Scene Understanding and Distribution Modeling with Mixed-Integer Scene Parsing

Gregory Izatt, Russ Tedrake
[gizatt,russt]@csail.mit.edu
MIT CSAIL
Cambridge, MA, USA

ABSTRACT

We present a probabilistic procedural modeling strategy for capturing the distribution of scenes of varying numbers of rigid objects, motivated by the need for such models to enable verification of robot behavior in the complex open world. Our model takes the form of a parameterized scene grammar over spatial objects, which we show admits maximum a posterior scene parsing via mixed-integer convex optimization. This parsing strategy can be used to explain and annotate unlabeled scenes with the structures specified by the model, and can be used to detect and explain outliers at a part level. We demonstrate our model's expressiveness on a cluttered 3D kitchen scene, and explore how different grammar design choices can impact the performance of this technique.

1 INTRODUCTION

How does one describe the set of all kitchen sinks that a dishwashing robot will ever face? Having a model of that set of environments is critical to verifying that a robot will succeed at its duties when deployed into the open world, but actually capturing that distribution over diverse worlds is a difficult problem. Even if one assumes the world is composed of rigid objects with known shapes, different worlds will contain different *numbers and classes* of objects in different configurations, where the presence, types, and pose of those objects may covary.

In this work, we present one approach for capturing this discrete and continuous variability: we describe the generation of these environments with a *scene grammar*. A scene grammar can represent the ideas of breaking a room down into parts (like a formal constituency grammar), or that objects might be placed relative to some "parent" object (like a formal dependency grammar). For any given class of scenes, one can try to describe its structure with a grammar by defining the scene's object types and their relationships in the form of grammar rules. Random draws from a grammar are scene trees, which capture that hierarchical structure in the scene (see Figure 1).

Scene grammars have a history in computer vision and graphics, but vary significantly in design. In this work, we put forward a particular flavor of scene grammar which focuses on capturing *spatial* structure of *objects* in the scene; we present a grammar structure that offers significant expressiveness while maintaining a simple and explicit functional form. We focus in particular on the problem of "parsing" an unlabeled scene with the grammar – that is, recovering reasonable scene trees that explain observed scenes while following the grammar rules. We design our grammar such that the maximum a-posteriori (MAP) parsing problem can be framed as a mixed-integer convex optimization, and show that parsing can be used as a powerful scene understanding tool that infers scene structure, detects out-of-distribution environments, and enables model-parameter estimation.

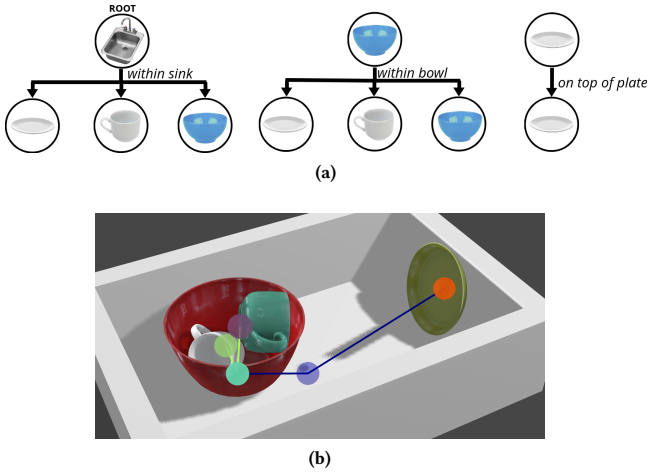

Figure 1: (a) A sketch of a scene grammar that describes the occurrence and placement of dishware in a cluttered sink. Any random draw a scene grammar will activate the available rules stochastically, generating a scene with a random number of objects in random poses. (b) An example sink annotated with a *scene tree* representing the hierarchy of relationships between objects corresponding to rules in the grammar.

In a set of experiments, we probe the limits of these tools. We observe that small changes in grammar design can have significant impacts on the tractability of parsing. Our parsing procedure includes algorithms for ameliorating those barriers to scaling in many cases. Leveraging those algorithms and design lessons, we show that we can apply these tools to model a realistically complex 3D kitchen sink scene.

2 RELATED WORK

Having a rich open world model that can simultaneously capture discrete *and* continuous variation in the world seems essential to making systems and robots robust in the diverse real world. In the case that only continuous parameters (e.g. friction parameters) of the world are unknown, this system identification problem is simpler, though still difficult: [Mehta et al. 2020] surveys one school of recent approaches that perform posterior Bayesian optimization of continuous dynamics parameters from observed rollouts. However,

in the open-world modeling case, it's not even clear how to parameterize this distribution over worlds, and what kinds of relationships in the world the model should strive to capture.

Fortunately, there is a large body of work on probabilistic modeling of scenes from which we can draw inspiration. L-systems [Prusinkiewicz et al. 1996] and shape grammars [Stiny 1980] literally describe the shape of objects by using rules to describe how to "grow" the object out of primitive geometry. Scene grammars instead describe the scene at the level of objects, though the amount of detail they aim to capture can vary: [Feng Han and Song-Chun Zhu 2009], [Zhao and Zhu 2011], and [Chua 2018] all describe scenes from objects all the way down to the polygon, line segment, or pixel level, while [Liu et al. 2014], [Kar et al. 2019], [Zinberg et al. 2019], and [Devaranjan et al. 2020] use objects as their terminal level of detail. One step above a scene grammar would be to instead describe a distribution over scene *graphs*, in which an object can have relationships to more than one other object. These models are more complex to do sample from and do inference over, although it's shown to be possible with MCMC methods [Yeh et al. 2012]). Scene graph models are well suited to producing variations on existing content [Fisher et al. 2012] and providing rich priors and structures for both object-level, pixel-level, and point cloud perception [Park et al. 2018] [Rosinol et al. 2021]. Even richer models – whether based on probabilistic programming [Ritchie 2016] [Kulkarni et al. 2015] [Mansinghka et al. 2013] or autoregressive neural models [Eslami et al. 2016] [Ritchie et al. 2019] – can in principle capture *any* distribution over *any* scenes, at the cost of significant model complexity, extremely difficult inference problems, and for deep neural models, data inefficiency.

In this work, we focus on using scene grammars as a modeling tool. Scene grammars have easily understandable and transparent structure and make strong independence assumptions, but as our results demonstrate, they can still describe complex robotics-relevant scenes. Scene grammars are easy to design (unlike many shape grammars) and sample from (unlike some scene graph models that only provide a joint density over scenes).

A significant amount of work has gone into exploring methods for *inverting* scene grammar models by way of "parsing." Unfortunately, common dynamic programming, chart-based, and traversal-based parsing techniques from formal grammars are hard to apply to scene grammars, as scene grammars typically have continuous attributes and produce unordered outputs. Heuristic parsing methods using a combination of top-down proposals and greedy bottom-up parse tree construction can get pretty far [Feng Han and Song-Chun Zhu 2009] [Izatt and Tedrake 2020], but can be inconsistent for grammars where intermediate structure is hard to guess. Reversible-jump MCMC provides an appealing framework with which to search the space of varying-size parse trees ([Talton et al. 2011], [Zhao and Zhu 2013], [Yeh et al. 2012], [Matheos et al. 2020]), but only works when one can engineer a good jump proposal. In this work, we observe that this parsing problem can be framed as a mixed-integer optimization as long as the scene grammar has a particular form.

Symbol	Meaning
o	An observed environment as a list of nodes, each of which has associated geometry.
T	A scene tree.
$\mathcal{N}$	Set of node types.
N	Single node type.
n	Single node instance; e.g. n^p, a parent node.
$\mathcal{R}$	A set of all rules.
R	A particular rule.
$n.x$	The pose x attribute belonging to node n.
$N.\mathcal{R}$	The set of rules that a node of type N can enact.

Table 1: Reference guide for some of the commonly-used notation used in this paper.

3 SYSTEM OVERVIEW

We present a stochastic grammar model that aims to capture the distribution of scenes of varying numbers and types of rigid objects. In this work, an observed scene o is a list of objects and their poses. We detail a probabilistic generative model $p_\theta(o, T)$ that supposes that one can generate new scenes o by drawing a scene tree T from a parameterized scene grammar; o is then extracted from T by collecting the parts of the tree that produce scene geometry, and throwing out the rest of the tree structure.

While drawing new worlds unconditionally from this generative model is easy, inverting the model – that is, explaining how the model would have generated a given scene – is hard. *Scene parsing* refers to this process of consuming an observed scene (as a list of objects with poses) o and producing a parse tree annotation T – which requires optimizing over both the discrete structure (edges) and continuous poses (nodes) of T. In Section 3.1, we present a scene grammar formulation tailored to make this inverse problem tractable while still maintaining enough flexibility to describe diverse scenes. In Section 4, we show how and when we can frame that scene parsing problem for our grammar as a mixed-integer convex program. In Section 5, we investigate the effectiveness and runtime scaling behavior of this parsing technique on two example grammars.

3.1 Spatial Scene Grammar Definition

A spatial scene grammar is a stochastic, attributed, parameterized context-free grammar $\{\mathcal{N}, \mathcal{R}, P_\theta^{\mathcal{N}}, P_\theta^{\mathcal{R}}, N_{root}, x_{root}\}$. This grammar is defined over node types $\mathcal{N}$ (i.e. symbols) that represent spatial entities. Each node type $N \in \mathcal{N}$ has a continuous pose attribute $N.x \in SE(3)$[1], and a boolean flag $N.has\_geometry$ indicating whether that node type will produce geometry in the final scene. The set of production rules $\mathcal{R}$ represent how a node (i.e. an instantiated symbol) can produce a child node. The rules are organized by their parent node: each node type $N \in \mathcal{N}$ has an associated list of production rules $N.\mathcal{R} = [R_0, ..., R_M]$, each of which describes the possible production of a single additional child object.

In this definition, a node can simultaneously activate multiple rules to generate multiple children. Specifically, a parent node n^p

[1]Throughout this paper, we use notation $A.B$ to indicate entity B is a property of entity A.

Algorithm 1: Forward sampling a scene from a grammar.

Input : a root node *root*
Output: a sampled scene tree

```
/* Build scene-tree top-down, starting from the
   supplied root node.                              */
node_queue = [root]
tree = Tree([root])
while len(node_queue) ≥ 0 do
    parent = node_queue.pop()
    /* Sample what this parent is going to
       produce.                                     */
    rules = parent.sample_rules()
    for rule in rules do
        /* Sample the pose of each child being
           produced.                                */
        child = rule.sample_child()
        node_queue.push(child)
        tree.add_node(child)
        tree.add_edge(parent, child)
/* Return only those nodes that produce scene
   geometry.                                        */
return [node for node in tree if node.has_geometry]
```

of type N activates a subset of its rules $\mathcal{R}_{active} \subset N.\mathcal{R}$ following a (possibly parameterized) discrete distribution over n_p's rules:

$$P_\theta^N(\mathcal{R}_{active}).$$

When active, a given rule R produces a child node n^c of fixed type $R.N$ with its pose drawn from a parameterized continuous distribution that is conditioned on the parent's pose

$$P_\theta^R(n^c.x|n^p.x).$$

This is expressive enough to describe, for example, a child whose pose is a distribution in the parent's frame of reference.

A draw from the grammar is a tree T of nodes, each having a concrete spatial pose. The tree is stored as a set of parent-ruleset pairs $T = \{n^p, \mathcal{R}_{active}\}$; for a given node n^p, $\mathcal{R}_{active}$ is n^p's set of active rules, which serves as the list of outgoing edges to child nodes $R.n^c$ for each rule $R \in \mathcal{R}_{active}$. A tree is drawn from the grammar by initializing a tree with a root node of type N_{root} at predetermined pose x_{root} and recursively sampling children from unexpanded nodes until no unexpanded nodes remain (see Algorithm 1). An observed scene o is constructed by collecting the geometry-producing nodes from T. The observed scene o does not always reveal the full structure of the tree, because only those nodes that produce geometry are observable, and there may be multiple trees that produce the same observations. (For example, the centers of clusters of objects may be represented with nodes, but only the actual objects in the cluster show up in the observed scene.) A node whose type produces geometry always produces the same fixed geometry – if a node is in the tree, then its associated geometry will appear in the scene at the pose determined by the node's random continuous variables.

The joint probability of a sampled tree T is then

$$P(T) = \prod_{n^p, \mathcal{R}_{active} \in T} P_\theta^{N^p}(\mathcal{R}_{active}|n^p) \prod_{R \in \mathcal{R}_{active}} P_\theta^R(R.n^c.x|n^p.x), \quad (1)$$

where the outer product iterates over parent nodes n^p and their actives rule sets $\mathcal{R}_{active}$ implied by their successors in the tree, where N^p indicates the node type of corresponding parent node n^p. The inner product iterates over the individual active rules R and their produced child nodes n^c associated with that parent.

The relationship between a parent node and its children is factored into the discrete choice (corresponding to P_θ^N) made by the parent to produce that child set, and the continuous choice (corresponding to P_θ^R) made to place each child at its pose relative to the parent. In the following sections, we enumerate additional structure and restrictions we impose on these distributions to keep our ultimate goal of posterior scene tree parsing tractable.

3.2 Node types to capture common discrete relationships

The kinds of children a parent creates are chosen by sampling a set of production rules from parent node. In general, this is a draw from a joint distribution over $M = |N.\mathcal{R}|$ binary variables representing the activation of each possible rule, requiring 2^M parameters to describe. In practice, we define a small set of parent node types that represent more structured patterns of rule activation that can be described with vastly fewer parameters:

(1) **AND** node type: The parent node provides a list of production rules, all of which are activated every time. This requires 0 parameters.
(2) **OR** node type: The parent node provides a list of production rules, one of which is chosen at random to be activated. This requires $M - 1$ parameters.
(3) **INDEPENDENT SET** node type: The parent node provides a list of production rules, each of which is activated randomly, independent of the other rules. This requires M parameters.
(4) **REPEATING SET** node type: The parent node provides a single production rule, which is activated $k \sim Categorical(1...M_{rep})$ times. This requires M_{rep} parameters.

This breakdown is strongly inspired by the AND-OR-SET node types popular in the scene grammar literature [Zhao and Zhu 2011][Park et al. 2018].

3.3 Rule types to capture common continuous relationships

Each production rule R under parent n^p that is activated produces a new child node n^c with random pose x drawn from a rule-specific conditional distribution $p_\theta^R(n^c.x|n^p.x)$. We separate this spatial relationship into conditional distributions over the translational and rotational components, which we constrain to be independent from one another. The separate translation and rotation parts of the pose are referred to as $n.t$ and $n.r$ respectively.

XYZ Rules

Name	Child translation expression	Log-density expression	Parameters
Fixed offset	$n^c.t = n^p.t + t_o$	0.	$t_o \in R^3$
World frame Normal	$n^c.t \sim Normal(\mu, \Sigma)$	$-0.5\delta^T\Sigma\delta - F(\Sigma)$ $\delta = n^c.t - \mu$	$\mu \in \mathbb{R}^3, \Sigma \in \mathbb{R}^3 > 0$
World frame Normal offset	$n^c.t \sim n^p.t + Normal(\mu, \Sigma)$	$-0.5\delta^T\Sigma\delta - F(\Sigma)$ $\delta = n^c.t - n^p.t - \mu$	$\mu \in \mathbb{R}^3, \Sigma \in \mathbb{R}^3 > 0$
Parent frame Normal offset	$n^c.t \sim n^p.t + n^p.r \times Normal(\mu, \Sigma)$	$-0.5\delta^T\Sigma\delta - F(\Sigma)$ $\delta = n^p.r^T(n^c.t - n^p.t - \mu)$	$\mu \in \mathbb{R}^3, \Sigma \in \mathbb{R}^3 > 0$

Rotation Rules

Name	Child rotation expression	Log-density expression	Parameters
Uniform rotation	$n^c.r \sim Uniform(SO(3))$	0.	None.
Fixed offset	$n^c.r = n^p.r \times R_o$	$-\log(\pi^2)$	$R_o \in SO(3)$
World frame Bingham distribution	$n^c.r \sim Bingham(M, Z)$	$ZM^T qq^T M - F(Z)$ $qq^t \leftarrow n^c.r$, see Section 4.3.4.	$M \in \mathbb{R}^{4\times4}, Z \in \mathbb{R}^4$
Parent frame Bingham distribution	$n^c.r \sim n^p.r \times Bingham(M, Z)$	$ZM^T qq^T M - F(Z)$ $qq^T \leftarrow (n^p.r)^T(n^c.r)$, see Section 4.3.4.	$M \in \mathbb{R}^{4\times4}, Z \in \mathbb{R}^4$

Table 2: The set of rule types supported by our grammar, along with their expression for forward sampling and log-density evaluation. Each rule type describes a parameterized, stochastic distribution on the translation and rotation of a child node n^c conditioned on the pose of its parent node n^p, in terms of each of node's translation t and rotation r components. Distinction is made between the different frames in which translational and rotational offsets are expressed, as offsets expressed in the parent's frame are bilinear in the participating pose variables and require special handling during scene parsing in Section 4. Terms $F(\Sigma)$ and $F(Z)$ represent normalizers that are a function of the indicated distribution parameters.

Translation distributions are captured using Normal distributions. Rotational distributions are captured using Bingham distributions, which are an analogue of Normal distributions over the unit sphere appropriate for describing distributions over unit quaternions [Bingham 1974]. Bingham distributions are parameterized by an orthogonal orientation matrix $M \in \mathbb{R}^{4\times4}$ and a concentration vector $Z \in \mathbb{R}^4$. A helpful overview of this distribution type is provided in [Glover and Kaelbling 2013].

The complete list of rule types we support during parsing is listed in Table 2. This particular set of rule types has arisen from efforts to apply this grammar formulation to a wide variety of scenes; empirically, each of these rule types is useful in different circumstances, and trades off expressiveness with complexity. In particular, expressing either translational or rotational pose distributions in the parent node frame (i.e. the "Parent frame" rules) introduces bilinear relationships between the pose variables, which must be handled as special cases in the MIP convex parsing procedure in Section 4.

3.4 Expressiveness

Despite these limitations on continuous distribution specification, this framework can be used to capture a wide variety of scenes. This grammar framework can be viewed as a generalized, multi-level Gaussian mixture model – that is, one where the number of modes and their hierarchy is itself stochastic – with special handling dedicated to describing distributions of 3D poses $\in SE(3)$. As a result, we inherit similar expressiveness: we can combine our primitive distributions together to build arbitrarily complex distributions over poses. We demonstrate its use for describing realistically cluttered arrangements of objects in household environments in Section 5.

4 PARSING

Given a grammar with parameters θ and an observed set of objects o, finding the maximum a posterior (MAP) tree $\arg\max_T p_\theta(T|o)$ is a general approach for "parsing" the scene – such a MAP tree would be the best possible explanation for the scene using this model. Being able to find this optimal scene tree would be broadly useful, but this posterior inference problem can be extremely hard. The best scene tree T for a given scene may be ambiguous; may require proposing new hidden structure; and generally requires searching over diverse paths for explaining the scene.

Because the set of observed objects is unordered and attributed, common approaches to parsing from formal grammars are not helpful. We instead approach this parsing problem by enumerating the space of all parse trees for our grammar with a set of binary and continuous variables, and writing a cost and constraint set to extract the MAP parse as a mixed-integer program (MIP).

The success of this optimization strongly depends on the nature of the grammar. As we will detail, certain continuous relationships between nodes lead to to nonconvex costs and constraints. In some cases, these nonconvexities can either be efficiency approximated or reparameterized away. In cases where the grammar contains significant continuous hidden structure (i.e. intermediate nodes with uncertain poses that produce no observable geometry), the nonconvexities cannot always be avoided, so we provide some studies of piecewise convex approximation strategies.

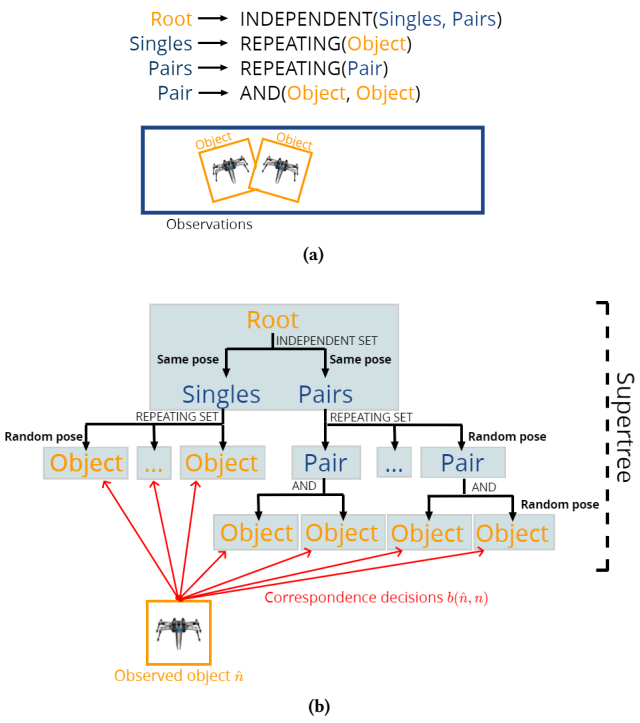

Figure 2: (a) The input to the parsing procedure is a grammar spec, along with a list of observed objects. Nodes shown in gold produce geometry (or have a-priori-fixed pose, in the case of the root). The complete grammar spec also includes additional information about how each node's pose is distributed relative to the parent, which is omitted for clarity. (b) For a given grammar, we generate a "supertree", for which any scene tree sampled from the grammar is a subtree. In this figure, edges leaving nodes are labeled with the parent node type, and edges arriving at nodes are labeled with the rule type dictating that node's pose relationship to its parent. The nodes of the supertree are partitioned into equivalent sets of nodes (gray boxes), such that one set of pose variables is sufficient to parameterize the pose of any node in the set. Given an observed set of nodes, we formulate parsing as a mixed-integer optimization that simultaneously chooses activations of each node in the supertree, poses for each equivalent set, and correspondences between observed nodes and their possible counterparts in the supertree.

4.1 Problem Setup

We motivate our formulation by rewriting the objective

$$\arg\max_T P_\theta(T|o) = \arg\max_T \log(P(o|T)P_\theta(T))$$

by taking a log and applying Bayes' rule. In this optimization, we might search over all trees T, where $p(o|T)$ is an indicator function that the observed nodes in T precisely match observed objects o and those trees that match the observed nodes are ranked by their tree score $P_\theta(T)$.

To search over the space of all trees, we form a "supertree" ST from the grammar, which is constructed in such a way that any tree T is a subgraph of ST[2]. The supertree is constructed by initialization a tree at N_{root}, and for a node of type N in the supertree, attaching one child for every rule R in $N.\mathcal{R}$ of type $R.N$ that would be produced by that rule. (See Figure 2b.) If the grammar is recursive, then we bound the depth of the supertree to keep it finite.

This supertree ST is a convenient data structure for parameterizing the space of all scene trees in the grammar. For each node n in ST, we add a binary variable $n.a$ indicating whether is active, and a continuous pose variable $n.x \in SE(3) = (n.t \in \mathbb{R}^3, n.r \in \mathbb{R}^{3\times3})$ representing the pose of that node if it is active. Any tree in the grammar (modulo recursion bounds) corresponds to a pattern of these node activation variables, and a particular choice of node poses; and given these variables, we can compute $P(T)$ and $P(o|T)$.

4.1.1 Supertree simplification via "equivalent sets". Because of the restrictions on node types in Section 3.2, representing complex logical relationships (e.g. a sink will contain a set of plates *and* a set of cups) requires chaining the primitive AND-OR-SET types together. This leads to supertree structures with sets of intermediate nodes whose poses are constant offsets from one another. (For example, the illustrative grammar in Figure 2 introduces intermediate "Pairs" and "Singles", which are purely logical nodes to handle production of varying numbers of their respective child types. The sink grammar in Figure 5 has many such intermediate nodes.) For each such "equivalence set" of nodes in the supertree, we can allocate one set of pose decision variables to describe the pose of *all* nodes in the group, and use [linear expressions of] those decision variables wherever each of the individual node's poses are needed. (Example equivalent sets are illustrated in Figures 2b and 5.) We detect these equivalent sets as partitions of the supertree by collecting the nodes in the supertree, connecting those parent and child pairs whose corresponding production rules are fixed offset rules, and taking the resulting set of connected subgraphs to be the set of equivalent sets. In practice, we maintain separate equivalent sets for rotation and translation.

4.2 Implementation of $P(o|T)$

To constrain that we only consider trees that match our observed set, for each node $\hat{n} \in o$, we add binary variables $b_{n,\hat{n}}$ for each $n \in ST$ of matching node type. A given $b_{n,\hat{n}}$ being active indicates that we're explaining the existence of observed node $\hat{n}$ with node n in the our parse tree. We impose constraints

$$\forall \hat{n} \in o \qquad \sum_{\text{same-type } n \in ST} b_{n,\hat{n}} = 1$$

$$\forall n \in ST \qquad \sum_{\text{same-type } \hat{n} \in o} b_{n,\hat{n}} \leq n.a$$

which enforce that all observed nodes are explained exactly once, no latent node is used to explain more than one observation, and a latent node that can produce geometry is active if and only if it explains an observation.

[2]This approach is similar in spirit to [Chua 2018], which forms an enormous factor graph over every possible (discretized) placement of every possible node type. In our case, we instead capture node poses with continuous attributes, leading to a much smaller tree over all generations.

When a correspondence $b_{n,\hat{n}}$ is active, we require that the latent node pose precisely matches the observed node pose. We implement this with a Big-M formulation:

$$\forall\{n, \hat{n}\} \text{ of matching type } :$$
$$|\hat{n}.t - n.t| \leq M_t * (1. - b_{n,\hat{n}})$$
$$|\hat{n}.r - n.r| \leq M_R * (1. - b_{n,\hat{n}})$$

which constrains a node translation and rotation to be elementwise equal to its corresponding observed node, but free to vary if not corresponded. $M_R = 2$ (since elements of the rotation are bounded in $[-1, 1]$) and M_t is larger than the largest reasonable distance between nodes of this type in the grammar.

4.3 Implementation of $P_\theta(T)$

Implementing $P_\theta(T)$ requires constraining these variables to only encode legal trees, and encoding the objective term $P_\theta(T)$ of those trees.

4.3.1 $x \in SE(3)$. We are optimizing over the poses of nodes in the scene tree. We parameterize poses with translation vector and rotation matrix components $n.x = (n.t \in \mathbb{R}^3, n.r \in \mathbb{R}^{3\times3})$. We impose the constraint

$$n.r \in SO(3): \qquad n.r^T n.r = \mathbb{1}, \quad det(n.r) = +1$$

with the mixed-integer convex outer approximation employed by [Dai et al. 2019]. These constraints allocate binary variables to partition the range of each element of $n.r$, and impose piecewise convex outer approximations (i.e. McCormick envelopes, [McCormick 1976]) on their bilinear products as they appear in the $SO(3)$ orthogonality constraints. As this is an expensive constraint, we are careful to only apply it where necessary: any node that produces geometry does not need this constraint applied, as that node's rotation will either be constrained to be equal to a matching-type observed node's rotation, or will not enter into the objective. (As discussed in Sections 4.1.1 and 4.3.5, in practice, we can actually avoid expressing this constraint in an even wider set of circumstances.)

4.3.2 Tree structure constraints. For any parent-child pair of nodes n^p, n^c in ST, we enforce

$$n^c.a \geq n^p.a$$
$$n_{root}.a = 1$$

which constrains that the child can only be active if the parent is active, and that the root of the supertree is always active.

Depending on the type of n^p, n^p being active may imply different constraints on the activation of the children, which we translate into constraints:

(1) **AND Node**: $n^c.a = n^p.a$: if the parent is active, all children are active.
(2) **OR Node**: $\sum_{n^c} n^c.a = n^p.a$: if the parent is active, exactly one child is active.
(3) **INDEPENDENT SET**: No constraints.
(4) **REPEATING SET**:
   (a) $\sum_{n^c} n^c.a \geq n^p.a$: if the parent is active, at least one child is active.
   (b) $n_i^c.a \geq n_{i+1}^c.a$: assign a consistent ordering n_i^c for the children in the child set; the children must activate in this order.
   (c) $n_i^c.t[0] \geq n_{i+1}^c.t[0]$: Under the consistent ordering of children, the x-component of the children translations must increase. This constraint is not necessary, but breaks a symmetry resulting from children of this rule being exchangeable, decreasing the number of equivalent solutions the MIP solver must sort through.

4.3.3 P_θ^N terms. Starting from Equation 1, $\log P_\theta(T)$ expands to a sum of densities:

$$\log P_\theta(T) =$$
$$\sum_{\{n^p, \mathcal{R}_{active}\} \in T} n^p.a \times \log P_\theta^N(\mathcal{R}_{active}|n^p)$$
$$\sum_{R, n^c \in \mathcal{R}_{active}} n^c.a \times \log P_\theta^R(n^c.x|n^p.x)$$

where multiplication by $n.a$ indicates that we only include densities for those nodes and rules active in the tree under consideration. To implement this in an MI context, we use convex reformulations of these terms.

For the discrete density terms $n^p.a \times \log P_\theta^N(\mathcal{R}_{active}|n^p)$, the implementation depends on the node type:

(1) **AND Node**: $\log(P_\theta^N)$ is always 0.
(2) **OR Node**: $\log(P_\theta^N) = \sum_i^{|N.\mathcal{R}|} \log(\theta^N[i]) * n_i^c.a$, where θ^N is the set of parameters controlling the Categorical probability of each rule activation for this node type.
(3) **INDEPENDENT SET**:

$$\log(P_\theta^N) = \sum_i^{|N.\mathcal{R}|} \log(\theta^N[i]) * n_i^c.a,$$

where θ^N is the set of parameters controlling Bernoulli probability of each rule activation for this node type.
(4) **REPEATING SET**:

$$\log(P_\theta^N) = \sum_i^{|N.\mathcal{R}|} (\log(\theta^N[i]) - \log(\theta^N[i-1])) \times n_i^c.a,$$

where $\theta^N[i]$ is the parameter controlling the Categorical probability of exactly i children being active, with $\log(\theta^N[0]) = 0$. Since $n_i^c.a \implies n_{i-1}^c$, this objective term will equal $\theta^N[i]$ if i children are active, or 0 if none are active (which indicates that the parent and all of its children are inactive and should not enter the objective).

4.3.4 P_θ^R terms. For the continuous density terms

$$n^c.a \times (\log P_\theta^R(n^c.x|n^p.x),$$

we need a way to deactivate the objective term $\log P_\theta^R$ when n^c is inactive. To avoid bilinear relationships between the activation variable and the node pose, we implement this deactivation with a slack formulation that adds variable $n^c.x_{slack}$ for each node:

$$|n^c.x_{slack} - n^c.x| \leq M(1. - n^c.a)$$

We can now express this cost term as

$$(1. - n^c.a) \times C + \log P^R_\theta(n^c.x_{slack} | n^p.x).$$

This term is designed such that when $n^c.a$ is deactivated, the objective takes 0 value. C is chosen to be the maximum value that can achieved by the rule probability $\log P^R_\theta(n^c.x_{slack}|n^p.x)$ when $n^c.x_{slack}$ is unconstrained; for our rule types, this is the same for any setting of $n^p.x$ and is easily computable from the log-density equations in Table 2. M is chosen to be sufficiently large to not restrict the pose when the child node is inactive. If the child is active, then the constant term falls away, the slack pose is constrained equal to the node pose, and the objective is the appropriate rule log probability. If the child is inactive, then the child pose is unconstrained, and so will be chosen to maximize $\log P^R_\theta$, which cancels with the constant term to result in zero overall density. This functionally removes this rule from $P_\theta(T)$ when the child is inactive, even if other constraints have been placed on $n^c.x$.

The exact functional form of $\log P^R_\theta(n^c.x_{slack}|n^p.x)$ depends on the rule type, as listed in Table 2. For most rule types, the rule log-density can be directly implemented as a quadratic objective, but there are two special cases:

(1) Bingham distribution log-densities are linear in the quaternion outer product qq^T; we represent the 10 unique terms in this outer product with intermediate variables. These outer product terms have linear relationships to elements of the rotation matrix according to the standard quaternion-to-rotation-matrix conversion formula, which we impose as linear constraints. See Appendix A for more details.
(2) Parent-frame Normal- and Bingham-distributed offsets involve a bilinear relationship with the parent node rotation $n^p.R$. When required, these bilinear relationships are approximated with piecewise McCormick Envelopes [McCormick 1976] that reuse the binary variables involved in constraining the relevant rotations $\in SO(3)$.

4.3.5 Short-circuiting in observed cases. The piecewise convex approximations to the $SO(3)$ constraints and bilinear relationships in the parent-frame rule cases are extremely undesireable: as shown in Section 5.1, having even a handful of these constraints in the parsing MIP can increase problem size and runtime by an order of magnitude. We can avoid these approximations by taking advantage of the fact that many nodes in the supertree are expected to produce geometry, and thus those nodes will either be corresponded to observed nodes or be inactive. This means that their pose can be described as taking the value of one of the observed nodes of matching type, rather than being directly optimized with a decision variable. In certain cases, this lets us rewrite non-convex cost terms in convex ways by taking advantage of our correspondence variables.

For each equivalent set of nodes in the supertree, we define that that equivalent set is *observable* if and only if any node in the equivalent set being active implies at least one observable node in the equivalent set is active. If an equivalent set is observable under this definition, then the pose of each node n in the equivalent set will take the value of one of a finite list of observed poses $n.X^o$. This list of poses is generated by, for every node n^o in the equivalent set that has geometry, accumulating $\hat{n}^o.x$ from all matching-type observed nodes $\hat{n}^o \in o$.

As an immediate consequence, any node in an observable equivalent set does not need its rotation constrained to be in $SO(3)$, as we know that if the node is active, its rotation will be constrained to be exactly equal to an observed rotation. But further, in the cases of the the bilinear relationships in parent-frame translation and rotation rules, we can rewrite the costs to avoid those bilinear relationships. Examining the Parent-frame Normal offset case, there is a bilinear relationship between node pose variables in the parent-frame offset term $\delta = n^p.r^T(n^c.t - n^p.t - \mu)$. If the parent node is in an observable equivalent set, then we can replace δ with a new intermediate variable δ_{slack}, which is constrained such that

$$\forall (r^o, t^o) \in n^p.X^o :$$
$$|\delta_{slack} - (r^o)^T(n^c.t - t^o - \mu)| \leq M * (1. - b_{n,\hat{n}}).$$

Here, we use the binary correspondence variable for observed node $\hat{n}$ to force δ_{slack} to be equal to a (linear expression of) that observed node's pose (r^o, t^o). This removes the bilinear relationship (and its corresponding loose piecewise outer convex approximation), in exchange for a list of linear constraints with Big-M activations. A very similar trick can be applied to handle the bilinear term $(n^p.r)^T(n^c.r)$ in the Parent-frame Bingham distribution case.

The combination of constraint short-circuiting and equivalent set formation is critical to scaling this system. Without either, the kitchen-sink grammar in Figure 5 would require hundreds of intermediate node poses, each requiring $SO(3)$ constraints; this leads to MIPs with excessive solve times even using powerful commercial solvers. By applying these simplifications to the MIP, the same scenes can be parsed in seconds because we have removed many or all of the nonconvex constraints.

4.4 Nonlinear optimization post-processing

Due to the approximations made to enforce rotation decision variables $\in SO(3)$ constraints, as well as McCormick Envelope approximations used to encode parent-frame translation and rotation rule probabilities, the solution to the MIP may not be precisely feasible or optimal under the true rotation constraints and objective. For that reason, we take the MIP solution and use it to seed a nonlinear program (NLP) of the same objective. The NLP is constructed identically to the MIP, minus the mixed-integer-convex-specific handling of the rotation constraints and bilinear cost terms, and with fixed parse tree structure equal to the structure uncovered by the MIP. This NLP is handed to a commercial solver and seeded with the MIP optimal solution; solutions are typically found in tens of milliseconds.

4.5 Multiple solutions

Because this optimization is a mixed-integer convex optimization, it can be solved to certified global optimality by a number of off-the-shelf solvers by an efficient (though still exponential in the worst-case) branch-and-bound algorithm. As a bonus, the same procedure can be used to extract the global top N best integer solutions (paired with their associated optimal continuous variable setting) as ranked by the objective. In our case, when there are no nonconvex constraints to approximate, different settings of the

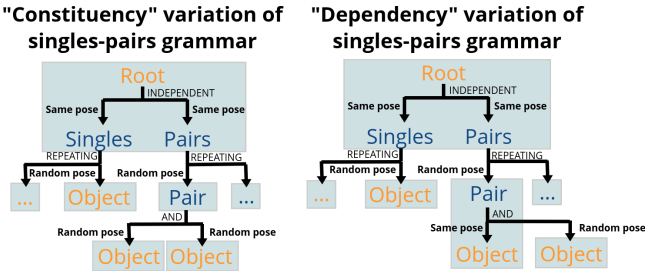

Figure 3: These two grammar variations, as represented by their supertrees, describe very similar distributions over objects that can appear either on their own or in spatially correlated pairs. The difference is in the spatial meaning of a Pair node: in the constituency variation, the objects in the pair are both randomly offset from the Pair node, making the Pair node's pose uncertain during parsing. In the dependency variation, the first object in the pair shares its pose with the Pair node, while only the second node is randomly offset. In this case, a Pair node's pose is *not* uncertain, as it is in an equivalent set with an Object since objects produce scene geometry. This second grammar variation proves significantly easier to parse in practice.

binary variables correspond to different parse tree structures – meaning that we can get diverse explanations of the scene from the MIP solver at the cost of more runtime. We explore some of these top N solutions in Section 5.1.

5 RESULTS

To demonstrate the expressiveness of this grammar setup and probe its performance, we focus on two examples.

(1) We demonstrate and benchmark our grammar's performance on the singles/pairs grammar used as an illustrative example in Section 4. In particular, we examine variations on this grammar to examine the effect including significant hidden continuous structure has on parsing performance.
(2) We demonstrate that our method scales to a complex 3D scene by designing a grammar to capture the arrangement of dishes in a cluttered kitchen sink, and show that our parsing technique can scale to this more realistic scenario.

For all experiments, we request the top 10 parses from the MIP solver, unless otherwise noted. We use Gurobi 9.0.2 [Gurobi Optimization, LLC 2021] as the MI convex optimizer and SNOPT 7.4 [Gill et al. 2018] as the nonlinear optimizer. We use Drake [Tedrake and the Drake Development Team 2019] for interfacing with these solvers, as well as kinematic calculations, simulation, and visualization.

5.1 Singles-pairs grammar

This grammar describes a set of oriented objects that appear either independently ("singles") or in pairs with related pose ("pairs"). As physical analogue, we consider modeling that aircraft might appear in the air either on their own, or in a formation as a pair, where the two aircraft in the formation tend to be close in position and orientation.

We provide two grammar variations to describe this scenario, inspired loosely by constituency and dependency parsing strategies from language. Single aircraft are produced with unit Normal translations and uniform random rotations. In the constituency grammar variation, pairs of aircraft are produced by sampling their cluster center from a unit Normal translation and uniform random rotation, and then sampling the offset of each aircraft from the cluster center in the cluster center's frame from a tightly-peaked Normal and Bingham distribution. In the dependency grammar variation, pairs of aircraft are produced by sampling the first object's location, and then sampling the other object offset from the first using the same tightly peaked distributions. These grammars are illustrated in Figure 3. *Critically, the only difference is in the relationships between the aircraft in a pair and their cluster center:* the minor change from the constituency variation to the dependency variation removes an unobserved equivalent set, which we will see has a significant impact on parsing performance.

5.1.1 Parsing a single example. Parsing the constituency variation of the grammar requires inferring the pose of the latent cluster centers, which produce no geometry and so must use piecewise approximations to the $SO(3)$ constraints and bilinear relationships between the cluster center pose and its constituent object poses. As seen in Figure 4 (top-left), the MIP-optimal parse tree has reasonable tree structure but only approximately correct node poses due to looseness in the approximations of the nonconvex constraints. The nonlinear refinement corrects the approximate poses to the true MAP parse tree configuration. Unfortunately, the top 10 solutions reported by the MIP solver are all variations on the same parse tree: the variations between the solutions are different settings of binary variables involved in the piecewise approximations that lead to very similar solutions, instead of actual variations in parse tree structure.

The dependency variation of the grammar captures the same set of scenes with the same relationships, but because the latent cluster center is in an observable equivalent set, we can use the observation-based short-circuiting trick from Section 4.3.5 to remove those expensive piecewise approximations. Both of the above shortcomings are improved in this situation: the MIP-optimal parse tree is already optimal in terms of node poses, and as illustrated in Figure 4 (bottom left), the top N solutions represent a diverse range of parse trees.

In the parsing example in Figure 4, both grammar variations could produce a maximum of 8 objects and generated supertrees with 15 nodes. Compared to the constituency parsing MIP, the dependency parsing MIP had vastly fewer continuous variables (417 vs 4680), binary variables (26 vs 214), and constraints (1496 vs 14063).

5.1.2 Parsing performance. Within each each grammar variation, we can examine the scaling behavior by increasing the maximum number of objects that can appear in the scene. Figure 4 (right) plots the distribution of runtimes of the parsing procedure across randomly generated scenes for both grammar variations, and compares how the runtime scales as grammar complexity increases.

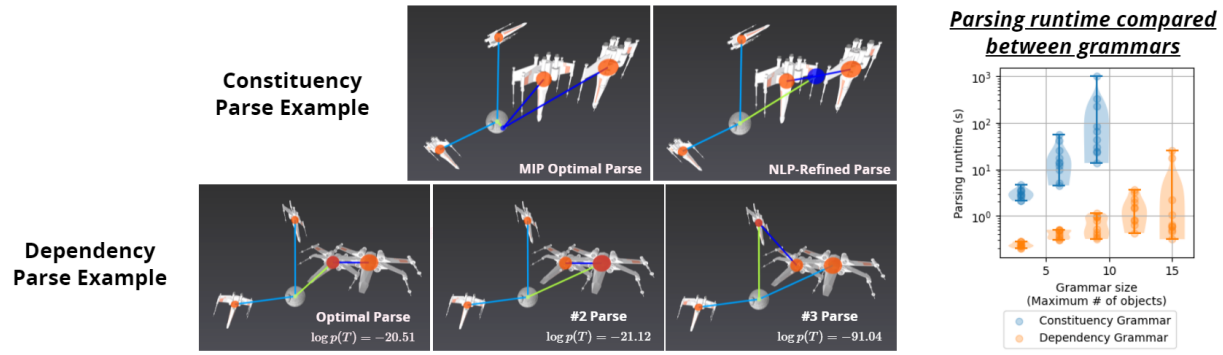

Figure 4: (Upper Left) The optimal parse of a representative singles-doubles scene using the constituency grammar variation. Post-processing with a nonlinear optimization pass resolves approximation errors in the MIP solution while keeping the optimal discrete configuration discovered in the MIP solve. (Lower Left) The top three parses of a representative singles-doubles scene using the dependency grammar variation. Nonlinear refinement is unnecessary in this case, as there are no nonconvex constraints with approximation gaps. (Right) A comparison of runtimes for random samples of scenes from both grammar variations as the grammar complexity increases.

5.1.3 Discussion. By comparing the results from the two equivalent grammars for this scene, it is evident that a grammar in which every equivalent set corresponds to some observable geometry is preferable in terms of parsing runtime, accuracy, and posterior sampling diversity. This result is not surprising: while we have demonstrated that the approximations to the nonconvex constraints lead to satisfactory MAP solutions, they are still expensive and complex for a MIP solver to resolve.

5.2 Cluttered sinks

To illustrate that our tool scales to meaningfully complex scenes, we demonstrate a grammar that captures some of the structure present in cluttered arrangements of objects in a sink. For motivation, consider the problem of verifying the performance of a robot meant to transfer objects from a sink into a dishwasher. Such a robot may be particularly sensitive to certain arrangements of objects, like objects packed into bowls or dishes tightly stacked on one another. We design a grammar that captures some of that structure, so that we might begin to quantify how often and with what spatial distributions those cases occur, and to detect those cases at runtime via scene parsing.

The grammar setup we employ is illustrated in Figure 5. To take advantage of the efficiency gains granted by the constraint short-circuiting described in Section 4.3.5 and the lessons learned from the singles-doubles grammar (Section 5.1), we architect our grammar so that each equivalent set of nodes contains observable geometry.

To test our parsing procedure, we hand-constructed a test dataset of 30 example scenes using a custom virtual reality scene construction tool. Our sink grammar structure and parameters were chosen to capture structure observed in these scenes.

5.2.1 Scene understanding and outlier detection. We show that we can use this grammar to parse out structure from observed scenes. In Figure 6, we illustrate parses of a handful of example scenes. We use edge colors to illustrate which relationships in the scene are

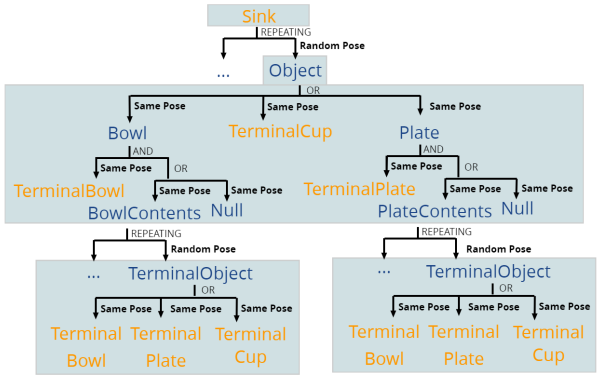

Figure 5: An abbreviated form of the grammar used to describe sinks. The sink produces a random random number of objects at random poses within it, using Normal and Bingham rules to dictate Object poses. Each object specializes into a Bowl, Plate, or TerminalCup. TerminalCups directly produce geometry, while Bowls and Plates have the option of additionally producing more objects relative to themselves using parent frame Normal and Bingham pose rules. (The "Null" object produces no geometry, and only exists to make sure it's possible to create a Bowl or Plate with no contents.) The full supertree has 625 nodes, but only 43 equivalent sets.

understood by the model to have high or low probability. One of these examples is an intentionally crafted "outlier" scene which was designed to violate the placement style used in the "inlier" scenes. It is apparent from the edge coloring of the parse of that "outlier" scene that parsing finds this scene unlikely under the grammar model. The parsing process indicates which *edge* is unlikely, offering a part-level explanation that this scene is an outlier because it it places a bowl too far outside of the sink.

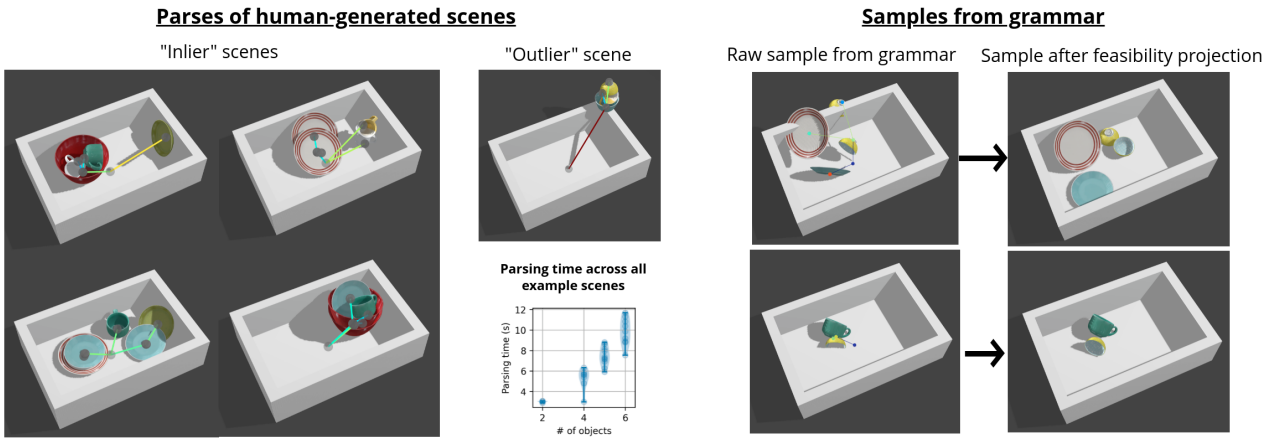

Figure 6: (Left) Human-generated example environments of varying dish models in a sink, along with their optimal parses from the MIP parsing procedure. In these plots, the edges of are colored according to the parsed rule probabilities: blue edges are more likely, green are moderately likely, and red edges are unlikely. The color range in all images are the same. An "outlier scene" handcrafted to intentionally violate model assumptions about the environment is also parsed for comparison. (Center bottom) A plot of parsing runtime for scenes in this dataset, organized by the number of objects in each scene. (Right) Example draws from the grammar before and after post-processing to make the samples physically feasible.

5.2.2 Sampling physically feasible scenes. Objects in physical environments like a sink are always nonpenetrating and usually in a static equilibrium. Unfortunately, these constraints are well beyond the scope of a context-free grammar like ours to express. This shortcoming does not affect the ability to *parse* scenes, but in order to produce samples from our grammar for visualization or simulation, we would like our samples to be physically reasonable. To achieve this, when we draw a list of objects from our model $\hat{o} \sim p(o)$, we then attempt to move those objects to the closest configuration that satisfies these constraints. We use SNOPT [Gill et al. 2018] and Drake [Tedrake and the Drake Development Team 2019] to find the closest non-penetrating arrangement, and then forward-simulate the resulting non-penetrating scene until it is stable. This operation biases samples away from the true distribution induced by the grammar, but produces satisfactory samples for visualization and simulation.

6 DISCUSSION

We have presented a scene grammar model that is rich enough to describe distributions over interesting scenes, but whose discrete and continuous rules are sufficiently simple to enable scene parsing by mixed-integer convex optimization. Our results indicate that this technique works especially well on scenes in which most parts of the scene grammar correspond to observed geometry, but we ensure that our tools are powerful enough to infer modest amounts of ambiguous continuous structure when needed. Unfortunately, scenes that have latent nodes for which rotations must be optimized rotations – e.g., latent oriented cluster centers – are fundamentally hard to parse due to the nonconvexity of $SO(3)$ and the nonconvex relationships those nodes can form with their children. While careful grammar design can avoid this issue, it deserves to be tackled head-on. One may consider mixed-integer nonlinear optimization, especially in combination with neural proposal schemes (e.g. [Ritchie et al. 2016]) for warm-starting, or other relaxations of the SE(3) constraints.

A major limitation of scene grammar models in general is that capturing scene-wide constraints – like all objects being in a physically feasible arrangement – is practically impossible to design into a context-free grammar. In this work, we rely on post-hoc modification of samples to prepare them for simulation or visualization, but an HMC-based method like [Ritchie et al. 2015] is relevant as a more principled alternative when unbiased conditioned samples are needed. Fortunately, this constraint issue does not interfere with the sort of density estimation, outlier detection, and scene parsing we've discussed in this work.

One pain point in deploying these grammars is determining their parameters. [Izatt and Tedrake 2020] demonstrates a variational EM strategy that, *given* access to a reasonable accurate and performant scene parsing procedure, can infer reasonable parameter estimates from unlabeled scenes using a variational EM strategy. We have performed pilot experiments that indicate that our parsing method plays very well with that EM approach, which lends hope to making design of these sorts of grammars more automatic, user-friendly, and data-driven. With that in mind, we hope that the tools we present in this work can move us closer to closing the system identification loop on these difficult open world environments.

REFERENCES

Christopher Bingham. 1974. An Antipodally Symmetric Distribution on the Sphere. *The Annals of Statistics* 2, 6 (Nov. 1974), 1201–1225. https://doi.org/10.1214/aos/1176342874 Publisher: Institute of Mathematical Statistics.

Jeroen Chua. 2018. *Probabilistic Scene Grammars: A General-Purpose Framework For Scene Understanding.* Ph. D. Dissertation. Brown University.

Hongkai Dai, Gregory Izatt, and Russ Tedrake. 2019. Global inverse kinematics via mixed-integer convex optimization. *The International Journal of Robotics Research* 38, 12-13 (2019), 1420–1441.

Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. 2020. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *European Conference on Computer Vision*. Springer, 715–733.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. 2016. Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems* 29 (2016), 3225–3233.

Feng Han and Song-Chun Zhu. 2009. Bottom-Up/Top-Down Image Parsing with Attribute Grammar. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 1 (Jan. 2009), 59–73. https://doi.org/10.1109/TPAMI.2008.65

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics* 31, 6 (Nov. 2012), 1–11. https://doi.org/10.1145/2366145.2366154

Philip E. Gill, Walter Murray, Michael A. Saunders, and Elizabeth Wong. 2018. *User's Guide for SNOPT 7.7: Software for Large-Scale Nonlinear Programming*. Center for Computational Mathematics Report CCoM 18-1. Department of Mathematics, University of California, San Diego, La Jolla, CA.

Jared Glover and Leslie Pack Kaelbling. 2013. Tracking 3-D rotations with the quaternion Bingham filter. (2013).

Gurobi Optimization, LLC. 2021. Gurobi Optimizer Reference Manual. https://www.gurobi.com

Gregory Izatt and Russ Tedrake. 2020. Generative Modeling of Environments with Scene Grammars and Variational Inference. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Paris, France, 6891–6897. https://doi.org/10.1109/ICRA40945.2020.9196910

Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. 2019. Meta-Sim: Learning to Generate Synthetic Datasets. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Seoul, Korea (South), 4550–4559. https://doi.org/10.1109/ICCV.2019.00465

Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. 2015. Picture: A probabilistic programming language for scene perception. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Boston, MA, USA, 4390–4399. https://doi.org/10.1109/CVPR.2015.7299068

Tianqiang Liu, Siddhartha Chaudhuri, Vladimir G. Kim, Qixing Huang, Niloy J. Mitra, and Thomas Funkhouser. 2014. Creating consistent scene graphs using a probabilistic grammar. *ACM Transactions on Graphics* 33, 6 (Nov. 2014), 1–12. https://doi.org/10.1145/2661229.2661243

Vikash K Mansinghka, Tejas D Kulkarni, Yura N Perov, and Josh Tenenbaum. 2013. Approximate bayesian image interpretation using generative probabilistic graphics programs. *Advances in Neural Information Processing Systems* 26 (2013), 1520–1528.

George Matheos, Alexander K Lew, Matin Ghavamizadeh, Stuart Russell, Marco Cusumano-Towner, and Vikash Mansinghka. 2020. Transforming Worlds: Automated Involutive MCMC for Open-Universe Probabilistic Models. (2020).

Garth P McCormick. 1976. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical programming* 10, 1 (1976), 147–175.

Bhairav Mehta, Ankur Handa, Dieter Fox, and Fabio Ramos. 2020. A User's Guide to Calibrating Robotics Simulators. *arXiv:2011.08985 [cs]* (Nov. 2020). http://arxiv.org/abs/2011.08985 arXiv: 2011.08985.

Seyoung Park, Bruce Xiaohan Nie, and Song-Chun Zhu. 2018. Attribute And-Or Grammar for Joint Parsing of Human Pose, Parts and Attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 7 (July 2018), 1555–1569. https://doi.org/10.1109/TPAMI.2017.2731842

Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomír Mech. 1996. *L-Systems: From The Theory To Visual Models Of Plants*.

Daniel Ritchie. 2016. *Probabilistic Programming for Procedural Modeling and Design*. Ph. D. Dissertation. https://dritchie.github.io/pdf/thesis.pdf

Daniel Ritchie, Sharon Lin, Noah D Goodman, and Pat Hanrahan. 2015. Generating Design Suggestions under Tight Constraints with Gradient-based Probabilistic Programming. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 515–526.

Daniel Ritchie, Anna Thomas, Pat Hanrahan, and Noah Goodman. 2016. Neurally-Guided Procedural Models: Amortized Inference for Procedural Graphics Programs using Neural Networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 622–630. http://papers.nips.cc/paper/6353-neurally-guided-procedural-models-amortized-inference-for-procedural-graphics-programs-using-neural-networks.pdf

Daniel Ritchie, Kai Wang, and Yu-An Lin. 2019. Fast and Flexible Indoor Scene Synthesis via Deep Convolutional Generative Models. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Long Beach, CA, USA, 6175–6183. https://doi.org/10.1109/CVPR.2019.00634

Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. 2021. Kimera: from slam to spatial perception with 3d dynamic scene graphs. *arXiv preprint arXiv:2101.06894* (2021).

George Stiny. 1980. Introduction to shape and shape grammars. *Environment and planning B: planning and design* 7, 3 (1980), 343–351.

Jerry Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Mech, and Vladlen Koltun. 2011. Metropolis Procedural Modeling. http://vladlen.info/papers/metropolis.pdf

Russ Tedrake and the Drake Development Team. 2019. Drake: Model-based design and verification for robotics. https://drake.mit.edu

Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D Goodman, and Pat Hanrahan. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Yibiao Zhao and Song-chun Zhu. 2011. Image Parsing with Stochastic Scene Grammar. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 73–81. http://papers.nips.cc/paper/4236-image-parsing-with-stochastic-scene-grammar.pdf

Yibiao Zhao and Song-Chun Zhu. 2013. Scene Parsing by Integrating Function, Geometry and Appearance Models. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Portland, OR, USA, 3119–3126. https://doi.org/10.1109/CVPR.2013.401

Ben Zinberg, Marco Cusumano-Towner, and K Mansinghka Vikash. 2019. Structured differentiable models of 3D scenes via generative scene graphs. (2019).

A BINGHAM DISTRIBUTION REPARAMETERIZATION

Given a 3×3 matrix of decision variables $r \in SO(3)$, we wish to penalize with cost corresponding to the Bingham log-density

$$ZM^T qq^T M.$$

Here, q is a quaternion corresponding to rotation r, and Z, M are Bingham distribution parameters matrices. Naively encoding this cost by adding decision variables for q leads to a nonconvex parameterization, since the conversion from r to q is not linear.

Instead, we introduce 10 decision variables for the 10 unique bilinear terms in the outer product qq^T:

$$qq^T = \begin{pmatrix} q_0^2 & q_0q_1 & q_0q_2 & q_0q_3 \\ q_0q_1 & q_1^2 & q_1q_2 & q_1q_3 \\ q_0q_2 & q_1q_2 & q_2^2 & q_2q_3 \\ q_0q_3 & q_1q_3 & q_2q_3 & q_3q_3 \end{pmatrix}$$

And enforce each of these constraints:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

$$r = \begin{pmatrix} 1 - (q_2^2 q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{pmatrix}$$

to ensure the (implied) quaternion is a unit quaternion that corresponds with the desired rotation. The matrix qq^T assembled out of these decision variables now enters linearly into the expression for the Bingham log-density.