Article

The International Journal of
Robotics Research
2017, Vol. 36(1) 86–104

DOI: 10.1177/0278364916683444
journals.sagepub.com/home/ijr

Goal-directed robot manipulation through axiomatic scene estimation

Zhiqiang Sui[1], Lingzhu Xiang[2], Odest C Jenkins[1] and Karthik Desingh[1]

Abstract

Performing robust goal-directed manipulation tasks remains a crucial challenge for autonomous robots. In an ideal case, shared autonomous control of manipulators would allow human users to specify their intent as a goal state and have the robot reason over the actions and motions to achieve this goal. However, realizing this goal remains elusive due to the problem of perceiving the robot's environment. We address and describe the problem of axiomatic scene estimation for robot manipulation in cluttered scenes which is the estimation of a tree-structured scene graph describing the configuration of objects observed from robot sensing. We propose generative approaches to scene inference (as the *axiomatic particle filter*, and the *axiomatic scene estimation by Markov chain Monte Carlo* based sampler) *of the robot's environment as a scene graph. The result from AxScEs estimation are axioms amenable to goal-directed manipulation through symbolic inference for task planning and collision-free motion planning and execution. We demonstrate the results for goal-directed manipulation of multi-object scenes by a PR2 robot.*

Keywords

Goal-directed robot manipulation, scene estimation for manipulation, integrated perception tasks planning and motion planning

1. Introduction

Performing robust goal-directed sequential manipulation is an ongoing and critical challenge for autonomous robots, for which perception has been the main bottleneck. In an ideal case, shared autonomous control of manipulators would allow human users to specify their intent as a goal state (i.e. the desired configuration of the world) without being required to specify how this goal should be achieved (either as motions or actions). Such human-expressed goals could then be realized autonomously by a robot through reasoning over sequences of actions and motion controls. There have been considerable advances in reasoning for robot decision making and purposeful robot motion, both of which are increasingly converging. However, robots still lack the general ability to perceive the world, especially in typical human environments with considerable clutter. This lack of perception greatly limits the ability and generality of robots to reliably make decisions, carry out manipulation actions, and learn from human users. From a practical perspective, the limited ability to perceive in common human environments often restricts robots to simulation and/or highly controlled environments.

Addressing perception for manipulation, we describe and address the problem of *axiomatic scene estimation (AxScEs)*, pronounced "access", for robot manipulation in cluttered scenes. Figure 1 shows goal-directed manipulation in action from an AxScEs estimate of a cluttered scene of eight objects with a Willow Garage PR2 robot. We phrase the problem of AxScEs as the estimation of a tree-structured scene graph describing the configuration of objects observed from robot sensing. Similar to their use in computer graphics, scene graphs are represented as parameterized axiomatic statements that assert relationships between objects in the scene graph and the poses and geometry of each object. Our generative approach to inference for problems of AxScEs iteratively hypothesizes possible scene configurations (as lists of axioms) and evaluates these hypotheses against the robot's observations (currently as depth images). The result of this inference is an

[1]Department of Electrical Engineering and Computer Science, University of Michigan, USA
[2]Institute for Aerospace Studies, University of Toronto, Canada

Corresponding author:
Zhiqiang Sui, Laboratory for Perception Robotics and Grounded Reasoning Systems, Department of Electrical Engineering and Computer Science, University of Michigan, 2260 Hayward St., Ann Arbor, MI, 48109-2121, USA.
Email: zsui@umich.edu

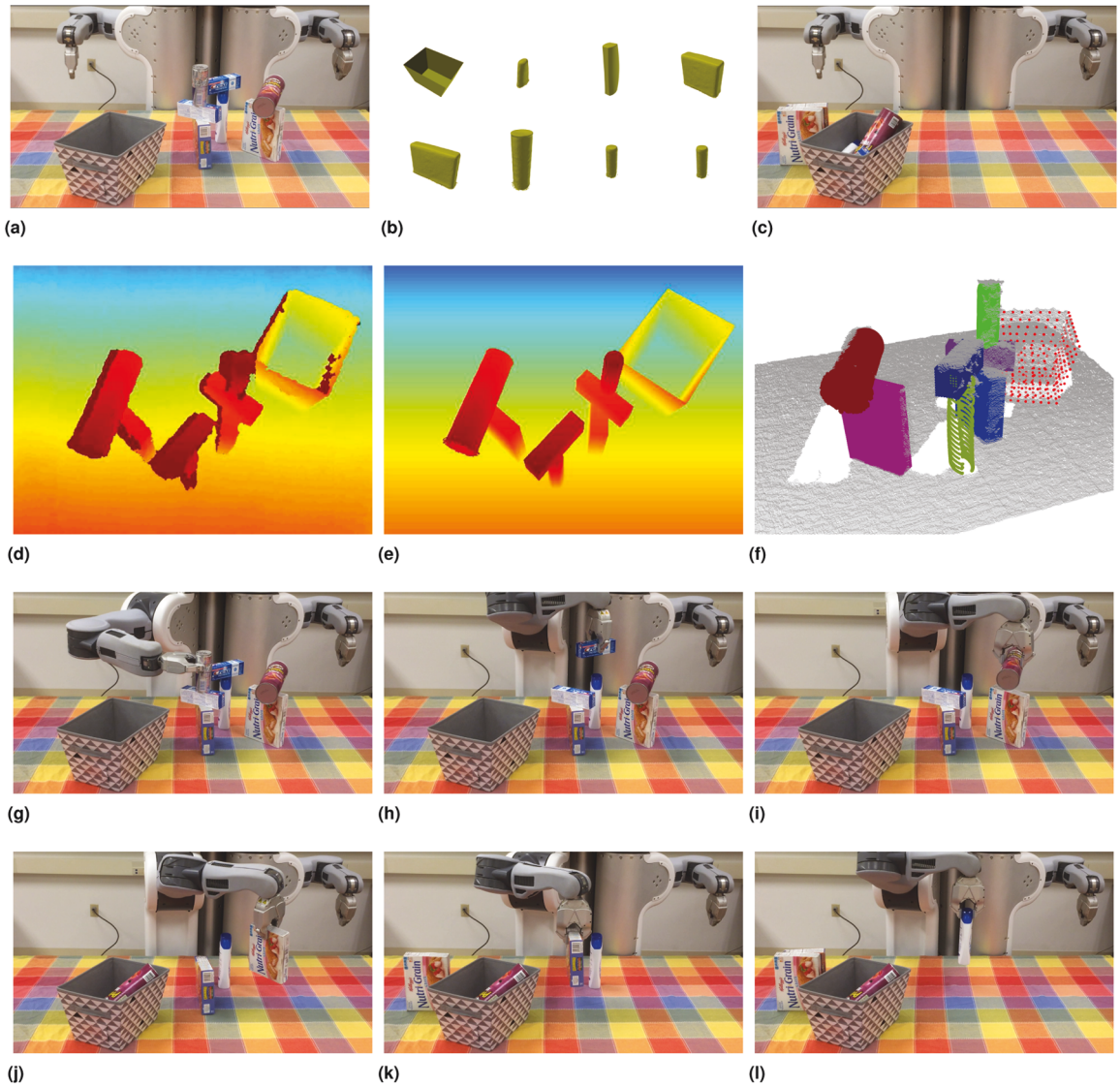

Fig. 1. An example of AxScEs estimation and goal-directed manipulation of a (a) cluttered scene with (b) eight objects to a (c) goal state of putting small objects in a bin and boxes to the side. AxScEs estimation generatively infers from robot observations (d) the maximally likely scene configuration (e) and (f). The resulting AxSeEs is used to plan and execute actions (g) to (l) to the user-specified goal state (c). (a) Robot in initial scene. (b) Known geometries for each object. (c) Achieved user-specified goal scene. (d) Depth image observation of initial scene. (e) Estimated initial scene in depth image. (f) Estimated initial scene in point cloud. (g) Action: Pick red bull, place bin. (h) Action: pick toothpaste box, place bin. (i) Action: pick pringles, place bin. (j) Action: Pick nutrigrain, place side region. (k) Action: Pick nature_valley, place side region. (l) Reach goal: Pick shampoo, place bin.

approximate posterior probability distribution over possible scenes, where the scene with maximum likelihood is taken as the estimate of the scene configuration. Though probabilistic in nature, a principal motivation for providing AxScEs estimates represented in axiomatic form is the "closing the loop" between goal-directed symbolic planners (Fikes and Nilsson, 1972; Laird et al., 1987) and modern robotics. Such planners reason over manipulation actions for the robot to execute from an AxScEs estimate towards realizing a given goal scene, also expressed axiomatically. Planned sequences of actions are then executed by motion control/planning systems for pick-and-place manipulation (Ciocarlie et al., 2014) or general manipulation affordances (Hart et al., 2015).

The remainder of this article describes the problem of axiomatic scene estimation, proposes instances of AxScEs estimators, and examines their use for goal-directed manipulation in scenes of increasing numbers of physically stacked objects. We describe a formulation for the problem of axiomatic scene estimation in Section 2. An analysis of the growth of possible tree-structured scene graphs is presented with respect to the number of stackable objects in a scene. Section 3 motivates the need for generative approaches to problems of AxScEs as a matter

of inclusion towards bridging probabilistic and symbolic inference for goal-directed manipulation. Section 4 covers related work in goal-directed robot manipulation in relation to shared autonomy, goal-directed control, perception, and decision making under uncertainty. In Section 5, we phrase the problem AxScEs as a probabilistic state estimation model that factors into inference of scene tree relations and object pose. Within the AxScEs model, we cast our previous work, the *axiomatic particle filter (AxPF)* (Sui et al., 2015), as an exhaustive search over scene tree relations with particle filter inference of poses. We also introduce *axiomatic scene estimation by Markov chain Monte Carlo (MCMC) sampling (AxMC)* using the Metropolis–Hastings algorithm (Hastings, 1970) to search scene tree relations with pose estimation likelihoods. Our GPU-optimized parallel implementation of AxScEs estimation for both the AxPF and AxMC are described in Section 6. This likelihood works directly with depth images from common ranging cameras without the need for computing discriminative features. Our experiments with this implementation are described in Section 7. These results indicate that AxScEs estimators are effective for manipulation-quality perception based on edit distance on scene graphs, estimation of currently manipulatable (clear) objects, and object pose estimation. More precisely, exhaustive search over scene graphs with AxPF estimates yields with high accuracy, but, due to computational complexity, is limited to relatively small collections of scenes. In contrast, AxMC estimates can be generated within tractability but with less accuracy. We conclude in Section 8 with a discussion of our current AxScEs estimators and a summary of directions for future work.

2. Problem statement

The objective of axiomatic scene estimation is to infer a symbolic description of the scene $\hat{\mathbf{S}}$ from partial observations z_t by a given robot at time t. This symbolic scene description can then be readily used by modern task and motion planners to generate sequential actions that will autonomously control the robot to achieve a user-expressed desired goal state $\mathbf{S}_G$.

Axiomatic state x_t at time t is defined as a collection of axioms expressing possible scenes $\mathbf{S}$. A scene is expressed as a scene graph $\mathbf{S}(\mathbf{W}(\mathbf{Q}, \mathbf{V}))$ as a set of axiomatic assertions describing the pose Q_i and geometry V_i of each object W_i and relations for object interactions and affordances. The planning domain definition language (PDDL) (McDermott et al., 1998) is used to model axiomatic state as a formal language, which implicitly defines a tree-structured scene graph. Shown in Figure 2, an example of a scene graph of a four object scene is represented in PDDL. To avoid ambiguity, we restrict the set of axioms to only spatial and physical expressions that can be tested geometrically or through physical simulation. These axioms assert the existence of an object W_i, as *(*W_i `object`*)*, with spatial geometry V_i, *(*`geom` W_i V_i*)*, and spatial pose configuration Q_i, *(*`pose` W_i Q_i*)*. Axioms also assert parent-child relationships between objects as whether an object W_i is inside another object W_j, *(*`in` W_i W_j*)*, or resting on another object W_k, *(*`on` W_i W_k*)*, as well as whether the object is the possession of a robot R, *(*`has` R W_i*)*. Each of these inter-object relations induces a spatial frame relation, where the frame of a child object is expressed in relation to its parent object.

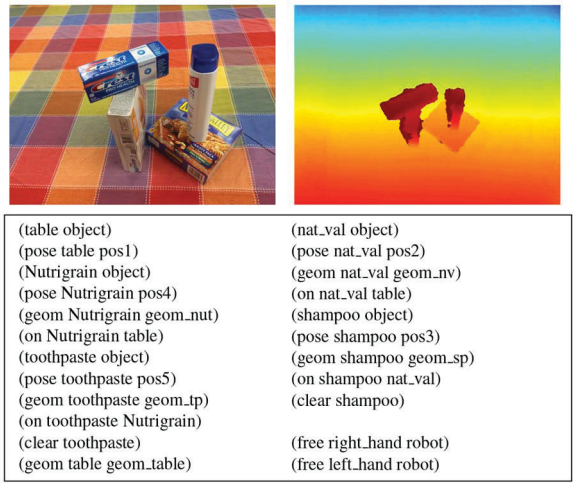

Fig. 2. Axiomatic scene estimation for an example four-object scene (top left), observed by the robot as a depth image (top right), will estimate the pose and spatial relations of objects as parameterized axiomatic assertions.

Given the AxScEs relations above, the relation *(*`clear` W_i*)* is asserted for each object W_i that is not supporting another object. The objects asserting this relation can be picked by the robot or used as a support surface for placing other objects. For general manipulation affordances, additional axioms can be created that describe assertions for preconditions and postconditions for actions associated with objects. Precondition and postconditions axioms are envisioned to resemble collision-based "trigger" conditions, widely used to script interactive behaviors in video games through programming languages such as Lua (Miller et al., 2009).

2.1. Assumptions

For the methods presented in this article, we address the problem of AxScEs for inferring the axioms in the scene $\mathbf{S}$ (Section 5.2) and the three degree-of-freedom (DOF) poses of each object $\mathbf{Q}$ (Section 5.4). The n objects comprising of $\mathbf{W}$ are assumed to have been uniquely identified with each having known spatial geometries contained in $\mathbf{V}$. This assumption is made based on using an ideal of common visual object recognition systems (Collet et al., 2011; Felzenszwalb et al., 2010) as a preprocessing step. Only the inter-object relation for stacking (`on`) is considered for AxScEs estimation, although the relations for enveloping (`in`) and grasping (`has`) are considered for task planning. Objects are assumed to be upright oriented and can take

on any pose on the support surface provided by its parent object. As such, the object poses of **Q** consists of a 2D position and yaw rotation (SE(2) group) in the coordinates provided by its parent object.

2.2. Scene graph enumeration

In the general case, scene estimation in this axiomatic form can lead to a very high dimensional belief space that would theoretically pose problems for probabilistic inference. For this work, we will assume scene graphs are tree-structured, have an implied base support plane, and consider only the stacking case (asserted by the "on" relation). Thus, a single object could physically support any number of other objects, but is itself physically supported by one other object. With these assumptions, let $T(n)$ be the number of possible scene graphs, given n objects in a scene. Then, the total number of scenes can be expressed recursively

$$T(n) = \sum_{k=1}^{n} {}^{n}C_k g(n-k, k) \tag{1}$$

where ${}^{n}C_k$ is the number of combinations for selecting a subset of k objects out of the n objects and $g(s,k)$ is the number of scene graphs possible for stacking s objects on top of a fixed scene on k base objects

$$g(s,k) = \sum_{s_1=0}^{s}\sum_{s_2=0}^{s}\ldots\sum_{s_i=0}^{s}\ldots\sum_{s_k=0}^{s} \frac{s!}{s_1!s_2!\ldots s_i!\ldots s_k!} T(s_1)\,T(s_2)\ldots T(s_i)\ldots T(s_k) \tag{2}$$

where $s = \sum_{i=1}^{k} s_i$.

When $n = 0$, $T(0) = 1$ as the number of scenes with no objects as one. Similarly when $n = 1$, $T(1) = 1$ expresses the number of scenes with one object. When $n = 2$, $T(2) = 3$ breaks down into two terms ${}^{2}C_1 * g(1,1)$ and ${}^{2}C_2 * g(0,2)$ with respect to the recursion in equation (2). The first term considers the number of ways one object can be placed on one supporting object $g(1,1) = 1$ times the number of ways each of the two objects to each of these stacking roles ${}^{2}C_1 = 2$. $g(0,2) = 1$ is the number of ways two objects can be placed on the table, for which there is only one combination for stacking. Relationally, $T(2) = 3$ expresses the three possible axiomatic scene graphs for two objects: Objects A and B are not stacked, object A is stacked on object B, object B is stacked on object A.

When $n = 3$, $T(3) = 16$ has three terms ${}^{3}C_1 * g(2,1)$, ${}^{3}C_2 * g(1,2)$ and ${}^{3}C_3 * g(0,3)$. The first term ${}^{3}C_1 * g(2,1)$ denotes the number of ways of choosing one object out of three objects, $g(2,1)$ denotes the number of ways one object can be placed on two supporting objects. Computing each of the terms turns out to be 9, 6, and 1 respectively and hence $T(3) = 16$.

Following this recursive expression of equation (2), $T(3) = 16$, $T(4) = 125$, $T(5) = 1296$, $T(6) = 16807$, $T(7) = 262144$, $T(8) = 4782969$, and so on. This recursive expansion provides an upper bound as it assumes each object is capable of providing a support surface for all other objects. However, we speculate that most of these scene possibilities are actually implausible physically and statistically improbable to be encountered in common manipulation settings and human environments. That stated, naively performing state estimation in such a huge state space becomes intractable quickly as the number of objects grows. Such inference can still be of considerable use for manipulation as tabletop environments can often consist of small stacks of objects. In such cases, a tabletop segmentation algorithm, such as for the PR2 interactive manipulation (Ciocarlie et al., 2014) can be used to identify clusters of stacked objects, each of which can be treated as their own scene.

3. A generative approach to AxScEs

To motivate the problem of AxScEs , consider the scene in Figure 3 observed from 3D point clouds captured from the robot's perspective. For this scene, assume the goal for the robot is to grab the bottom green block to give to a human user. It can be clearly observed that *block1* (the top block) and *block2* (the bottom block) are two distinct objects from the perspective of human perception. A naive perception of this scene, common to most robots, would instead perceive the objects that are physically touching as a single object as shown in the right. From the perspective of common segmentation methods for 3D point clouds, estimates of the scene as two smaller objects or a single larger object are equally likely parsings of the robot's observations.

To capture this uncertainty, our approach to problems of AxScEs is to maintain a distribution across plausible scene graph hypotheses supported by the robot's point cloud observations. These generated hypotheses form an approximate probability distribution (or belief) over possible states of the scene. This ambiguity over possible scenes can be resolved at a later time with further information, such as after a robot action to grasp one of the objects. In addition, by maintaining diverse perspectives, the robot can use either one of these hypotheses as an estimate of the scene state to plan and execute a current course of action. If the chosen state estimate was incorrect, the alternate hypothesis of the scene should still be represented in the diversity of the belief distribution. Assuming the result of the action resolved the ambiguity, this alternate state hypothesis will now have a greater likelihood given the new point cloud observation. This distribution will now clearly distinguish the alternate as the true scene state estimate from which the robot's plan can be recomputed. This approach to scene estimation is implemented by a system architecture, whose details are described in Section 6.1.

Our approach to AxScEs aims to emulate and scale the highly effective and now ubiquitous pattern of decoupled decision making and probabilistic inference for

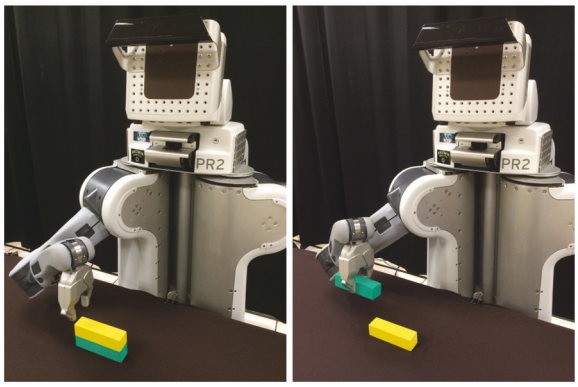

Fig. 3. Example of a robot needing to grasp an object (the green block) in a simple stack scene. The robot needs to estimate the scene but faces ambiguity about whether there are two stacked objects or one larger object. Once estimated, the robot needs to perform a sequence of actions to move the yellow block and then, once cleared, grasp the green block.

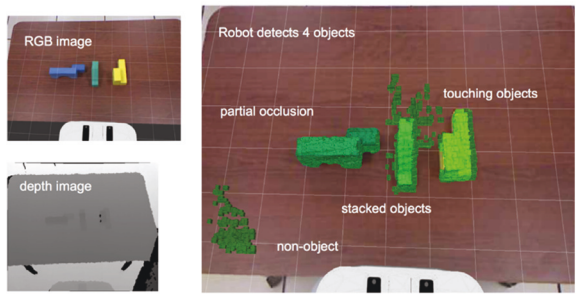

Fig. 4. Sources common of perception errors for physically interacting objects, as shown for depth-based object segmentation using the PR2 interactive manipulation (Ciocarlie et al., 2014). Due to issues of uncertainty in open-loop perception, reliable autonomous robot manipulation is currently limited to distinctly separated objects on flat tabletops. For example, consider the scene of six simple blocks on a tabletop and its depth image from a Kinect RGBD camera on the head of a PR2 robot. The robot can find the tabletop, assuming it is the largest flat object in the scene. However, it cannot distinguish (and thus grasp directly) any of the individual objects due to occlusions, physical interaction, and false positives. This uncertainty motivates the need for both further action-oriented information in the form of affordances and human-in-the-loop control to handle errors and ambiguity.

autonomous navigation (Biswas and Veloso, 2013; Dellaert et al., 1999) in the domain of robot manipulation. Specifically, probabilistic perception and symbolic planning are treated as independent processes, allowing each to focus on what they do best. These processes interoperably interface through communication of the maximally likely axiomatic state estimate and selected robot action (or operator). While avoiding the intractability of planning in the space of beliefs, this decoupling assumes state estimates are a plausibly accurate representation of the current state of the world.

Unlike autonomous navigation, AxScEs estimation faces a drastically large state space where generative inference must balance estimation accuracy and computational tractability. Our first efforts for AxScEs estimation proposed the *axiomatic particle filter (AxPF)* (Sui et al., 2015). This original description of the AxPF performs particle filtering (Dellaert et al., 1999) over an exhaustive enumeration of scene graph structures and demonstrated our core approach to goal-directed manipulation. While theoretically applicable to highly cluttered scenes of n objects, our initial AxPF faced a number of practical and computational challenges for general AxScEs problems. First, this implementation of the AxPF used a non-optimized serial pipeline for evaluating scene hypotheses, as particles. As we discuss later, such implementation issues can be addressed and scaled to large numbers of particle hypotheses using multi-core processing and GPU-based 3D processing.

More importantly, the scene dimensionality grows rapidly towards intractability as the number of objects in the scene increases, exceeding factorial growth. Such expansive state spaces prohibit exhaustive search over scenes, even with an optimized processing pipeline. Furthermore, our search space consists of state variables with mixed types over both non-binary tree structures and real-valued object pose parameters. As described later, we explore sampling based algorithms, including Markov chain Monte Carlo (MCMC) and particle filtering, suited to the diverse types and high-dimensionality of axiomatic scenes.

4. Related work

4.1. Shared autonomy for manipulation

In order for autonomous robots to interact fluidly with human partners, a robot must be able to interpret scenes in the context of a human user's model of the world. The challenge is that many aspects of the human's world model are difficult or impossible for the robot to sense directly. We posit the critical missing component is the grounding of symbols that conceptually tie together low-level perception and high-level reasoning for extended goal-directed autonomy. We specifically face the problem of anchoring (Coradeschi and Saffiotti, 2003), a case of symbol grounding (Harnad, 1990), to associate physical objects in the real world and relationships between these objects with computationally assertable facts (or axioms), from the robot's perception of the world. Anchoring and symbol grounding are at the heart of the emerging area of semantic mapping (Kuipers, 2000) and its accelerated growth due to advancements in 3D RGBD mapping (Herbst et al., 2011; Rusu et al., 2008a). With a working memory of grounded axioms about the world, robot manipulators will be able to flexibly and autonomously perform goal-directed tasks that require reasoning over sequential actions (illustrated in Figure 1). Just as important, human users will be able to more intuitively specify goals for robots, as desired states of the world, through spatial configurations.

In the greater context of shared autonomy, goal-directed manipulation offers the opportunity to extend the boundaries of the "neglect curve" (Goodrich et al., 2001). The neglect curve is a conceptual expression of robot effectiveness with respect to delegation (or user neglect), codifying tradeoffs between the extremes of full autonomy and manual teleoperation. While teleoperation can often yield high-levels of robot effectiveness, the performance of such systems rely heavily upon the training, aptitude, and stamina of a human operator. Conversely, systems for autonomous robots place much less burden on a human operator but are often limited to generalized trajectories over controls (Akgun et al., 2012; Calinon and Billard, 2007; Jenkins and Matarić, 2004; Pastor et al., 2011), reactive policies (Chernova and Veloso, 2009; Crick et al., 2011; Grollman and Jenkins, 2008; Niekum and Barto., 2012; Platt et al., 2010; Vondrak et al., 2012), or goals as combinations of hardcoded features (Abbeel and Ng, 2004; Atkeson and Schaal, 1997; Kober and Peters, 2011; Nicolescu and Matarić, 2003; Smart and Kaelbling, 2002). As evidenced during the recent DARPA robotics challenge (Yanco et al., 2015), shared autonomy is especially onerous and error prone for control of humanoids and mobile manipulators due to the complexity of goal-directed control. Similar to the pointing work of Kemp et al. (2008), our long-term conjecture is that shared autonomy through the expression of goals will greatly reduce the complexity for human operation of robots, improving robot effectiveness during periods of delegation.

4.2. Goal-directed manipulation

Our aim is to estimate axiomatic state representations that will allow robotics to build on the body of work in sequential planning algorithms, which have over a five-decade history. Described in early work, such as Stanford Research Institute Problem Solver (STRIPS; Fikes and Nilsson, 1972) and SHRDLU (SHRDLU is an natural language understanding computer program which contains planner that can answer queries; Winograd, 1972), classical planning algorithms adapted theorem-provers to "prove" conclusions about goals based on symbolic axioms that describe the world through assertable logical statements. A classical planner can compute actions for a physical robot to perform arbitrary sequential tasks assuming full perception of the environment, which is often an untenable assumption in general.

However, in structured perceivable environments, systems based on classical planning have demonstrated the ability to reliably perform goal-directed manipulation. Recent work by Kirk and Laird (2013); Mohan et al. (2012) uses the Soar cognitive architecture for teaching a robot arm to play games such as tic-tac-toe, Connect-4, and Towers of Hanoi through language-based expressions. Similar in spirit to our AxScEs estimators, Soar uses an axiomatic scene graph representation (Wintermute and Laird, 2008). We posit AxScEs estimates could also be used within broader cognitive architectures, such as ACT-R/E (Trafton et al., 2013), that are suited to axiomatic rather than strictly metric spatial representations. Chao et al. (2011) perform taskable symbolic goal-directed manipulation with a focus on associating observed robot percepts with knowledge categories. This method uses background subtraction to adaptively build appearance models of objects and obtain percepts but with sensitivity to lighting and object color. Narayanaswamy et al. (2011) perform scene estimation and goal-directed robot manipulation for cluttered scenes of toy parts for flexible assembly of structures.

The KnowRob system of Tenorth and Beetz (2013) performs taskable goal-directed sequential manipulation at the scale of entire buildings by automatically synthesizing sources from the semantic web and Internet. Leveraging the community of perception modules available in the robot operating system (ROS) (Quigley et al., 2009), KnowRob focuses uncertainty at the symbolic level and relies on hard and complete state estimates from hardcoded software components. Similarly, Srivastava et al. (2013) rely on hardcoded perception systems to perform the joint task and motion planning, taking advantage of modifications in controlled environments, which include green screens and augmented reality tags.

4.3. Perception for manipulation

While domains with uncertainty are traditionally problematic for classical planning, we posit that advances in robot perception and manipulation with new approaches to anchoring can overcome this uncertainty for goal-directed robot control. There have been a number of discriminative methods proposed to perceive exact single estimates of scene state for manipulation, which both complement and inspire probabilistic AxScEs estimation. Based on the semantic mapping work of Rusu et al. (2008b), the canonical manipulation baseline is the PR2 interactive manipulation pipeline (Ciocarlie et al., 2014) (Figure 4). The grasp planner from this pipeline is used for results presented in this article. This pipeline is able to perform relatively reliable pick-and-place manipulation for non-touching objects in flat tabletop settings. This pipeline relies upon the estimation of the largest flat surface, by the clustering of computed surface normals. Any contiguous mass of points extruding from this support surface is considered a single object, leading to many false positives in object recognition and pose estimation. Rosman and Ramamoorthy (2011) addressed such point cloud segmentation issues in relational scene graph estimation by detecting contact points between objects that can be directly observed from depth. Collet et al. (2009) proposed a system for recognition and pose registration of common household objects from a single image by using local descriptors. In aims similar to this

article, Papazov et al. (2012) perform sequential pick-and-place manipulation using a bottom-up approach of matching known 3D object geometries to point clouds using Random sample consensus (RANSAC) and retrieval by hashing methods. Cosgun et al. (2011) presented a novel algorithm for placing objects by performing a sequence of manipulation actions in cluttered surfaces, like the tabletop. Beyond the scope of this paper, our work aims to use the robust grasp estimation methods of ten Pas and Platt (2014), which are able to localize graspable points in highly unstructured scenes of diverse unknown objects.

In the real world the robot's attempts to perceive are dominated by uncertainty in the robot's sensing and action. As such, the hard estimates from discriminative perception can prove a difficult match for classical planning. Uncertainty is a result of both measurements by the sensors and performance by the motors that control the robot. For example, sensor measurements are frequently inadequate for segmentation of objects in contact, or identification of occluded or partially visible objects (Figure 4). The resulting noisy and incomplete descriptions of scene state are unsuitable inputs for existing classical planning algorithms, affecting both the robot's axiomatic representation of that world and its ability to perform effectively.

The AxMC method we propose in this article is very similar to the recently proposed knowledge-supervised (KS) MCMC method of Liu et al. (2015). KSMCMC uses MCMC to sample over scene graph structures represented axiomatically to estimate objects as oriented bounding boxes. Pose estimation is performed using image features for alignment. Joho et al. (2012) use a generative model to cluster objects on a flat surface into semantically meaningful categories. In a similar manner, Dogar and Srinivasa (2011); Dogar et al. (2012) consider active manipulation of highly occluded non-touching objects on flat surfaces.

4.4. Manipulation under uncertainty

Generative inference provides a means to address uncertainty probabilistically. In our case, a generative approach maintains a distribution over all possible scene graphs and is not reliant upon selecting and maintaining a hard (potentially incorrect) state estimate for perception. Possible world states can be hypothesized to explain possibilities for the true world state that could have generated the robot's observations. These generated hypotheses form an approximate probability distribution (or belief) over possible states of the world.

Recent work by Choi and Christensen (2013) used OpenGL interoperating with CUDA which is a parallel computing platform, for fast particle generation and likelihood evaluation for single object pose tracking with ground truth initialization. Their likelihood evaluation was feature based, with color, 3D point coordinates, and surface normals extracted from depth measurements, similar to work by Fallon et al. (2012) for depth-based localization. In contrast, likelihood evaluation in our proposed system uses a direct method based on photometric error minimization between depth camera measurements and particle filter estimates, without extracting hard features from the depth images. In the implementation, we overcome the inability to access depth values from CUDA by deriving depth values in OpenGL shaders, and employ modern scene rendering techniques to improve rendering performance.

In terms of generative inference, recent work by Zhang and Trinkle (2012) formulated a physics-informed particle filter, Grasping-Simultaneous Localization, and Modeling, and Manipulation (G-SLAM), for grasp acquisition in occluded scenes. While well-suited for grasp acquisition, we posit an axiomatic representation is needed for moving to manipulation tasks where reasoning over sequential actions is required.

It is tempting to characterize the entire problem of goal-directed manipulation as belief space planning within a Partially observable Markov decision process (POMDP; Kaelbling et al., 1998). The state of the world is only partially observable in the POMDP formulation, and the process of a robot making a decision and then acting is formed as a Markov process over the space of all possible world states. POMDPs provide a complete conceptual model for the problem of goal-directed manipulation under uncertainty. However, this completeness comes at the cost of computational infeasibility for all but a small number of discrete-state problems. For robotic manipulation, Lang et al. (2012) attempt to overcome the limitations of the POMDP through online relational reinforcement learning, using physical simulation for exploration. Particle filtering has also been used to combine the symbolic and statistical approaches (Manfredotti et al., 2010; Zettlemoyer et al., 2007) in structured domains. In their recent work in robot manipulation, Kaelbling and Lozano-Pérez (2013) build on similar notions to blend probabilistic and symbolic inference into a single process. In this work, belief space planning occurs over logical assertions that each generates distributions of probability and combine hierarchically to solve for combined task and motion plans.

5. AxScEs scene estimation methods

In this section, we present our methods for axiomatic scene estimation, which are used with in the system architecture (Section 6.1) for goal-directed manipulation. For axiomatic scene estimation, we represent the configuration of a scene at given time $\mathbf{S}_t$ as a random state variable x_t to be inferred from robot observations z_t. This scene state variable $x_t = [g_t, q_t]$ is comprised of both real-valued object poses, as random variable $q_t \in \Re^3$, and set-valued lists of axioms, as random variable g_t. In our case, the axioms g_t define the topology of objects in a scene as a tree. We only have a few options to perform inference with a state variable dimensions of mixed type. Among these inference options, one

option is our original brute force method, the *axiomatic particle filter (AxPF)* (Sui et al., 2015). The AxPF exhaustively marginalizes over combinations of scene axioms g_t and performs inference over object poses q_t through particle filtering on robot observations z_t. In the context of the *AxPF*, we additionally explore object pose estimation using MCMC sampling.

Avoiding exploration over all possible scenes, another approach to AxScEs inference is to search over scenes with algorithms amenable to general data structures, such as a hill climbing optimization or MCMC algorithm. Similar to Liu et al. (2015), such an inference procedure samples over possible scenes g_t where pose estimation on q_t, on robot observations z_t, is performed for each sampled scene. Our proposed *axiomatic scene estimation by MCMC sampling (AxScEs MCMCs)* takes this form. AxMC performs scene inference of g_t with the MCMC-based Metropolis–Hastings algorithm and pose inference of q_t with a particle filter. AxMC works directly with depth images without the need for discriminative features, as used by Liu et al. (2015) or Collet et al. (2011). Further, AxMC provides distributions over both scene structure and object poses, which conceptually allows for update over time as we consider future work.

5.1. Axiomatic particle filter

For the Axiomatic Particle Filter (AxPF), we modeled the inference of axiomatic state x_t from a history of robot observations $z_{1:t}$ as a sequential Bayesian filter. This model consists of updating a prior belief from time $t-1$ with a dynamic resampling and likelihood evaluation to form a new posterior belief at time t

$$p(x_t|z_{1:t}) \propto p(z_t|x_t) \int p(x_t|x_{t-1},u_{t-1})\, p(x_{t-1}|z_{1:t-1})\, \mathrm{d}x_{t-1} \quad (3)$$

Although results presented are for observations of static scenes, the dynamics term $p(x_t|x_{t-1},u_{t-1})$ in this formulation is to retain generality for tracking the scene as the robot performs an action u_{t-1}. As described by Dellaert et al. (1999), the sequential Bayesian filter in equation (3) is commonly approximated by a collection of N weighted particles, $\{x_t^{(j)}, w_t^{(j)}\}_{j=1}^{N}$, with weight $w_t^{(j)}$ for particle $x_t^{(j)}$, expressed as

$$p(x_t|z_{1:t}) \approx p(z_t|x_t) \sum_j w_{t-1}^{(j)} p(x_t|x_{t-1}^{(j)}, u_{t-1}) \quad (4)$$

Over successive iterations, inference in the particle filter is performed by drawing N scene hypotheses by importance sampling, evaluating the likelihood of each hypothesis, and normalizing the weights to sum to one

$$x_t^{(j)} \sim \sum_j w_{t-1}^{(i)} p(x_t|x_{t-1}^{(i)}, u_{t-1}) \quad (5)$$

$$w_t^{*(j)} = p(z_t|x_t^{(j)}) \quad (6)$$

$$w_t^{(j)} = \frac{w_t^{*(j)}}{\sum_k w_t^{*(k)}} \quad (7)$$

Within the likelihood $p(z_t|x_t^{(j)})$, the scene $\mathbf{S}^{(j)}$ associated with each particle $x_t^{(j)}$ is rendered into a depth image $\hat{z}_t^{(j)}$, through the z-buffer of a 3D graphics engine, for comparison with the robots current observation z_t

$$p(z_t|x_t^{(j)}) = e^{-\lambda_r \cdot \mathrm{SSD}(z, \hat{z}_t^{(j)})} \quad (8)$$

where λ_r is a constant scaling factor and $\mathrm{SSD}(I, I')$ is the sum of squares distance function (SSD) between depth images I and I'

$$\mathrm{SSD}(I, I') = \sum_{(a,b)\in z} (I(a,b) - I'(a,b))^2 \quad (9)$$

where a and b are 2D image indices. Once the posterior distribution converges about a single scene hypothesis, the scene $\hat{\mathbf{S}}_t$ from the most likely particle $\hat{x}_t$ is taken as the scene estimate

$$\hat{\mathbf{S}}_t = \arg\max_{x_t^{(j)}} p(x_t^{(j)}|z_{1:t}) \quad (10)$$

This axiomatic scene estimate $\hat{\mathbf{S}}_t$ is used for planning robot actions and motion towards a given goal state $\mathbf{S}_G$, which is also expressed in axiomatic form.

5.2. AxScEs formulation

Assuming for a moment computing with unbounded resources, the AxPF described above still lacks concepts that define operations for resampling, diffusion, and dynamics on mixed-type scene state. We decompose the AxScEs problem as an expression of the probability of a scene into pose q_t and scene tree structure g (assuming a static scene for clarity)

$$p(x_t|z_{1:t}) = p(g, q_t|z_{1:t}) \quad (11)$$

$$= p(g|z_{1:t})\, p(q_t|g, z_{1:t}) \quad (12)$$

$$\propto p(g|z_{1:t})\, p(g, z_t|q_t) \quad (13)$$

$$\int p(q_t|q_{t-1}, u_{t-1})\, p(q_{t-1}|g, z_{1:t-1})\, \mathrm{d}q_{t-1}$$

$$\approx p(g|z_{1:t})\, p(g, z_t|q_t) \sum_j w_{t-1}^{(j)} p(q_t|q_{t-1}^{(j)}, u_{t-1}) \quad (14)$$

The expressions decompose the problem of AxScEs into a scene tree factor $p(g|z_{1:t})$ and an object pose factor $p(q_t|g, z_{1:t})$. For inference, we assume the scene tree factor is unknown and treat the object pose factor as a likelihood of a pose being given a scene tree. Inference of the scene tree structure allows maintenance of the distribution over scenes in relation to object poses over time, approximated

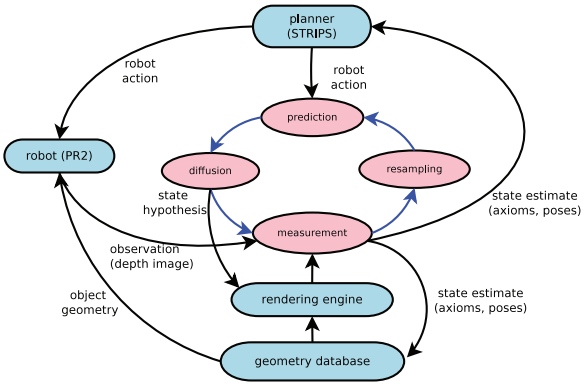

Fig. 5. Architecture diagram for pose estimation within goal-directed manipulation with respect to the AxPF . Pose estimation and manipulation components are respectively highlighted in red and blue.

by particle filtering. For goal-directed planning, our primary concern is obtaining a scene estimate $\hat{\mathbf{S}}$ from these distributions, which leads to our formulation of AxScEs

$$\hat{\mathbf{S}} = \arg\max_{x_t} p(x_t|z_{1:t}) \tag{15}$$

$$= \arg\max_{g,q_t} p(g|z_{1:t})\, p(q_t|g,z_{1:t}) \tag{16}$$

$$\approx \arg\max_{g,q_t} p(g|z_{1:t}) \tag{17}$$

$$\left[\arg\max_{q_t} p(g,z_t|q_t) \sum_j w_{t-1}^{(j)} p(q_t|q_{t-1}^{(j)}, u_{t-1})\right]$$

In this AxScEs formulation, we cast the *AxPF* as assuming the distribution of scene trees $p(g|z_{1:t})$ to be uniform. Consequently, pose inference over q_t on factor $p(q_t|g,z_{1:t})$ is performed on each possible scene, as assignments of g to exhaustive combinations of tree axioms. Pose inference for each tree is performed with independent particle filters, as described in equation (4), executing in parallel. The AxScEs estimate $\mathbf{S}_t$ is taken as the scene and pose associated with the maximally likely estimate produced across all of these particle filters.

5.3. Axiomatic Monte Carlo Markov chain

We now describe AxMC as a method to perform axiomatic scene estimation using MCMC and particle filtering over, respectively, scene trees g and object poses q_t. With respect to equation (12), we cast inference of the unknown scene tree factor $p(g|z_{1:t})$ as the target distribution for MCMC sampling. Just as in the AxPF, particle filtering is performed on the pose factor $p(q_t|g,z_{1:t})$ and is treated as a likelihood for a sampled scene tree. The AxScEs estimate $\mathbf{S}_t$ is taken as the scene tree and pose associated with the maximally likely sample from the AxMC process.

MCMC sampling uses the single-site Metropolis–Hastings algorithm to approximate the target distribution. In each iteration of Metropolis–Hastings, a proposal distribution $p'(g^*|g^{(i)})$ is used to generate a new sample g^* local to the previous sample $g^{(i)}$. As we describe in Section 5.4, a common instance of Metropolis–Hastings for a real-valued vector space has this local sampling occurring using a normally distributed proposal.

For our tree-valued variable g, generation of a proposal sample $g^* \sim \mathcal{T}(g^{(i)})$ occurs with respect to a tree kernel $\mathcal{T}(g^{(i)})$ that performs a single random edit to the tree $g^{(i)}$. The sampling of $\mathcal{T}(g^{(i)})$ randomly selects two different nodes, a and b, of $g^{(i)}$ to perform one of three permutation operations, also selected at random:

(a) swap a and b;

(b) move a to be the child of b;

(c) move b to the child of a.

These operations are carried out by changing the `on` relations for objects associated with tree nodes for a and b.

The newly sampled scene tree g^* is either accepted or rejected with probability

$$A(g^{(i)}, g^*) = \min\left\{1, \frac{\arg\max_{q_t} p(q_t|g^*, z_{1:t})}{\arg\max_{q_t} p(q_t|g^{(i)}, z_{1:t}))}\right\} \tag{18}$$

based on the respective maximally likely pose estimates for each scene tree. In other words, the sample g^* is accepted if $A(g^{(i)}, g^*)$ is greater than a uniformly generated random number between zero and one. After accepting a fixed number of N samples $G^* = \{g^{(i)}\}_{i=1}^N$, the scene tree estimate $\hat{g}$ is taken as the sample with the highest likelihood with respect to the likelihood over pose estimates q_i, as expressed in equation (17).

5.4. MCMC pose estimation

As an alternative to particle filtering, we have additionally investigated an MCMC approach to pose estimation of q_i as a likelihood for known scene graph g. This pose inference used a single-site Metropolis–Hastings algorithm to approximate the target distribution $p(q|z,g)$ as a sampled Markov chain, where g is a known set of scene graph axioms. In each iteration of Metropolis–Hastings, a proposal distribution $p'(q^*|q^{(i)})$ is used to generate a new sample $q^* \sim \mathcal{N}(q^{(i)}, \Sigma_q)$ from a normal distribution centered on the previous sample $q^{(i)}$ with covariance Σ_q over the space of all pose dimensions. Alternatively, this sampling can be done per object with normal distributions in the space of DOFs for each object. The newly sampled particle q^* is either accepted or rejected with acceptance probability

$$A(q^{(i)}, q^*) = min\left\{1, \frac{w(q^*)}{w(q^{(i)})}\right\} \tag{19}$$

where $w(q)$ is the likelihood of pose state q, as specified in equations (6) and (8).

After accepting a fixed number of N samples $Q^* = \{q^{(i)}\}_{i=1}^N$, the pose estimate $\hat{q}$ is taken as the sample with the highest likelihood with respect to the likelihood w

$$\hat{q} = \arg\max_{Q^{*(i)}} w(Q^{*(i)}) \tag{20}$$

After several rounds of informal testing, we chose to focus on pose inference by particle filtering due to significantly better accuracy in estimation. We attribute this preference to the relative ease of tuning parameters of the particle filter predictive density in comparison to MCMC proposal covariance Σ_q. The remainder of the discussion in this article will assume pose estimation by particle filtering, although MCMC pose estimation can be performed instead without loss of generality.

6. Implementation

Our implementation for AxScEs perception of scenes and goal-directed manipulation is discussed in this section. This implementation follows the architecture outlined in Figure 5. The core of this implementation is the particle filter object pose estimation that follows a module flow of *prediction*, *diffusion*, *measurement* and *resampling*. The distribution over object poses is represented as a mixture model of particles. The *measurement* module additionally performs comparisons of relative likelihood across estimations from other scenes. We store object geometries in a database, which are expressed with respect to the parent object frame once retrieved. We further assume that invalid samples, where the center of mass for a child object is outside the support surface of its parent, is disregarded.

6.1. System architecture

As shown in Figure 5 the *measurement* module gets the observation from the robot and hypothesized particles generated from the rendering engine. Robot observations are in the form of depth images from a Microsoft Kinect depth camera mounted on the head of a Willow Garage PR2 robot. The likelihood of a particle is calculated by comparing the depth images of the observation and a graphical rendering of the axiomatic state hypothesized by a particle. The comparison function is a sliding window sum of squared distance (SSD) on two images. The z-buffer of an OpenGL-based graphics rendering engine is used to generate depth images from axiomatic states. We assume a known intrinsic calibration and extrinsic pose for the Kinect camera and that all the object geometries are known and stored in a geometry database.

The principal output of the *measurement* module is the posterior distribution representing the distribution of belief for the current state of the world. If the particles converge within a threshold, the *planner* takes the maximum likely state estimate and computes a plan of action for the robot to execute. In parallel, the *resampling* module takes in the posterior distribution and performs important sampling over their states to give the new distribution of particles to the *prediction* module. Based on the robot action decided by the *planner* the *predict* module updates the state of the particles. The *diffusion* module adds noise randomly to this distribution of particles and *measurement* is performed again with a new observation from the robot. The *diffusion* module also updates the rendering engine with a new set of axiomatic states to generate particles.

A STRIPS-based system (Fikes and Nilsson, 1972) was used for sequential planning in our manipulation system. With the goal and the current state of the world, the *planner* would compute a sequence of actions towards the goal and outputs the next immediate action to the robot. Actions from the planner will be pick-and-place actions for a specific object in the scene. Given this object's pose and geometry, from the *geometry database*, PR2 tabletop manipulation (Ciocarlie et al., 2014) is used to execute these manipulation actions.

6.2. Parallelized likelihood evaluation

As described in Section 2.2, the complexity of scene graph enumeration quickly grows beyond computational tractability. However, in practice, we can address this inference computationally viable through parallelization and constraining the space of physically viable scene estimates (Desingh et al., 2016). Described below is one method for parallel sample generation and likelihood evaluation through leveraging hardware graphics rasterizers in modern GPUs. This parallelization provides performance beyond what is offered through general-purpose computing on graphics processing units.

We consider parallel rendering based on OpenGL (Shreiner and Group, 2009) to simulate depth cameras and generate scene estimation particles rapidly. This renderer sets up the rendering pipeline using camera extrinsic and intrinsic parameters, object geometries and estimated object transformations. During each particle filter iteration, the OpenGL renderer renders all particles in parallel onto a single render buffer, which is then passed to CUDA kernels for computing the objective metrics of particles.

A particle is a scene consisting of objects with the same geometries but of different transformations. Each particle is specified by a draw call for its object geometries and transformations. Then, draw calls for rendering each particle are separated by viewport specifications `glViewport()` which set the positions and sizes of sub-images for particle hypotheses in the output render buffer. With credit to internal GPU work scheduling, all particles are rendered in parallel and viewport specification does not reduce the parallelism of issued draw calls.

The transformations in OpenGL draw calls include model matrices, the view matrix, and the projection matrix. Model matrices specify the transformations of the objects without changing the geometry of the objects. Model matrices are constantly updated per iteration according to the

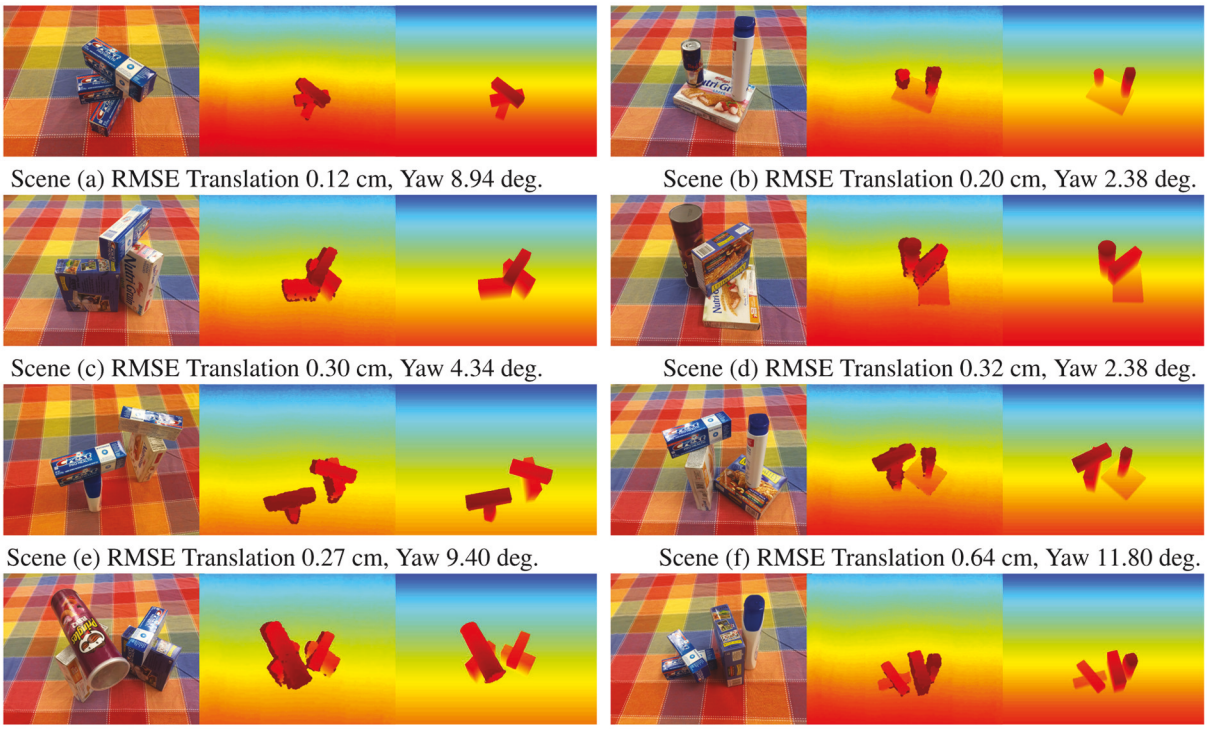

Fig. 6. Pose estimation results for known axiom sets in scenes containing three to four objects. Each subfigure shows the RGB (left) and depth (middle) from a RGBD camera mounted on the head of the PR2 and estimated scene graph as a depth image (right), as well as the RMSE for translation and rotation error.

changing estimates in particles. The view matrix is the extrinsic transformation of the camera within the world coordinate and can be derived using the position of the camera and the direction of the camera. The projection matrix can be derived using the camera intrinsic parameters, including the focal length f_x,f_y, the principal point c_x, c_y, and the OpenGL clipping near and far distances, z_n, z_f. We derive the following matrix P for perspective projection

$$P = \begin{pmatrix} \frac{2f_x}{W} & 0 & 1-\frac{2c_x}{W} & 0 \\ 0 & \frac{2f_y}{H} & 1-\frac{2c_y}{H} & 0 \\ 0 & 0 & -\frac{z_f+z_n}{z_f-z_n} & -\frac{2z_f z_n}{z_f-z_n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad (21)$$

where W and H are the image width and height.

Similar to Choi and Christensen (2013), we attach the output render buffer to a frame buffer object (FBO) for efficient off screen rendering. However, we use render buffer objects (RBOs) instead of textures because multisampling features in textures are not useful for our purposes and only add overhead. We also attach a depth render buffer to the FBO which is required for depth enabled rendering. In contrast to previous research where only color information is used, we are interested in the depth information. However, RBOs in depth format are not supported by CUDA and cannot be accessed from CUDA kernels via the OpenGL interoperation interfaces. We propose an efficient multi-step process for this by modifying the OpenGL fragment shader to compute the depth values and output as float point color values in a color formatted `GL_R32F` RBO, which can be accessed from CUDA. Note that the depth rendering process described here is similar to a part of the known deferred shading pipeline where the depth is saved in an intermediate result called geometric buffers (Saito and Takahashi, 1990).

Fragment shaders have access to a built-in variable gl_FragCoord $=(x, y, z, 1/w)$ in which w is the extra dimension of the clip-space homogeneous coordinate of the fragment. Using the perspective projection matrix P, a point $[X, Y, Z, 1]^{\mathrm{T}}$ in camera coordinate will be projected to clip-space coordinate $[x, y, z, w]^{\mathrm{T}}$ in which $w = -Z$, and Z is the distance from the point to the X-Y plane of the camera coordinate. $w = -Z$ has a negative sign because $[x, y, z]^{\mathrm{T}}$ is converted from the right-handed camera coordinate to the left-handed normalized device coordinate. The depth in the camera coordinate is then represented by $Z = w$. Thus the depth values can be computed in the fragment shader with `color = 1/gl_FragCoord.w;`. By leveraging the fragment shader that is already part of the existing rendering pipeline, this approach obtains depth values in one pass and eliminates the overhead of extra copying from a depth RBO to a color RBO.

The color RBO containing depth values is then passed to CUDA kernels through memory mapping with no data

transfer and minimal overhead. The CUDA kernels compute the squared error objective for each pixel, and rearrange the memory layout to compute the sums of errors for each particle. The sums are then normalized and used as weights in particle filter resampling. Actually, the square error objectives can also be computed in the fragment shader prior to CUDA kernels, which provides some flexibility, although it should not have a big difference on the overall performance. The process of computing scores for 625 particles takes 0.027 seconds.

To maximize the performance of OpenGL rendering, we adopt several scene rendering best practices (Tavenrath and Kubisch, 2013). We use `glVertexAttribDivisor()` to specify vertex attributes format, making it only require a single model matrix for all vertices of an object. We also use the OpenGL extension `ARB_multidraw_indirect` which allows drawing of multiple objects in a scene with a single draw call provided with parameters of multiple draw commands. With this extension, more objects in a scene would no longer require more draw calls, and all object geometries in all scenes/particles and their draw commands can be constructed and uploaded to GPU as static data during initialization. During particle filter iterations, only the model matrices will need to be updated, and the draw calls with fixed parameters reissued.

To validate the correctness of the depth value obtained by the OpenGL renderer, we also implemented a separate renderer based on the Nvidia OptiX (Parker et al., 2010) ray tracing engine. We use the OptiX Prime API to implement the renderer which solely executes on GPU compute nodes without the help of hardware rasterizers. The OptiX renderer submits parallel rendering queries which contain the scene geometries, transformations, and ray specifications corresponding to each pixel. Experiments with a toy scene of three cubes on a table show less than 10^{-5} (meter) average error in the results between OpenGL and OptiX renderers, which can be mostly attributed to floating number error. However, the performance of the OptiX renderer is much worse than the OpenGL renderer. To render 1000 images of 512 $\times$ 424 resolution, the OptiX renderer took 0.124 seconds, while for 1024 images of the same resolution, the OpenGL renderer took less than 0.005 seconds. This is because the OpenGL renderer takes advantage of the power of hardware rasterizers, while the OptiX renderer is limited to per pixel computation on GPU compute processors.

7. Results

In this section, we examine our *AxScEs* estimators, the AxPF and the AxMC, with respect to 20 test scenes of interacting objects from a depth camera. These objects are common to households and vary in dimensions and geometries, as shown in Figure 1(b). We first report the results of particle filter inference on object poses, which serves as the foundation for the inference methods over scene graphs by both AxPF and AxMC . Results are then presented for exhaustive search by the AxPF over scene graph which yields estimates with high accuracy in small collections of scenes. AxMC results are then presented that demonstrate tractable inference with less accuracy. All the experiments are tested on a Linux PC with Intel Core i7, 32 GB memory and a NVidia GeForce GTX Titan X Graphic Card with CUDA 7.5.

Next, we conduct three sets of experiments to demonstrate our goal-directed manipulation system with AxMC axiomatic scene estimation. In our baseline manipulation experiment, we evaluate the manipulation system in a scenario of three blocks stacked and rotated (Figure 8). We then consider a more complex scenario of three stacked blocks along with a basket in the scene (Figure 9). At last, to test the limit of AxScEs , we conduct an experiment with an eight object scene as shown in Figure 1. The PR2 robot was successful in achieving the goal scene: The *nature_valley* and *nutrigrain* boxes cleared to a side of the table and place all other objects into the basket.

7.1. Object pose estimation

In order to validate the accuracy of particle filter pose estimation, we first started by evaluating our GPU-optimized likelihood function for estimating object poses given known scene graphs. For each scene, 40 estimation trials were performed with 400 particle filter iterations with 1250 particles. As the render buffer size supported by the graphics card is 163484 $\times$ 16384 and the size of the depth image is 640 $\times$ 480, so the maximum number of images the graphics card can render at a time is 16384 / 640 = 625. Thus, our choice is of 1250 particles as two times 625.

The root mean square error (RMSE) on both translation (x and y) and rotation (yaw) are computed and are denoted in each scene in Figure 6 and Figure 7. The translation error remains very low for each scene (under 1 cm) but the rotation error seems a bit high. The large angular error is primarily due to the less accurate estimation of occluded supporting objects, and not due to the accounting of object symmetry. Supporting objects, higher in the scene graph, are occluded by the top objects and, thus, have fewer pixels in the observation depth image. Further, the standing objects also have fewer observed pixels, due to taking observations directly from the robots first person viewpoint, which leads to a larger angular error. Regardless, these errors are within our observed estimate of tolerable error for grasping with the PR2. The time taken for each particle filter iteration is 0.022 s and varies with different rendering objects. The total computation time for each scene is around 9.08 s.

This experiment demonstrates our particle filter can estimate the object poses with high accuracy and can serve as a likelihood function for scene graph estimation methods.

Fig. 7. Pose estimation results for known axiom sets in scenes containing five to seven objects. Each subfigure, shows the RGB (left) and depth (middle) from a RGBD camera mounted on the head of the PR2 and estimated scene graph as a depth image (right), as well as the RMSE for translation and rotation error.

Table 1. Metrics calculated for AxPF scene estimation. The first two columns are the mean and variance of the tree edit distance which is the minimum number of node operations to transform one tree to the other. From the 3rd column to the 5th column, the accuracy, precision and recall of the leaf node classification are reported. The last two columns are the pose error of translation (x and y) and *yaw* a correctly classified leaf node. N is the number of objects in each scene.

Scene	N	Tree edit distance		Leaf node classification			RMS pose error	
		Mean	Var	Accuracy	Precision	Recall	Translation	Yaw
Scene (a)	3	0.00	0.00	1.00	1.00	1.00	0.47	0.99
Scene (b)	3	0.00	0.00	1.00	1.00	1.00	0.38	6.47
Scene (c)	3	0.80	1.07	0.73	0.80	0.80	3.04	11.73
Scene (d)	3	0.00	0.00	1.00	1.00	1.00	0.21	0.90
Scene (e)	4	0.00	0.00	1.00	1.00	1.00	0.23	1.69
Scene (f)	4	1.40	2.71	0.80	0.75	0.90	0.62	8.56
Scene (g)	4	1.10	1.10	0.72	0.72	0.72	0.82	0.73
Scene (h)	4	0.00	0.00	1.00	1.00	1.00	0.46	8.84

7.2. Metrics for AxScEs

For evaluating the results of our *AxScEs* methods, we used metrics related to scene graph structure, tree edit distance (Zhang and Shasha, 1989), and leaf node classification, as the correct identification of currently manipulatable objects. The tree edit distance is the minimum number of node operations to transform one scene tree to the other. This distance uses three edit operations: Replace a node, insert a node, and delete a node. The tree edit distance is used to compute the distance between an estimated scene graph tree and ground truth scene graph tree, where, smaller values mean two trees are closer to each other.

In a cluttered scene, the directly manipulable objects provide support for no other objects are immediately available to be picked or placed upon. These objects, asserted by the clear relation, are the leaves in a scene graph tree. We care more about these objects than the support objects higher in a scene graph from an estimation perspective because they are unoccluded. As leaf node objects are picked up and moved away, the scene will become less cluttered and the supported objects will become clearer in the eye of the robot. Towards properly estimating leaf node objects, we introduce leaf node classification which identifies whether a node in the estimation is a correct leaf node or not. We report the accuracy, precision, and recall for this manipulation-oriented classification, as well as their pose estimation accuracy.

7.3. Scene graph estimation

7.3.1. AxPF. For each of the 20 test scenes, we then ran the exhaustive search over scene graph (Section 5.1) with 625 particles for each scene. Due to the prohibitive computational complexity, the AxPF was not considered for scenes with more than four objects. From Table 1, mean and variance of the tree edit distance remain very low for all the scenes tested. The computation time of the exhaustive set of particle filters is relatively high. For scenes with three objects, the exhaustive particle filters averaged 110.80 s and for four object scenes, the time grew to 1318.14 s on average.

7.3.2. AxMC. We ran MCMC with 200 iterations. In each MCMC iteration, a particle filter estimates object poses with 625 particles over 400 iterations. We ran the experiment ten times and the results in Table 2 is averaged over these experiments. Based on these results, we interpreted the AxMC to perform well for scenes of up to six objects. The tree edit distance grows linearly with the number of objects in the scene. The average accuracy of leaf node classification is 0.78 which means on average only one leaf node object would be wrong for each scene as there are maximal four objects on the top. The RMS yaw error of the leaf nodes is relatively smaller than the errors from Section 7.1 which are computed over all the objects in the scene. This indicates that the robot can grasp the top objects more robustly. For scenes with greater numbers of objects, we found that at least one object was estimated correctly in each trial. This gives room for an active approach to perception and manipulation. From an AxScEs estimate, the leaf object with the highest likelihood can be grasped and moved to decrease ambiguity for another round of AxScEs estimation.

7.4. Manipulation results

In this set of manipulation experiments, AxMC estimation is evaluated within the goal-directed manipulation system described in the previous section. The AxMC will first estimate the scene and get the scene graph and the pose for each object after convergence. Given the pose, object geometry, and transformations, the system would reconstruct the scene graph in 3D points in camera view. The planner computes a sequence of actions towards the goal axiomatic state, and executes the first action in this plan. After performing this action, the robot re-estimates and re-plans for the resulting scene to take its next action. This process loop continues until the goal scene state is achieved.

7.4.1. Three stacked and rotated blocks. The first manipulation experiment is to rearrange three stacked and rotated blocks into a straight stack with reversed order. The observation depth image is shown in Figure 8(a). Figure 8(b) shows the estimation result in depth image and Figure 8(c) shows the reconstructed scene in point cloud. The STRIPS planner planned a sequence of actions towards the goal with the estimated scene graph and sent them to the robot. Then the robot executed them sequentially, as shown from Figure 8(d) to Figure 8(i).

7.4.2. Extraction of middle block. The second manipulation experiment is to extract the middle block from the three block sequence into a basket which shows our system can handle complex geometries. The remaining two blocks are rearranged into a straight stack aside from the basket. Figure 9(a) shows the perceived depth image and Figures 9(b) and (c) show the estimated blocks with the basket. From Figures 9(d) and (e), the robot picked up the top block and placed it on the table. Then the middle block was picked by the robot and placed it into a basket which is shown in Figures 9(f) and (g). Finally, the bottom block was picked and placed onto the top block as shown in Figures 9(h) and (i).

7.4.3. Manipulation in a cluttered environment. To test the limit of our approach, we conducted the manipulation experiment in a much cluttered environment with eight objects in it. The goal of the task is to place nature_valley and nutrigrain boxes to a side of the table and to put all other objects into the basket. Note that as the robot gripper is not wide enough to pick up the large boxes lying on the table, nature_valley and nutrigrain are standing vertically on the table. Figures 1(e) and 1(f) show the estimation results and

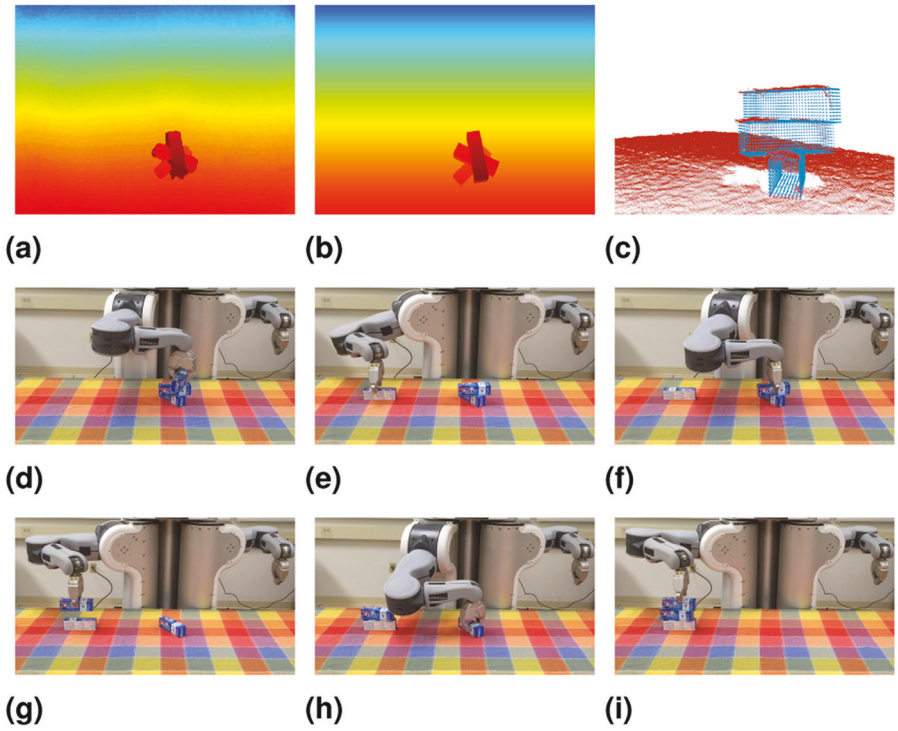

Fig. 8. Manipulation experiment of rearrangement of three rotated and stacked blocks. Observation and estimated depth image along with reconstructed point cloud (top row). Frames of robot performing stacking actions to rearrange toothpaste boxes into a straight stack (bottom rows). (a) Observation. (b) Estimated result. (c) Reconstructed scene. (d) Pick up block1. (e) Place block1. (f) Pick up block2. (g) Place block2. (h) Pick up block3. (i) Place block3.

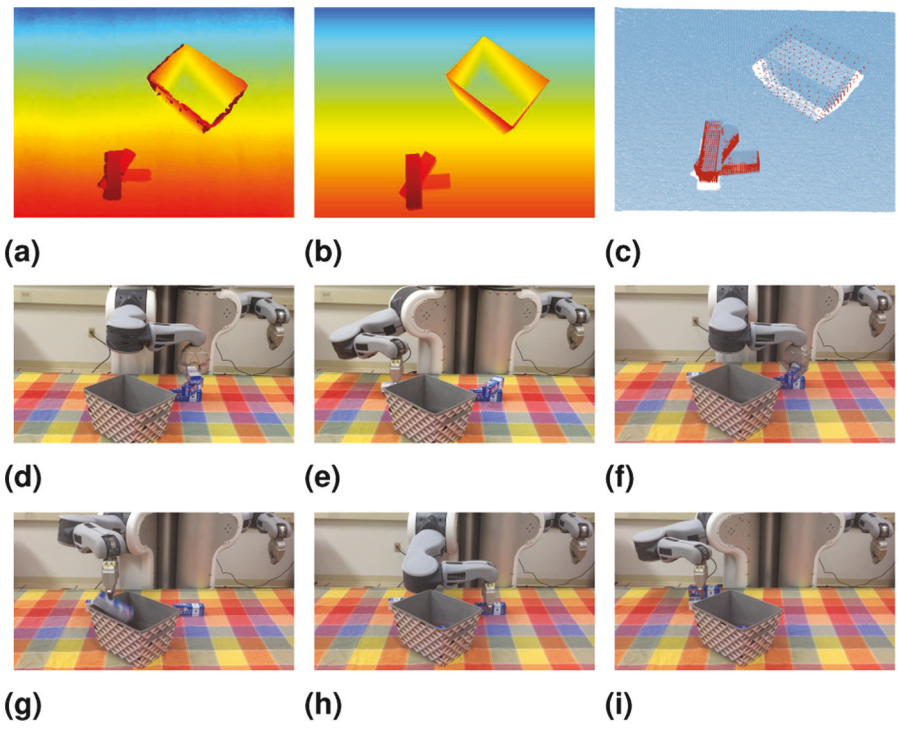

Fig. 9. Manipulation experiment of extraction of the middle block into the basket. Observation and estimated depth image along with reconstructed point cloud (top row). Frames of robot performing stacking actions to extract the middle block into the basket (bottom rows). (a) Observation. (b) Estimated result. (c) Reconstructed scene. (d) Pick up block1. (e) Place block1. (f) Pick up block2. (g) Place block2. (h) Pick up block3. (i) Place block3.

Table 2. Metrics calculated for AxMC scene estimation. The first two columns are the mean and variance of the tree edit distance which is the minimum number of node operations to transform one tree to the other. From the 3rd column to the 5th column, the accuracy, precision and recall of the leaf node classification are reported. The last two columns are the pose error of translation (x and y) and *yaw* a correctly classified leaf node. N is the number of objects in each scene.

Scene	N	Tree edit distance		Leaf node classification			RMS pose error	
		Mean	Var	Accuracy	Precision	Recall	Translation	Yaw (Degree)
Scene (a)	3	0.00	0.00	1.00	1.00	1.00	0.41	1.35
Scene (b)	3	0.30	0.90	1.00	1.00	1.00	0.38	5.23
Scene (c)	3	1.40	0.93	0.53	0.65	0.65	1.20	18.72
Scene (d)	3	1.00	1.11	1.00	1.00	1.00	0.70	0.55
Scene (e)	4	1.20	2.84	0.82	0.81	0.85	1.16	5.97
Scene (f)	4	2.20	2.18	0.72	0.68	0.85	0.81	7.33
Scene (g)	4	2.00	0.67	0.65	0.62	0.75	3.13	3.74
Scene (h)	4	1.40	1.82	0.97	0.96	1.00	0.91	7.34
Scene (i)	5	3.00	2.00	0.72	0.60	0.90	1.20	1.04
Scene (j)	5	3.30	1.34	0.66	0.71	0.73	1.72	7.60
Scene (k)	5	2.40	1.60	0.82	0.70	0.95	0.56	1.97
Scene (l)	5	2.40	1.38	0.82	0.69	1.00	2.02	0.51
Scene (m)	6	3.90	0.77	0.73	0.68	0.87	2.31	6.77
Scene (n)	6	4.30	4.23	0.59	0.58	0.70	5.88	5.92
Scene (o)	6	3.20	2.84	0.76	0.71	0.89	3.18	13.31
Scene (p)	6	3.20	1.29	0.78	0.78	0.95	3.09	7.31
Scene (q)	7	4.30	2.01	0.60	0.62	0.75	7.81	8.80
Scene (r)	7	4.10	5.88	0.81	0.79	0.93	6.30	32.54
Scene (s)	7	5.80	1.51	0.63	0.55	0.70	5.58	25.73
Scene (t)	7	5.60	2.49	0.66	0.57	0.80	4.62	15.78
Scene	8	5.80	2.34	0.50	0.50	1.00	6.94	7.85

scene in point cloud view. The actions performed by the robot is shown from Figures 1(g) to 1(l).

8. Conclusion

In this article, we proposed generative approaches to address the problem of *axiomatic scene estimation* (*AxScEs*) as the estimation of scenes for goal-directed robot manipulation. In AxScEs estimation, a generative model maintains a distribution across plausible scene graph hypotheses supported by the robot's point cloud observations. These generated hypotheses form an approximate probability distribution (or belief) over possible states of the scene. We cast the problem of AxScEs as factors for estimating a scene graph as a tree and poses. Our AxPF method performs inference in this model as a brute force exhaustive search over combinations of scenes. We additionally proposed the MCMC-base AxMC method to avoid exploration over all possible scenes by random walk sampling. A parallelized GPU-optimized version of these inference methods was described and implemented. Our results indicate that AxScEs estimators are effective for manipulation-quality perception based on edit distance on scene graphs, estimation of currently manipulatable (clear) objects and object pose estimation.

In addressing problems of AxScEs, one of our primary aims is to enable axiomatic perception that will enable a greater convergence of symbolic inference for task planning and collision-free motion planning and execution. Overcoming the divides between perception, planning and action is a critical challenge for realizing the next generation of task-oriented mobile manipulators. In this regard, our AxScEs estimators are only a step towards this goal. There are still many issues to address given the computational and spatial complexity that limit our current AxPF and AxMC methods. Our methods focus on the space of potential scenes. We have yet to exploit the space of plausible scenes, where the constraints of physics and space could bring scene inference into tractability. Ideally, such scene inference could occur in real-time, similar to localization for modern autonomous navigation. While our models incorporate notions of dynamics for tracking, we have left exploration of this issue as future work.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by Office of Naval Research (grant number N00014-08-1-0910) and NASA (grant number NNX13AN07A).

References

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: *Proceedings of the twenty-first international conference on machine learning*, ICML '04, New York, USA, p.1. ACM.

Akgun B, Cakmak M, Jiang K, et al. (2012) Keyframe-based learning from demonstration. *International Journal of Social Robotics* 4(4): 343–355.

Atkeson CG and Schaal S (1997) Robot learning from demonstration. In: *Proceedings of the fourteenth international conference on machine learning*, ICML '97, San Francisco, CA, pp.12–20. Morgan Kaufmann Publishers Inc.

Biswas J and Veloso MM (2013) Localization and navigation of the cobots over long-term deployments. *International Journal of Robotics Research* 32(14): 1679–1694.

Calinon S and Billard A (2007) Incremental learning of gestures by imitation in a humanoid robot. In: *Second conference on human-robot interaction (HRI)*, Arlington, VA, pp. 255–262. IEEE.

Chao C, Cakmak M and Thomaz AL (2011) Towards grounding concepts for transfer in goal learning from demonstration. In: *Proceedings of the joint IEEE international conference on development and learning and on epigenetic robotics (ICDL-EpiRob)*. Frankfurt, Germany, pp. 1–6. IEEE.

Chernova S and Veloso M (2009) Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research* 34(1): 1–25.

Choi C and Christensen HI (2013) RGB-d object tracking: A particle filter approach on GPU. In: *Intelligent robots and systems (IROS), 2013 IEEE/RSJ international conference on*, Tokyo, Japan, pp.1084–1091. IEEE.

Ciocarlie M, Hsiao K, Jones EG, et al. (2014) Towards reliable grasping and manipulation in household environments. In: *Experimental Robotics*. Heidelberg: Springer Berlin, pp.241–252.

Collet A, Berenson D, Srinivasa SS, et al. (2009) Object recognition and full pose registration from a single image for robotic manipulation. In: *IEEE international conference on robotics and automation (ICRA)*, Kobe, Japan. pp. 48–55. IEEE.

Collet A, Martinez M and Srinivasa SS (2011) The moped framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research* 30(10): 1284–1306.

Coradeschi S and Saffiotti A (2003) An introduction to the anchoring problem. *Robotics and Autonomous Systems* 43(2): 85–96.

Cosgun A, Hermans T, Emeli V, et al. (2011) Push planning for object placement on cluttered table surfaces. In: *2011 IEEE/RSJ international conference on intelligent robots and systems*, San Francisco, CA, pp.4627–4632. IEEE.

Crick C, Osentoski S, Jay G, et al. (2011) Human and robot perception in large-scale learning from demonstration. In: *Proceedings of the 6th ACM/IEEE international conference on human-robot interaction (HRI)*. New York, NY, pp. 339–346. ACM.

Dellaert F, Fox D, Burgard W and Thrun S (1999) Monte carlo localization for mobile robots. In: *IEEE international conference on robotics and automation*. Detroit, MI, pp. 1322–1328. IEEE.

Desingh K, Jenkins OC, Reveret L, et al. (2016) Physically plausible scene estimation for manipulation in clutter. In: *IEEE-RAS international conference on humanoid robots (Humanoids 2016)*. 15–17 November, Cacun, Mexico.

Dogar M and Srinivasa S (2011) A framework for push-grasping in clutter. In: *Proceedings of robotics: Science and systems*, Los Angeles, CA, pp. 65–72.

Dogar MR, Hsiao K, Ciocarlie MT, et al.(2012) Physics-based grasp planning through clutter. In: *Robotics: Science and systems*. p. 504. MIT Press.

Fallon M, Johannsson H and Leonard J (2012) Efficient scene simulation for robust monte carlo localization using an rgb-d camera. In: *Robotics and automation (ICRA), 2012 IEEE international conference on*, St. Paul, MN, pp.1663–1670. IEEE.

Felzenszwalb PF, Girshick RB, McAllester D, et al. (2010) Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(9): 1627–1645.

Fikes RE and Nilsson NJ (1972) Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3): 189–208.

Goodrich MA, Jr DRO, Crandall JW, et al. (2001) Experiments in adjustable autonomy. In: *Proceedings of the IJCAI workshop on autonomy, delegation and control: Interacting with intelligent agents*, Seattle, WA, pp.1624–1629. IJCAI.

Grollman DH and Jenkins OC (2008) Sparse incremental learning for interactive robot control policy estimation. In: *International conference on robotics and automation (ICRA 2008)*, 19–23 May, Pasadena, CA, pp.3315–3320. IEEE.

Harnad S (1990) The symbol grounding problem. *Physica D: Nonlinear Phenomena* 42(1): 335–346.

Hart S, Dinh P and Hambuchen K (2015) The affordance template ros package for robot task programming. In: *Robotics and automation (ICRA), 2015 IEEE International conference on*, Seattle, WA pp.6227–6234. IEEE.

Hastings WK (1970) Monte carlo sampling methods using markov chains and their applications. *Biometrika* 57(1): 97–109.

Herbst E, Ren X and Fox D (2011) Rgb-d object discovery via multi-scene analysis. In: *Intelligent robots and systems (IROS), 2011 IEEE/RSJ international conference on*, San Francisco, CA, pp.4850–4856. IEEE.

Jenkins OC and Matarić MJ (2004) Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion. *International Journal of Humanoid Robotics* 1(2): 237–288.

Joho D, Tipaldi GD, Engelhard N, et al. (2012) Nonparametric bayesian models for unsupervised scene analysis and reconstruction. In: *Proceedings of robotics: Science and systems*, Sydney, Australia. pp. 161–168.

Kaelbling LP, Littman ML and Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2): 99–134.

Kaelbling LP and Lozano-Pérez T (2013) Integrated task and motion planning in belief space. *The International*

Journal of Robotics Research 32(9–10): 1194–1227. Doi: 10.1177/0278364913484072.

Kemp CC, Anderson CD, Nguyen H, et al. (2008) A point-and-click interface for the real world: Laser designation of objects for mobile manipulation. In: *Human-robot interaction (HRI), 2008 3rd ACM/IEEE international conference on*, 12–15 March, Amsterdam, The Netherlands, pp.241–248. IEEE.

Kirk JR and Laird JE (2013) Learning task formulations through situated interactive instruction. In: *Proceedings of the second annual conference on advances in cognitive systems*, 12–14 December, Baltimore, Maryland, pp.219–236.

Kober J and Peters J (2011) Policy search for motor primitives in robotics. *Machine Learning* 84(1-2): 171–203.

Kuipers B (2000) The spatial semantic hierarchy. *Artificial Intelligence* 119(1): 191–233.

Laird JE, Newell A and Rosenbloom PS (1987) Soar: An architecture for general intelligence. *Artificial Intelligence* 33: 1–64.

Lang T, Toussaint M and Kersting K (2012) Exploration in relational domains for model-based reinforcement learning. *Journal of Machine Learning Research* 13(1): 3725–3768.

Liu Z, Chen D, Wurm KM, et al. (2015) Table-top scene analysis using knowledge-supervised mcmc. *Robotics and Computer-Integrated Manufacturing* 33: 110–123.

Manfredotti CE, Fleet DJ, SZ, et al. (2010) Relational particle filtering. In: *Monte Carlo methods for modern applications, 2010 NIPS workshop*, Whistler, Canada.

McDermott D, Ghallab M, Howe A, et al. (1998) *Pddl - the planning domain definition language*. Technical Report TR-98-003, Yale Center for Computational Vision and Control, USA.

Miller FP, Vandome AF and McBrewster J (2009) *Lua (Programming Language)*. London, UK. Alpha Press.

Mohan S, Mininger AH, Kirk JR, et al. (2012) Acquiring grounded representations of words with situated interactive instruction. In: *Advances in Cognitive Systems* pp. 113–130. Citeseer.

Narayanaswamy S, Barbu A and Siskind JM (2011) A visual language model for estimating object pose and structure in a generative visual domain. In: *Robotics and automation (ICRA), 2011 IEEE international conference on*, Shanghai, China, pp.4854–4860. IEEE.

Nicolescu M and Matarić MJ (2003) Natural methods for robot task learning: Instructive demonstration, generalization and practice. In: *2nd international joint conference on autonomous agents and multi-agent systems (AAMAS)*, 14–18 July, Melbourne, Australia, pp.241–248.

Papazov C, Haddadin S, Parusel S, et al. (2012) Rigid 3d geometry matching for grasping of known objects in cluttered scenes. *The International Journal of Robotics Research* 31(4): 538–553. Doi: 10.1177/0278364911436019.

Parker SG, Bigler J, Dietrich A, et al. (2010) Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics* 29(4): 66:1–66:13. Doi: 10.1145/1778765.1778803.

Pas AT and Platt R (2014) Localizing handle-like grasp affordances in 3d point clouds. In: *International symposium on experimental robotics (ISER)*, pp. 623–638. Springer International Publishing.

Pastor P, Kalakrishnan M, Chitta S, et al. (2011) Skill learning and task outcome prediction for manipulation. In: *Proceedings of the 2011 IEEE international conference on robotics & automation (ICML 2011)*, Shanghai, China, pp. 3828–3834. IEEE.

Platt RJ, Fagg AH and Grupen RA (2010) Null-space grasp control: Theory and experiments. *IEEE Transactions on Robotics* 26(2): 282–295.

Quigley M, Conley K, Gerkey BP, et al. (2009) Ros: an open-source robot operating system. In: *ICRA workshop on open source software*, Kobe, Japan, volume 3, p. 5. ICRA.

Rosman B and Ramamoorthy S (2011) Learning spatial relationships between objects. *International Journal on Robotics Research* 30(11): 1328–1342.

Rusu RB, Marton ZC, Blodow N, et al. (2008a) Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems* 56(11): 927–941.

Rusu RB, Marton ZC, Blodow N, et al. (2008b) Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems* 56(11): 927–941.

Niekum S GK, Osentoski S and Barto AG (2012) Learning and generalization of complex tasks from unstructured demonstrations. *Intelligent Robots and Systems*, Vilamoura, Portugal, pp. 5239–5246. IEEE.

Saito T and Takahashi T (1990) Comprehensible rendering of 3-d shapes. In: *Proceedings of the 17th annual conference on computer graphics and interactive techniques*, SIGGRAPH '90, New York, USA, pp.197–206. ACM.

Shreiner D, Sellers G, Kessenich J and Licea-Kane B (2013) OpenGL programming guide: The official guide to learning OpenGL, Version 4.3, 8th ed, Upper Saddle River, NJ. Addison-Wesley Professional.

Smart WD and Kaelbling LP (2002) Effective reinforcement learning for mobile robots. In: *Robotics and automation, 2002. Proceedings. ICRA '02. IEEE international conference on*, volume 4, Washington, DC, pp.3404–3410. IEEE.

Srivastava S, Riano L, Russell S, et al. (2013) Using classical planners for tasks with continuous operators in robotics. In: *Proceedings of the ICAPS workshop on planning and robotics (PlanRob)*, Beijing, China.

Sui Z, Jenkins OC and Desingh K (2015) Axiomatic particle filtering for goal-directed robotic manipulation. In: *International conference on intelligent robots and systems (IROS 2015)*, 28 September–2 October, Hamburg, Germany, pp.4429–4436.

Tavenrath M and Kubisch C (2013) Advanced scenegraph rendering pipeline. In: *GPU technology conference*, March, San Jose, CA.

Tenorth M and Beetz M (2013) Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *International Journal of Robotics Research* 32(5): 566–590.

Trafton G, Hiatt L, Harrison A, et al. (2013) Act-r/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction* 2(1): 30–55.

Vondrak M, Sigal L, Hodgins J, et al. (2012) Video-based 3D motion capture through biped control. *ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH)* New York, NY, pp. 27:1–27:12. ACM.

Winograd T (1972) Understanding natural language. *Cognitive psychology* 3(1): 1–191.

Wintermute S and Laird JE (2008) Bimodal spatial reasoning with continuous motion. In: *Proceedings of the 23rd national conference on artificial intelligence - Volume 3*, AAAI'08, pp.1331–1337. AAAI Press.

Yanco H, Norton A, Ober W, et al. (2015) Analysis of human-robot interaction at the darpa robotics challenge trials. *Journal of Field Robotics* 32(3): 420–444.

Zettlemoyer LS, Pasula HM and Kaelbling LP (2007) Logical particle filtering. In: *Dagstuhl seminar on probabilistic, logical and relational learning*, Dagstuhl, Germany. IBFI.

Zhang K and Shasha D (1989) Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing* 18(6): 1245–1262.

Zhang L and Trinkle J (2012) The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing. In: *Robotics and automation (ICRA), 2012 IEEE international conference on*, St. Paul, MN, pp.3805–3812. IEEE.