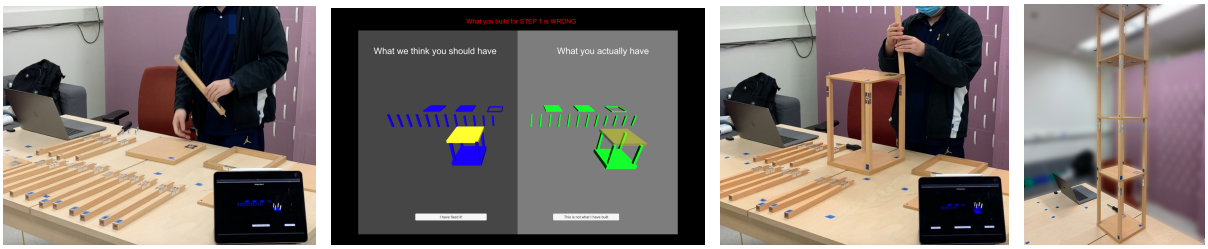
# StructureSense: Inferring Constructive Assembly Structures from User Behaviors

XINCHENG HUANG, University of Michigan, USA and University of British Columbia, Canada
KEYLONNIE L MILLER, University of Michigan, USA
ALANSON P. SAMPLE, University of Michigan, USA
NIKOLA BANOVIC, University of Michigan, USA






(a) (b) (c) (d)

Fig. 1. *StructureSense* infers constructive assembly structures from user behaviors: a) the user starts an assembly task (e.g., building a lamp) while *StructureSense* tracks the structure and provides the next step instructions according to the user's progress (the interface is mirrored to the tablet in the bottom-right corner), b) *StructureSense* detects that the user has used the wrong (top) shelf instead of the correct (middle) shelf and is alerting the user about the error, c) after fixing the error from the previous step, the user proceeds with assembly, and d) the final furniture structure that the user assembled.

Recent advancements in object-tracking technologies can turn mundane constructive assemblies into Tangible User Interfaces (TUI) media. Users rely on instructions or their own creativity to build both permanent and temporary structures out of such objects. However, most existing object-tracking technologies focus on tracking structures as monoliths, making it impossible to infer and track the user's assembly process and the resulting structures. Technologies that can track the assembly process often rely on specially fabricated assemblies, limiting the types of objects and structures they can track. Here, we present StructureSense, a tracking system based on passive UHF-RFID sensing that infers constructive assembly structures from object motion. We illustrated StructureSense in two use cases (as guided instructions and authoring tool) on two different constructive sets (wooden lamp and Jumbo Blocks), and evaluated system performance and usability. Our results showed the feasibility of using StructureSense to track mundane constructive assembly structures.

CCS Concepts: • **Human-centered computing → Interactive systems and tools**; *Ubiquitous and mobile computing systems and tools.*

Authors' addresses: Xincheng Huang, xchuang@umich.edu, University of Michigan, Ann Arbor, Michigan, USA and University of British Columbia, Vancouver, British Columbia, Canada; Keylonnie L Miller, keylonnm@umich.edu, University of Michigan, Ann Arbor, Michigan, USA; Alanson P. Sample, apsample@umich.edu, University of Michigan, Ann Arbor, Michigan, USA; Nikola Banovic, nbanovic@umich.edu, University of Michigan, Ann Arbor, Michigan, USA.


2474-9567/2022/12-ART204 $15.00
https://doi.org/10.1145/3570343

Additional Key Words and Phrases: tangible user interfaces, TUI, RFID, user modeling, bayesian inference.

**ACM Reference Format:**
Xincheng Huang, Keylonnie L Miller, Alanson P. Sample, and Nikola Banovic. 2022. StructureSense: Inferring Constructive Assembly Structures from User Behaviors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 204 (December 2022), 25 pages. https://doi.org/10.1145/3570343

## 1 INTRODUCTION

Coupling interactivity with everyday mundane physical objects, such as constructive assembly objects and connective building blocks, can turn any such object into Tangible User Interfaces (TUI) media [20]. People use constructive assemblies and connectable blocks to quickly build both permanent structures (e.g., playground structures, IKEA furniture) and temporary structures (e.g., camping tents, LEGO bricks). Making mundane physical objects interactive enables timely feedback that can help people to build quickly while avoiding building errors [12, 25, 40], engage in learning kinetically [32, 38, 44], and rapidly prototype and share designs [27, 33, 41].

Turning constructive assemblies into TUI media requires maintaining a steady connection between the physical structure of the assembled object and its virtual representation. However, most existing structure detection approaches use assembly-specific fabricated connectors [1, 2, 4, 10, 20, 25, 27, 30], which do not generalize to other constructive sets and assemblies. Touchless tracking methods (e.g., using Computer Vision [15, 24, 33, 41, 44, 45]) offer a more generalizable alternative, but are sensitive to environmental factors (e.g., occlusion and clutter). Tagging objects using passive ultrahigh-frequency (UHF) radio-frequency identification (RFID) [17, 18, 29, 39] are another generalizable alternative, but at a reduced sensitivity to environmental factors. However, existing tag-based methods can only track individual objects, but not the structures that such individual objects form.

In this paper we present *StructureSense* (Fig. 1), a robust, low-cost system for tracking constructive assembly structures based on passive UHF-RFID sensing. *StructureSense* enables tracking and inference of individual objects that make up the structure of arbitrarily different constructive assemblies. *StructureSense* infers constructive assembly structures from movements of individual objects. Each time the user moves one or more objects, our system samples likely structures that the user has built from the posterior probability distribution of all possible structures using Markov Chain Monte Carlo (MCMC) sampling [8, 13]. The system can then present the structures it sampled, ordered based on their relative probability, in a simulator to the user to enable a variety of applications from interactive assembly instructions to support rapid prototyping.

We illustrated our method in two distinct example scenarios: 1) *interactive guidance*, corresponding to building in a constrained space of possible and predetermined final structure(s), and 2) *authoring*, corresponding to building in an unconstrained space of undefined final structures. Our first scenario (building a floor shelf lamp) served to establish a baseline structure tracking performance. The second scenario (creating a virtual model with Jumbo Blocks–large physical snap-together connective blocks) was meant to push the limits of our system. For each scenario, we implemented a prototype with distinct sets of constructive assemblies in Unity3D[1] that used *StructureSense* to track what the user is building. We validated the prototypes in terms of their system performance (using benchmarking) and usability (via simplified user testing [36]).

Our findings showed that *StructureSense* can accurately and quickly track predetermined final structures. We also identified a set of challenges of using *StructureSense* in the authoring scenario with a large sample space of unknown final structures, with concrete steps to address those challenges. Our work contributes a generalizable Bayesian method for design and interaction [43] for tracking structures built from mundane constructive assemblies and connective building objects. This is the first step towards future UHF-RFID-based structure tracking systems that will turn any sets of mundane physical objects into TUI media.

[1] https://unity.com/

## 2 RELATED WORK

Here, we review computer technology that turns physical constructive sets into Tangible User Interfaces (TUIs) [20]. TUIs based on constructive assemblies have shown their potential in various fields (e.g., education [38, 44], crowd-sourced fabrication [25], rapid 3D digital content creation [27, 33, 41]). We reviewed existing techniques that enable tracking of individual objects and structures created using physical constructive sets.

### 2.1 Specially Fabricated Assemblies

Specially fabricated assemblies with embedded sensors have been used to track and identify the structure geometry during assembly tasks with high accuracy. Existing work shows that magnetic edge connectors [14], modified DC power connectors [1, 2], and infrared LEDs and phototransistors [3] can recover the structure of a constructive assembly by detecting the pair-wise connection between assemblies. To reduce the cost of embedding sensors into every individual object, recent approaches [4, 10, 19, 30] have highlighted how specially designed assemblies can pass down structural information of stacked-together blocks through capacitance to a single sensing device at the bottom. However, existing tracking approaches based on specially fabricated assemblies lack generalizability and cannot easily be applied to existing mundane constructive assemblies and building blocks.

### 2.2 Touchless Tracking

Touchless tracking methods, such as those based on radar sensing [45, 46] or Computer Vision (CV) [15, 22, 24, 33, 41, 44], enable tracking any mundane individual objects of a constructive assembly without requiring specially fabricated assemblies or instrumentation of objects they are tracking. For example, existing CV-based methods utilize a combination of color-based and depth-based computer vision tracking to detect individual objects and structures. Although some methods (e.g., [22, 44]) can track structures, they track them as monoliths (i.e., they do not track the process of creating the structure).

Therefore, it is not immediately clear how to use such methods for tracking the assembly process. Methods such as DuploTrack [15], CubeBuilder [41], and AR-based assembly support systems [24] can track structure creation process by comparing the point clouds of structures between steps. However, their higher computational overhead may introduce delayed responses to user inputs. CV-based methods are also sensitive to environmental clutter, occlusions, and fast-moving objects. This often leads to having to track one object at a time, which can affect the richness of the interactive experience.

### 2.3 Tag-based Tracking

Existing methods for mundane object tracking [9, 28, 29, 39] which utilize low-cost, off-the-shelf RFID technology [23] are impervious to environmental clutter, occlusions, and fast moving objects. For example, the can provide real-time standalone object state tracking—IDSense [29] and RapID [39] can track if an object is still, moving, swiped, or covered [29] and can report the object's instant velocity [39]. However, they lack the ability to track individual objects that make up a structure, including tracking how the individual objects are connected and assembled together; something that is crucial for tracking and inferring what structure the user is building.

Affixing UHF-RFID tags to connective joints or surfaces of individual objects as contact switches and detecting when an RFID tag is covered could be used to track which individual objects are assembled together using RapID [39]. However, applying such techniques to arbitrary mundane building assemblies is challenging due to the strict requirements regarding tag positioning for proper contact detection. For example, commonplace building assemblies, such as Lego bricks, that are connected with studs and tubes typically have minimal contact areas between them where it is difficult to affix RFID contact switches. RFID contact switches also do not work for building objects connected by metal connectors (e.g., furniture parts) as metals can interfere with RF signals.

Although embedding RFID-based contact switches into specifically manufactured assemblies [17, 18] alleviates the RFID tag positioning problem, such approaches (much like supplementing RFID technology with embedded sensors [16]) resemble specially fabricated assemblies and share the same challenges. Location tracking [7, 11, 31, 34] and gesture sensing [21] capabilities of RFIDs could add some of the capabilities of touch-less sensing to RFID-based tracking. However, those existing technologies currently focus on tracking single or few objects in controlled environments, making it difficult to track the building process of constructive assemblies.

## 3 DESIGN SPACE FOR TRACKING AND INFERRING CONSTRUCTIVE ASSEMBLY STRUCTURES

Here, we first define the design space for systems that can track constructive assembly structures. We identified the dimensions of the design space based on existing design features and considerations from prior literature. We then used the design space to call out the design trade-offs that all designs must consider, as there is not one single design that can address all design considerations and outperform all other designs along all the dimensions. We identified the following six design dimensions that make up our design space:

- *Generalizability*: The ability of a tracking method to track arbitrarily different constructive assemblies ranging from special constructive assemblies to any mundane object. The ability to generalize to arbitrary constructive sets is an often ignored yet important feature for assembly tracking systems. The higher generalizability of a tracking system makes the system immediately applicable to more real-world scenarios.
- *Compositionality*: The ability of a tracking method to track structures from monolithic structures (i.e., without regard for different individual objects that make up the structure) to polylithic structures (i.e., being able to track individual objects that make up a structure including how the individual objects are connected and assembled together). Assembly task tracking systems with higher compositionality are capable of providing more informative feedback to users, enabling applications such as assembly guidance [15] and tangible rapid 3D modeling [27, 33, 41].
- *Creative Freedom*: The ability of a tracking method to track structures ranging from pre-specified structures to previously unspecified structures. Existing constructive set tracking systems [15, 38, 44] highlighted creative freedom as a key design consideration for applications in entertainment and educational domains.
- *Robustness*: The ability of a system to perform against environmental factors (e.g., occlusion), ranging from those requiring specific and perfect lab conditions (feeble) to those unaffected by real-life environments (robust). Existing work [17, 18, 29, 39] has shown that highly robust tracking systems contribute to ease-of-deployment and ease-of-use, as well as system stability.
- *Responsiveness*: The time needed to provide a response to the user's interaction with the assemblies and maintain a virtually seamless building process. Responsiveness of interactive systems significantly impacts the user experience [36], where the response times range from having a delayed response (responding in more than 10 seconds) to reacting instantaneously (responding in less than 0.1 seconds) [37].
- *Accuracy*: The ability of a system to identify the correct structure. This is the inverse of the error rate, and ranges from inaccurate (no better than random coin toss) to oracles (100% accuracy). All the existing work has identified accuracy as a key design consideration as it has a direct impact on user experience.

Dimensions such as generalizability, compositionality, and creative freedom are considerations specific to structure sensing systems, while other dimensions such as robustness, responsiveness, and accuracy are considerations shared by general TUIs. Note the existing work places emphasis on low-cost solutions [29, 39]; however, we excluded cost as a dimension for comparison. While we recognize the importance of low-cost solutions for adoptability of tracking systems, we found that most existing methods have comparable costs that are impacted by a variety of factors. For example, specifically fabricated assemblies may be costly to manufacture; individual RFID tags may be inexpensive, but they require installation of costly RFID readers; computer vision-based systems do not require instrumentation of individual objects, but require costly camera equipment, *etc.*.

## 4 STRUCTURESENSE DESIGN OVERVIEW

Here, we detail our design of *StructureSense* along the design dimensions we have identified. Note that there are always design trade-offs; although any given design can be optimized to outperform all of the existing methods along a single dimension, no design can outperform all the other designs along all dimensions. With such trade-offs in mind, we designed *StructureSense* with the goal of increasing the compositionality of existing tag-based systems to track unspecified, polylithic structures for mundane constructive assemblies. We tried to preserve the generalizability, robustness, responsiveness, and accuracy of existing tag-based tracking systems.

### 4.1 Generalizability

To achieve high generalizability, *StructureSense* infers the progress of assembly tasks and the structures built from the motion of each individual object. *StructureSense* tracks movement of individual objects using RapID [39], thus matching the generalizability of existing tag-based tracking systems (assuming tracked objects are instrumented with commercially-available UHF-RFID tags). Given the wide range of RFID tag types available in the market, typically it is not difficult to find tags with the right shape and size to fit most constructive assembly objects.

### 4.2 Compositionality

Existing tag-based tracking methods [29, 39] usually trade-off compositionality for high generalizability. For example, RapID [39] provides two types of object state tracking: 1) the velocity of an object, and 2) whether an object is covered. Object velocity tracking falls strictly under the category of monolithic (atomic) object tracking resulting in low compositionality. On the other hand, tracking whether an object is covered could reveal some information about the geometric relationship between different individual objects in simple structures (e.g., "2.5D" Tic-Tac-Toe board). However, it is not immediately clear how RapID can be extended to track more complex assemblies (including 3D assemblies) without reducing to tracking specially fabricated assemblies.

To mimic the high *compositionality* of specifically fabricated assemblies, *StructureSense* uses a software-level knowledge of the assembly objects that includes: 1) a one-to-one mapping between constructive assembly objects and their attached RFID-tags, 2) the connective properties of the objects (e.g. for interlocking bricks, such as LEGO, studs connect to tubes, but studs do not connect each other), 3) the geometrical properties (e.g., block shapes, and sizes) of each type of object to help the system detect structures that are physically possible, and 4) geometrical key points for the assemblies to compare different structures. By default, our system automatically adds key points between any pair of objects with connective properties that indicate that the objects can be physically connected together. The users can then manually add more key points to reduce the number of possible structures (e.g., when the positions of connective joints do not uniquely identify each structure in the assembly).

### 4.3 Creative Freedom

Unlike existing tag-based tracking systems that can only track pre-specified structures, our goal was to enable tag-based tracking of unspecified structures. We designed *StructureSense* to infer the structure that the user is building from the real-time motion profile of different objects in the constructive set. Our inference model is based on the following insights: 1) objects that the user moves reveal information about possible objects in a structure (e.g., the user is either installing the moving object or inspecting it), 2) the order in which the user moves the objects implies the structure the user is assembling (e.g., some structures are more likely than others given a certain sequence of object movements), and 3) some structures are inherently more probable than others given the nature of assembly tasks (e.g., users may tend to close up as many (if not all) connective joints in a structure, structures that have better balance are generally more probable than the ones that easily fall over). For example, short individual object movement is more likely to be an inspection than an installation, while repeated movements of different objects in unison could signal that the user assembled the objects together.

### 4.4 Robustness

*StructureSense* preserves the robustness of existing tag-based tracking systems, such as IDSense [29] and RapID [39]. The UHF-RFID system that we use for RFID tag detection and tracking gathers radio signal strength (RSS) and phase through radio signal making it robust for tracking occluded assembly objects. However, the size of the construction area and the number of possible constructive assembly objects is constrained by the per-tag read rate of the RFID readers and the range of antennas. Also, similar to other tag-based tracking systems, *StructureSense* is limited to tracking constructive assemblies that are made of non-metallic materials, although it generally tolerates small metal connective parts such as nails and screws that are not in direct contact.

### 4.5 Responsiveness and Accuracy

Since *StructureSense* uses RapID [39] for tracking motion of individual objects, its performance is proportional to the high responsiveness and accuracy of RapID [39]. However, in addition to tracking the motion of individual objects, *StructureSense* also needs to estimate the probability of different possible structures. Thus, the responsiveness of *StructureSense* is further impacted by the speed of the structure sampling method and the number of samples required to estimate the posterior probability of different possible structures. The high accuracy of RapID [39] minimizes the impact of compounding errors for constructive assembly structures with many individual objects. However, the uncertainty of which structure the user is building (computed as the entropy of the posterior probability distribution of possible structures) increases with the number of individual objects the user moves as part of the building process. High structure uncertainty could be mitigated by asking the user to disambiguate which structure they have built at different stages of the building process.

## 5 STRUCTURESENSE SYSTEM IMPLEMENTATION

We implemented *StructureSense* as three separate modules: 1) constructive assembly movement tracking module based on RapID [39], 2) structure inference module using MCMC sampling [8, 13], and 3) applications programming interface (API) module that enables interfacing between the other two modules and any user applications.

### 5.1 Movement Detection and Tracking

For the purpose of our system, we only need to detect if an object is *moving* or *still*. Thus, we adapted the velocity tracking functionality of RapID [39] to enable near real-time movement tracking for individual constructive assembly objects. RapID [39] estimates the velocity of objects based on the phase difference between consecutive reads of a tag with a maximal latency of 200 milliseconds. We further smooth the velocity results reported by applying a median filter with a non-overlapping slide window. The median of the velocities estimated in each window will be the smoothed velocity. We empirically set the size of the median filter slide window to 10. With a per-tag read rate of 60 reads/sec, such a window size allowed us to get a good velocity estimation at 6 times/sec, which still feels instantaneous to the users [37]. Therefore, we employed an empirical velocity threshold of 30cm/s. This threshold can be customized according to desired experimental settings and movement sensitivity.

Our movement tracking module then uses real-time movement tracking to build and maintain a "movement history"—information about when the user moved each object throughout the assembly process (Fig. 2). Each time it detects a movement, *StructureSense* adds information about all objects that moved at that time to a dictionary where the keys are the unique identifiers for each object and the values are lists of movement steps. Note that the movement steps are not timestamps; we do not keep track of the actual time that elapsed between different movements. Instead, we simply keep track of movement count, which the system increments by 1 whenever it detects that one or more objects' motion status changes from *still* to *moving*.

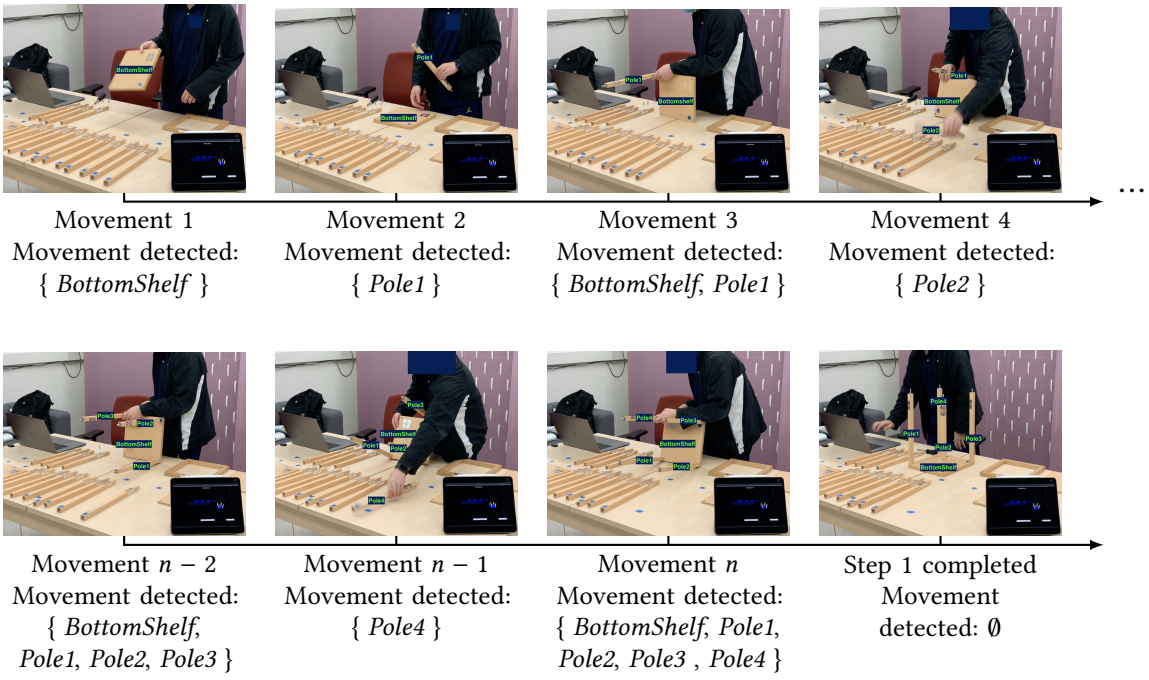



Fig. 2. An example of movement detection as the user progress through an assembly task object by object. Each time the system detects that one or more objects have moved it increments the movement time step by 1 and updates the movement history with all the objects that moved at that time.

## 5.2 Structure Inference

The structure inference module infers possible structures given the observed movement history. To describe the complex process of inferring structures, we refer to the current state of the entire constructive assembly set as *structures*, which is a set of constructive assembly structures that changes as the user assembles individual objects. During assembly, *structures* can contain uninstalled individual objects which we refer to as *atomics*, and partially assembled groups of objects, which we refer to as *composites*. We refer to connective joints (e.g., studs and tubes) as *connectors*. We then define *configurations* as possible ways that the user can assemble objects in a *composite* using different *connectors*.

Then, the set of all possible *structures* given the observed movement history is defined by any combination of different possible *configurations*. Note that the number of all possible *configuration*, and thus number of possible *structures*, increases exponentially with the number of *atomics* that the user interacted with and their *connectors*. Therefore, it is computationally infeasible to enumerate all possible structures at each step. Instead, our structure inference algorithm explores the state space of all possible structures at each movement step by sampling from the posterior probability distribution of possible structures given movement history.

*5.2.1 Sampling Possible Structures.* Since computing the posterior probability of possible structures given the current movement history is computationally intractable, the inference module uses Bayesian inference [5, 6] to estimate the probability of different possible structures by sampling from the posterior probability distribution of possible structures given the object movement information using Markov Chain Monte Carlo sampling [8, 13].

Here, we leverage the insight that the possible orientation of each building object can take in a structure is limited for rigid constructive assemblies. This reduces the need for RFID-based localization [7, 11, 31, 34] and orientation sensing [21] for simultaneously tracking multiple objects in real environments. Thus, we sample the probability distributions of building object placements and orientations in a structure with properly designed prior and likelihood functions. At each time step $t$, our model for MCMC is formulated in Eq. 1, where $M_t$ is the movement history from the start of the assembly until the current movement time-step $t$, and $S_t$ represents the assembled structures at time-step $t$.

$$\begin{aligned} P(S_t|M_t) &= \frac{P(M_t|S_t)P(S_t)}{P(M_t)} \\ &\propto P(M_t|S_t)P(S_t) \end{aligned} \tag{1}$$

Although common assembly tasks usually end up with a single structure at the end, users can create multiple "intermediate" sub-structures during the assembly process. To account for this, each structure in $S_t$ can contain multiple disjointed sub-structures.

Note that our sampling approach does not require full *a priori* knowledge of the state space of all possible structures (e.g., the size of state space, the configurations of all possible structures). Instead, to explore the state space using sampling, our method uses a proposal function that is capable of uniformly drawing from the state space by simulating the combination of assemblies at the software level (see Section 5.2.3 for details) and enough samples (determined by the MCMC algorithm chain length) to accurately estimate the posterior probability distribution. However, the MCMC chain length is application dependent and needs to be determined according to the constructive set it tracks (a choice that we illustrate later in Section 7).

*5.2.2 Representing Configurations.* Configurations are virtual representations of physical composites or atomics, which allow our system to construct, modify, compare, and eventually sample the structures being built on the software level. Each configuration stores the following information about a composite or atomic:

- The types and quantities of objects that each composite or atomic contains.
- The types and quantities of connectors.
- The connections between objects are stored as an undirected graph where each node represents an object and each edge represents a connection (or a single node for atomics).
- The coordinates of key points of the composite or atomic.

Adding two configurations together is straightforward as we can simply combine information about assemblies and connectors, and merge the graph representation of connections by adding edges. The merge between keypoint coordinates is done by modifying the coordinates stored in one configuration relative to the other. To compare two configurations, we first compare whether the two configurations contain the same number of individual objects with the same types. For example, a composite containing only one individual object is different from another composite containing two individual objects. We then compare the types of connectors used and the number of connections made. If both return true, we compare the key points. Note that we only use object types and quantities instead of their affixed RFID tags' ids for comparison, such that two configurations are considered the same as long as they look the same in reality. At the end of each structure inference, we traverse the resulting samples and compare and merge any duplicates.

*5.2.3 Structures Proposal Function.* MCMC requires a proposal function to draw samples for the sampling process. Our proposal function constructs and returns a random *structures*. Since any update to structures can only come from its previous state, to generate new proposed *structures*, we randomly pick *structures* from previously sampled structures (or initial *structures* with all constructive objects represented as *atomics* for the first step of the assembly task) and combine two of its composites or atomics together. We assume that the user is only capable of assembling

two parts together with one movement using both of their hands. If the combination generates a physically possible new composite, we return the updated *structures* as a proposal.

*5.2.4 Structures Prior Distribution.* To sample from the posterior distribution of structures, MCMC needs a prior distribution for structures. Our method is modular in that it allows for interchangeable priors (e.,g., the prior could be specific to a particular constructive assembly or the prior could consider a general property of constructive assembly structures). To illustrate our method, we used a general prior distribution that is applicable across different types of constructive assemblies. To construct our prior, we considered two inherent properties of structures: 1) the number of connectors left open (i.e. users close connectors as they build), and 2) the physical balance of composites (i.e. users tend to build things that do not fall over). Assuming these two properties are independent of each other, we define our prior $P(S_t)$ in Eq. 2, where $P_c(S_t)$ is the probability of structures having a certain number of open connectors, and $P_b(S_t)$ is the probability distribution considering the balance.

$$P(S_t) = P_c(S_t) \cdot P_b(S_t) \tag{2}$$

We model $P_c(S_t)$ (Eq. 3) as a zipf distribution on the number of total open connectors (denoted as *ConnOpen*) offsetting the minimal possible open connectors (denoted as *minConnOpen*). The $\alpha$ in Eq. 3 is a shape parameter that can be set to any real number bigger than 1, which controls the decreasing rate of the function. In our applications, we set the $\alpha$ here to 2.

$$P_c(S_t) = \text{zipf}(\text{ConnOpen}, \alpha, -\text{minConnOpen}) \tag{3}$$

The balance prior will only concern the composites. We calculate the distance between the central gravity point and the center of the composite's bottom surface. We calculate $d_{balance}$ as projected to the ground. The balance priori is then calculated as a normal distribution on $d_{balance}$ in Eq. 4, where the mean is 0 and the standard deviation $\sigma$ is adjustable as a hyper-parameter. We applied the connector prior to our furniture guidance application and applied both in the virtual Jumbo Block model creation application. For the latter, the standard deviation is set as the distance between two adjacent connectors on the same object (which is the same across the entire set of Jumbo Block).

$$P_b(S_t) = \mathcal{N}(d_{balance}, 0, \sigma^2) \tag{4}$$

*5.2.5 Structures Likelihood.* The likelihood function $P(M_t|S_t)$ models the probability of an observed movement given structures. We developed our likelihood functions based on the observation that recently moved objects are more likely to participate in a new installation. Therefore, we design our likelihood functions to be probabilities on "activeness". We define activeness for a composite or an atomic based on when it is last updated before the current movement step. Specifically, we traversed the movement profile of each individual object in a composite (or just the object itself for atomics), and use the most recent time-step to evaluate its activeness. With the $mostRecentTimeStep$ obtained, we calculate the value of $recency$ variable as in Eq. 5.

$$recency = currentTimeStep - mostRecentTimeStep \tag{5}$$

We developed two probability measures for $recency$ as shown in Eq. 6 and Eq. 7. The former favors higher activeness and the latter favors the opposite. The $\beta$ and $\gamma$ here are again the shape parameters for zipf distribution and can be adjusted as hyper-parameters to determine the system's sensitivity to activeness and idleness.

$$P_{active} = \text{zipf}(recency, \beta, 0) \tag{6}$$

$$P_{idle} = \begin{cases} \text{zipf}(-recency, \gamma, -10) & \text{if } recency < -10 \\ 1 & \text{if } recency \geq -10 \end{cases} \tag{7}$$

Recall that our proposal function is able to generate samples of *structures* in two different cases, a case where the movement leads to an installation, and a case where it does not. The first case, which we call "installation", describes a situation where an object's movement eventually leads to an update to the structures. We refer to the second case as "inspection", which describes the opposite situation where a moving object is eventually put back down without being connected to anything. We designed two different likelihood functions respectively for each of these two cases.

For cases of *installation*, we observed that installation is more likely to happen on the more active objects (i.e. atomics or composites) that have recently been moved. Therefore, intuitively, at a movement time-step, if an installation happens, the newly installed composite should come from recently moved atomics or composites, while the other atomics and composites that did not participate in this installation should be less active. Therefore, our likelihood function contains two activeness measures respectively for the two composites/atomics used for the installation, denoted as $S_1$ and $S_2$, and one idleness measure for the rest components as a whole denoted as $S_3$. The final likelihood is the product of the three measures as shown in Eq. 8.

$$L_{installation} = P_{active}(S_1) \cdot P_{active}(S_2) \cdot P_{idle}(S_3) \tag{8}$$

$$L_{inspection} = P_{idle}(S_{inspect}) \tag{9}$$

For cases of "inspection", inspecting the same object repeatedly and consecutively without making installation is rare. Therefore, our likelihood function for "inspection", shown in Eq. 9 is an idleness measure on the atomics or composites inspected, denoted as $S_{inspect}$. In our pilots of *StructureSense*, we observed that the users are more likely to install constructive objects than to inspect them. Therefore, we adjusted the shape parameters in Eq. 8 and Eq. 9 such that our system favors installations over inspections. Specifically, we set the $zipf$ shape parameters in Eq. 8 to 1.1 (such that the probability drops slower as the inactive time increases), and the $zipf$ shape parameters in Eq. 9 to 3.

### 5.3 Application Programming Interface (API)

Once the MCMC sampling is complete, *StructureSense* stores the resulting accepted samples of *structures* and their probabilities. Our system then makes these samples available to an application through an API. In the next section, we illustrate the use of the API in two example applications that we developed.

## 6 EXAMPLE APPLICATIONS

In order to illustrate *StructureSense* and evaluate the performance and usability of our system, we developed two example applications using Unity3D: 1) a guidance system for a furniture assembly task, and 2) an authoring system for virtual design creation with Jumbo Blocks.

### 6.1 Guidance Application

Our guidance application provides interactive instructions for building pre-specified constructive assembly structures. Interactive assembly instructions provide the user with step-by-step instructions according to their assembly progress, making it easier to keep track of the current step. Our application can also detect potential errors during the assembly process and guide the user on how to fix them. For our illustration, we implemented instructions for a shelf floor lamp (Fig. 3). The floor lamp we used consisted of 12 long poles, 4 short poles, 1 bottom shelf, 3 middle shelves, and one top frame. We designed our interactive instructions based on the paper

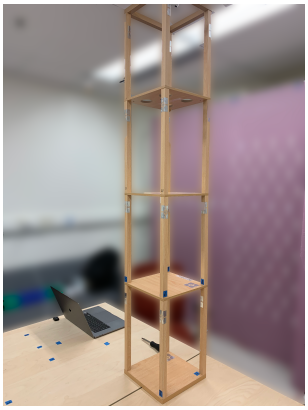
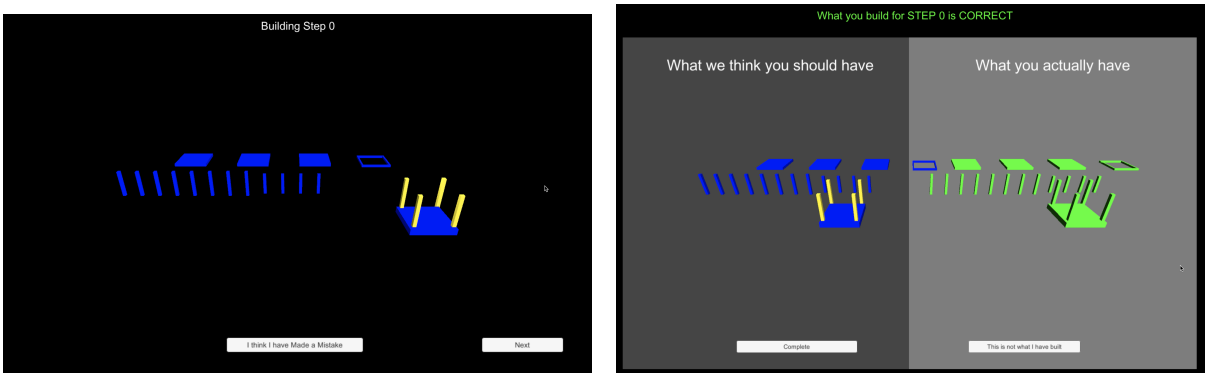


Fig. 3. An example constructive assembly (shelf floor lamp) that we used in our guided assembly application. Each object (lamp part) in the assembly is tagged with an RFID.




(a) (b)

Fig. 4. The guidance application in two modes: a) the instruction mode, and b) the verification mode.

instructions that came with the product and broke the assembly procedure into 8 steps. Each of the steps involves assembling between 1 and 4 furniture parts.

The user interface for our application (Fig. 4) had two modes: 1) the instruction mode (Fig. 4a), and 2) the verification mode (Fig. 4b). The instruction mode presented the user with the instruction for the current assembly step. When the user clicks on the "next" button to indicate that they completed the current step, the application uses the *StructureSense* API to sample the most likely structures and checks whether the user has assembled the structure in the step correctly by comparing the structures in the instructions with the most likely structure that *StructureSense* returned. In addition to automatically entering the verification mode after completing a major step, the user can enter the verification mode anytime by clicking on the "I think I have made a mistake" button if they have any doubts about what they have built.

*6.1.1 Helping Users Recognize and Recover from Errors.* Assembly errors are inevitable on both the user's and the system's part. When the guidance application catches such errors, it alerts the user when they enter the verification mode (Fig. 5). During an assembly task and after the user completes an assembly step (Fig. 5a), our

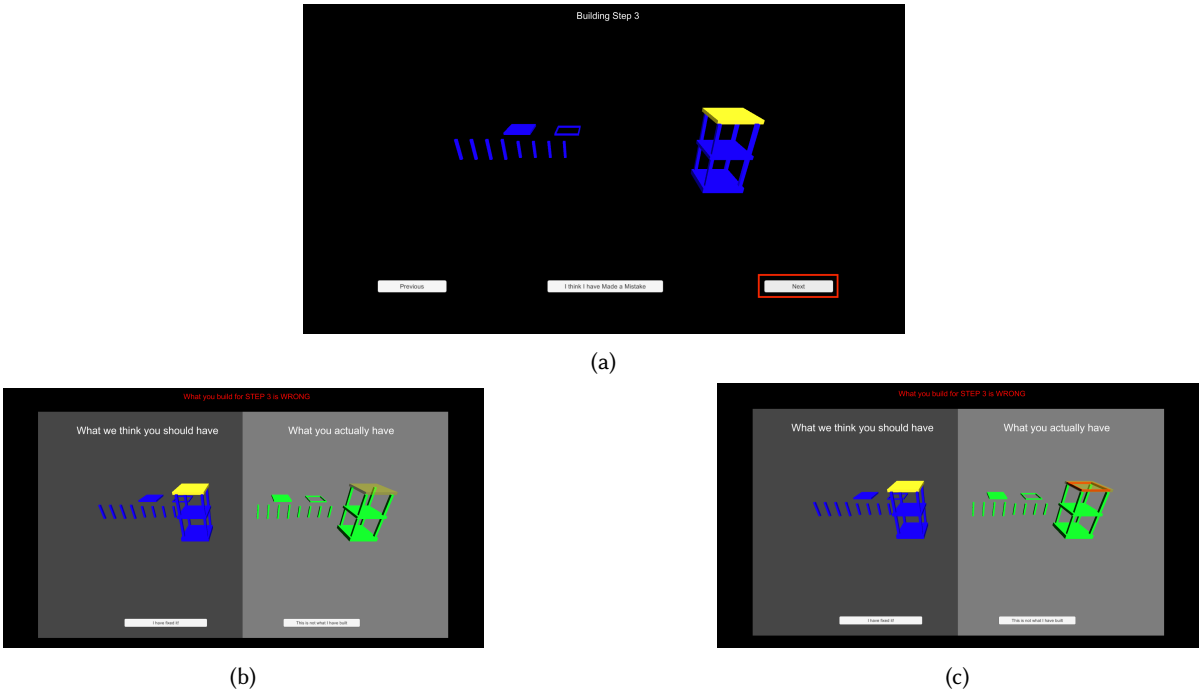


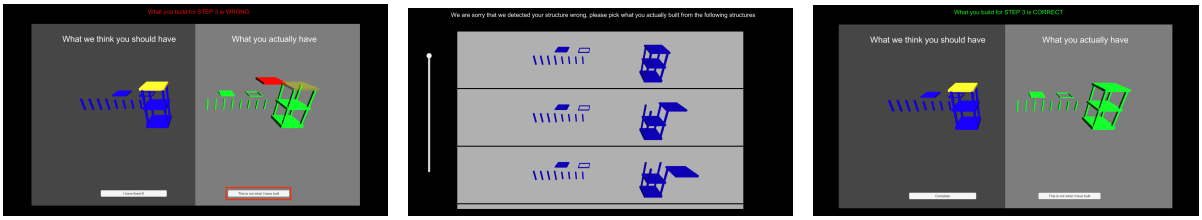

Fig. 5. Upon completing a step in the instruction mode and clicking on the *next* button (highlighted with a red rectangle) (a), the user enters the verification mode. The system detects one of two possible types of errors: (b) the user skipped or missed a component (missing component highlighted in transparent yellow), or (c) the user used a different component than instructed (the wrong component highlighted in red and the correct component in transparent yellow). After the user has fixed the error, they can click on "*I have fixed it!*" button on the verification interface and proceed with the assembly task.

guidance application is capable of detecting the following two types of errors: 1) when the user skipped or missed a component (Fig. 5b), and 2) when the user used a different component than instructed (Fig. 5c). In the verification mode, correctly assembled parts are displayed in green, the missing parts in transparent yellow, and the mistakenly installed parts in red.

The system may also make a wrong prediction about the most likely structure the user is building: 1) the system may wrongly detect an installation error when the user installed the object correctly, or 2) the system may fail to detect an installation error. Fig. 6 illustrates how the users can navigate themselves out of a system error. When they notice that there is a difference between what they built and what the system infers, they can click on the "This is not what I built" button in the verification mode (Fig. 6a). The system will then list all the possible structures in the current state space (Fig. 6b), in the order of their inferred probability, allowing the user to correct the system by selecting the actual structure they built from the list (Fig. 6c).

## 6.2 Authoring Application

Our authoring application gives the user creative freedom to assemble previously unspecified structures using constructive virtual assemblies. Instead of providing guided instructions, the authoring application enables the user to rapidly prototype and share designs. For our illustration, we used Jumbo Blocks, a bricks constructive set resembling large LEGO bricks. Fig. 7 shows an example structure built with Jumbo Blocks.




(a) (b) (c)

Fig. 6. Our system allows the user to indicate the right structure they have built when the system structure prediction is wrong. For example, when the verification page wrongly alerts the user that the top shelf (highlighted in red) is wrongly installed (a), the user can click on the "*this is not what I built*" button (highlighted in the red rectangle). The system will then prompt the user with a list of all the possible structures from the state space ordered by their relative probability (b). The user can click on the one that correctly represents what they actually built, and the system then returns to the verification mode (c), where the user can now click on the "*Complete*" button and proceed with the assembly task.

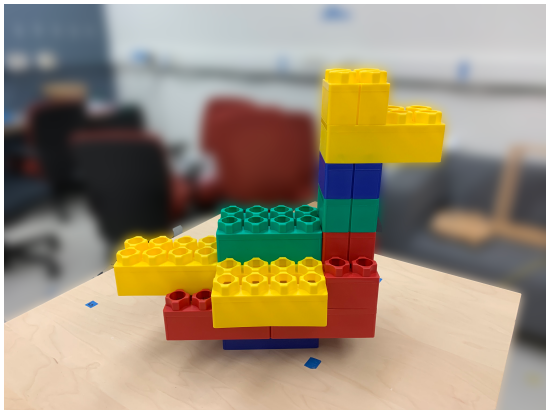


Fig. 7. An example constructive assembly (a Jumbo Blocks duck) that we used in our authoring application. Each object (Jumbo Block brick) in the assembly is tagged with an RFID tag.

The user interfaces for our example application (Fig. 8) feature a split screen showing: 1) a virtual representation of the most likely current structures the user has assembled on the left, and 2) a list of other possible structures ordered by their probability in descending order on the right. The application uses callbacks from the *StructureSense* API, which fire each time that the system detects that the user added an object to the physical structure, to update the two split screens with the latest results from *StructureSense*.

The user can then inspect the models in the split screens using direct manipulation (e.g., zoom in/out and rotate the models). As long as the most likely structure on the left is what the user has built, they can simply proceed with assembly. However, if the user notices that *StructureSense* has made a mistake tracking the current structure, the user can search through the list of possible structures on the right and select the correct one. At any point in the assembly process, the user can save their virtual structures model (e.g., to create interactive instructions for our first application).

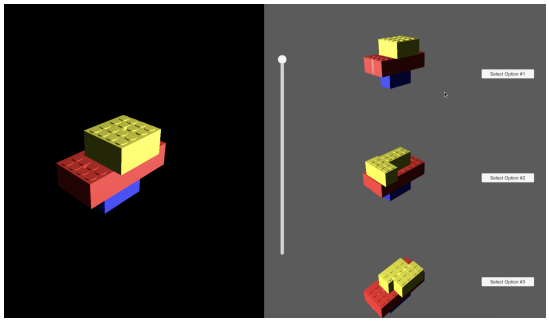


Fig. 8. Authoring application interface. The left half of the interface provides a detailed view and the right half of the interface lists all the possible structures

## 7 SYSTEM EVALUATION

We used our two applications to evaluate *StructureSense* along the design dimensions we identified in Section 3. We first benchmarked the responsiveness and accuracy of our system for different types of structure predictions and used the results to fine-tune our system (e.g., selected MCMC chain length). We used the fine-tuned system to perform user evaluation of *StructureSense* using the simplified user testing method [36]. We then performed a comparison of *StructureSense* with existing work along the dimensions of our design space.

### 7.1 System Benchmarking

One of our design goals is to enable high performance (i.e., high responsiveness and accuracy) of *StructureSense*. The performance of *StructureSense* is affected by: 1) the number of objects in the constructive assembly, 2) the number of possible connections between the objects, and 3) the number of samples required to perform structure inference (i.e. MCMC chain length). *StructureSense* has no control over the first two potential bottlenecks—the more individual objects there are and the more connectors objects have, the more possible configurations they can be assembled into. Although *StructureSense* has control over the MCMC chain length (i.e., the number of samples), the larger the space of possible structures, the more samples are needed to obtain a reasonable estimation of the posterior probability distribution of structures. This highlights the trade-off between processing times (responsiveness) and the uncertainty of resulting structures (which affects accuracy).

Here, we first focus on benchmarking the responsiveness of *StructureSense*. Though ideally, any tracking system would react in a way that appears instantaneous to the user (e.g., in less than 0.1 seconds [37]), our goal is to support interactions that enable the user to keep their attention on the assembly task (i.e., optimize the system to respond in closer to a second, but no more than 10 seconds [37]). We benchmarked the two modules that our system responsiveness is most affected by separately by measuring: 1) the time it takes to detect the movement of different individual objects, and 2) the time to perform structure inference.

We benchmarked our system's performance on simulated user movements from the two applications we described in the previous section. We simulated user movements using repeated measures of actual, practiced, expert movement data from scripted assembly tasks that one of the authors conducted using the two applications. To obtain the eventual "simulated data", we excluded inspections (i.e., movement histories that do not lead to new structures) and removed any RapID tracking errors (e.g., false-positive movement detection) from the expert movement data. We used such simulated data to avoid any measurement noise due to variance in user performance. Feeding such simulated data back to the system allowed us to automate the system benchmarking process without asking users to repeat the assembly processes in prolonged user study sessions.

We implemented each of the three *StructureSense* modules and enabled communication between them using network protocols. We conducted all our benchmarking experiments on a MacBook Pro 16-inch 2019 with a 2.3GHz 8-core CPU and 16GB of memory. We used an *Impinj Speedway Revolution R420 UHF-RFID Reader*[2] and an *Impinj Far Field RFID Antenna* with a maximal sensing range of approximately 10 meters.

*7.1.1 Benchmarking Movement Detection.* We empirically determined per-tag read rate to ensure movement tracking performance. During early testing of our system, we observed a performance decrease when the per-tag read dropped below 25 read/sec. The assembly tasks we used in our experiments contained up to 20 constructive objects, with which we have a 60 read/sec per-tag read rate by setting the reader mode to "Max throughput" (FM0 encoding). We empirically estimated that our system can simultaneously track between 30 and 40 assemblies without saturating the RF system.

The physical configuration of the environment has minimal effect on our system setup. The system is functional as long as the space for the assembly tasks tracked is within the sensing range of the RFID antenna. We conducted our assembly experiments on a $1.5m \times 2m$ table and placed RFID antenna $1m$ from the table. We added clutter and additional "distractor" tags into the environment to simulate adverse environmental factors; however, we primarily evaluated the robustness of the system when the user was obstructing the "line of sight" between the antenna and the objects in the course of the assembly process. Our final configuration resulted in movement detection that took on average 190.38 *milliseconds* ($std$ = 964.7 *milliseconds*) to detect the movement state of all individual objects in our applications (while maintaining RapID's detection accuracy [39]).

*7.1.2 Benchmarking MCMC Chain Length.* The first step in configuring structure inference is to determine the number of samples to generate. To benchmark our system's sampling performance, we ran our structure inference algorithm with five different MCMC chain lengths (ranging from 1,000 to 5,000 samples) and repeated the runs five times. We then calculated the average time that our system took to sample from the posterior probability distribution of structures for each movement in our "simulated data" (Fig. 9).

Our results showed that the sampling time is linearly correlated with the MCMC chain length (i.e., each sample takes constant time, as expected). However, the sampling on average took longer to complete on data from the authoring application (Fig. 9b) than data from the guided application (Fig. 9a) due to the former having a large number of individual objects and possible ways to assemble them together. This is because the sampling step includes generating possible structures using a proposal function which takes longer when the state space of possible final structures is large.

Note that different applications may require sampling at different times during the assembly process or at different time intervals. For example, our guidance application only needs to present the results of structure inference when the user enters the verification mode but can keep inferring structures as the user proceeds with the assembly. Similarly, in the authoring application, the user may build and explore a few different structures before selecting and saving the final structure they built (which may provide even more time between different interactions with the application).

Thus, we set the length of the MCMC chain to 2,000 for the guidance application and 4,000 for the authoring application, which resulted in on average approximately 1 *second* and 5 *seconds* to complete respectively (both under the upper bound of 10 *seconds* in accordance with our design goals). We selected the longer chain length for the authoring application because we hypothesized that the users will have longer times between interactions, and it could benefit from a longer chain length to explore the larger state space of possible final structures.

*7.1.3 Benchmarking Structure Inference.* To benchmark the responsiveness of our structure inference (now with our selected MCMC chain length), we simulated two scenarios for each example application: 1) *with user disambiguation*—the best case scenario where the user indicates correct structure they are building, and 2)

[2]https://www.impinj.com/products/readers/impinj-speedway

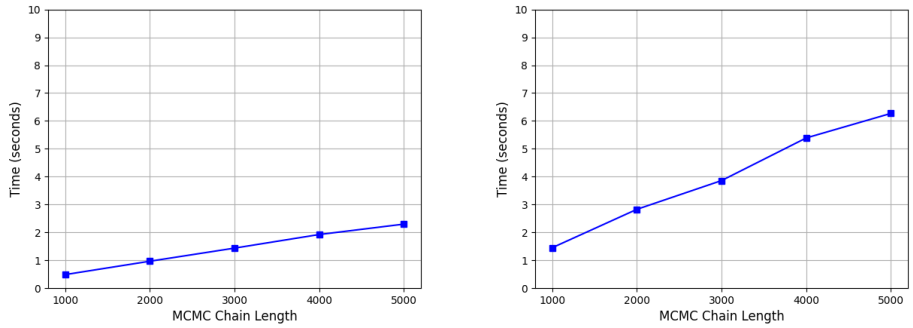



Fig. 9. Average system response time vs. MCMC chain length for: a) the guidance application, and b) the authoring application.

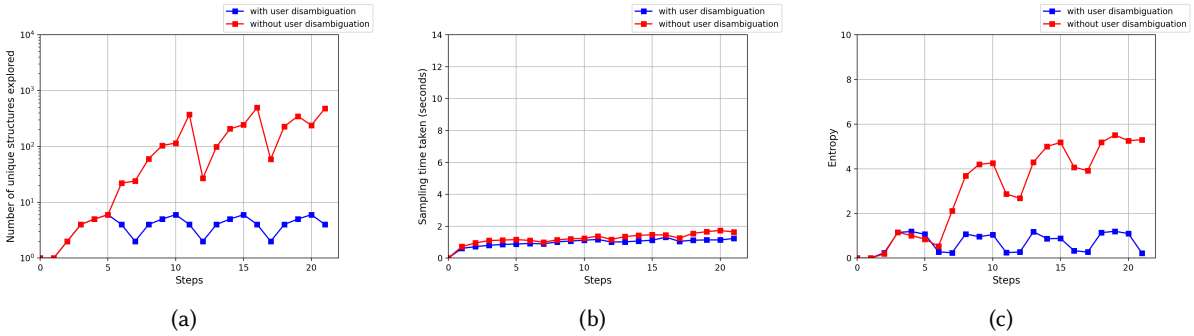



Fig. 10. Guidance application benchmarks at each assembly step: a) mean number of unique MCMC sampled structures, b) mean sampling time, and c) entropy of the estimated posterior distribution. The figure shows results with user disambiguation in blue and without user disambiguation in red.

*without user disambiguation*—the worst case scenario in which the user provides no feedback about what they have built to the system. The presence of user disambiguation allows the system to constrain the number of possible configurations in future steps, thus improving system response time. However, in the absence of user disambiguation, the system's performance will likely degrade as the number of possible configurations increases.

For each movement step in our simulated data, we repeated our structure inference ten times and computed the mean number of different unique structures sampled using MCMC, the mean time to perform the inference, and the mean entropy of the structures' posterior probability distribution (Eq. 10). Note that the entropy measures how much information the estimated posterior distribution provides about the structure that the user is building. Thus, high entropy corresponds to high uncertainty that could potentially lead to low inference accuracy. In the disambiguation condition, we simulated the disambiguation action every 6 movements (corresponding to an average step size in the guidance application).

$$H(X) = -\sum_{i=0}^{n} P(x_i)\log(P(x_i)) \tag{10}$$

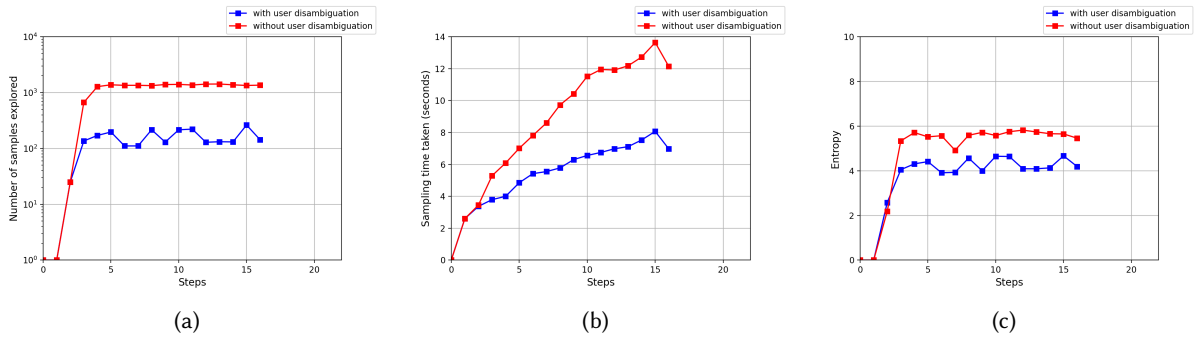



Fig. 11. Authoring application benchmarks at each assembly step: a) mean number of unique MCMC sampled structures, b) mean sampling time, and c) entropy of the estimated posterior distribution. The figure shows results with user disambiguation in blue and without user disambiguation in red.

Our system was highly responsive when inferring structures from the simulated guided assembly data (Fig. 10; best case: mean sampling time=0.97s, std=0.27s; worst case: mean sampling time=1.22s, std=0.36s). The entropy in the best-case scenario remained low because the user structures disambiguation constrains the structure sample space at each next step. However, as expected, when the user disambiguation was absent, the large structure space resulted in higher entropy, which in turn could result in lower inference accuracy.

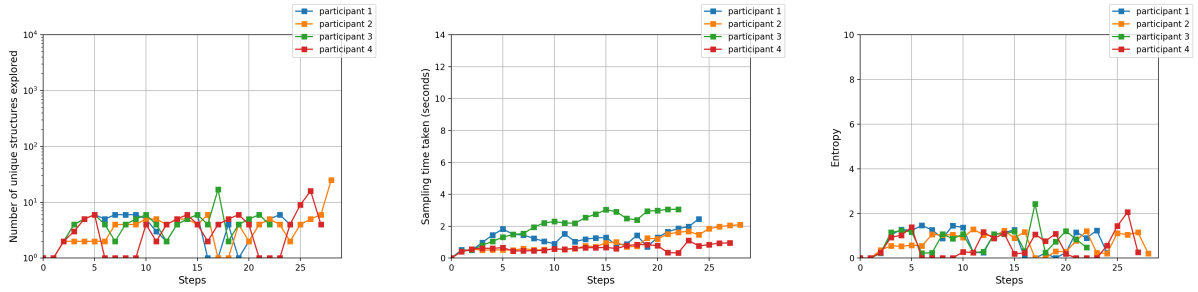
However, the large structure space and lack of pre-specified final structures when running structure inference on simulated authoring application data showed the limits of our system (Fig. 11). The number of different structures that the user could build at each assembly step grew exponentially. With hundreds of possible structures at each step, our system took longer to sample structures (mean processing time=5.39s, std=2.02s) and had higher entropy even in the best-case scenario than in the simulated guided assembly application.

Although the response time in the best-case scenario was still within our limit of 10 *seconds*, the mean response time in the worst-case scenario quickly exceeded that time limit. This showed that our earlier assumption that the authoring task could benefit from a longer MCMC chain length than the guided task was not correct. However, our further investigation showed that reducing the MCMC chain length to increase the responsiveness of the system in the authoring application increases the entropy, which could in turn negatively impact structure inference accuracy. Thus, we kept our MCMC chain lengths the same before proceeding to user evaluation.

## 7.2 User Evaluation

We then continued our evaluation and conducted simplified user testing [36] where participants interacted with our two example applications. We recruited 6 participants (2 female, 4 male). All of them interacted with the guidance application, and five of them interacted with the authoring application. One of the participants left the study early and did not interact with the authoring application. The study on average took 1 hour to complete, and we compensated the participants $15 for taking part in the study.

*7.2.1 Tasks and Procedures.* Participants arrived at our lab, and before taking part in the study, they read and understood the consent form, which detailed the study and the tasks. Only participants that consented to be part of the study could proceed with the study task. The participants could also optionally consent to be video recorded. The investigator then answered any questions from participants before proceeding. The investigator then showed the participants how to interact with the applications.




(a) (b) (c)

Fig. 12. Guidance application performance at each assembly step: a) mean number of unique MCMC sampled structures, b) mean sampling time, and c) entropy of the estimated posterior distribution. Each line corresponds to a different participant in our study. Note that the steps include inspections and wrong installations.

The participants first performed the guided assembly of the floor lamp. When done with the guided task, participants performed the authoring task, where the investigator instructed the participants to create a virtual model of a duck (Fig. 7). The participants could provide structure disambiguation in both tasks. This was implicit in the guided application, and the investigator showed the participants how to do it in the authoring application.

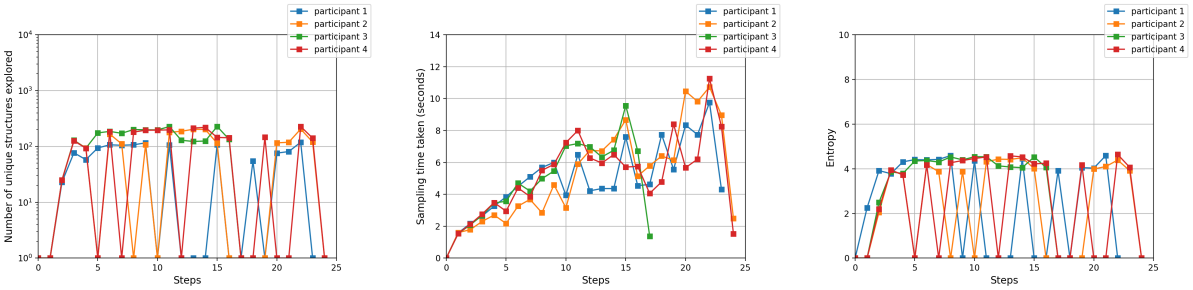
The investigator instructed the participants to perform a think-aloud during the assembly tasks. Throughout the participants' assembly procedure, the investigator noted: 1) any usability issues that participants experienced, and 2) whether the system correctly responded to the participant's actions. The study software logged individual block movements and the time the participants took to complete the assembly tasks. Upon the completion of each assembly task, the investigator followed up with clarifying questions regarding participants' user experience, any prediction errors that they encountered, and any other concerns they had.

We then analyzed individual participant data. Movement data from 3 participants were corrupted due to logging failure, which resulted in movement data from 4 participants in each condition. We aggregated the participants' think-aloud reflections and responses to the follow-up questions (referring to the recorded study sessions after the user experiments when needed).

*7.2.2 System Performance Results.* Here, we reproduced our system benchmarks using actual participant data (Fig. 12 and Fig. 13). Note that the participants took more steps than in the idealized assembly tasks from simulated data because participant data contained inspections and wrong installations.

The participants took on average 576 *seconds* ($std$ = 126 *seconds*) to complete the guided assembly task. At each assembly step, the system remained responsive and completed structure inference in around 2 *seconds*, with an exception of P3 where it took slightly longer (Fig. 12b). To complete the authoring task, the participants on average took 782 *seconds* ($std$ = 67 *seconds*). The median response time for each step was 5 *seconds*, with inference only exceeding the 10 *seconds* threshold we set for our designs towards the end of the assembly task for two participants.

*7.2.3 Qualitative Evaluation Results.* In guided assembly tasks, all the participants were able to assemble the lamp with minimal errors. Participants made only two installation errors, and in both cases, our application alerted them to the error, which they fixed. Our application wrongly alerted the participants about structure differences when they completed the instructed steps correctly twice. In both cases, the system wrongly detected object movements as "inspections". However, the application enabled the users to quickly recover from the errors. In all cases, the correct structure ranked second or third in the list of alternative structures.




(a) (b) (c)

Fig. 13. Authoring application performance at each assembly step: a) mean number of unique MCMC sampled structures, b) mean sampling time, and c) entropy of the estimated posterior distribution. Each line corresponds to a different participant in our study. Note that the steps include inspections and wrong installations.

We followed up with the two participants who encountered the system errors (*P2* and *P4*) regarding their experience. Both of them reflected that it was somewhat confusing seeing the system predicting a structure different than what they built. However, they also acknowledged that the system made it straightforward to navigate out of such system errors.

At the end of the guided assembly tasks, we followed up with the participants regarding their experience with the system response time, and all of the participants reported that they did not notice any latency while interacting with the application. We attributed this to the fact that the system starts the structure inference immediately at the beginning of each installation (i.e., when the user starts interacting with a component). Therefore, in the guidance application, most structure predictions are finished before the corresponding installation, giving users the experience of instant responses.

However, all five participants that interacted with the authoring application reported during their think-aloud that system response delay was noticeable in most assembly steps. When we followed up with the participants regarding their concerns about the delayed system response time, they commented that the experience was "unnatural" as they needed to wait for the system to update what they were building after each assembly step.

At each step of the authoring process, the system provided median 62 alternative structures to select from. This forced the participants to confirm their desired structures after each step, which significantly impacted the usability of the application. The median rank of the participant-selected alternative structure was 16.5. At times, toward the end of the assembly tasks, the participants could not find the exact virtual representation of what they built because the number of possible structures has grown too large for the proposal function to guarantee full coverage of the structures space. When this happened, the participants selected the closest approximation.

## 7.3 Comparison of *StructureSense* with Existing Systems

Here, we used our findings to place *StructureSense* in the design space we defined in Section 3 and compare it with other existing tracking systems. In Fig. 14, we illustrate the placement of *StructureSense* and representative prior systems in the design space. Note that while our design space considers all papers from our related work, our illustration in Fig. 14 only contains a subset of representative work that allows us to clearly place *StructureSense* within our design space. Specifically, in Fig. 14, we show the placement of representative systems from specifically fabricated assemblies (CapStone [10] and RFIBricks [18]), touchless tracking systems (DuploTrack [15] and Norilla [44]), and tag-based tracking systems (IDSense [29] and RapID [39]) to illustrate the dimensions.

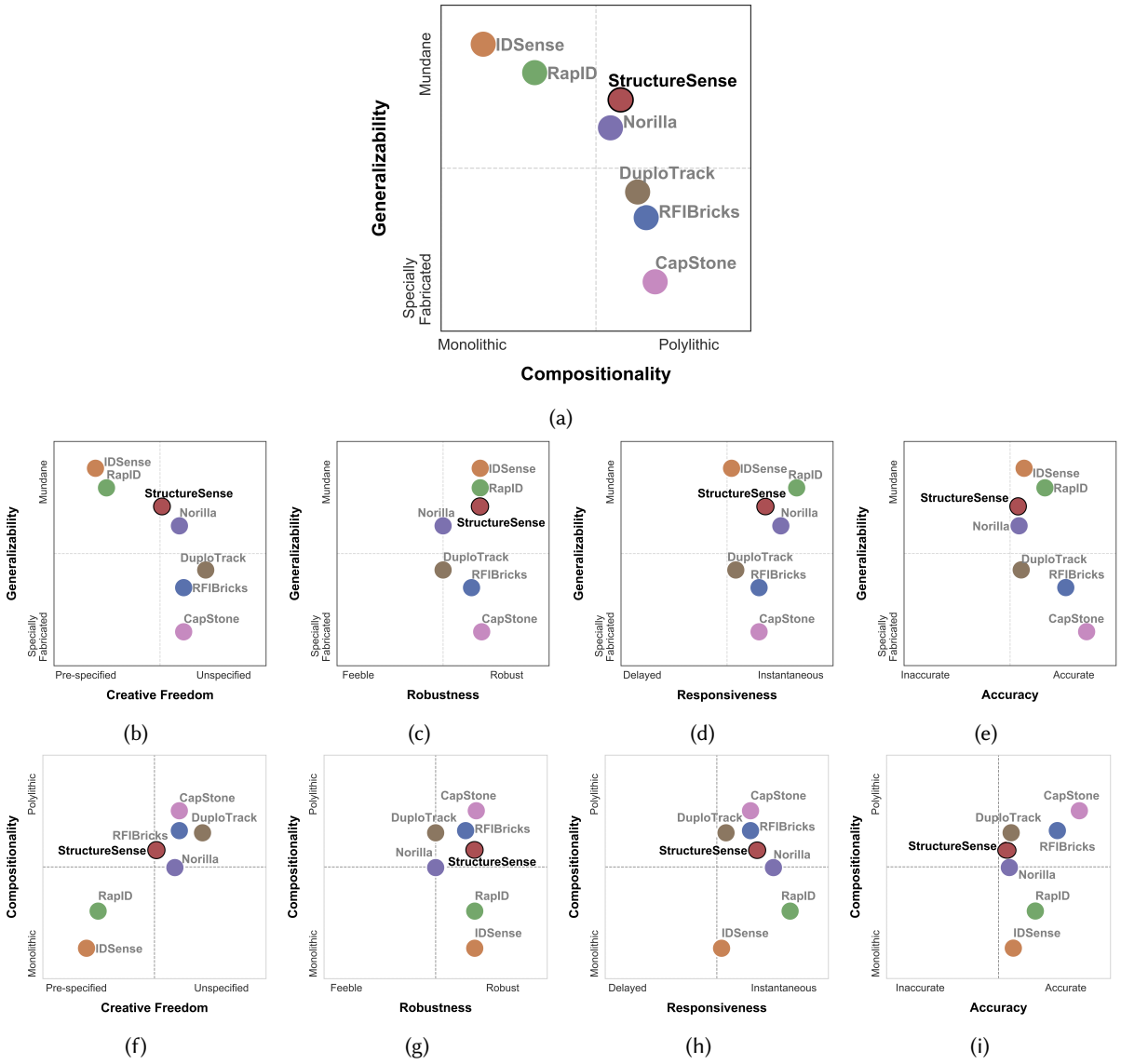



Fig. 14. Comparison of *StructureSense* with existing systems along the dimensions of our design space.

We decided on the placements of *StructureSense* and prior systems along all dimensions (both continuous and ordinal) based on the demonstrating applications and experiments reported in the prior work. In particular, we decided on the placement for the qualitative design dimensions by first converting them to ordinal values by mapping them to a Likert scale (similar to the method by Lambrichts et al [26]). For each dimension, the authors rated the systems on a 10-point Likert scale ranging from the extreme values as reported on the axes in Fig 14. For example, for the *Compositionality* scale, the values ranged from *Monolithic* (1) to *Polylithic* (10).

Our main design goal for *StructureSense* was to achieve a balance between generalizability and compositionality by enabling tracking and inference of highly polylithic structures for mundane constructive assemblies. Most existing assembly tracking systems would either prioritize generalizability or compositionality. For example, in the second quadrant, tracking systems based on specially fabricated assemblies [10, 18] are highly polylithic (i.e., they enable easy tracking of individual objects in a structure), but lack generalizability (Fig. 14a). The downside of prioritizing compositionality over generalizability with specially fabricated assemblies is the effort of setting up such systems for a different set of constructive assemblies is infeasible.

On the other hand, in the fourth quadrant, existing tag-based tracking methods [28, 29, 39] have limited ability to track polylithic structures, but they are highly generalizable to tracking different kinds of objects (e.g., they can track any building object or a monolithic structure to which one can affix an RFID tag) (Fig. 14a). *StructureSense* conducts its structure inference based on tag-based movement detection, thus inheriting the high generalizability from RapID [39]. Statistically inferring structures from movement history enables our system to track polylithic structures. Therefore, we placed *StructureSense* closer to CV-based systems [15, 44] in approximately the first quadrant, which has a higher balance between generalizability and compositionality than the rest of the systems.

Enabling high creative freedom is important for authoring applications (e.g., [18, 41, 44]), but systems that can only track pre-defined structures are still valuable for interactive instruction applications (e.g., [24]). Existing tracking methods based on specially fabricated assemblies and touchless tracking methods tend to handle unspecified polylithic structures better than existing tag-based methods, which specialize in tracking mundane monolithic objects and structures (Fig. 14f). Specifically fabricated assemblies such as RFIBricks [18] and CapStone [10] can have versatile connectors that place few limitations on what kinds of structures they can be combined into.

On the other hand, touchless tracking methods such as [15] are mostly decoupled from the objects they track and the kinds of structures those objects can form, leading to high creative freedom. We demonstrated that StructureSense can track the creation of unspecified structures with the authoring application given user disambiguation. However, we also showed that StructureSense may suffer from a delayed response time and less satisfactory accuracy as the unspecified structure it tracks gets larger. Therefore, the creative freedom for *StructureSense* is higher than the systems that use tag-based methods [29, 39], but lower than the ones with specially fabricated assemblies and touchless methods.

Unlike touchless tracking methods, both specifically fabricated assemblies and tag-based tracking methods are highly robust (Fig. 14c & Fig. 14g). Close coupling of objects and tracking system enables high robustness of both specifically fabricated assemblies and tag-based tracking systems, but at the cost of generalizability (Fig. 14c) and compositionality (Fig. 14g) respectively. Most touchless tracking methods are highly impacted by environmental factors—they must have a clear line-of-sight to the structure they are tracking, struggle to track objects that move too fast, and tend to have higher restrictions on minimum object size relative to other methods. Due to being based on tag-based methods [29, 39], *StructureSense* has shown robustness to various environmental factors (e.g., occlusion, clutter); thus, making *StructureSense* one of the more robust assembly tracking systems.

Most existing tracking systems are responsive enough to allow the user to keep their attention on the task, with RapID [39] approaching the instantaneous time limit of 0.1 seconds [37]. Naturally, the responsiveness of all systems deteriorates with the increase in the number of objects they need to track. When the responsiveness drops below 1 second (the limit required for the user to have an uninterrupted flow [37]), the effects of delayed responses could be mitigated by providing feedback about the system progress (e.g., using a percent-done indicator [35]). Our system benchmarking and user evaluation showed that our guidance application has a high response time (which further overlapped with the user's installation time) leading to a practically seamless user experience. However, our authoring application showed that *StructureSense* has a delayed response rate facing more complicated structures. Due to such a dichotomy, we mapped *StructureSense* based on its responsiveness in the guided application (which was similar to that of the evaluation of DuploTrack [15]), with an acknowledgment that its poor performance in the authoring application would place *StructureSense* as the slowest response system.

Most of the existing systems have high accuracy. However, their high accuracy comes at the expense of either generalizability (Fig. 14e) or compositionality (Fig. 14i). Specially fabricated assemblies are highly accurate because connections between individual objects are deterministic. The accuracy of touchless tracking systems could vary drastically and is affected by both the hardware (e.g., the resolution of cameras) and the environmental factors (see *Robustness*). Existing tag-based tracking systems have high accuracy, but this is primarily because they are limited to tracking monolithic structures or pre-specified structures made up of only a few individual objects. Although *StructureSense* demonstrated satisfactory accuracy while tracking guided assembly tasks with a pre-specified goal, we acknowledge that its accuracy has plenty of room for improvement when tracking unspecified structures.

## 8 DISCUSSION

We have shown that our *StructureSense* design and implementation was able to track and infer polylithic structures made from mundane constructive assemblies. Our evaluation showed that *StructureSense* offers a robust solution for tracking mundane constructive assembly instrumented only using adhesive RFID tags. Virtual specification of properties of constructive assemblies enabled *StructureSense* to track highly polylithic structures. Thus, our results show that *StructureSense* enables increased compositionality of tag-based tracking systems.

Two main contributing factors in enabling the tracking of polylithic structures using a tag-based tracking system were our virtual specification of constructive assembly properties (information that would otherwise require specially fabricated assemblies) and our Bayesian-based structures sampling method (which replaced the need to localize individual objects using CV). However, our approach still requires virtual specifications for different constructive assembly sets. Some of this effort could be alleviated by creating libraries of common building object definitions that span multiple constructive sets (e.g. generalizing Jumbo Blocks to LEGO bricks).

We illustrated the generalizability of *StructureSense* on two arbitrarily different constructive sets (furniture parts and toy building blocks). *StructureSense* uses a principled sampling method that is not coupled with any particular constructive set which aids its generalizability. In our demonstration, we have shown examples of generalizable proposal functions, prior distribution, and likelihood. Our examples had good performance when constructing pre-defined structures. However, our naive proposal function and prior distribution resulted in high uncertainty of possible structures for our example with large structures space. Constructive set providers could supply specific proposals, prior distribution, and likelihood function implementations corresponding to their constructive sets, which could decrease the time to sample structures and increase inference accuracy.

We found that *StructureSense* was able to successfully track pre-specified structures with high responsiveness and accuracy. However, the performance of the system quickly deteriorated with an increase in structure space when authoring previously unspecified structures. While the performance of our system still precludes us from achieving the high creative freedom required for authoring structures in large structures spaces, our system is well suited for interactive instruction applications. Such guided interactive constructive assemblies instruction applications can help people to build quickly while avoiding building errors. Our system performance and usability evaluation of our example guided assembly application has shown that *StructureSense* is capable of detecting instances when the user makes an assembly mistake and aiding the user in recovering from the error.

Relying solely on the movement history of individual objects contributed to the generalizability and compositionality of our system. We also showed the feasibility of adding creative freedom to tag-based tracking systems. However, relying only on movement history when tracking creative assemblies with large structures spaces may not be sufficient. Obtaining more information even at the expense of responsiveness (e.g., using relative RFID object localization [42]) could greatly aid the accuracy of our system in large structures spaces. Such information could reduce the entropy of the posterior probability of possible structures and have a positive impact on the responsiveness of the system when tracking structures in large structures spaces.

## 9 CONCLUSION AND FUTURE WORK

In this work, we presented *StructureSense*, a tag-based system for tracking and inferring constructive assembly structures. Our system extends the ability of existing tag-based systems to track mundane constructive assembly objects to tracking polylithic constructive assembly structures. Our demonstration has shown that *StructureSense* could support applications for guided interactive constructive assembly instruction. We have also explored the applicability of our system to support authoring "design-by-demonstration" applications. Our lab experiments are the first necessary step to ensure that the system works before any real-world deployment.

Our findings inform several promising directions for future work. For example, future work should explore how adding additional information about constructive assemblies (e.g., object localization) could make *StructureSense* more applicable for authoring structures in large structures spaces. Our findings also call for an investigation of how a library of commonly used curated constructive sets containing virtual object definitions, proposal functions, and prior distributions could benefit the responsiveness and accuracy of a tag-based structures tracking system. Such future advances in tag-based structure tracking systems could enable the coupling of interactivity with everyday mundane physical objects to enable quick and accurate structure assembly and rapid TUI prototyping.

## ACKNOWLEDGMENTS

We thank the reviewers and the associate editors for their insightful comments that helped us improve this paper. We also thank Yang-Hsi Su from the University of Michigan for his help with the UHF-RFID hardware and API.

## REFERENCES

[1] David Anderson, James L. Frankel, Joe Marks, Aseem Agarwala, Paul Beardsley, Jessica Hodgins, Darren Leigh, Kathy Ryall, Eddie Sullivan, and Jonathan S. Yedidia. 2000. Tangible Interaction + Graphical Interpretation: A New Approach to 3D Modeling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 393–402. https://doi.org/10.1145/344779.344960

[2] David Anderson, James L. Frankel, Joe Marks, Darren Leigh, Eddie Sullivan, Jonathan Yedidia, and Kathy Ryall. 1999. Building Virtual Structures with Physical Blocks. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology* (Asheville, North Carolina, USA) *(UIST '99)*. Association for Computing Machinery, New York, NY, USA, 71–72. https://doi.org/10.1145/320719.322587

[3] Masahiro Ando, Yuichi Itoh, Toshiki Hosoi, Kazuki Takashima, Kosuke Nakajima, and Yoshifumi Kitamura. 2014. StackBlock: Block-Shaped Interface for Flexible Stacking. In *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST'14 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 41–42. https://doi.org/10.1145/2658779.2659104

[4] Patrick Baudisch, Torsten Becker, and Frederik Rudeck. 2010. Lumino: Tangible blocks for tabletop computers based on glass fiber bundles, Vol. 2. 1165–1174. https://doi.org/10.1145/1753326.1753500

[5] James O Berger. 1985. *Statistical Decision Theory and Bayesian Analysis*. Springer New York.

[6] Jose Bernardo and Adrian Smith. 2000. *Bayesian Theory*. Vol. 15. https://doi.org/10.2307/2983298

[7] Kévin Bouchard, Dany Fortin-Simard, Sebastien Gaboury, Bruno Bouchard, and Abdenour Bouzouane. 2014. Accurate Trilateration for Passive RFID Localization in Smart Homes. *International Journal of Wireless Information Networks* 21 (03 2014), 1–18. https://doi.org/10.1007/s10776-013-0234-4

[8] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. 2011. *Handbook of Markov Chain Monte Carlo*. CRC press.

[9] Michael Buettner, Richa Prasad, Matthai Philipose, and David Wetherall. 2009. Recognizing Daily Activities with RFID-Based Sensors. In *Proceedings of the 11th International Conference on Ubiquitous Computing* (Orlando, Florida, USA) *(UbiComp '09)*. Association for Computing Machinery, New York, NY, USA, 51–60. https://doi.org/10.1145/1620545.1620553

[10] Liwei Chan, Stefanie Mueller, Anne Roudaut, and Patrick Baudisch. 2012. CapStones and ZebraWidgets: Sensing Stacks of Building Blocks, Dials and Sliders on Capacitive Touch Screens. https://doi.org/10.1145/2207676.2208371

[11] Kirti Chawla and Gabriel Robins. 2011. An RFID-based object localisation framework. *IJRFITA* 3 (2011), 2–30.

[12] Markus Funk, Thomas Kosch, and Albrecht Schmidt. 2016. Interactive Worker Assistance: Comparing the Effects of in-Situ Projection, Head-Mounted Displays, Tablet, and Paper Instructions. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. Association for Computing Machinery, New York, NY, USA, 934–939. https://doi.org/10.1145/2971648.2971706

[13] W.R. Gilks, S. Richardson, and D. Spiegelhalter. 1995. *Markov Chain Monte Carlo in Practice.* Taylor & Francis.
[14] Matthew Gorbet, Maggie Orth, and Hiroshi Ishii. 1998. Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography. 49–56.
[15] Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. 2012. DuploTrack: A real-time system for authoring and guiding duplo block assembly. *UIST'12 - Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, 389–402. https://doi.org/10.1145/2380116.2380167
[16] Toshiki Hosoi, Kazuki Takashima, Tomoaki Adachi, Yuichi Itoh, and Yoshifumi Kitamura. 2014. A-Blocks: Recognizing and Assessing Child Building Processes during Play with Toy Blocks. In *SIGGRAPH Asia 2014 Emerging Technologies* (Shenzhen, China) *(SA '14)*. Association for Computing Machinery, New York, NY, USA, Article 1, 2 pages. https://doi.org/10.1145/2669047.2669061
[17] Meng-Ju Hsieh, Rong-Hao Liang, Jr-Ling Guo, and Bing-Yu Chen. 2018. RFIDesk: an interactive surface for multi-touch and rich-ID stackable tangible interactions. In *SIGGRAPH Asia 2018 Emerging Technologies* (Tokyo, Japan) *(SA '18, Article 11)*. Association for Computing Machinery, New York, NY, USA, 1–2.
[18] Meng-Ju Hsieh, Rong-Hao Liang, Da-Yuan Huang, Jheng-You Ke, and Bing-Yu Chen. 2018. RFIBricks: Interactive Building Blocks Based on RFID. 1–10. https://doi.org/10.1145/3173574.3173763
[19] Koshi Ikegawa, Masaya Tsuruta, Tetsuya Abe, Arika Yoshida, Buntarou Shizuki, and Shin Takahashi. 2016. Lightweight Capacitance-based Block System for 3D Space Interaction. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces* (Niagara Falls, Ontario, Canada) *(ISS '16)*. Association for Computing Machinery, New York, NY, USA, 307–312.
[20] Hiroshi Ishii and Brygg Ullmer. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '97)*. Association for Computing Machinery, New York, NY, USA, 234–241. https://doi.org/10.1145/258549.258715
[21] C. Jiang, Y. He, X. Zheng, and Y. Liu. 2018. Orientation-Aware RFID Tracking with Centimeter-Level Accuracy. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 290–301. https://doi.org/10.1109/IPSN.2018.00057
[22] Ricardo Jota and Hrvoje Benko. 2011. Constructing Virtual 3D Models with Physical Building Blocks. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI EA '11)*. Association for Computing Machinery, New York, NY, USA, 2173–2178. https://doi.org/10.1145/1979742.1979915
[23] Mandeep Kaur, Manjeet Sandhu, Neeraj Mohan, and Parvinder Sandhu. 2011. RFID Technology Principles, Advantages,Limitations and Its Applications. *International Journal of Computer and Electrical Engineering* 3 (01 2011), 151–157. https://doi.org/10.7763/IJCEE.2011.V3.306
[24] Bui Minh Khuong, Kiyoshi Kiyokawa, Andrew Miller, Joseph J. La Viola, Tomohiro Mashita, and Haruo Takemura. 2014. The effectiveness of an AR-based context-aware assembly support system in object assembly. In *2014 IEEE Virtual Reality (VR)*. 57–62. https://doi.org/10.1109/VR.2014.6802051
[25] Benjamin Lafreniere, Tovi Grossman, Fraser Anderson, Justin Matejka, Heather Kerrick, Danil Nagy, Lauren Vasey, Evan Atherton, Nicholas Beirne, Marcelo H. Coelho, Nicholas Cote, Steven Li, Andy Nogueira, Long Nguyen, Tobias Schwinn, James Stoddart, David Thomasson, Ray Wang, Thomas White, David Benjamin, Maurice Conti, Achim Menges, and George Fitzmaurice. 2016. Crowdsourced Fabrication. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 15–28. https://doi.org/10.1145/2984511.2984553
[26] Mannu Lambrichts, Raf Ramakers, Steve Hodges, Sven Coppers, and James Devine. 2021. A Survey and Taxonomy of Electronics Toolkits for Interactive and Ubiquitous Device Prototyping. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 2, Article 70 (jun 2021), 24 pages. https://doi.org/10.1145/3463523
[27] Danny Leen, Raf Ramakers, and Kris Luyten. 2017. StrutModeling: A Low-Fidelity Construction Kit to Iteratively Model, Test, and Adapt 3D Objects. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 471–479.
[28] Hanchuan Li, Eric Brockmeyer, Elizabeth J. Carter, Josh Fromm, Scott E. Hudson, Shwetak N. Patel, and Alanson Sample. 2016. PaperID: A Technique for Drawing Functional Battery-Free Wireless Interfaces on Paper. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 5885–5896. https://doi.org/10.1145/2858036.2858249
[29] Hanchuan Li, Can Ye, and Alanson P. Sample. 2015. IDSense: A Human Object Interaction Detection System Based on Passive UHF RFID. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 2555–2564. https://doi.org/10.1145/2702123.2702178
[30] Rong-Hao Liang, Liwei Chan, Hung-Yu Tseng, Han-Chih Kuo, Da-Yuan Huang, De-Nian Yang, and Bing-Yu Chen. 2014. GaussBricks. 587–590. https://doi.org/10.1145/2559206.2574776
[31] Haishu Ma and Kesheng Wang. 2017. Fusion of RSS and Phase Shift Using the Kalman Filter for RFID Tracking. *IEEE Sensors Journal* 17, 11 (2017), 3551–3558. https://doi.org/10.1109/JSEN.2017.2696054
[32] Timothy McNerney. 2004. From turtles to Tangible Programming Bricks: Explorations in physical language design. *Personal and Ubiquitous Computing* 8 (09 2004), 326–337. https://doi.org/10.1007/s00779-004-0295-6

[33] Andrew Miller, Brandyn White, Emiko Charbonneau, Zach Kanzler, and Joseph J LaViola, Jr. 2012. Interactive 3D model acquisition and tracking of building block structures. *IEEE Trans. Vis. Comput. Graph.* 18, 4 (April 2012), 651–659.

[34] Andrea Motroni, Paolo Nepa, Alice Buffi, and Bernardo Tellini. 2019. A Phase-Based Method for Mobile Node Localization through UHF-RFID Passive Tags. In *2019 IEEE International Conference on RFID Technology and Applications (RFID-TA).* 470–475. https://doi.org/10.1109/RFID-TA.2019.8892264

[35] Brad A. Myers. 1985. The Importance of Percent-Done Progress Indicators for Computer-Human Interfaces. *SIGCHI Bull.* 16, 4 (April 1985), 11–17. https://doi.org/10.1145/1165385.317459

[36] Jakob Nielsen. 1989. Usability Engineering at a Discount. In *Proceedings of the Third International Conference on Human-Computer Interaction on Designing and Using Human-Computer Interfaces and Knowledge Based Systems (2nd Ed.)* (Boston, Massachusetts, USA). Elsevier Science Inc., USA, 394–401.

[37] Jakob Nielsen. 1994. *Usability Engineering.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[38] Hayes Solos Raffle, Amanda J Parkes, and Hiroshi Ishii. 2004. Topobo: a constructive assembly system with kinetic memory. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04).* Association for Computing Machinery, New York, NY, USA, 647–654.

[39] Andrew Spielberg, Alanson Sample, Scott E. Hudson, Jennifer Mankoff, and James McCann. 2016. RapID: A Framework for Fabricating Low-Latency Interactive Objects with RFID Tags. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16).* Association for Computing Machinery, New York, NY, USA, 5897–5908. https://doi.org/10.1145/2858036.2858243

[40] Anna Syberfeldt, Oscar Danielsson, Magnus Holm, and Lihui Wang. 2015. Visual Assembling Guidance Using Augmented Reality. *Procedia Manufacturing* 1 (2015), 98–109. https://doi.org/10.1016/j.promfg.2015.09.068 43rd North American Manufacturing Research Conference, NAMRC 43, 8-12 June 2015, UNC Charlotte, North Carolina, United States.

[41] Jeppe U. Walther, J. Andreas Bærentzen, and Henrik Aanæs. 2016. Tangible 3D Modeling of Coherent and Themed Structures. *Comput. Graph.* 58, C (Aug. 2016), 53–65. https://doi.org/10.1016/j.cag.2016.05.004

[42] Jue Wang, Fadel Adib, Ross Knepper, Dina Katabi, and Daniela Rus. 2013. RF-Compass: Robot Object Manipulation Using RFIDs. In *Proceedings of the 19th Annual International Conference on Mobile Computing; Networking* (Miami, Florida, USA) *(MobiCom '13).* Association for Computing Machinery, New York, NY, USA, 3–14. https://doi.org/10.1145/2500423.2500451

[43] John Williamson, Antti Oulasvirta, Per Ola Kristensson, and Nikola Banovic (Eds.). 2022. *Bayesian Methods for Interaction and Design.* Cambridge University Press. https://doi.org/10.1017/9781108874830

[44] Nesra Yannier, Scott Hudson, and Kenneth Koedinger. 2020. Active Learning is About More Than Hands-On: A Mixed-Reality AI System to Support STEM Education. *International Journal of Artificial Intelligence in Education* (02 2020). https://doi.org/10.1007/s40593-020-00194-3

[45] Hui-Shyong Yeo, Ryosuke Minami, Kirill Rodriguez, George Shaker, and Aaron Quigley. 2018. Exploring Tangible Interactions with Radar Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 200 (dec 2018), 25 pages. https://doi.org/10.1145/3287078

[46] Hui-Shyong Yeo and Aaron Quigley. 2017. Radar Sensing in Human-Computer Interaction. *Interactions* 25, 1 (dec 2017), 70–73. https://doi.org/10.1145/3159651