frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 31 July 2020
doi: 10.3389/frobt.2020.00100

Symbolic Learning and Reasoning With Noisy Data for Probabilistic Anchoring

Pedro Zuidberg Dos Martires[1]*, *Nitesh Kumar*[1], *Andreas Persson*[2]*, *Amy Loutfi*[2] and *Luc De Raedt*[1,2]

[1] Declaratieve Talen en Artificiele Intelligentie (DTAI), Department of Computer Science, KU Leuven, Leuven, Belgium, [2] Center for Applied Autonomous Sensor Systems (AASS), Department of Science and Technology, Örebro University, Örebro, Sweden

Robotic agents should be able to learn from sub-symbolic sensor data and, at the same time, be able to reason about objects and communicate with humans on a symbolic level. This raises the question of how to overcome the gap between symbolic and sub-symbolic artificial intelligence. We propose a semantic world modeling approach based on bottom-up object anchoring using an object-centered representation of the world. Perceptual anchoring processes continuous perceptual sensor data and maintains a correspondence to a symbolic representation. We extend the definitions of anchoring to handle multi-modal probability distributions and we couple the resulting symbol anchoring system to a probabilistic logic reasoner for performing inference. Furthermore, we use statistical relational learning to enable the anchoring framework to learn symbolic knowledge in the form of a set of probabilistic logic rules of the world from noisy and sub-symbolic sensor input. The resulting framework, which combines perceptual anchoring and statistical relational learning, is able to maintain a semantic world model of all the objects that have been perceived over time, while still exploiting the expressiveness of logical rules to reason about the state of objects which are not directly observed through sensory input data. To validate our approach we demonstrate, on the one hand, the ability of our system to perform probabilistic reasoning over multi-modal probability distributions, and on the other hand, the learning of probabilistic logical rules from anchored objects produced by perceptual observations. The learned logical rules are, subsequently, used to assess our proposed probabilistic anchoring procedure. We demonstrate our system in a setting involving object interactions where object occlusions arise and where probabilistic inference is needed to correctly anchor objects.

Keywords: semantic world modeling, perceptual anchoring, probabilistic anchoring, statistical relational learning, probabilistic logic programming, object tracking, relational particle filtering, probabilistic rule learning

OPEN ACCESS

Edited by:
Fabrizio Riguzzi,
University of Ferrara, Italy

Reviewed by:
Nicos Angelopoulos,
University of Essex, United Kingdom
Riccardo Zese,
University of Ferrara, Italy

***Correspondence:**
Pedro Zuidberg Dos Martires
pedro.zudo@kuleuven.be
Andreas Persson
andreas.persson@oru.se

Specialty section:
This article was submitted to Computational Intelligence in Robotics, a section of the journal Frontiers in Robotics and AI

Received: *15 June 2019*
Accepted: *24 June 2020*
Published: *31 July 2020*

Citation:
Zuidberg Dos Martires P, Kumar N, Persson A, Loutfi A and De Raedt L (2020) Symbolic Learning and Reasoning With Noisy Data for Probabilistic Anchoring. Front. Robot. AI 7:100. doi: 10.3389/frobt.2020.00100

1. INTRODUCTION

Statistical Relational Learning (SRL) (Getoor and Taskar, 2007; De Raedt et al., 2016) tightly integrates predicate logic with graphical models in order to extend the expressive power of graphical models toward relational logic and to obtain probabilistic logics than can deal with uncertainty. After two decades of research, a plethora of expressive probabilistic logic reasoning languages and

systems exists (e.g., Sato and Kameya, 2001; Richardson and Domingos, 2006; Getoor, 2013; Fierens et al., 2015). One obstacle that still lies ahead in the field of SRL (but see Gardner et al., 2014; Beltagy et al., 2016), is to combine symbolic reasoning and learning, on the one hand, with sub-symbolic data and perception, on the other hand. The question is how to create a symbolic representation of the world from sensor data in order to reason and ultimately plan in an environment riddled with uncertainty and noise. In this paper, we will take a probabilistic logic approach to study this problem in the context of perceptual anchoring.

An alternative to using SRL or probabilistic logics would be to resort to deep learning. Deep learning is based on *end-to-end learning* (e.g., Silver et al., 2016). Although exhibiting impressive results, deep neural networks do suffer from certain drawbacks. As opposed to probabilistic rules, it is, for example, not straightforward to include prior (symbolic) knowledge in a neural system. Moreover, it is also often difficult to give guarantees for the behavior of neural systems, cf. the debate around safety and explainability in AI (Huang et al., 2017; Gilpin et al., 2018). Although not free from this latter shortcoming, this is less of a concern for symbolic systems, which implies that bridging the symbolic/sub-symbolic gap is therefore paramount. A notion that aims to bridge the symbolic/sub-symbolic gap is the definition of *perceptual anchoring*, as introduced by Coradeschi and Saffiotti (2000, 2001). Perceptual anchoring tackles the problem of creating and maintaining, in time and space, the correspondence between symbols and sensor data that refer to the same physical object in the external world (a detailed overview of perceptual anchoring is given in section 2.1). In this paper, we particularly emphasize sensor-driven *bottom-up anchoring* (Loutfi et al., 2005), whereby the anchoring process is triggered by the sensory input data.

A further complication in robotics, and perceptual anchoring more specifically, is the inherent dependency on time. This means that a probabilistic reasoning system should incorporate the concept of time natively. One such system, rooted in the SRL community, is the probabilistic logic programming language *Dynamic Distributional Clauses* (DDC) (Nitti et al., 2016b), which can perform probabilistic inference over logic symbols and over time. In our previous work, we coupled the probabilistic logic programming language DDC to a perceptual anchoring system (Persson et al., 2020b), which endowed the perceptual anchoring system with probabilistic reasoning capabilities. A major challenge in combining perceptual anchoring with a high-level probabilistic reasoner, and which is still an open research question, is the administration of *multi-modal* probability distributions in anchoring[1]. In this paper, we extend the anchoring notation in order to handle additionally multi-modal probability distributions. A second point that we have not addressed in Persson et al. (2020b), is the learning of probabilistic rules that are used to perform probabilistic logic reasoning. We show that, instead of hand-coding these probabilistic rules, we can adapt existing methods present in the body of literature of SRL to learn them from raw sensor data. In other words, instead of providing a model of the world to a robotic agent, it learns this model in form of probabilistic logical rules. These rules are then used by the robotic agent to reason about the world around it, i.e., perform inference.

In Persson et al. (2020b), we showed that enabling a perceptual anchoring system to reason further allows for correctly anchoring objects under object occlusions. We borrowed the idea of encoding a *theory of occlusion* as a probabilistic logic theory from Nitti et al. (2014) (discussed in more detail in section 2.3). While Nitti et al. operated in a strongly simplified setting, by identifying objects with AR tags, we used a perceptual anchoring system instead—identifying objects from raw RGB-D sensor data. In contrast to the approach presented here, the theory of occlusion was not learned but hand-coded in these previous works and did not take into account the possibility of multi-modal probability distributions. We evaluate the extensions of perceptual anchoring, proposed in this paper, on three showcase examples, which exhibit exactly this behavior: (1) we perform probabilistic perceptual anchoring when object occlusion induces a multi-modal probability distributions, and (2) we perform probabilistic perceptual anchoring with a learned theory of occlusion.

We structure the remainder of the paper as follows. In section 2, we introduce the preliminaries of this work by presenting the background and motivation of used techniques. Subsequently, we discuss, in section 3, our first contribution by first giving a more detailed overview of our prior work (Persson et al., 2020b), followed by introducing a probabilistic perceptual anchoring approach in order to enable anchoring in a multi-modal probabilistic state-space. We continue, in section 4, by explaining how probabilistic logical rules are learned. In section 5, we evaluate both our contributions on representative scenarios before closing this paper with conclusions, presented in section 6.

2. PRELIMINARIES

2.1. Perceptual Anchoring

Perceptual anchoring, originally introduced by Coradeschi and Saffiotti (2000, 2001), addresses a subset of the symbol grounding problem in robotics and intelligent systems. The notion of perceptual anchoring has been extended and refined since its first definition. Some notable refinements include the integration of *conceptual spaces* (Chella et al., 2003, 2004), the addition of *bottom-up anchoring* (Loutfi et al., 2005), extensions for *multi-agent systems* (LeBlanc and Saffiotti, 2008), considerations for non-traditional sensing modalities and *knowledge based anchoring* given full scale knowledge representation and reasoning systems (Loutfi, 2006; Loutfi and Coradeschi, 2006; Loutfi et al., 2008), and *perception and probabilistic anchoring* (Blodow et al., 2010). All these approaches to perceptual anchoring share, however, a number

[1]A multi-modal probability distribution is a continuous probability distribution with strictly more than one local maximum. The key difference to a uni-modal probability distribution, such as a simple normal distribution, is that summary statistics do not adequately mirror the actual distribution. In perceptual anchoring these multi-modal distributions do occur, especially in the presence of object occlusions, and handling them appropriately is critical for correctly anchoring objects. This kind of phenomenon is well known when doing filtering and is the reason why particle filters can be preferred over Kalman filters.

of common ingredients from Coradeschi and Saffiotti (2000, 2001), including:

- A *symbolic system* (including: a set $\mathcal{X} = \{x_1, x_2, \dots\}$ of *individual symbols*; a set $\mathcal{P} = \{p_1, p_2, \dots\}$ of *predicate symbols*).
- A *perceptual system* [including: a set $\Pi = \{\pi_1, \pi_2, \dots\}$ of *percepts*; a set $\Phi = \{\phi_1, \phi_2, \dots\}$ of *attributes* with values in the domain $D(\phi_i)$].
- *Predicate grounding relations* $g \subseteq \mathcal{P} \times \Phi \times D(\Phi)$ that encode the correspondence between unary predicates and values of measurable attributes (i.e., the relation g maps a certain predicate to compatible attribute values).

While the traditional definition of Coradeschi and Saffiotti (2000, 2001) assumed *unary* encoded perceptual-symbol correspondences, this does not support the maintenance of anchors with different attribute values at different times. To address this problem, Persson et al. (2017) distinguishes two different types of attributes:

- *Static attributes* ϕ, which are unary within the anchor according to the traditional definition.
- *Volatile attributes* ϕ_t, which are individually indexed by time t, which are maintained in a set of attribute instances φ, such that $\phi_t \in \varphi$.

Without loss of generality, we assume from here on that all *attributes stored in an anchor* are volatile, i.e., that they are indexed by a time step t. Static attributes are trivially converted to volatile attributes by giving them the same attribute value in each time step.

Given the components above, an *anchor* is an internal data structure α_t^x, indexed by time t and identified by a unique individual symbol x (e.g., `mug-1` and `apple-2`), which encapsulates and maintains the correspondences between percepts and symbols that refer to the *same physical object*, as depicted in **Figure 1**. Following the definition presented by Loutfi et al. (2005), the principal functionalities to create and maintain anchors in a *bottom-up fashion*, i.e., functionalities triggered by a perceptual event, are:

- *Acquire*—initiates a new anchor whenever a candidate object is received that does not match any existing anchor α_t^x. This functionality defines a structure α_t^x, indexed by time t and identified by a unique identifier x, which encapsulates and stores all perceptual and symbolic data of the candidate object.
- *Re-acquire*—extends the definition of a matching anchor α_t^x from time $t - k$ to time t. This functionality ensures that the percepts pointed to by the anchor are the most recent and adequate perceptual representation of the object.

Based on the functionalities above, it is evident that an *anchoring matching function* is essential to decide whether a candidate object matches an existing anchor or not. Different approaches in perceptual anchoring vary, in particular in how the matching function is specified. For example, in Persson et al. (2020b), we have shown that the anchoring matching function can be approximated by a learned model trained with manually labeled samples collected through an *annotation interface* (through which the human user can interfere with the anchoring process and provide feedback about which objects in the scene match previously existing anchors).

In another recently published work on anchoring, Ruiz-Sarmiento et al. (2017) focus on spatial features and distinguish *unary* object features, e.g., the position of an object, from *pairwise* object features, e.g., the distance between two objects, in order to build a graph-based world model that can be exploited by a *probabilistic graphical model* (Koller and Friedman, 2009) in order to leverage contextual relations between objects to support 3-*D* object recognition. In parallel with our previous work on anchoring, Günther et al. (2018) have further exploited this graph-based model on spatial features and propose, in addition, to learn the matching function through the use of a *Support Vector Machine* (trained on samples of object pairs manually labeled as *"same or different object"*), in order to approximate the *similarity* between two objects. The assignment of candidate objects to existing anchors is, subsequently, calculated using prior similarity values and a *Hungarian* method (Kuhn, 1955). However, in contrast to Günther et al. (2018), the matching function introduced in Persson et al. (2020b) do not only rely upon spatial features (or attributes), but can also take into consideration visual features (such as color features), as well as semantic object categories, in order to approximate the anchoring matching problem.

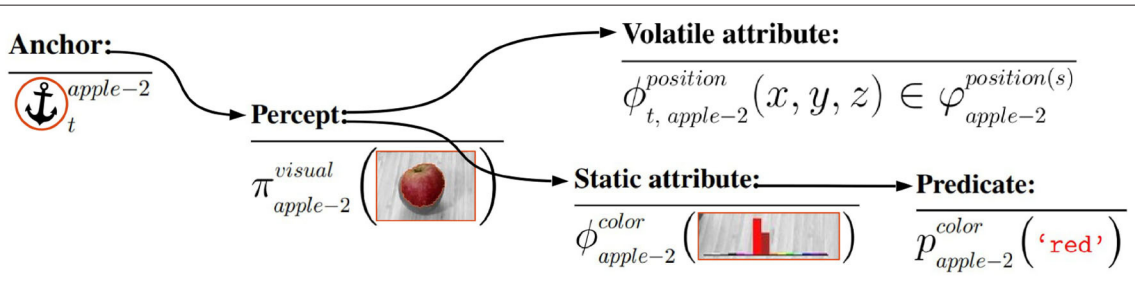

FIGURE 1 | A conceptual illustration of the internal data structure that constitutes a single anchor, and which is first initiated by a percept π from a raw image. The volatile and static attributes are derived from this percept, while predicates such as red, are derived from static attributes (which are not indexed by time), e.g., the static color histogram attribute.

2.2. Dynamic Distributional Clauses

Dynamic Distributional Clauses (DDC) (Nitti et al., 2016b) provide a framework for probabilistic programming that extends the logic programming language *Prolog* (Sterling and Shapiro, 1994) to the probabilistic domain. A comprehensive treatise on the field of probabilistic logic programming can be found in De Raedt and Kimmig (2015) and Riguzzi (2018). DDC is capable of representing discrete and continuous random variables and to perform probabilistic inference. Moreover, DDC explicitly models time, which makes it predestined to model dynamic systems. The underpinning concepts of DDC are related to ideas presented in Milch et al. (2007) but embedded in logic programming. Related ideas of combining discrete time steps, Bayesian learning and logic programming are also presented in Angelopoulos and Cussens (2008, 2017).

An *atom* $p(t_1, \ldots, t_n)$ consists of a predicate p/n of arity n and terms $t_1, \ldots, t_n$. A *term* is either a constant (written in lowercase), a variable (in uppercase), or a function symbol. A *literal* is an atom or its negation. Atoms which are negated are called *negative atoms* and atoms which are not negated are called *positive atoms*.

A distributional clause is of the form $\mathsf{h} \sim \mathcal{D} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n$, where $\sim$ is a predicate in infix notation and b_i's are literals, i.e., atoms or their negation. h is a term representing a random variable and $\mathcal{D}$ tells us how the random variable is distributed. The meaning of such a clause is that each grounded instance of a clause $(\mathsf{h} \sim \mathcal{D} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n)\theta$ defines a random variable $\mathsf{h}\theta$ that is distributed according to $\mathcal{D}\theta$ whenever all literals $b_i\theta$ are true. A grounding substitution $\theta = \{\mathsf{V}_1/\mathsf{t}_1, \ldots, \mathsf{V}_n/\mathsf{t}_n\}$ is a transformation that simultaneously substitutes all logical variables V_i in a distributional clause with non-variable terms t_i. DDC can be viewed as a language that defines conditional probabilities for discrete and continuous random variables: $p(\mathsf{h}\theta|\mathsf{b}_1\theta, \ldots, \mathsf{b}_n\theta) = \mathcal{D}\theta$.

Example 1: Consider the following DDC program:

```
n~poisson(6).
pos(P):0~uniform(0,100) ← n~=N, between(1,N,P).
pos(P):t+1~gaussian(X+3, Σ) ← pos(P):t~=X.
left(O1,O2):t~finite([0.99:true, 0.01:false]) ←
    pos(O1):t~=P1, pos(O2):t~=P2, P1<P2.
```

The first rule states that the number of objects n in the world is distributed according to a Poisson distribution with mean 6. The second rule states that the position of the n objects, which are identified by a number P between 1 and n, are distributed according to a uniform distribution between 0 and 100. Here, the notation `n~=N` means that the logical variable N takes the value of our random variable n. The label 0 (resp. t) in the program denotes the point in time. So, `pos(P):0` denotes the position of object P at time 0. Next, the program describes how the position evolves over time: at each time step the object moves three units of length, giving it a velocity of 3 [*length*]/[*time*]. Finally, the example program defines the `left` predicate, through which a relationship between each object is introduced at each time step. DDC then allows for querying this program through its built in predicate:

```
query((left(1,2):t~=true, pos(1):t>0), Probability)
```

`Probability` in the second argument unifies with the probability of object `1` being to the left of object `2` and having a positive coordinate position.

A DDC program $\mathbb{P}$ is a set of distributional and/or definite clauses (as in Prolog). A DDC program defines a probability distribution $p(x)$ over possible worlds x.

Example 2: One possible world of the uncountably many possible worlds encoded by the program in Example 1. The sampled number `n` determines that 2 objects exist, for which the ensuing distributional clauses then generate a position and the `left/2` relationship:

```
n~= 2.
pos(1):t~= 30.5.
pos(2):t~= 63.2.
pos(1):t+1~= 32.4.
pos(2):t+1~= 58.8.
left(1,2):t~= true.
left(2,1):t~= false.
```

When performing inference within a specific time step, DDC deploys importance sampling combined with backward reasoning (SLD-resolution), likelihood weighting and Rao-Blackwellization (Nitti et al., 2016a). Inferring probabilities in the next time given the previous time step is achieved through particle filtering (Nitti et al., 2013). If the DDC program does not contain any predicates labeled with a time index the program represents a *Distributional Clauses* (DC) (Gutmann et al., 2011) program, where filtering over time steps is not necessary.

2.3. Occlusions

Object occlusion is a challenging problem in visual tracking and a plethora of different approaches exist that tackle different kinds of occlusions; a thorough review of the field is given in Meshgi and Ishii (2015). The authors use three different attributes of an occlusion to categorize it: the *extent* (partial or full occlusion), the *duration* (short or long), and the *complexity* (simple or complex)[2]. Another classification of occlusions separates occlusions into *dynamic occlusions*, where objects in the foreground occlude each other, and *scene occlusions*, where objects in the background model are located closer to the camera and occlude target objects by being moved between the camera and the target objects[3].

Meshgi and Ishhii report that the majority of research on occlusions in visual tracking has been done on partial, temporal and simple occlusions. Furthermore, they report that none of the approaches examined in the comparative studies of Smeulders et al. (2013) and Wu et al. (2013), handles either partial complex occlusions or full long complex occlusions. To the best of our knowledge, our previous paper on combining bottom-up anchoring and probabilistic reasoning, constitutes the first tracker that is capable of handling occlusions that are full, long and complex (Persson et al., 2020b). This was achieved

[2] An occlusion of an object is deemed complex if during the occlusion the occluded object considerably changes one of its key characteristics, e.g., position, color, size). An occlusion is simple if it is not complex.

[3] Further categories exist, we refer the reader to Vezzani et al. (2011) and Meshgi and Ishii (2015).

by declaring a *theory of occlusion* (ToO) expressed as dynamic distributional clauses.

Example 3: An excerpt from the set of clauses that constitute the ToO. The example clause describes the conditions under which an object is considered a potential `Occluder` of an other object `Occluded`.

```
occluder(Occluded,Occluder):t+1~finite(1.0:true) ←
    observed(Occluded):t,
    \+observed(Occluded):t+1,
    position(Occluded):t~= (X,Y,Z),
    position(Occluder):t+1~= (XH,YH,ZH),
    D is sqrt((X-XH)^2+(Y-YH)^2), Z<ZH, D<0.3.
```

Out of all the potential `Occluder`'s the actual occluding object is then sampled uniformly:

```
occluded_by(Occluded,Occluder):t+1 ←
    sample_occluder(Occluded):t+1~= Occluder.
sample_occluder(Occluded):t+1~uniform
    (ListOfOccluders) ← findall(O,occluder
    (Occluded, O):t+1, ListOfOccluders).
```

Declaring a theory of occlusion and coupling it to the anchoring system allows the anchoring system to perform *occlusion reasoning* and to track objects not by directly observing them but by reasoning about relationships that occluded objects have entered with visible (anchored) objects. The idea of declaring a theory of occlusion first appeared in Nitti et al. (2013), where, however, the data association problem was assumed to be solved by using AR tags.

As the anchoring system was not able to handle probabilistic states in our previous work, the theory of occlusion had to describe unimodal probability distributions. In this paper, we repair this deficiency (cf. section 3.2). Moreover, the theory of occlusion had to be hand-coded (also the case for Nitti et al., 2013). We replace the hand-coded theory of occlusion by a learned one (cf. section 4).

Considering our previous work from the anchoring perspective, our approach is most related to the techniques proposed in Elfring et al. (2013), who introduced the idea of *probabilistic multiple hypothesis anchoring* in order to match and maintain probabilistic tracks of anchored objects, and thus, maintain an adaptable *semantic world model*. From the perspective of how occlusions are handled, Elfring. et al's and our work differs, however, substantially. Elfring et al. handle occlusions that are due to *scene occlusion*. Moreover, the occlusions are handled by means of a multiple hypothesis tracker, which is suited for short occlusions rather then long occlusions. The limitations with the use of multiple hypothesis tracking for world modeling, and consequently also for handling object occlusions in anchoring scenarios (as in Elfring et al., 2013), have likewise been pointed out in a publication by Wong et al. (2015). Wong et al. reported instead the use of a clustering-based *data association* approach (opposed to a tracking-based approach), in order to aggregate a consistent semantic world model from multiple viewpoints, and hence, compensate for partial occlusions from a single viewpoint perspective of the scene.

3. ANCHORING OF OBJECTS IN MULTI-MODAL STATES

In this section, we present a *probabilistic anchoring framework* based on our previous work on conjoining probabilistic reasoning and object anchoring (Persson et al., 2020b). An overview of our proposed framework, which is implemented utilizing the libraries and communication protocols available in the Robot Operating System (ROS)[4], can be seen in **Figure 2**. However, our prior *anchoring system*, seen in **Figure 2**–(2.), was unable to handle probabilistic states of objects. While the *probabilistic reasoning module*, seen in **Figure 2**–(3.), was able to model the position of an object as a probability distribution over possible positions, the anchoring system only kept track of a single deterministic position: the expected position of an object. Therefore, we extend the anchoring notation toward a probabilistic anchoring approach in order to enable the anchoring system to handle multi-modal probability distributions.

3.1. Requirements for Anchoring and Semantic Object Tracking

Before presenting our proposed probabilistic anchoring approach, we first introduce the necessary requirements and assumptions (which partly originate in our previous work, Persson et al., 2020b):

1. We assume that unknown anchor representations, α_t^y, are supplied by a *black-box* perceptual processing pipeline, as exemplified in **Figure 2**–(1.). They consist of extracted *attribute measurements* and corresponding grounded *predicate symbols*. We further assume that for each perceptual representation of an object, we have the following attribute measurements: (1) a *color attribute* (ϕ_y^{color}), (2) a *position attribute* (ϕ_y^{pos}), and (3) a *size attribute* (ϕ_y^{size}).

   Example 4: In this paper we use the combined Depth Seeding Network (DSN) and Region Refinement Network (RNN), as presented by Xie et al. (2019), for the purpose of segmenting arbitrary object instances in tabletop scenarios. This two-stage approach leverages both RGB and depth data (given by a Kinect V2 RGB-D sensor), in order to first segment rough initial object masks (based on depth data), followed by a second refinement stage of these object masks (based on RGB data). The resulting output for each segmented object, is then both a 3-*D spatial percept* ($\phi_y^{spatial}$), as well as a 2-*D visual percept* (ϕ_y^{visual}). For each segmented *spatial percept*, and with the use of the Point Cloud Library (PCL), are both a *position attribute* measured as the 3-*D* geometrical center, and a *size attribute* measured as the 3-*D* geometrical bounding box. Similarly, using the Open Computer Vision Library (OpenCV), a *color attribute* is measured as the *discretized color histogram* (in HSV color-space) for each segmented visual percept, as depicted in **Figure 3**.

[4]The code can be found online at: https://bitbucket.org/reground/anchoring

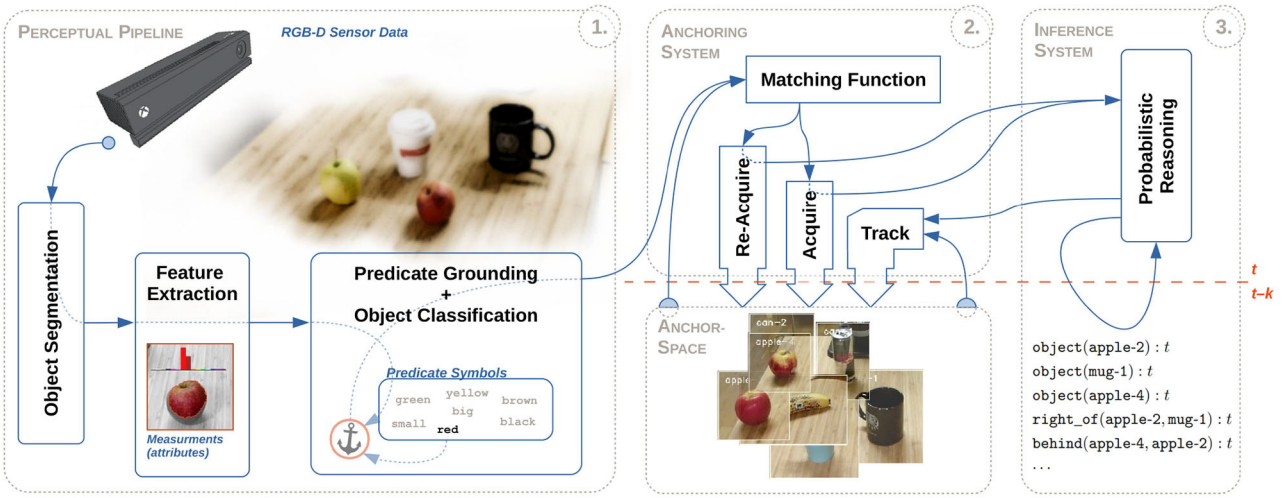

FIGURE 2 | The overall framework architecture is divided into three basic sub-systems (or modules): (1.) an initial *perceptual processing pipeline* for detecting, segmenting and processing perceived objects, (2.) an *anchoring system* for creating and maintaining updated and consistent representations (anchors) of perceived objects, and (3.) an *inference system* for aiding the anchoring system and logically tracking objects in complex dynamic scenes.

2. In order to semantically categorize objects, we further assume that a Convolutional Neural Network (CNN), such as the GoogLeNet model (Szegedy et al., 2015), is available. In the context of anchoring, the inputs for this model are segmented *visual percepts* (π_y^{visual}), while resulting object categories, denoted by the predicate $p_y^{category} \in \mathcal{P}$, are given together with the predicted probabilities $\phi_y^{category}$ (cf. section 2.1)[5].

   Example 5: For this work, we have used the same fine-tuned model as used in Persson et al. (2020b), which is based on the network architecture of the 1 K GoogLeNet model, developed by Szegedy et al. We have, however, fine-tuned the model to classify 101 objects categories that are only relevant for a household domain, e.g., `mug`, `ball`, `box`, etc., where the model was trained for a *top-1 accuracy* of 73.4% (and a *top-5 accuracy* of 92.0%). An example of segmented objects together with the *3-top best object categories*, given by the integrated GoogLeNet model, is illustrated in **Figure 4**.

   In addition, this integrated model is also used to enhance the traditional *acquire* functionality such that a unique identifier x is generated based on the object category symbol $p^{category}$. For example, if the anchoring system detects an object it has not seen before and classifies it as a `cup`, a corresponding unique identifier $x =$ `cup-4` could be generated (where the 4 means that this is the forth distinct instance of a `cup` object perceived by the system).
3. We require the presence of a *probabilistic inference system* coupled to the *anchoring system*, as illustrated in

[5]The anchoring framework is conceived in a modular fashion. This would for example allow us to replace the GoogLeNet-based classifier by more recent and memory efficient architecture such as residual neural networks (He et al., 2016). The modularity of the anchoring framework is presented in a separate demo paper (Persson et al., 2020a).

FIGURE 3 | Examples of measured *color attribute* (measured as the *discretized color histogram* over each segmented object).

Figure 2–(3.). The anchoring system is responsible for maintaining objects perceived by the sensory input data and for maintaining the observable part of the world model. Maintained anchored object representations are then treated as observations in the inference system, which uses relational object tracking to infer the state of occluded objects through their relations with perceived objects in the world. This inferred belief of the world is then sent back to the anchoring system, where the state of occluded objects is updated. The feedback-loop between the anchoring system and the probabilistic reasoner results in an additional anchoring functionality (Persson et al., 2020b):

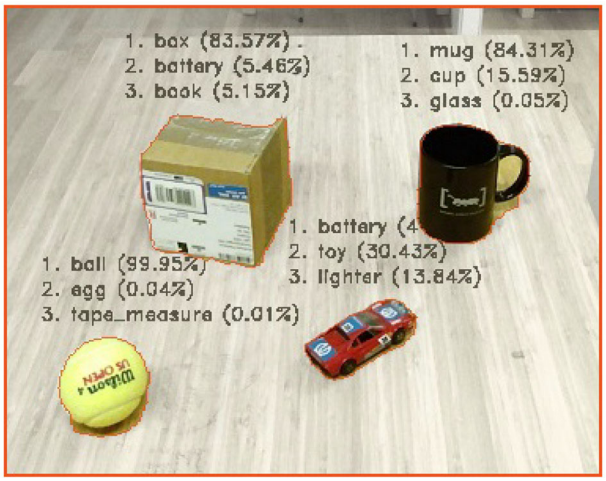

FIGURE 4 | Examples of *semantically categorized objects* (depicted with the *3-top best object categories* for each segmented object).

- *Track*—extends the definition of an anchor α^x from time $t-1$ to time t. This functionality is directly responding to the state of the probabilistic object tracker, which ensures that the percepts pointed to by the anchor are the adequate perceptual representation of the object, even though the object is currently not perceived.

Even though the mapping between measured attribute values and corresponding predicate symbols is an essential facet of anchoring, we will not cover the *predicate grounding* in further detail in this paper. However, for completeness, we will refer to **Figure 3** and exemplify that the *predicate grounding relation* of a *color attribute* can, intuitively, be expressed as the encoded correspondence between a specific peek in the color histogram and certain *predicate symbol* (e.g., the symbol `black` for the `mug` object). Likewise, a future greater ambition of this work is to establish a practical framework through which the spatial relationships between objects are encoded and expressed using symbolic values, e.g., *object A* is `underneath` *object B*.

3.2. Probabilistic Anchoring System

The entry point for the anchoring system, seen in **Figure 2**–(2.), is a learned *matching function*. This function assumes a bottom-up approach to perceptual anchoring, described in Loutfi et al. (2005), where the system constantly receives candidate anchors and invokes a number of attribute specific matching similarity formulas (i.e., one matching formula for each measured attribute). More specifically, a set of attributes Φ_y of an unknown candidate anchor α_t^y (given at current time t) is compared against the set of attributes Φ_x of an existing anchor α_{t-k}^x (defined at time $t-k$) through attribute specific similarity formulas. For instance, the similarity between the *positions attributes* ϕ_y^{pos} of an unknown candidate anchor, and the last updated position $\phi_{t-k,x}^{pos}$ of an existing anchor, is calculated according to the L^2-norm (in 3-D space), which is further mapped to a *normalized similarity value* (Blodow et al., 2010):

$$d^{pos}(\phi_{t-k,x}^{pos}, \phi_{t,y}^{pos}) = e^{-L^2(\phi_{t-k,x}^{pos}, \phi_{t,y}^{pos})} \tag{1}$$

Hence, the similarity between two *positions attributes* is given in interval $[0,1]$, where a value of 1 is equivalent with perfect correspondence. Likewise, the similarity between two *color attributes* are calculated by the *color correlation*, while the similarity between *size attributes* is calculated according to the *generalized Jaccard similarity* (for further details regarding similarity formulas, we refer to our previous work Persson et al., 2020b). The similarities between the attributes of a known anchor and an unknown candidate anchor are then fed to the learned matching function to determine whether the matching function classifies the unknown anchor to be *acquired* as a new anchor, or *re-acquired* as an existing anchor. This matching function is utilized by a support vector machine, which has been trained with the use of 5,400 samples of *humanly annotated data* (i.e., human users have provided feedback about what they think is the appropriate anchoring action for objects in various scenarios), to a *classification accuracy* of 96.4%. It should, however, be noted that the inputs for this classifier are the various similarity values between attributes (cf. Equation 1), and that the classifier learns to interpret, combine and weight different similarity values between attributes in order to correctly determine whether a new anchor should be acquired, or if an existing anchor should be re-acquired. By omitting similarity values of specific attributes during training, we can also estimate the importance of different attributes. For example, excluding the similarity values between *color attributes* during training reduces the classification accuracy to 92.5%, while excluding the similarity values between *position attributes*, instead, decreases the accuracy to 72.8%. This illustrating example of the importance of the position of an object, in the context of anchoring, is a further motivation for reasoning about possible states once the position of an object changes during the absence of observations (e.g., in the case of movements during occlusions).

In our prior work on anchoring, the attribute values have, in addition, always been assumed to be deterministic within a single time step. This assumption keeps the anchoring system de facto deterministic even though it is coupled to a probabilistic reasoning module. We, therefore, extend the anchoring notation with two distinct specifications of (volatile) attributes:

1. An attribute $\phi_t \in \varphi$ is *deterministic at time t* if it takes a single value from the domain $D(\phi_t)$.
2. An attribute $\phi_t \in \varphi$ is *probabilistic at time t* if it is distributed according to a probability distribution $Pr(\phi_t)$ over the domain $D(\phi_t)$ at time step t.

Having a probabilistic attribute value ϕ_t (e.g., $\phi_{t-k,x}^{pos}$ in Equation 1), means that the similarity calculated with the probabilistic attribute values (e.g., the similarity value d^{pos}), will also be probabilistic. Next, in order to use an anchor matching function together with probabilistic similarity values, two extensions are possible: (1) extend the anchor matching function to accept random variables (i.e., probabilistic similarity values), or (2) retrieve a point estimate of the random variable.

We chose the second option as this allows us to reuse the anchor matching function learned in Persson et al. (2020b) without the additional expense of collecting data and re-training the anchor matching function. The algorithm to produce the set of matching similarity values that are fed to the anchor matching function is given in Algorithm 1, where lines 4–5 are the extension proposed in this work.

Algorithm 1 Attribute Compare

Input: Φ_x, Φ_y – *sets of anchor attribute values*
Output: $\mathcal{D}_{x,y}$ – *set of matching similarity values*

1: **function** ATTRIBUTECOMPARE(Φ_x, Φ_y)
2: $\quad\mathcal{D}_{x,y} \leftarrow$ *empty set*
3: $\quad$**for each** $\phi_{t,x} \in \Phi_x$ **and** $\phi_{t,y} \in \Phi_y$ **do**
4: $\quad\quad$**if** $\phi_{t-1,x}$ is probabilistic **then**
5: $\quad\quad\quad\mathcal{D}_{x,y} \overset{+}{\leftarrow} \underset{\phi_{t-1,x}}{point\_estimate}(d(\phi_{t-1,x}, \phi_{t,y}))$
6: $\quad\quad$**else** ▷ deterministic case
7: $\quad\quad\quad\mathcal{D}_{x,y} \overset{+}{\leftarrow} d(\phi_{t-k,x}, \phi_{t,y})$
8: $\quad$**return** $\mathcal{D}_{x,y}$

The *point_estimate* function in Algorithm 1 (line 5) is attribute specific [indicated by the subscript ($\phi_{t-1,x}$)], i.e., we can chose a different point estimation function for *color attributes* than for *position attributes*. An obvious attribute upon which reasoning can be done is the position attribute, for example, in the case of possible occlusions. In other words, we would like to perform probabilistic anchoring while taking into account the *probability distribution of an anchor's position*. A reasonable goal is then to match an unknown candidate anchor with the most likely anchor, i.e., with the anchor whose position attribute value is located at the highest mode of the probability distribution of the position attribute values. This is achieved by replacing line 5 in Algorithm 1 with:

$$\mathcal{F}_x^{pos} \leftarrow \left\{ \phi_{t-1,x}^{pos} \,\middle|\, \frac{\partial Pr(\phi_{t-1,x}^{pos})}{\partial \phi_{t-1,x}^{pos}} = 0 \right\} \tag{2}$$

$$\mathcal{D}_{x,y} \overset{+}{\leftarrow} \max_{\phi^{pos} \in \mathcal{F}_x^{pos}} (d^{pos}(\phi^{pos}, \phi_{t,y}^{pos})) \tag{3}$$

$\mathcal{F}_x^{pos}$ is the set of positions situated at the modes of the probability distribution $Pr(\phi_{t-1,x}^{pos})$. In Equation (3) we take the max as the co-domain of the position similarity value d^{pos} is in $[0, 1]$, where 1 reflects perfect correspondence (cf. Equation 1).

In Persson et al. (2020b), we approximated the probabilistic state of the world in the *inference system* (cf. **Figure 2**–(3.)) by N particles, which are updated by means of particle filtering. The precise information that is passed from the inference system to the anchoring system is a list of N particles that approximate a (possible) multi-modal belief of the world. More specifically, an anchor α_t^x is updated according to the N particles of possible states of a corresponding object, maintained in the inference system, such that N possible 3-D positions are added to the volatile *position attributes* φ_x^{pos}. In practice we assume that samples are only drawn around the modes of the probability distribution, which means that we can replace line 5 of Algorithm 1 with:

$$\mathcal{D}_{x,y} \overset{+}{\leftarrow} \max_i \left(d^{pos}(\phi_{t-1,x,i}^{pos}, \phi_{t,y}^{pos}) \right) = \max_i \left(e^{-L^2(\phi_{t-1,x,i}^{pos}, \phi_{t,y}^{pos})} \right) \tag{4}$$

Where $\phi_{t-1,x,i}$ is a sampled position and i ranges from 1 to the number of samples N.

Performing probabilistic inference in the coordinate space is a choice made in the design of the probabilistic anchoring system. Instead, the probabilistic tracking could also be done in the HSV color space, for instance. In this case, the similarity measure used in Algorithm 1 would have to be adapted accordingly. It is also conceivable to combine the tracking in coordinate space and color space. This introduces, however, the complication of finding a similarity measure that works on the coordinate space and the color space at the same time. A solution to this would be to, yet again, learn this similarity function from data (Persson et al., 2020b).

4. LEARNING DYNAMIC DISTRIBUTIONAL CLAUSES

While several approaches exist in the SRL literature that learn probabilistic relational models, most of them focus on parameter estimation (Sato, 1995; Friedman et al., 1999; Taskar et al., 2002; Neville and Jensen, 2007) and structure learning has been restricted to discrete data. Notable exceptions include the recently proposed hybrid relational formalism by Ravkic et al. (2015), which learns relational models in a discrete-continuous domain but has not been applied to dynamics or robotics, and the related approach of Nitti et al. (2016b), where a relational tree learner DDC-TL learns both the structure and the parameters of distributional clauses. DDC-TL has been evaluated on learning action models (pre- and post-conditions) in a robotics setting from before and after states of executing the actions. However, there were several limitations of the approach. It simplified perception by resorting to AR tags for identifying the objects, it did not consider occlusion, and it could not deal with uncertainty or noise in the observations.

A more general approach to learning distributional clauses, extended with *statistical models* proposed in Kumar et al. (2020)[6]. Such a statistical model relates continuous variables in the body of a distributional clause to parameters of the distribution in the head of the clause. The approach simultaneously learns the structure and parameters of (non-dynamic) distributional clauses, and estimates the parameters of the statistical model in clauses. A DC program consisting of multiple distributional clauses is capable of expressing intricate probability distributions over discrete and continuous random variables. A further shortcoming of DDC-TL (also tackled by Kumar et al.) is the inability of learning in the presence of background knowledge—that is, additional (symbolic) probabilistic information about objects in the world and relations (such as spatial relations)

[6]https://github.com/niteshroyal/DreaML

among the objects that the learning algorithm should take into consideration.

However, until now, the approach presented in Kumar et al. (2020) has only been applied to the problem of autocompletion of relational databases by learning a (non-dynamic) DC program. We now demonstrate with an example of how this general approach can also be applied for learning dynamic distributional clauses in a robotics setting. A key novelty in the context of perceptual anchoring is that we learn a DDC program that allows us to reason about occlusions.

Example 6: Consider again a scenario where objects might get fully occluded by other objects. We would now like to learn the ToO that describes whether an object is occluded or not given multiple observations of the before and after state. In DDC we represent observations through facts as follows

```
pos(o1_exp1):t~= 2.3.
pos(o1_exp1)t+1~= 9.3.
pos(o2):t~= 2.2.
pos(o2):t+1~= 9.2.
occluded_by(o1_exp1,o2_exp1):t+1.
pos(o3_exp1):t~= 8.3.
```

⋮

For the sake of clarity, we have considered only one-dimensional positions in this example.

Example 7: Given the data in form of dynamic distributional clauses, we are now interested in learning the ToO instead of relying on a hand-coded one, as in Example 3. An excerpt from the set of clauses that constitute a learned ToO is given below. As in Example 3, the clauses describe the circumstances under which an object (`Occluded`) is potentially occluded by an other object (`Occluder`).

```
occluder(Occluded,Occluder):t+1~finite(1.0:false)
    ← occluded_by(Occluded,Occluder):t,
    observed(Occluded):t+1.
occluder(Occluded,Occluder):t+1~finite(0.92:true,
    0.08:false) ← occluded_by(Occluded,Occluder):
    t,\+observed(Occluded):t+1.
occluder(Occluded,Occluder):t+1~finite(P1:true,P2:
    false) ← \+occluded_by(Occluded,Occluder):t,
    \+observed(Occluded):t+1,
    distance(Occluded,Occluder):t~=Distance,
    logistic([Distance],[-16.9,0.8],P1),
    P2 is 1-P1.
```

Note that, in the second but last line of the last clause above the arbitrary threshold on the `Distance` is superseded by a learned statistical model, in this case a logistic regression, which maps the input parameter `Distance` to the probability `P1`:

$$\texttt{P1} = \frac{1}{1 + e^{16.9 \times \texttt{D-0.8}}} \tag{5}$$

Replacing the hand-coded `occluder` rule with the learned one in the theory of occlusion allows us to track occluded objects with a partially learned model of the world.

In order to learn dynamic distributional clauses, we first map the predicates with subscripts that refer to the current time step `t` and the next time step `t+1` to standard predicates, which gives us an input DC program. For instance, we map `pos(o1_exp1):t` to `pos_t(o1_exp1)`, and `occluder(o1_exp1,o2_exp2):t+1` to `occluder_t1(o1_exp1,o2_exp2)`. The method introduced in Kumar et al. (2020) can now be applied for learning distributional clauses for the target predicate `occluder_t1(o1_exp1,o2_exp2)` from the input DC program.

Clauses for the target predicate are learned by inducing a distributional logic tree. An example of such a tree is shown in **Figure 5**. The key idea is that the set of clauses for the same target predicate are represented by a distributional logic tree, which satisfies the mutual exclusiveness property of distributional clauses. This property states that if there are two distributional clauses defining the same random variable, their bodies must be mutually exclusive. Internal nodes of the tree correspond to atoms in the body of learned clauses. A leaf node corresponds to a distribution in the head and to a statistical model in the body of a learned clause. A path beginning at the root node and proceeding to a leaf node in the tree corresponds to a clause. Parameters of the distribution and the statistical model of the clause are estimated by maximizing the expectation of the log-likelihood of the target in partial possible worlds. The worlds are obtained by proving all possible groundings of the clause in the input DC program. The structure of the induced tree defines the structure of the learned clauses. The approach requires declarative bias to restrict the search space while inducing the tree. Note that the fragment of programs that can be learned by the algorithm described in Kumar et al. (2020) does not include recursive programs, as only tree structured programs can be learned.

In summary, the *input* to the learner of Kumar et al. (2020) is a DC program consisting of

- background knowledge, in the form of DC clauses;
- observations, in the form of DC clauses—these constitute the training data;
- the declarative bias, which is necessary to specify the hypothesis space of the DC program (Adé et al., 1995);
- the target predicates for which clauses should be learned.

The *output* is:

- a set of DC clauses represented as a tree for each target predicate specified in the input.

In contrast to learning algorithms that tackle discrete data, the declarative bias used to learn rules with continuous random variables has to additionally specify whether a random variable is distributed according to a discrete probability distribution or a continuous probability distribution. In other words, the declarative bias specifies whether a leaf in the learned tree represents continuous or a discrete probability distribution. Currently the algorithm of Kumar et al. (2020) only supports normal distributions for continuous random variables and finite categorical distributions for discrete random variables.

Once the clauses are learned, predicates are mapped back to predicates with subscripts to obtain dynamic distributional clauses. For instance, `occluder_t1(Occluded,Occluder)` in the learned clauses is mapped back to `occluder(Occluded,Occluder):t+1`.

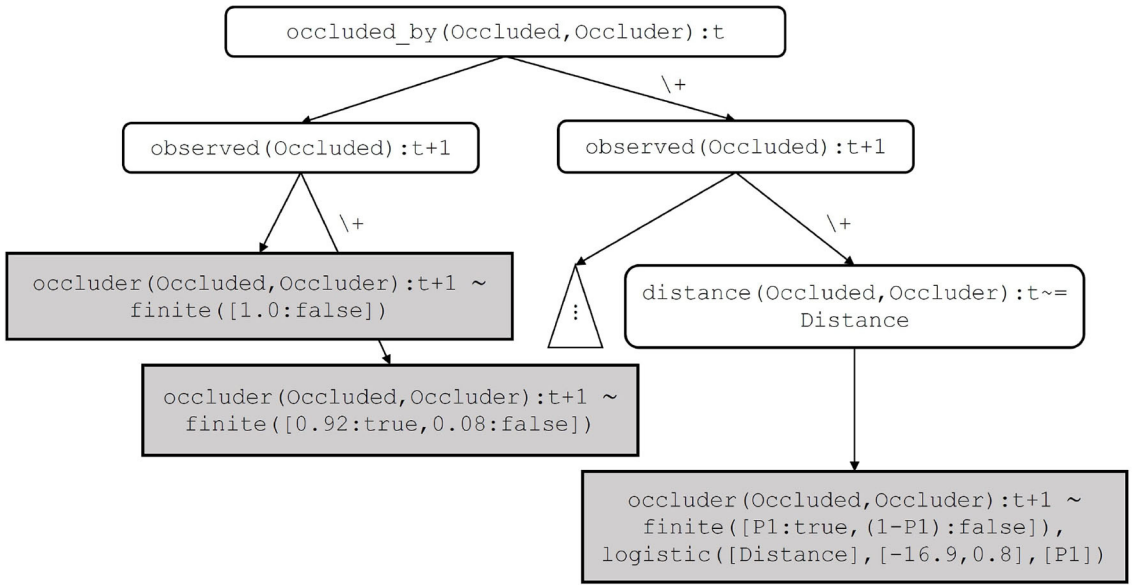

FIGURE 5 | A distributional logic tree that represents learned clauses for the target `occluder(Occluded,Occluder):t+1`. The leftmost path corresponds to the first clause, the rightmost path corresponds to the last clause for `occluder(Occluded,Occluder):t+1` in Example 6. Internal nodes such as `occluder(Occluded,Occluder):t` and `observed(Occluded):t+1` are discrete features, whereas, internal nodes such as `distance(Occluded,Occluder):t+1~=Distance` is a continuous feature.

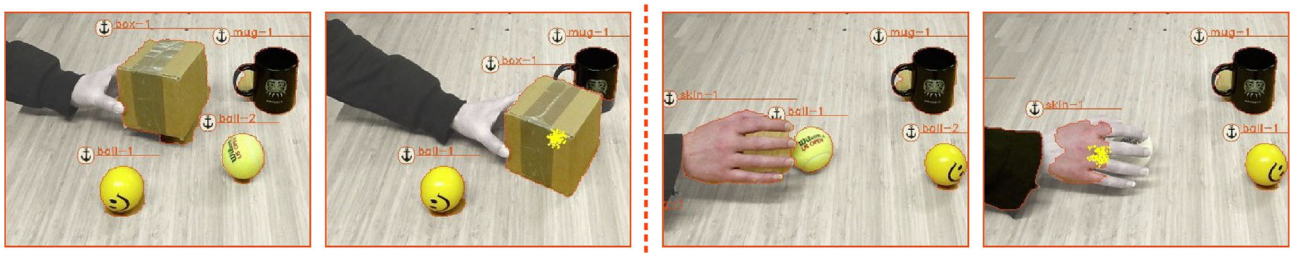

FIGURE 6 | Depicted are two training points in the data set that were used to learn the transition rule of an object to another object. The panels on the left show a **ball** that is being occluded by a **box**, and on the right, the same **ball** that is being grabbed by a hand (or a **skin** object, as we have only trained our used GoogLeNet model to recognize general human skin objects instead of particular human body parts, cf. section 3.1). The plotted dots on top of the occluding object represent samples drawn from the probability distribution of the occluded object, in other words the object that is labeled in the data set to transition into the occluding counterpart.

The data used for the learning of the theory of occlusion consists of training points of before-after states of two kinds. The first kind are pairs describing a transition of an object from being observed to being occluded. Here, the data set contained 58 data point pairs, with 13 pairs describing the transition of an object from being observed to being occluded and the remaining 45 describing situations with an object being observed in the before state, as well as in the after state. Examples of two raw data points for the first kind can be seen in **Figure 6**. The second kind of data pairs describe an object being occluded in the before state as well as in the after state. Here we had 425 positively labeled data pairs, i.e., an object was occluded in the after state. For 416 of these pairs the labeling was correct while the remaining were mislabeled (the occluded object in the after state was labeled as observed in the before state). For negative data points (objects not occluded in the after state) we had 1,152 data pairs. For 473 of these pairs the non-occluded object was labeled as not occluded in the before state as well, for 2 it was labeled as occluded and for the remaining there was no label in the before state. While for the first kind we did not have any mislabeled data, the data points for the second kind did exhibit a small percentage of inaccurately labeled data pairs, for example approximately $\approx$ 3% for positive data pairs. Noise in the data was also present in the position of the objects—originating from the perceptual anchoring system.

The predicate specifying whether an object is occluded in the after state or not was the target predicate of the learner. In the declarative bias we specified the predicates to be used as features. These included predicates specifying whether an object is occluded in the before state, position predicates,

and distance predicates between objects. Furthermore, the rule learner automatically decides on which statistical models, if any, to use in the learned rules. The available statistical models are linear, softmax, and logistic models.

The processed data that was fed to the distributional clauses learner is available online[7] as well as models with the learned theory of occlusion[8]. The learned theory of occlusions is conceptually close to the one shown in Example 7. The first-order nature of the learned rules enable the usage of the rules in situations of object occlusions with specific objects that were not present in the training data set.

5. EVALUATION

A probabilistic anchoring system that is coupled to an inference system (cf. section 3.2) is comprised of several interacting components. This turns the evaluation of such a combined framework, with many integrated systems, into a challenging task. We, therefore, evaluate the integrated framework as a whole on representative scenarios that demonstrate our proposed extensions to perceptual anchoring. In section 5.1, we demonstrate how the extended anchoring system can handle probabilistic multi-modal states (described in section 3). In sections 5.2 and 5.3, we show that semantic relational object tracking can be performed with the probabilistic logic rules (in form of a DDC program) instead of handcrafted ones.

5.1. Multi-Modal Occlusions

We present the evaluation in the form of screenshots captured during the execution of a scenario where we obscure the stream of sensor data. We start out with three larger objects (two `mug` objects and one `box` object), and one smaller `ball` object. During the occlusion phase, seen in **Figure 7**–(1.), the RGB-D sensor is covered by a human hand and the smaller `ball` is hidden underneath one of the larger objects. In **Figure 7**–(1.), it should also be noted that the anchoring system preserves the latest update of the objects, which is here illustrated by the outlined contour of each object. At the time that the sensory input stream is uncovered, and there is no longer any visual perceptual input of the `ball` object, the system can only speculate about the whereabouts of the missing object. Hence, the belief of the `ball`'s position becomes a multi-modal probability distribution, from which we draw samples, as seen in **Figure 7**–(2.). At this point, we are, however, able to track the smaller `ball` through its probabilistic relationships with the other larger objects. During all the movements of the larger objects, the probabilistic inference system manages to track the modes of the probability distribution of the position of the smaller `ball`. The probability distribution for the position of the smaller `ball` (approximated by N samples) is continuously fed back to the anchoring system. Consequently, once the hidden `ball` is revealed and reappears in the scene, as seen in **Figures 7**–(3.), (4.), the `ball` is correctly *re-acquired* as the initial `ball-1` object. This would not have been possible with a non-probabilistic anchoring approach.

5.2. Uni-Modal Occlusions With Learned Rules

The conceptually easiest ToO is one that describes the occlusion of an object by another object. Using the method described in section 4, we learned such a ToO, which we demonstrate in **Figure 8**. Shown are two scenarios. In the one in the upper row a `can` gets occluded by a `box`—shown in the second screenshot. The `can` is subsequently tracked through its relation with the observed `box` and successfully re-anchored as `can-1` once it is revealed. Note that in the second screenshot, the `mug` is also briefly believed to be hidden under `box`, shown through the `black` dots, as the `mug` is temporally obscured behind the `box` and not observed by the vision system. However, once the `mug` is again observed the black dots disappear.

In the second scenario, we occlude one of two `ball` objects with a `box` and track the `ball` again through its relation with the `box`. Note that some of the probability mass accounts for the possibility for the occluded `ball` to be occluded by the `mug`. This is due to the fact that the learned rule is probabilistic.

In both scenarios, we included background knowledge that specifies that a `ball` cannot be the an occluder of an object (it does not *afford* to be the occluder). This is also why we see a probability mass of the occluded `ball` at the `mug`'s location and not at the observed `ball`'s location in the second scenario.

5.3. Transitive Occlusions With Learned Rules

Learning (probabilistic) rules, instead of a black-box function, has the advantage that a set of rules can easily be extended with further knowledge. For example, if we would like the ToO to be recursive, i.e., objects can be occluded by objects that are themselves occluded, we simply have to add the following rule to the DDC program describing the theory of occlusion:

```
occluded_by(Occluded,Occluder):t+1 ←
    occluded_by(Occluded,Occluder):t,
    \+observed(Occluded):t+1,
    \+observed(Occluder):t+1,
    occluded_by(Occluder,_):t+1.
```

Extending the ToO from section 5.2 with the above rule, enables the anchoring system to handle recursive occlusions. We demonstrate such a scenario in **Figure 9**. Initially, we start this scenario with a `ball`, a `mug` and a `box` object (which in the beginning is miss-classified as `block` object, cf. **Figure 4**). In the first case of occlusion, seen in **Figure 9**–(1.), we have the same type of uni-modal occlusion as described in the previous section 5.2, where the `mug` occludes the `ball` and, subsequently, triggers the learned relational transition (where plotted `yellow` dots represent samples drawn from the probability distribution of the occluded `ball` object). In the second recursive case of occlusion, seen in **Figure 9**–(2.), we proceed by also occluding the `mug` with the `box`. Above rule administers this *transitive occlusion*—triggered when the `ball` is still hidden underneath the `mug` and the `mug` is occluded by the `box`. This is illustrated here

[7]https://bitbucket.org/reground/anchoring/downloads/
[8]https://bitbucket.org/reground/anchoring

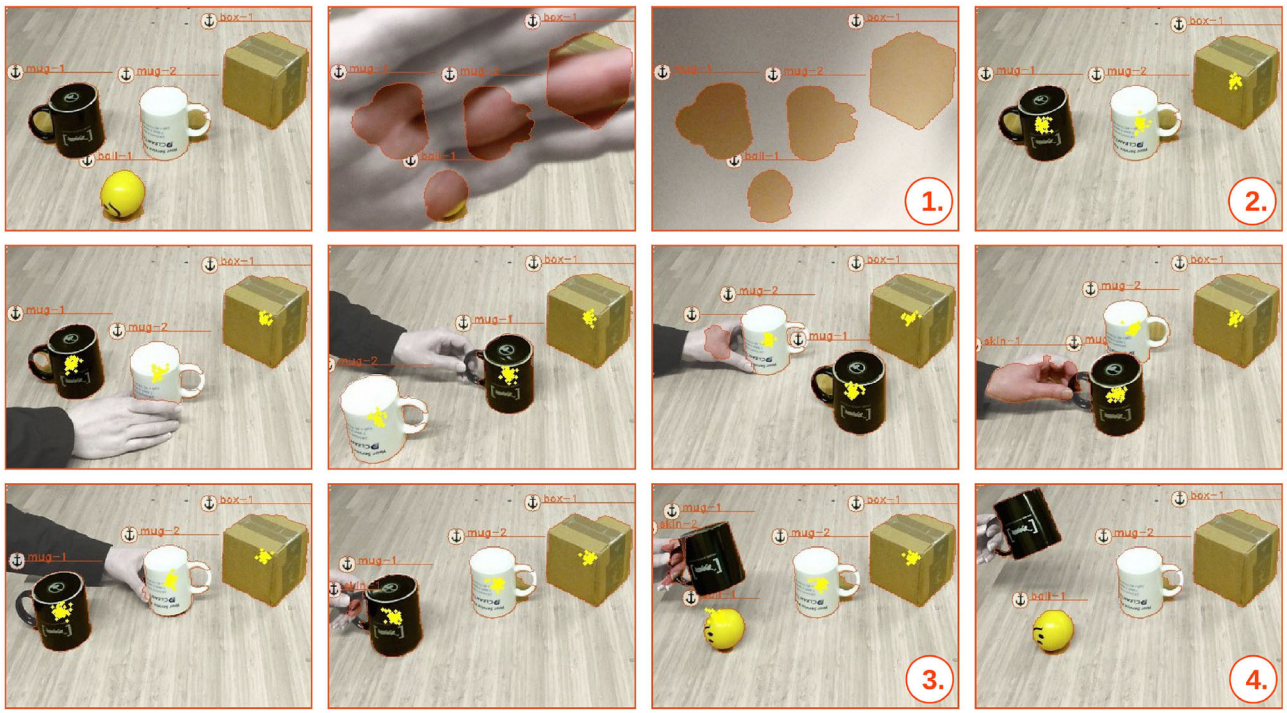

FIGURE 7 | Screen-shots captured during the execution of a scenario where the stream of sensor data is obscured. Visually perceived anchored objects are symbolized by a unique anchor identifiers (e.g., mug-1), while occluded hidden objects are depicted by plotted particles that represent possible positions of the occluded object in the inference system. The screenshots illustrate a scenario where the RGB-D sensor is covered and a ball is hidden under either one of three larger objects. These larger objects are subsequently shuffled around before the whereabouts of the hidden ball is revealed.

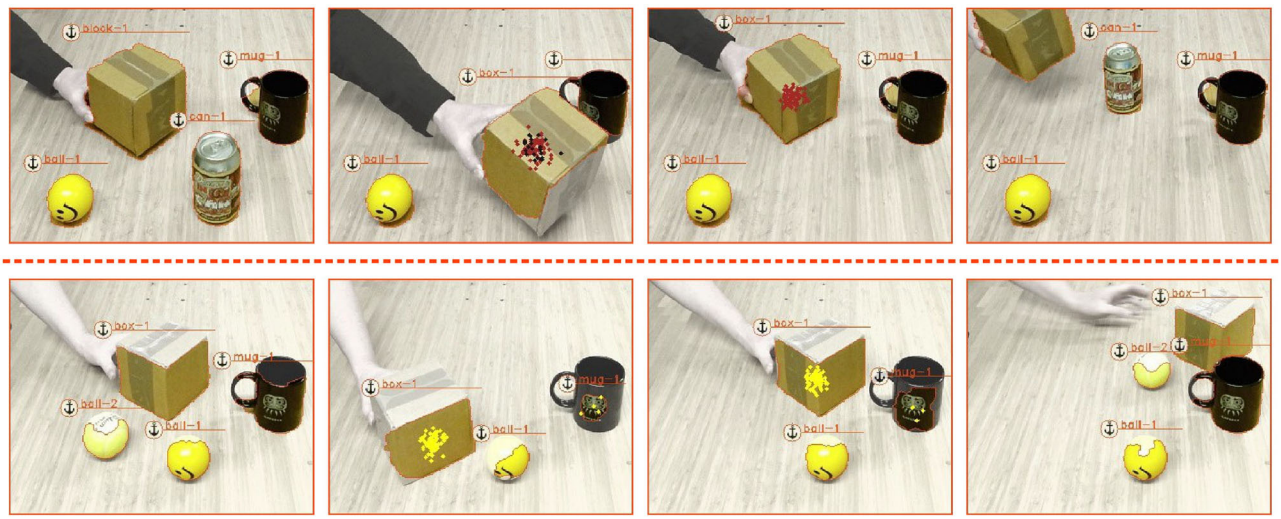

FIGURE 8 | The two scenario show how a learned ToO is used to perform semantic relational object tracking. In both scenarios, an object is occluded by a box and successfully tracked before the occluded object is being revealed and again *re-acquired* as the same initial object.

by both `yellow` and `black` plotted dots that represent samples drawn from the probability distributions of occluded `mug` and the transitively occluded `ball` object, respectively. Consequently, once the `box` is moved, both the `mug` and the `ball` are tracked through the transitive relation with the occluding `box`. Reversely, it can be seen, in **Figure 9**–(3.), that once the `mug` object is revealed the object is correctly *re-acquired* as the same `mug-1` object, while the relation between the `mug` and the occluded `ball`

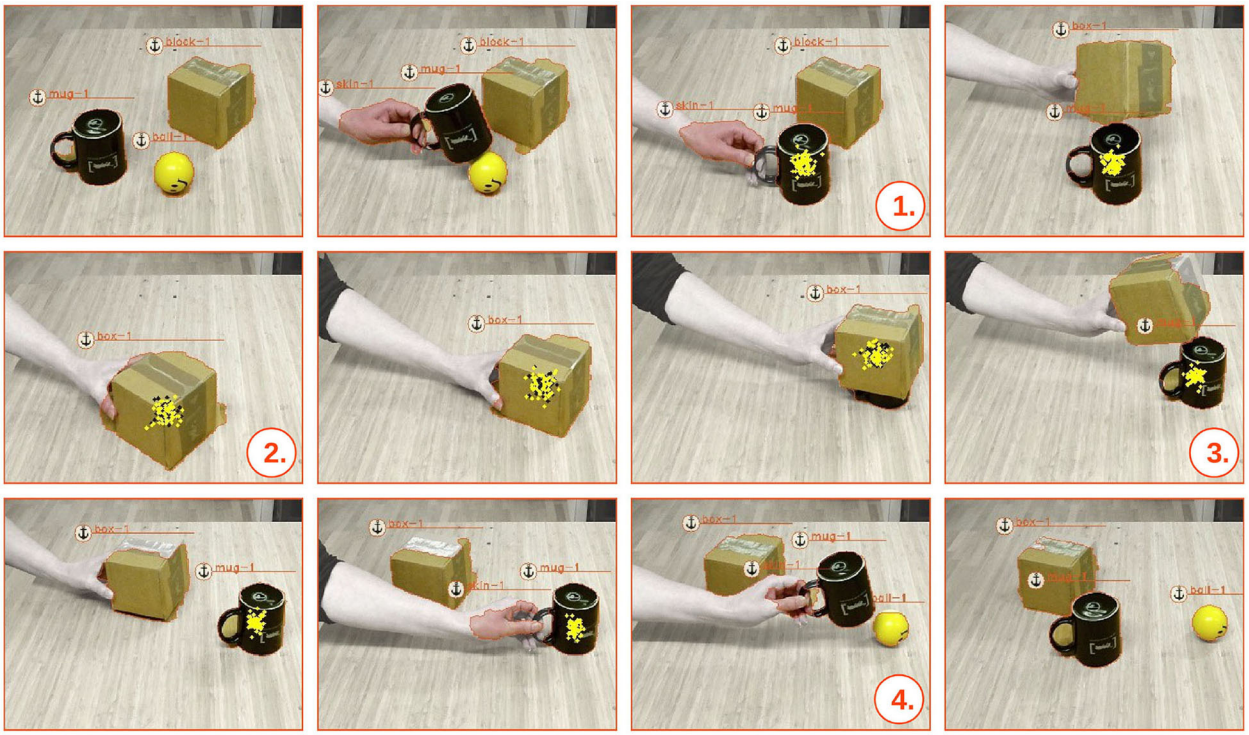

FIGURE 9 | A scenario that demonstrates transitive occlusions based on learned rules for handling the theory of occlusions. First the ball is occluded by the mug (indicated by the yellow dots) and subsequently the mug is occluded in turn by the box (indicated by the black dots). Once the mug is observed again the ball is still believed to be occluded by the mug.

object is still preserved. Finally, as the `ball` object is revealed, in **Figure 9**–(4.), it can be also seen that the object is, likewise, correctly *re-acquired* as the same `ball-1` object.

6. CONCLUSIONS AND FUTURE WORK

We have presented a two-fold extension to our previous work on semantic world modeling (Persson et al., 2020b), where we proposed an approach to couple an anchoring system to an inference system. Firstly, we extended the notions of perceptual anchoring toward the probabilistic setting by means of probabilistic logic programming. This allowed us to maintain a multi-modal probability distribution of the positions of objects in the anchoring system and to use it for matching and maintaining objects at the perceptual level—thus, we introduce probabilistic anchoring of objects either directly perceived by the sensory input data or logically inferred through probabilistic reasoning. We illustrated the benefit of this approach with the scenario in section 5.1, which the anchoring system was able to resolve correctly only due to its ability of maintaining a multi-modal probability distribution. This also extends an earlier approach to relational object tracking (Nitti et al., 2014), where the symbol-grounding problem was solved by the use of AR tags.

Secondly, we have deployed methods from statistical relational learning to the field of anchoring. This approach allowed us to learn, instead of handcraft, rules needed in the reasoning system. A distinguishing feature of the applied rule learner (Kumar et al., 2020) is its ability to handle both continuous and discrete data. We then demonstrated that combining perceptual anchoring and SRL is also feasible in practice by performing relational anchoring with a learned rule (demonstrated in section 5.2). This scenario did also exhibit a further strength of using SRL in anchoring domains, namely that the resulting system becomes a highly modularizable system. In our evaluation, for instance, we were able to integrate an extra rule into the ToO, which enabled us to resolve recursive occlusions (described in section 5.3).

A possible future direction would be to exploit how anchored objects and their spatial relationships, tracked over time, facilitate the learning of both the function of objects, as well as object affordances (Kjellström et al., 2011; Moldovan et al., 2012; Koppula et al., 2013; Koppula and Saxena, 2014). Through the introduction of a probabilistic anchoring approach, together with the learning of the rules that express the relation between objects, we have presented a potential framework for future studies of spatial relationship between objects, e.g., the spatial-temporal relationships between objects and human hand actions to learn the function of objects (cf. Kjellström et al., 2011; Moldovan et al., 2012). Such a future direction would tackle a similar question, currently discussed in the neural-symbolic community (Garcez et al., 2019), namely how to propagate back symbolic information to sub-symbolic representations of the world. A recent piece of work that combines SRL and neural methods is, for instance, Manhaeve et al. (2018).

Another aspect of our work that deserves future investigation is probabilistic anchoring, in itself. With the approach presented in this paper we are merely able to perform MAP inference. In order to perform full probabilistic anchoring, one would need to render the anchor matching function itself fully probabilistic, i.e. the anchor matching function would need to take as arguments random variables and again output probability distributions instead of point estimates—ideas borrowed from multi-hypothesis anchoring (Elfring et al., 2013) might, therefore, be worthwhile to consider for future work.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

PZ and AP outlined the extension of the framework to include probabilistic properties and multi-modal states. PZ and NK integrated SRL with perceptual anchoring. PZ, AP, and NK performed the experimental evaluation. AL and LD have developed the notions and the ideas in the paper together with the other authors. PZ, NK, AP, AL, and LD have all contributed to the text. All authors contributed to the article and approved the submitted version.

ACKNOWLEDGMENTS

The work was supported by the Swedish Research Council (Vetenskapsrådet, grant number: 2016-05321), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. PZ was supported by the Research Foundation-Flanders and Special Research Fund of the KU Leuven. This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant number 694980, SYNTH). This work has been supported by the ReGround project (http://reground.cs.kuleuven.be), which is a CHIST-ERA project funded by the EU H2020 framework program.

REFERENCES

Adé, H., De Raedt, L., and Bruynooghe, M. (1995). Declarative bias for specific-to-general ilp systems. *Mach. Learn.* 20, 119–154. doi: 10.1007/BF00993477

Angelopoulos, N., and Cussens, J. (2008). Bayesian learning of Bayesian networks with informative priors. *Ann. Math. Artif. Intell.* 54, 53–98. doi: 10.1007/s10472-009-9133-x

Angelopoulos, N., and Cussens, J. (2017). Distributional logic programming for Bayesian knowledge representation. *Int. J. Approx. Reason.* 80, 52–66. doi: 10.1016/j.ijar.2016.08.004

Beltagy, I., Roller, S., Cheng, P., Erk, K., and Mooney, R. J. (2016). Representing meaning with a combination of logical and distributional models. *Comput. Linguist.* 42, 763–808. doi: 10.1162/COLI_a_00266

Blodow, N., Jain, D., Marton, Z. C., and Beetz, M. (2010). "Perception and probabilistic anchoring for dynamic world state logging," in *2010 10th IEEE-RAS International Conference on Humanoid Robots* (Nashville, TN), 160–166. doi: 10.1109/ICHR.2010.5686341

Chella, A., Coradeschi, S., Frixione, M., and Saffiotti, A. (2004). "Perceptual anchoring via conceptual spaces," in *Proceedings of the AAAI-04 Workshop on Anchoring Symbols to Sensor Data* (San Jose, CA), 40–45.

Chella, A., Frixione, M., and Gaglio, S. (2003). Anchoring symbols to conceptual spaces: the case of dynamic scenarios. *Robot. Auton. Syst.* 43, 175–188. doi: 10.1016/S0921-8890(02)00358-5

Coradeschi, S., and Saffiotti, A. (2000). "Anchoring symbols to sensor data: preliminary report," in *Proceedings of the 17th AAAI Conference* (Menlo Park, CA: AAAI Press), 129–135. Available online at: http://www.aass.oru.se/~asaffio/

Coradeschi, S., and Saffiotti, A. (2001). "Perceptual anchoring of symbols for action," in *International Joint Conference on Artificial Intelligence (IJCAI)* (Seattle, WA), 407–416.

De Raedt, L., Kersting, K., Natarajan, S., and Poole, D. (2016). Statistical relational artificial intelligence: logic, probability, and computation. *Synth. Lect. Artif. Intell. Mach. Learn.* 10, 1–189. doi: 10.2200/S00692ED1V01Y201601AIM032

De Raedt, L., and Kimmig, A. (2015). Probabilistic (logic) programming concepts. *Mach. Learn.* 100, 5–47. doi: 10.1007/s10994-015-5494-z

Elfring, J., van den Dries, S., van de Molengraft, M., and Steinbuch, M. (2013). Semantic world modeling using probabilistic multiple hypothesis anchoring. *Robot. Auton. Syst.* 61, 95–105. doi: 10.1016/j.robot.2012.11.005

Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., et al. (2015). Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theor. Pract. Logic Programm.* 15, 358–401. doi: 10.1017/S1471068414000076

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999). "Learning probabilistic relational models," in *IJCAI, Vol. 99* (Stockholm), 1300–1309.

Garcez, A. D., Gori, M., Lamb, L. C., Serafini, L., Spranger, M., and Tran, S. N. (2019). Neural-symbolic computing: an effective methodology for principled integration of machine learning and reasoning. *arXiv [preprint]*. arXiv:1905.06088.

Gardner, M., Talukdar, P., Krishnamurthy, J., and Mitchell, T. (2014). "Incorporating vector space similarity in random walk inference over knowledge bases," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha), 397–406. doi: 10.3115/v1/D14-1044

Getoor, L. (2013). "Probabilistic soft logic: a scalable approach for Markov random fields over continuous-valued variables," in *Proceedings of the 7th International Conference on Theory, Practice, and Applications of Rules on the Web, RuleML'13* (Berlin; Heidelberg: Springer-Verlag). doi: 10.1007/978-3-642-39617-5_1

Getoor, L., and Taskar, B. (2007). *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA: The MIT Press. doi: 10.7551/mitpress/7432.001.0001

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). "Explaining explanations: an overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)* (Rome), 80–89. doi: 10.1109/DSAA.2018.00018

Günther, M., Ruiz-Sarmiento, J., Galindo, C., González-Jiménez, J., and Hertzberg, J. (2018). Context-aware 3D object anchoring for mobile robots. *Robot. Auton. Syst.* 110, 12–32. doi: 10.1016/j.robot.2018.08.016

Gutmann, B., Thon, I., Kimmig, A., Bruynooghe, M., and De Raedt, L. (2011). The magic of logical inference in probabilistic programming. *Theor. Pract. Logic Programm.* 11, 663–680. doi: 10.1017/S1471068411000238

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV). doi: 10.1109/CVPR.2016.90

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. (2017). "Safety verification of deep neural networks," in *International Conference on Computer Aided Verification* (Heidelberg: Springer), 3–29. doi: 10.1007/978-3-319-63387-9_1

Kjellström, H., Romero, J., and Kragić, D. (2011). Visual object-action recognition: inferring object affordances from human demonstration. *Comput. Vis. Image Understand.* 115, 81–90. doi: 10.1016/j.cviu.2010.08.002

Koller, D., and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press.

Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from RGB-D videos. *Int. J. Robot. Res.* 32, 951–970. doi: 10.1177/0278364913478446

Koppula, H. S., and Saxena, A. (2014). "Physically grounded spatio-temporal object affordances," in *European Conference on Computer Vision* (Zurich: Springer), 831–847. doi: 10.1007/978-3-319-10578-9_54

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Res. Logist. Quart.* 2, 83–97. doi: 10.1002/nav.3800020109

Kumar, N., Kuzelka, O., and De Raedt, L. (2020). Learning distributional programs for relational autocompletion. *arXiv:2001.08603*.

LeBlanc, K., and Saffiotti, A. (2008). "Cooperative anchoring in heterogeneous multi-robot systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)* (Pasadena, CA), 3308–3314. doi: 10.1109/ROBOT.2008.4543715

Loutfi, A. (2006). *Odour recognition using electronic noses in robotic and intelligent systems* (Ph.D. thesis). Örebro University, Örebro, Sweden.

Loutfi, A., and Coradeschi, S. (2006). Smell, think and act: a cognitive robot discriminating odours. *Auton. Robots* 20, 239–249. doi: 10.1007/s10514-006-7098-8

Loutfi, A., Coradeschi, S., Daoutis, M., and Melchert, J. (2008). Using knowledge representation for perceptual anchoring in a robotic system. *Int. J. Artif. Intell. Tools* 17, 925–944. doi: 10.1142/S0218213008004229

Loutfi, A., Coradeschi, S., and Saffiotti, A. (2005). "Maintaining coherent perceptual information using anchoring," in *Proc. of the 19th IJCAI Conference* (Edinburgh), 1477–1482.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). "Deepproblog: neural probabilistic logic programming," in *Advances in Neural Information Processing Systems* (Montreal, QC), 3749–3759.

Meshgi, K., and Ishii, S. (2015). The state-of-the-art in handling occlusions for visual object tracking. *IEICE Trans. Inform. Syst.* 98, 1260–1274. doi: 10.1587/transinf.2014EDR0002

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D. L., and Kolobov, A. (2007). "Chapter 13: BLOG: Probabilistic models with unknown objects," in *Statistical Relational Learning*, eds L. Getoor and B. Takar (Cambridge, MA: MIT Press), 373–397.

Moldovan, B., Moreno, P., Van Otterlo, M., Santos-Victor, J., and De Raedt, L. (2012). "Learning relational affordance models for robots in multi-object manipulation tasks," in *2012 IEEE International Conference on Robotics and Automation* (St Paul, MN), 4373–4378. doi: 10.1109/ICRA.2012.6225042

Neville, J., and Jensen, D. (2007). Relational dependency networks. *J. Mach. Learn. Res.* 8, 653–692.

Nitti, D., De Laet, T., and De Raedt, L. (2013). "A particle filter for hybrid relational domains," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Tokyo), 2764–2771. doi: 10.1109/IROS.2013.6696747

Nitti, D., De Laet, T., and De Raedt, L. (2014). "Relational object tracking and learning," in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (Hong Kong), 935–942. doi: 10.1109/ICRA.2014.6906966

Nitti, D., De Laet, T., and De Raedt, L. (2016a). Probabilistic logic programming for hybrid relational domains. *Mach. Learn.* 103, 407–449. doi: 10.1007/s10994-016-5558-8

Nitti, D., Ravkic, I., Davis, J., and De Raedt, L. (2016b). "Learning the structure of dynamic hybrid relational models," in *Proceedings of the Twenty-second European Conference on Artificial Intelligence* (The Hague: IOS Press), 1283-1290.

Persson, A., Länkvist, M., and Loutfi, A. (2017). Learning actions to improve the perceptual anchoring of objects. *Front. Robot. AI* 3:76. doi: 10.3389/frobt.2016.00076

Persson, A., Zuidberg Dos Martires, P., Loutfi, A., and De Raedt, L. (2020a). "ProbAnch: a modular probabilistic anchoring framework," in *IJCAI* (Yokohama). doi: 10.24963/ijcai.2020/771

Persson, A., Zuidberg Dos Martires, P., Loutfi, A., and De Raedt, L. (2020b). Semantic relational object tracking. *IEEE Trans. Cogn. Dev. Syst.* 12, 84–97. doi: 10.1109/TCDS.2019.2915763

Ravkic, I., Ramon, J., and Davis, J. (2015). Learning relational dependency networks in hybrid domains. *Mach. Learn.* 100, 217–254. doi: 10.1007/s10994-015-5483-2

Richardson, M., and Domingos, P. (2006). Markov logic networks. *Mach. Learn.* 62, 107–136. doi: 10.1007/s10994-006-5833-1

Riguzzi, F. (2018). *Foundations of Probabilistic Logic Programming*. Gistrup: River Publishers. doi: 10.1145/3191315.3191319

Ruiz-Sarmiento, J. R., Günther, M., Galindo, C., González-Jiménez, J., and Hertzberg, J. (2017). "Online context-based object recognition for mobile robots," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)* (Coimbra), 247–252. doi: 10.1109/ICARSC.2017.7964083

Sato, T. (1995). "A statistical learning method for logic programs with distribution semantics," in *Proceedings of the 12th International Conference on Logic Programming* (Tokyo), 715–729.

Sato, T. and Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *J. Artif. Intell. Res.* 15, 391–454. doi: 10.1613/jair.912

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature* 529:484. doi: 10.1038/nature16961

Smeulders, A. W., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., and Shah, M. (2013). Visual tracking: an experimental survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 1442–1468. doi: 10.1109/TPAMI.2013.230

Sterling, L., and Shapiro, E. Y. (1994). *The Art of Prolog: Advanced Programming Techniques*. Cambridge, MA: MIT Press.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Boston, MA), 1–9. doi: 10.1109/CVPR.2015.7298594

Taskar, B., Abbeel, P., and Koller, D. (2002). "Discriminative probabilistic models for relational data," in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence* (Edmonton, AB: Morgan Kaufmann Publishers Inc), 485–492.

Vezzani, R., Grana, C., and Cucchiara, R. (2011). Probabilistic people tracking with appearance models and occlusion classification: the ad-hoc system. *Pattern Recogn. Lett.* 32, 867–877. doi: 10.1016/j.patrec.2010.11.003

Wong, L. L., Kaelbling, L. P., and Lozano-Pérez, T. (2015). Data association for semantic world modeling from partial views. *Int. J. Robot. Res.* 34, 1064–1082. doi: 10.1177/0278364914559754

Wu, Y., Lim, J., and Yang, M.-H. (2013). "Online object tracking: a benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Columbus, OH), 2411–2418. doi: 10.1109/CVPR.2013.312

Xie, C., Xiang, Y., Mousavian, A., and Fox, D. (2019). "The best of both modes: separately leveraging RGB and depth for unseen object instance segmentation," in *Conference on Robot Learning (CoRL)* (Osaka).

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.