, and \( q^v \) expressed in the frame of object \( w^i \). Then, the spatial relation between \( w^j, w^i \) can be encoded by \( \text{has}(w^j, w^i), \text{in}(w^j, w^i) \), where \( \text{has}(w^j, w^i) \) asserts that \( w^j \) has an virtual object \( w^v \) attached to its frame. When the parent object \( w^j \) is in a new location, the robot can adapt to the new scenario by placing the child object \( w^j \) within the region of \( w^v \) attached to the frame of \( w^i \).

We assume that one object is supported by a single object, versus that one object is supported by multiple objects at the same time. In each 3D object model, the z-axis is the gravitational axis when the object stands upright, the x-axis is the gravitational axis when the object lies on a ground, and y-axis is the cross product of the other two axes. The dimensions \( \{h_x, h_y, h_z\} \) of the 3D box that encloses each object model are provided. In order to decide whether object \( w^i \) is being supported by another object, two heuristics are tested: 1) if one of the object axes (e.g. x-axis) is aligned with the gravitational axis, then the height \( h_z \) of the 3D volume occupied by the object equals to the corresponding dimension (e.g. \( h_z \)) of the provided 3D enclosing box. A simple rule \( z^i - h_{table} > 0.5H^i \) implies whether object \( w_i \) is being supported by another object; 2) if none of the object
axes are aligned with the gravitational axis, then $w_i$ is being supported by another object.

The set of objects that are being supported by other objects are denoted as $O_s$, and the rest of the objects are being denoted as $O_t$. For object $w^j \in O_s$, a heuristic measure is used to determine which object $w^j \in O_t$ is supporting $w^j$,

$$\arg\max_j m(r_p(w^j), r_i(w^j))$$

where $m(r_1, r_2)$ measures the overlapping area of two regions $r_1, r_2$. $r_p(w), r_i(w)$ represent the projected region on the horizontal plane of the top and bottom surface of object $w$, respectively. With identified supporting relation between a pair of objects $w^i, w^j$, the corresponding axiomatic assertion is expressed as either $on(w^i, w^j)$ or $in(w^i, w^j)$, depending on the geometry type of the supporting object $w^j$ being convex or concave.

V. IMPLEMENTATION

A. RCNN object detector

We employ RCNN [8] as our discriminative object detector for DIGEST. RCNN first generates object proposals given an image and then classifies the proposals using a deep convolutional neural network. For the sake of efficiency and performance, we replaced the original selective search [28] with EdgeBox [29] for object proposal generation.

B. Particle Filtering and parallelization

The implementation of particle filtering pose estimation method consists of three modules: measurement, resampling and diffusion. For each object in each particle $x_t(i)$, it is initialized by the candidate $C_{i_t}$ in the hypothesized scene, the object label $l_i$ determines which object to sample, and the initial pose is uniformly sampled from the bounding box $B_{j_t}$. Then these particles are given as input to a parallel graphics engine which generates scene estimation depth images rapidly. The measurement module takes a preprocessed point cloud as the observation and compares it with the point cloud back projected from the rendered depth image. Resampling and diffusion will give the new posterior distribution and add noise to the current state. We set the fixed particle filter iteration to 400 and employ 625 particles for all the experiments.

In the particle filtering process, the pose of each object is estimated sequentially. For example, if there are four hypothesized objects and 400 iterations for particle filtering, the pose of the object with the largest recognition confidence is estimated in the first 100 iterations. Then the pose of the object with the second largest recognition confidence is estimated in the next 100 iterations, with the first object fixed at the converged estimated pose. We iterate the same estimation process for the rest objects.

C. Planning

Given a demonstrated goal scene, the robot estimates the goal scene state, i.e., the object poses and desired inter-object relations, and stores the goal scene state by PDDL [16]. Similarly the robot estimates and stores the start scene state by PDDL. Note, that each pose label in PDDL is associated with an object pose in the robot base frame, s.t. the robot knows where to pick or place objects during action execution. For example, as shown in the lower right part of Figure 4, lipton is at pos_lipton, where pos_lipton is a pose label for lipton that is associated with the estimated lipton pose in the robot base frame. With sets of PDDL that describe the start and goal scene, the robot uses symbolic planner to plan a series of actions to reorganize the start scene to match the goal scene step-by-step. We use breadth first search for the symbolic planner in our experiments.

D. Manipulation Execution

To execute the planned actions, the robot relies on existing motion planning library to generate collision-free arm trajectories. We use Moveit! [25] in our experiment. In our experiment, the object manipulation actions are essentially a sequence of pick and place actions. To pick up an object, we
pre-defined a set of possible pre-grasp and grasp end-effector poses w.r.t the object, i.e., grasping from the top and from the side. Affordance template [10] can be incorporated to extend our pipeline to deal with more complex manipulation behavior.

VI. EXPERIMENTS

We evaluated 1) the accuracy of our cluttered scene estimator DIGEST in public dataset and our dataset, and 2) the performance the overall goal-directed PbD pipeline in tasks of setting up a tray.

A. DIGEST: Cluttered Scene Estimation

In this section, we examine our DIGEST estimator in two datasets: household occlusion dataset by Aldoma et al. [2] and our own scene estimation dataset with more objects and higher occlusion scenario. For the household occlusion dataset, we compare our method with D2P [20] under the same assumption and heuristics. For our own dataset, we trained a RCNN object detector and evaluated our work with the same metrics. All the experiments are ran on a Ubuntu 14.04 system with an Titan X Graphics card and CUDA 7.5.

1) Household Occlusion Dataset: We evaluate generative power of DIGEST with D2P, the state-of-art multi-object identification and 3 Dof localization method by Narayanan et al. [20] on the household occlusion dataset [2]. The dataset contains 36 household objects with 3D geometry models and 22 RGBD test scenes with 80 objects in total. There are at most four objects in the scene. All objects are standing upright in the dataset, thus in this experiment, only 3 Dof object pose \((x,y,\theta)\) is estimated instead of 6 Dof, where \(x,y\) are the 2D locations of the object and \(\theta\) is the yaw angle of the object.

Here, we take the same assumption as D2P: known object number and object categories. We also take exactly the same deep learning heuristics generated from D2P and use it in the same way. For each ROI in the depth image, candidates are generated only from the known object in the scene with high confidence score. The threshold is set to 0.2 as in D2P.

Pose estimation accuracy is percentage of correctly localized objects over the total number of objects in the dataset. An object is correctly localized if the pose error falls into a certain error bound. The pose error is computed the same way as in D2P. As shown in Figure 3, DIGEST performs better than D2P in small error bounds (position error smaller than 1cm and 10cm) and performs on par with D2P in larger position error bound. For position error bound with 1cm, DIGEST performs much better result than D2P since that 1) DIGEST explores the state space a lot more than D2P as we do not discretize the state space, and 2) DIGEST does not use ICP for local search, which D2P employs for their pose estimation step. DIGEST takes around 30 seconds (varying with the number of objects and the size of object geometries), which is faster than 139.74 seconds by D2P.

2) Our Dataset: We collected our own RGBD dataset for 15 household objects and 3D models. It has 20 test scenes with more objects and occlusions in each scene. There are 104 objects in total in all the scenes. We also generated ground truth pose by manually matching geometry models with observed point cloud in Blender. As we only estimate 3 Dof object pose, we take the ground truth value of the other three dimensions \((z,\text{roll},\text{pitch})\).

We also employ the same criteria from the previous experiment to evaluate the pose errors, as shown in Figure 6. The increasing tail in the plots is because some objects with complex geometry like waterpot or spray bottle sometimes has a flipped orientation in the estimation. The proposed method DIGEST has the ability to resolve false positives from the discriminative object detectors by leveraging the robustness of generative estimation, as shown in Figure 7.

B. PbD Pipeline: Set up a Tray

We design our experiments with the服务 robot scenario in mind, as illustrated in Figure 1. The robot needs to prepare a tray of food such that it can be delivered to an user. We tested our system on scenes of three to five objects plus a tray, with objects stacking, and/or placed next to each other.

The goal of the robot is to reproduce the topological layout of the goal scene, i.e., to reproduce the same inter-object relations, and the object poses can vary from the goal scene as long as the inter-object relations are satisfied.

1) Proof of Concept: To provide a proof of concept on our overall goal-directed PbD paradigm, we first tested our system with objects that are mostly primitive shaped, and with the assumption that the categories of the objects are known, and each object is standing upright, as in D2P [20]. Given the assumptions above, we used a previous method, i.e., APF [26] for 3 Dof object pose and scene graph estimation. In the later section, we relaxed these assumptions and used DIGEST for 6 Dof object pose estimation.

As summarized in Figure 8, we considered object stacking scenario and objects that are placed near each other. Each object has a set of pre-defined grasp pose, i.e., grasping from the top and from the side. During the manipulation experiments, the robot iterates through the provided grasp...
poses, and select the one that it can plan a trajectory for.

2) **PbD Enabled by DIGEST:** We relax the assumptions in the previous section, thus the object categories are not known, and the objects can be in arbitrary poses. The test scenes are composed by objects that we have trained the object detector on. We used DIGEST to estimate the 6 Dof object poses, and then scene graph is derived from the pose estimation results.

As shown in Figure 10, the goal and start scenes are well estimated, as a collection of 6 Dof poses of objects. Based on the scene graph inferred from the object pose estimates, the robot generated a sequence of symbolic actions to transit the scene from the start state to the goal state. The pose estimation does fall into local minima sometimes, resulting in flipped orientation of objects that is highly symmetric, e.g., spray bottle and clorox bottle.

In these experiments, the grasp pose for each object is defined by the user instead of using existing work for grasp pose localization on objects. The reason for the user-defined grasp pose is that when the robot picks up an object, not all the valid grasp poses will be equivalently good for a later placement action. An example of the robot manipulation action sequence is as shown in Figure 9. A video of the experiments are available at [https://youtu.be/H1OMNQ1qDA](https://youtu.be/H1OMNQ1qDA).

**VII. CONCLUSION**

We demonstrate a PbD paradigm for users to easily program a robot to complete manipulation tasks, where is goal of the task is represented as workspace scene graphs. We address the scene perception problem using the proposed DIGEST method, which is a discriminatively-informed generative method that recognizes objects and estimates their 6 Dof poses for manipulation. We show that DIGEST performs better than D2P in public dataset for 3 Dof pose estimation and also performs well in our own dataset. We also show that users can use our overall PbD pipeline building on DIGEST to successfully program a Fetch robot to complete manipulation tasks of setting up a tray.

**REFERENCES**


Fig. 9: Example of robot manipulation action sequence that transits the scene from the start state to the goal state via our scene-level PbD.

Fig. 10: We tested our pipeline for 5 different goal scenes. For each goal scene, two different starting scenes were tested. The rendered scene based on the object pose estimates are shown as depth images. The first four goal scenes involve object stacking, and the final goal scene involves objects placed near each other. In the experiments, the robot has successfully transited the scene from the start state to the goal state through manipulation actions.