Semantic Relational Object Tracking

Andreas Persson, Pedro Zuidberg Dos Martires, Amy Loutfi, and Luc De Raedt

arXiv:1902.09937v1 [cs.RO] 26 Feb 2019

***Abstract*—This paper addresses the topic of semantic world modeling by conjoining probabilistic reasoning and object anchoring. The proposed approach uses a so-called bottom-up object anchoring method that relies on the rich continuous data from perceptual sensor data. A novel anchoring matching function method learns to maintain object entities in space and time and is validated using a large set of trained humanly annotated ground truth data of real-world objects. For more complex scenarios, a high-level probabilistic object tracker has been integrated with the anchoring framework and handles the tracking of occluded objects via reasoning about the state of unobserved objects. We demonstrate the performance of our integrated approach through scenarios such as the shell game scenario, where we illustrate how anchored objects are retained by preserving relations through probabilistic reasoning.**

***Index Terms*—Semantic World Modeling, Perceptual Anchoring, Probabilistic Reasoning, Probabilistic Logic Programming, Object Tracking, Relational Particle Filtering**

I. Introduction

CONSIDER the classical shell game where a ball is hidden under one of three identical cups. The performer of the game rapidly moves the cups and the task of the observer is to follow the movement of the cups and to identify under which cup the ball is located. For an observer to successfully identify the right cup, he/she must successfully handle a number of subtasks. First, despite that each of the cups are visually similar, the observer must create an individual notion of each cup as a unique object so that it can be identified (e.g., "the cup in the middle"). Likewise, the observer must recognize the ball as a unique object. Secondly, even though the ball is hidden under one of the cups, the observer makes the assumption that although the ball is not perceived it should still be present under the cup. Third, as the performer rapidly moves the cups, the observer should track the cup under which the ball is hidden. And finally, the observer also needs to realize that cups can contain balls, and therefore as the cup moves, so does the ball. Depending on the level of skill of the performer (and perhaps some additional tricks) the shell game can be a difficult one to solve.

A. Persson and A. Loutfi are with the Center for Applied Autonomous Sensor Systems (AASS), Dept. of Science and Technology, Örebro University, 701 82 Örebro, Sweden.

P. Zuidberg Dos Martires and L. De Raedt are with Declaratieve Talen en Artificiele Intelligentie (DTAI), Department of Computer Science, KU Leuven, 3001 Heverlee, Belgium.

Manuscript ... July 15, 2018

978-1-5386-5541-2/18/$31.00 ©2019 IEEE

Imagine, how could an autonomous agent handle this task as the observer? For this to be achieved, autonomous agents in real-world scenarios need to maintain consonance between the perceived world (through sensory capabilities) and their internal representation of the world. One way to contend with this challenge is to create a *semantic world model* (i.e., a semantic object centered model of the world). As discussed by [1], a semantic world model is a way to represent objects not only by their numeric properties such as a position vector, but also by semantic properties which share a meaning with humans. Moreover, such an internal model of objects could also be used to obtain additional facts about objects and their affordances (e.g., cups can contain balls).

In this paper, we present an approach to create a semantic world model where object entities are maintained and tracked over time. We further integrate a probabilistic reasoning component which is able to support the tracking of objects in case of occlusions due to limitation in perception or due to interactions with other objects. Our approach is based on perceptual anchoring which, traditionally, has been considered as a special case of *symbol grounding* [2], used within the mobile robotics community. Perceptual anchoring, by definition, handles the problem to create and maintain, in time and space, the correspondence between symbols and sensor data that refer to the same physical object in the external world [3]. This problem has, subsequently, been defined as the *anchoring problem* [4]. We use *bottom-up* anchoring [5], whereby anchors (object representations) can be created by perceptual observations derived from interactions with the environment. For a practicable bottom-up anchoring system it is, however, essential to have a robust anchoring matching function that accurately matches perceptual observations against the perceptual data of previously maintained anchors. For this purpose, we introduce a novel method that replaces a traditionally hand-coded anchoring matching function by a learned model (c.f. [6]).

Furthermore, we integrate our bottom-up anchoring framework with a probabilistic reasoning system, Dynamic Distributional Clauses (DDC)[7], to facilitate reasoning about object entities at a symbolic level. DDC is an extension of the probabilistic logic programming language ProbLog [8], which can handle continuous random variables in addition to discrete ones. This predestines DDC to be utilized for reasoning within robotics, where the world is inherently continuous and uncertain. Integrating a reasoning system into the anchoring framework allows us to dynamically feed back information to the anchoring system and update the retained semantic world model with information stemming from our probabilistic model of the world. The novelty of our approach is that we encode the affordance of objects at a high level of abstraction. This allows us to not only track objects but

to also represent the interaction between objects. A standard object tracker could handle occlusions, but in cases where an object is first occluded but then moved as it has been encapsulated by another object, a standard tracker would fail. Particularly, if this type of object interaction would occur over an extended period of time. Therefore, we believe that by eventually learning affordances, we could further automate this reasoning process of objects. For now, we do not learn the affordance but, instead, encode relations in the probabilistic reasoner.

The remainder of the paper is structured as follows. In Section II we introduce previously presented works relevant to object anchoring in relation to probabilistic tracking, data association and reasoning. In Section III, we state the general background of used approaches and give an overview of both object anchoring and Dynamic Distributional Clauses, a probabilistic logic programming language. Our novel framework for improving object anchoring and tracking through the integration of probabilistic reasoning into object anchoring is, subsequently, presented in Section IV. In Section V, we present our result of learning the anchoring matching function, utilizing human annotated data and machine learning techniques. Moreover, we introduce the proof-of-concept of how the use of probabilistic reasoning and object tracking is employed in order to preserve a consistent world model. Finally, we conclude the paper with a summary and a discussion regarding possible directions of future work, presented in Section VI.

The work presented in this paper has been carried out as part of a larger project titled ReGROUND[1] [9], which strives towards the greater ambition of using relational symbol grounding for the purpose of affordance learning in robotics. More specifically, the ReGROUND project hypothesizes that *the grounding process should consider the full context of the environment*, which consists of multiple objects as well as their relationships and properties, and how the state of these objects changes through actions and over time. The framework that we introduce in this paper (presented in Section IV), constitutes one of the corner pillars of this project – namely, the groundwork that provides the basic data about the objects and their properties and relations.

II. Related Work

The importance of data association and object tracking in relation to perceptual anchoring was widely discussed by LeBlanc in his thesis on the topic of cooperative anchoring [10]. Around the same time, and as an alternative to traditional anchoring, early work on perceptual and probabilistic anchoring was presented by Blodow et al. in [11]. The history of objects was maintained as computationally complex scene instances and the approach was, therefore, mainly intended for solving the problem of anchoring and maintaining coherent instances of objects in object kidnapping scenarios, i.e., when an object disappears from the scene and later reappears in a different location.

[1]http://reground.cs.kuleuven.be/

The idea of probabilistic anchoring was subsequently further explored by Elfring et al., which introduced probabilistic multiple hypothesis anchoring [1]. This approach utilizes Multiple Hypothesis Tracking-based data association (MHT)[12], in order to maintain changes in anchored objects, and thus, maintain an adaptable world model. In similarity with their work, we acknowledge that a proper data association is important for object anchoring, and we support the requirements identified by the authors for a changing and adaptable world modeling algorithm, which are formulated to include: 1) *appropriate anchoring*, 2) *data association*, 3) *model-based object tracking*, and 4) *real-time execution*. Contrary to the work of Elfring et al., however, we address the tasks of appropriate anchoring and data association in a holistic fashion by introducing a learned anchoring matching function that administers both tasks. Moreover, in contrast to the work presented in [1], we do not encourage a highly integrated approach that supports a tight coupling between object anchoring/probabilistic data association and object tracking. Instead, our approach maintains a loose coupling, which is motivated by the fact that a MHT procedure will inevitably suffer from the *curse of dimensionality* [13]. A purely *probabilistic anchoring* approach, as presented in [1], will, therefore, further propagate the curse of dimensionality into the concept of anchoring.

The limitation in the use of MHT for world modeling has also been acknowledged in a recent publication on the topic of data association for semantic world modeling [14]. While this work inherits the same problem formulation, it substantially differs in approach. The authors discuss and exemplify issues related to the use of a tracking-based approach for world modeling, such as intractable branching of the tree-structured tracks of possible hypothesis, and instead, suggests a *clustering* approach based on Markov Chain Monte Carlo Data Association (MCMCDA)[15]. In the same work on data association for semantic world modeling, Wong et al. also pointed out some characteristics that differentiate world modeling from target tracking. For example, an appropriate world modeling algorithm should take into consideration that the state of most objects do not change between frames, while a tracking algorithm must consider unchanged objects as possible valid targets. For the approach presented in this work, we are assuming similar characteristics through high-level tracking of objects that are not directly perceived by the sensory input data.

From the relational point of view, which enables us to carry out reasoning, some research has been conducted on utilizing relations to improve the tracking of real world entities and state estimation ([16, 17, 18, 19]). The most expressive of theses approaches is by Nitti et al. in [19]. They utilize a relational particle filter, expressed in DDC, to carry out the tracking of objects and handle occlusions. In a box packaging scenario, where boxes are placed inside each other, they showed that binary predicates like `inside/2` are helpful when tracking objects that are not directly observable. However, Nitti et al. assumed the data association problem to be solved by identifying the objects (boxes) by augmented reality (AR) tags – hence, very strictly limiting the usage of their framework in real-world scenarios.

III. Background

In this section, we outline the general background of our used methods. We will both present the traditional definition found within the concept of *perceptual anchoring*, as well as an overview of *Dynamic Distributional Clauses*. We will further outline the problem domain of existing definitions and methods that address the problem of object anchoring. The same outline provides the motivation for our suggested approach, presented in subsequent Section IV.

A. Background on Perceptual Anchoring

Perceptual anchoring, originally presented by Coradeschi and Saffiotti in [3], has been defined in an attempt to address a subset of the symbol grounding applied to robotics and intelligent systems. The notion of perceptual anchoring has undergone several extensions and refinements since its first definition. Some notable works in this regard include the addition of bottom-up anchoring [5], extensions for multi-agent systems [20], considerations for non-traditional sensing modalities and knowledge based anchoring given full scale knowledge representation and reasoning systems [21, 22, 23], and integration of large-scale knowledge bases together with of common-sense reasoning in distributed anchoring [24]. All those works have been presented with a common ingredient of a number of prerequisite components from [3], including: a *symbolic system* (including: a set $\mathcal{X} = \{x_1, x_2, \dots\}$ of individual symbols (variables and constants); 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), and *predicate grounding relation* $g \subseteq \mathcal{P} \times \Phi \times D(\Phi)$, which encodes the correspondence between (unary) predicates and values of measurable attributes. The relation g maps a certain predicate to compatible attribute values. An overview of the anchoring components and their relations is exemplified in Example 1.

Example 1: Consider the captured camera image with segmented image regions, seen in Figure 1. Each segmented region corresponds to an individual *percept* captured by the *perceptual system*, see Figure 1 – № 1. We denote the percepts π_1, π_2 and π_3, which corresponds to observed physical objects *banana*, *apple* and *mug*, respectively. For each percept is, subsequently, a number of *attributes* measured, e.g., *color*, *size*, etc. One such attribute is a *color attribute* measured as a normalized color histogram over the masked area of *percept* π_2, illustrated in Figure 1 – № 2. For clarity, we denote the measured *color attribute* as attribute ϕ_2^{color}, which have values in a *domain* that is equal to the n number of histogram bins. Finally, the *predicate grounding relation* g, illustrated in Figure 1 – № 3, for the aforementioned *color attribute* can be seen as the encoded correspondence between specific peeks in the color histogram and certain *predicate symbols*, e.g.:

$$g(\texttt{'red'}, color, \arg\max_{i=1\dots n}(\phi_{2,i}^{color})) \textit{ iff } i = 6$$

An anchor is, consequently, an internal data structure α_t^x, indexed by time t and identified by a unique individual symbol x (e.g., `'coffee-mug-1'`, `'box-4'`, etc.), which encapsulates and maintains the correspondences between percepts and symbols that refer to the same physical object. Following the definition presented by [5], the principle 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^x. This functionality defines a structure α_t^x, index 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^x from time $t - k$ to time t. This functionality assures that the percepts pointed to by the anchor are the most recent and adequate perceptual representation of the object.

Given the defined functionalities above, it is evident that an initial matching function is essential to determine if a candidate object is matching an existing anchor or not. In previously reported work on perceptual anchoring, the problem of matching anchors has mostly been addressed through a simplified approach based on the use of symbolic values (or left out entirely), and where the predicate grounding relation mapping between symbolic predicate values and measured

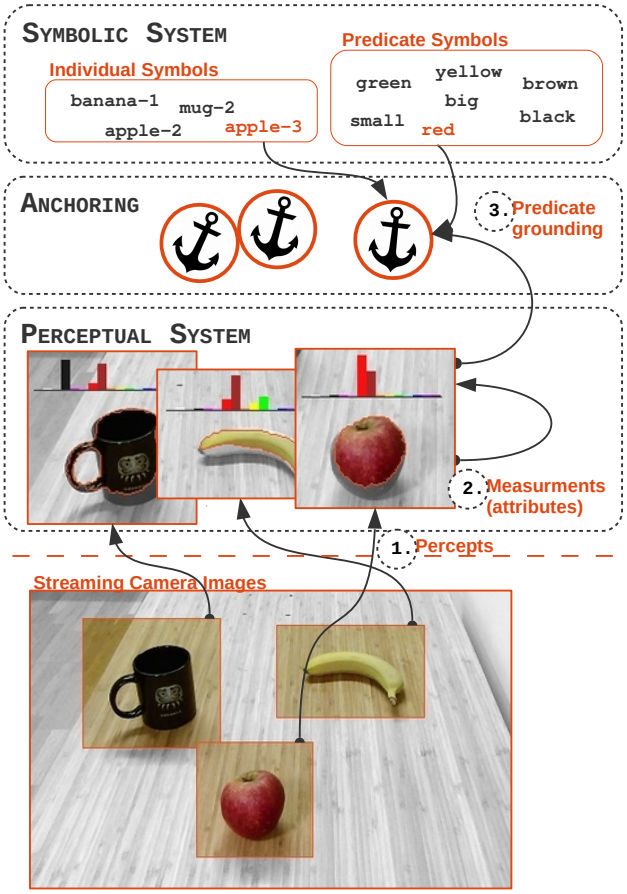

Fig. 1. A graphical illustration of the anchoring components and their interconnections.

attribute values are commonly facilitated by the use of conceptual spaces [25]. Conceptual spaces can be thought of as a discretization of the continuous perceptual attribute space, which consequently also result in a loss of information. The procedure of creating and maintaining anchors, based on the discretized symbolic values, must accordingly either be handled by a probabilistic system, as in the case of [1], or through the use of additional knowledge and the use of a reasoning system, as in the case of [24]. In this paper, we move the matching function down to the perceptual level by presenting a novel matching approach that facilitates the rich information found within the measured attribute values, further presented in Section IV-C.

Moving the anchoring matching function to the perceptual level will, inevitably, also introduce another level of complexity since measured attributes must be compared based on continuous attribute values. The system must, subsequently, both recognize previously observed objects and detect (or anchor) new previously unknown objects based on the result of the initial matching function. In the case of open world scenarios without a fixed number of possible objects that the system might encounter, this is undoubtedly a challenging issue. In this paper, we address this issue and present an evaluation that addresses the problem of learning how to determine if an object has previously been perceived (or not), in the context of bottom-up perceptual anchoring, presented in Section V-A.

Traditionally ([3, 4]), there has also been a third anchoring functionality; a *track* functionality[2]. This track functionality has recently been revised and explored in the interest of integrating object tracking into the concept of anchoring, introduced in [6], which suggested a tracking functionality highly integrated with a point cloud based particle filter object tracking approach on the initial perceptual sensor level [26]. However, the performance of the previously suggested framework was, consequently, affected by the computational load of the used object tracking approach, which requires tracking of computational demanding 3-D point cloud data. In this paper, we further explore the integration between object tracking and perceptual anchoring, presented in Section IV-D. This integration is based upon the belief that the integration should be loosely coupled in order to sustain the benefits of both the ability to maintaining individual instances on objects in a larger scale, as in the case of perceptual anchoring, as well as efficiently and logically track object instances over time, as in the case of probabilistic reasoning. In particular, we utilize *Dynamic Distributed Clauses* in order to track objects on a higher conceptual level and, hence, also enable reasoning about the objects perceived on the anchoring level, as presented in Section V-B.

B. Overview of Dynamic Distributional Clauses

Dynamic Distributional Clauses is an extension of the logic programming language Prolog [27] that is capable of handling discrete and continuous random variables at the same time. Programs written in DDC allow for high-level (discrete variables) reasoning with low-level sensor input (continuous variables).

In logic programming, reasoning happens through the usage of symbols. These are either terms or predicates. The latter are often referred to as relations. Terms can be constants, logical variables or n-ary functors applied to an n-tuple of terms. Constants can only have one assigned value to them, which means that only one interpretation is possible. This is in contrast to logical variables, which have multiple interpretations. More concretely, a logical variable `X` is a variable ranging over the set of all possible ground terms. Lastly, we also have terms of the form $\mathtt{p}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ with $\mathtt{p}/n$ being an n-ary predicate and all $\mathtt{t}_i$'s being terms themselves. This last kind of terms are dubbed atoms.

In the static case, i.e., when there is no explicit dependency on time in the terms, DDC programs reduce to *Distributional Clauses* (DC) [28, 29] programs. A distributional clause is of the form $\mathtt{h} \sim \mathcal{D} \leftarrow \mathtt{b}_1, \ldots, \mathtt{b}_n$, where $\sim$ is a predicate in infix notation and $\mathtt{b}_i$'s are literals, i.e., atoms or their negation. $\mathtt{h}$ is the name of a random variable and $\mathcal{D}$ tells us how the random variable is distributed – both are formally terms. The meaning of such a clause is that each grounded instance of a clause $(\mathtt{h} \sim \mathcal{D} \leftarrow \mathtt{b}_1, \ldots, \mathtt{b}_n)\theta$ defines a random variable $\mathtt{h}\theta$ that is distributed according to $\mathcal{D}\theta$. A grounding substitution $\theta = \{\mathtt{V}_1/\mathtt{t}_1, \ldots, \mathtt{V}_n/\mathtt{t}_n\}$ is a transformation that simultaneously substitutes all logic variables $\mathtt{V}_i$ in a distributional clause with non-variable terms $\mathtt{t}_i$. Here we see that random variables and distributions are themselves not necessarily grounded by definition. The mean of a normal distribution can for instance depend on random variables. For the atom $\mathtt{h}\theta$ to be defined it is necessary that all atoms $\mathtt{b}_i\theta$ in the distributional clause evaluate to true. Labeling a distributional clause with time indexes allows for declaring dynamic models via defining a transition model from time step $\mathtt{t}$ to time step $\mathtt{t}+1$.

Inference in DC is conducted through importance sampling combined with backward reasoning, likelihood weighting and Rao-Blackwellization [7], while filtering in the dynamic case is carried out through particle filtering [29].

To sum up, DDC is a template language that defines conditional probabilities for discrete and continuous random variables: $p(\mathtt{h}\theta|\mathtt{b}_1\theta, \ldots, \mathtt{b}_n\theta) = \mathcal{D}\theta$.

IV. Novel Method: Combined Object Anchoring and Tracking Framework

Our suggested combined framework architecture, seen in Figure 2, is a modularized architecture that utilizes libraries and communication protocols available in the Robot Operating System (ROS)[3]. Each module/sub-system (described in details below) has a dedicated task, while the overall goal of the combined framework is to create and maintain coherent and accurate representations (anchors) of perceived real-world objects. The same anchor representations also provides the historical background information, i.e., information about objects last perceived at time $t-k_x$, which is used by the anchoring system

[2]The *track* functionality was formally integrated with the *re-acquire* functionality for the extension to sensor-driven bottom-up anchoring [5], such that no distinction was made between extending the definition for an anchor from time $t-1$ or $t-k$.

[3]http://www.ros.org/

to process and anchor objects perceived at the present time t, illustrated in Figure 2 – № 2. Furthermore, the framework utilizes an inference system, illustrated in Figure 2 – № 3, which aids the anchor system in complex dynamic scenes. Finally, the architecture is a sensor-driven architecture that is triggered by perceptual data, i.e. sensor readings, which initially are pre-processed by a perceptual pipeline, illustrated in Figure 2 – № 1.

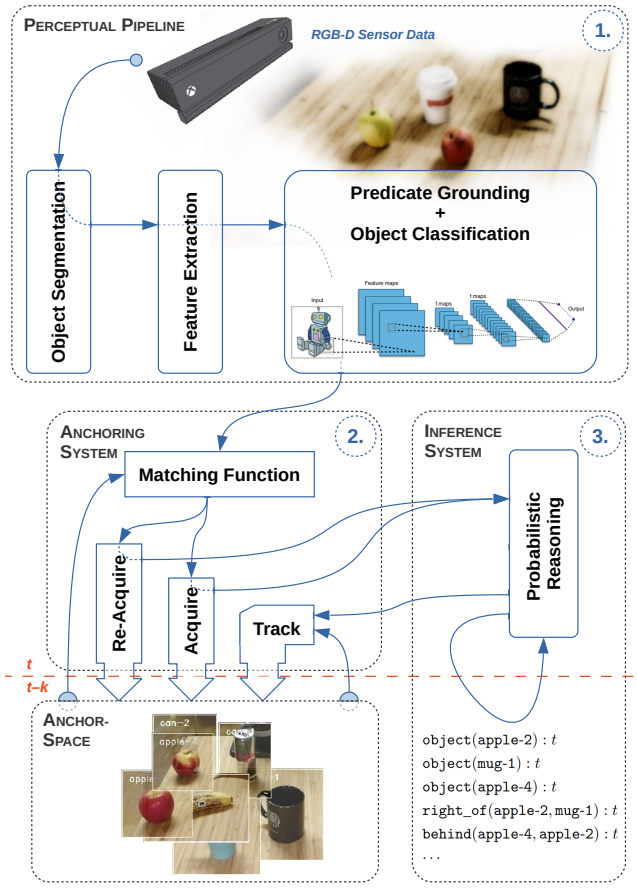

Fig. 2. Overview of our combined framework architecture. The overall framework is modularized architecture that consist of three core sub-systems (each described in further details in this section): *1)* an initial *perceptual pre-processing pipeline* with the purpose of detecting, segmenting and processing perceived objects in the physical world, *2)* an *anchoring system* with the purpose of creating and maintaining updated and consistent representations (anchors) of perceived objects, and *3)* an *inference system* with the purpose aiding the anchor system and track objects in complex dynamic scenarios.

A. Implementation Details: Pre-processing Pipeline

The overall background *pre-processing pipeline*, with the goal of detecting and segmenting objects, extract features from detected objects, classifying and symbolically grounding each object instances, is illustrated in Figure 2 – № 1. For this purpose, our system setup relies upon publicly available core libraries and systems, including: the Point Cloud Library[4] (PCL), the Open Computer Vision library[5] (OpenCV), and the Robot Operating System (ROS). It should also be noted, all methods and techniques covered in this section are considered to be replaceable *black-box* approaches that are used for the means of providing background for the subsequent theoretical Section IV-B. For example, the used object segmentation method could be replaced with a convolutional network-based semantic segmentation approach [30]. This requires, however, an adequate dataset of pixel-wise mask annotations for training the network to detect the objects of interest, which is something that is not always publically available and must, therefore, further be addressed, e.g. through weakly supervised learning [31]. Nevertheless, the details on used techniques, for the presented architecture, are here covered for completeness and reproducibility.

More specifically, the initial step of our pre-processing pipeline is an *object segmentation* method, which is performed with the purpose of detecting arbitrary objects of interest in the scene. The deployed object segmentation method is based on organized point cloud data (i.e., the organization of point cloud data is identical to the rows and columns of the imagery data from which the point cloud originates), which are given as input data by a Kinect2 RGB-D sensor. To establish the initial connection between the ROS environment and the Kinect2 sensor has further the ROS-Kinect2 bridge [32] been integrated and used together with the presented framework architecture. Hence, the segmentation procedure can briefly be described using the following steps:

- Estimate 3-D surface normals based on integral images [33]. This function uses the algorithm for calculating average 3-D gradients over six integral images, where the horizontal and vertical 3-D gradients are used to compute the normal as the cross-product between two gradients.
- Planar segmentation based on the calculated surface normals.
- Object segmentation through clustering of the remaining points (points that are not part of the detected planar surfaces). This segmentation uses a connected component segmentation, presented by [34], where a Euclidean comparison function is used to connect the components that constitute the cloud cluster of an individual object.

Moreover, provided that object instances have been segmented based on the full spectrum of available RGB-D data (as described above), we are further able to exploit the advancements in deep learning for *image classification* in the final *object classification* procedure of our pre-processing pipeline. The Convolutional Neural Networks (CNN) architecture used in this case is based on the 1 K GoogLeNet model, developed by [35], and which originally was trained on the ILSVRC 2012 visual challenge dataset [36]. For this work, we have, however, extracted a subset of the ImageNet database [37] and fine-tuned the model for 101 objects categories that are relevant for a household domain, e.g., `'mug'`, `'spoon'`, `'banana'`, `'tomato'`, etc., where the model was trained for a *top-1* accuracy of 73.4% (and a *top-5* accuracy of 92.0%).

B. Theoretical Aspects: Precepts, Attributes and Symbols

The resulting output of the object segmentation is m point cloud clusters (where m varies between frames). For

[4]http://pointclouds.org/
[5]http://opencv.org/

consistency with the definition of anchoring, we denote segmented clusters as percepts $\{\pi_1^{cloud}, \pi_2^{cloud}, \dots \pi_m^{cloud}\}$, which each corresponds to the spatial 3-D point cloud data of an individual object. Posterior to the segmentation of the point cloud clusters, the same RGB-D data is also used for segmenting corresponding visual 2-D imagery data of each detected object. This image segmentation is entirely based on the prior point cloud clusters and a projection between the 3-D point cloud frame and the 2-D visual RGB frame of the RGB-D sensor. Also, we denote visual data as percepts $\{\pi_1^{image}, \pi_2^{image}, \dots \pi_m^{image}\}$, which each corresponds to the visual 2-D imagery data of a segmented object.

Next, both segmented perceptual 3-D point cloud data and 2-D visual data are further forwarded to a *feature extraction* procedure. The first step of this feature extraction procedure is to extract both a *position attribute* as the point at the geometrical center of each segmented percept π_y^{cloud}, and a *size attribute* as the 3-D bounding box around each percept π_y^{cloud}, where $\pi_{y|y=1,2\dots m}^{cloud} \in \{\pi_1^{cloud}, \pi_2^{cloud}, \dots \pi_m^{cloud}\}$. The extracted *position attribute* is here denoted $\phi_y^{pos} \in \mathbb{R}^3$, while the corresponding *size attribute* is denoted $\phi_y^{size} \in \mathbb{R}^3$. Furthermore, a *color attribute* ϕ_y^{color} is extracted for each visual percept π_y^{image}, which is measured as a color histogram (in the HSV color space).

Finally, extracted attributes together with the perceptual data are further forwarded to a combined *predicate grounding* and *object classification* procedure. This procedure has the purpose of both grounding and associating a symbolic value with each extracted attribute, as well as classifying each object and further associating each object with an object category label. The predicate grounder is here responsible for grounding each measured attribute ϕ_y (of the set Φ_y, that originates from the same physical object) to a predicate grounding symbol p_y. For example, a certain peek in a color histogram, measured as a ϕ_y^{color} attribute, is grounded to the symbol `'red'`, such that p_y^{color}=‘red’ (cf. Example 1). In the context of anchoring, we further assume that all trained object categories of used GoogLeNet model are part of the set of possible predicate symbols $\mathcal{P}$. The inputs for the object classification procedure are, subsequently, the segmented visual percepts π_y^{image}, while resulting object categories together with predicted category probabilities are denoted by $p_y^{class} \in \mathcal{P}$ and ϕ_y^{class}, respectively.

C. Anchoring Management

The entry point for the *anchoring system*, seen in Figure 2 – № 2, is a *matching function*. This function assumes a bottom-up approach to perceptual anchoring, described in [5], where the system constantly receives candidate anchors and invokes a number of different matching algorithms (one matching algorithm for each measured attribute in the set $\Phi_y = \{\phi_y^{class}, \phi_y^{color}, \phi_y^{size,} \phi_y^{pos}\}$) in order to determine if an anchor, α_x, has previous been perceived or not.

1) Matching Function: More specifically, an unknown set of attributes Φ_y is compared against the set of attributes Φ_x of an existing anchor α^x. The combined result of all individual invoked matching algorithm determines, subsequently, if an anchored object has previously been perceived or not. In details, a classification attribute ϕ_y^{class} and symbol p_y^{class} of a candidate object is firstly compared against the classification attribute and symbol of a previously stored anchor according to:

$$d_{x,y}^{class}(\phi_x^{class}, \phi_y^{class}) = \begin{cases} \exp\left(-\frac{|\phi_x^{class} - \phi_y^{class}|}{\phi_x^{class} + \phi_y^{class}}\right) & \textbf{if } p_x^{class} \equiv p_y^{class} \\ 0 & \textbf{else} \end{cases} \quad (1)$$

We interpret the d^{class} as the exponentially decaying relative L^1-distance between the two attribute values ϕ_x^{class} and ϕ_y^{class}. This means that we exponentially penalize the distance between two objects in the class attribute space.

Secondly, the color histogram of a color attribute ϕ_y^{color} of a candidate object is compared (assuming normalized color histograms) according to the *color correlation*:

$$d_{x,y}^{color}(\phi_x^{color}, \phi_y^{color}) = \frac{1}{2}\left(1 + \frac{\sum_{i=1}^{n}(\phi_{x,i}^{color} - \mu_x)(\phi_{y,i}^{color} - \mu_y)}{\sqrt{\sum_{i=1}^{n}(\phi_{x,i}^{color} - \mu_x)^2 \sum_{i=1}^{n}(\phi_{y,i}^{color} - \mu_y)^2}}\right) \quad (2)$$

Where n is the number of histogram bins and the index i gives the i-th histogram bin value of ϕ_x^{color} and ϕ_y^{color}, respectively. μ_x and μ_y are the *color mean value* of each histogram, given according to:

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n}\phi_{x,i}^{color}, \mu_y = \frac{1}{n}\sum_{i=1}^{n}\phi_{y,i}^{color}$$

Next, the position attribute ϕ_y^{pos}, and the (last updated) position ϕ_x^{pos} of a previously stored anchor α^x, is calculated according to the L^2-distance (in 3-D spatial space). Inspired by the work presented by [11], this distance is then mapped to a *normalized similarity distance* according to:

$$d_{x,y}^{pos}(\phi_y^{pos}, \phi_x^{pos}) = e^{-L^2(\phi_y^{pos}, \phi_x^{pos})} \quad (3)$$

Furthermore, the size attribute ϕ_y^{size} of a candidate object is compared according to the *generalized Jaccard similarity* (for the bounding boxes in 3-D space):

$$d_{x,y}^{size}(\phi_x^{size}, \phi_y^{size}) = \frac{\sum_{i=1}^{3} min(\phi_{x,i}^{size}, \phi_{y,i}^{size})}{\sum_{i=1}^{3} max(\phi_{x,i}^{size}, \phi_{y,i}^{size})} \quad (4)$$

Motivated by the importance of the time t within the concept of anchoring, the difference in time since last recorded observation of a previously stored anchor α^x, defined at $t - k_x$, is finally mapped to a similar normalized distance according to:

$$d_{x,y}^{time}(t, t - k_x) = \frac{2}{1 + e^{t-(t-k_x)}} = \frac{2}{1 + e^{k_x}} \quad (5)$$

Consequently, all given matching distance values, Eqs. (1) to (5), are given in the interval $[0.0, 1.0]$, and all distance values are, therefore, also commensurable.

2) Creating and Maintaining Anchors: Combining all matching distance values, given by Eqs. (1) to (5), and determining whether a candidate anchor has previously been perceived or not is a non-trivial task. Especially in the context of *bottom-up* anchoring with both continuous possibilities of objects, and continuous similarity distances given by the initial *matching function*. The matching distance values can be combined in many different ways, e.g.: through a *min* or *max* function, by the *weighted average* with different weights, etc. Nonetheless, a threshold value is ultimately required in order to determine if the combined result is to be considered as a match (or not). In this paper, we will shed some light upon this issue and present our work on the topic of learning the anchoring matching function, which determines if an object has previously been observed or not.

At this point we would like to stress that the architecture of the anchoring system is completely agnostic towards how the matching function was learned. This means that the anchoring system considers the matching function to be an exchangeable *black-box* approximation of the true anchor-percept matching. In Section V-A we compare different classifiers to each other that could be potentially used to approximate the matching.

Regardless of the used classification algorithm, the process of the anchoring is to ultimately create or maintain anchors through either one of the two principal functionalities: *acquire* and *re-acquire*, respectively (further described in Section III-A). The *anchor-space*, in which the anchors are maintained and stored, is in this case expressed as a *permanent world model* (PWM). We further enhance the traditional *acquire* functionality by utilizing the deep learning classifier such that a unique identifier x is further generated based on the classification symbol p^{class}, e.g., for an object classified as a `'cup'`, a corresponding unique identifier could be generated as $x =$ `'cup-4'`.

D. Integration of the Inference System

In order to prevent the curse of dimensionality from propagating from a probabilistic inference system into the perceptual anchoring system, we opt for only loosely integrating the *inference system* with the *anchoring system*. This linkage has as a consequence that we need to maintain two distinct databases for representing our belief of the world.

The above-described anchoring system database plays the role of maintaining a *permanent world model* (PWM), remembering all objects that have appeared over time $(t - k_x)$. By contrast, the database of the inference system, implemented in DDC, operates on a *temporary world model* (TWM). The latter does, however, not only retain which objects are present in the scene but also how the single objects relate to each other. A `'cup'` might, for example, be remembered as standing at a certain point in 3-D space but also by the fact that it stands left to some other `'cup'`. This representation of the real world is obviously a lot more expensive but adds valuable information to the scene description when carrying out high-level object tracking. The relational nature of the TWM enables us to reason about the world and additionally to track objects on a high level.

By working with two distinct databases we take advantage of the databases for specialized tasks, e.g., reasoning with a relational database. However, it also means that we need to maintain two distinct databases, and more importantly, the databases have to reflect the same world.

Therefore, in order to retain the integrated framework in a coherent state, we need to place the loosely coupled inference system and anchoring system in a tight feedback loop. This feedback loop is represented by the incoming and outgoing arrows in panel № 3, Figure 2. It hence consists of two distinct steps:

1) Sending anchor information from the *anchoring system* to the *inference system* and initiating or updating the belief of the world in the *inference system*.
2) Sending back the updated belief of the world in the *inference system* and update the belief of the world in the *anchoring system*, accordingly.

The initial belief in the TWM is initiated by the belief of the PWM of the scene. The anchor information, originating from the PWM, is treated as observations in the TWM. For each initial observation, DDC clauses are added to the TWM database, which constitutes the temporary internal representation of the world. For a `'cup'` in the scene, for example, a rule is added that describes the initial belief of its position and velocity:

$$\texttt{pos(cup)}_0 \sim \texttt{gaussian}(\vec{\mathrm{R}}, \vec{0}, \vec{\Sigma}) \leftarrow \texttt{obs(percept_pos(cup))}_0 \sim\!= \vec{\mathrm{R}}. \tag{6}$$

Where $\vec{\mathrm{X}}$ is the observed 3-D position of the geometric center of the percept (the `'cup'` in this case). $\vec{\Sigma}$ is the covariance matrix that specifies the Gaussian and the $\vec{0}$ corresponds to the initial velocity which is set to 0 in each dimension.

For all the following time steps we define an observation model that takes into account uncertainty in the measurement process itself. We adopt the approach of [19] of expressing the measurement model as the product of Gaussian densities around the position of each object. Assuming independently and identically distributed measurements of the objects allows for this factorization. This idealization assumes that observing an object does not depend on the observation of any other object.

$$\texttt{obs(percept_pos(cup))}_{t+1} \sim \texttt{gaussian}(\vec{\mathrm{R}}, \Sigma_{obs}) \leftarrow \texttt{pos(cup)}_t \sim\!= (\vec{\mathrm{R}}, _). \tag{7}$$

In the case that all objects in the scene are observed, i.e., none of the objects is occluded by another one, the TWM and PWM are now in a state of cognitive consonance, as we used the anchor information as observations for the inference system. If, however, objects get occluded due to manipulations of the world, the occluded objects do not produce any perceptual data anymore. Hence, the perceptual anchoring system can no longer update its belief of the world, and no updated belief is sent to the inference system. In this case, the inference

system needs to reason about what might have happened to the object that is not perceived anymore by the anchoring system. Considering the world at time step $t-1$, and the observations of the world at time step t, we can speculate about the world in time step t. We infer the state of an occluded object through its relations with perceived objects in the world. This inferred updated belief of the world is then sent back to the anchoring system where the state of occluded objects is also updated.

This approach allows us to propose a modified high-level anchoring *track* functionality (cf. [6]), such that:

- *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 assures that the percepts pointed to by the anchor are the adequate perceptual representation of the object, even though the object is currently not perceived.

With the introduction of the anchoring *track* functionality, we also need to ensure cognitive consonance at the anchoring side by updating the PWM based on the updated belief of the world, established by the inference system. More specifically, the 3-D *position* attribute ϕ_x^{pos} of an anchor α_t^x is updated according to the inferred position of the corresponding object maintained in the TWM of the inference system. This exchange of information between both systems, as described in this section, is further facilitated by sharing the unique identifier x of an anchored object. Hence, we are able to differentiate between specific instances of objects and we can express the rules, given by Eq. 6 and 7, for an object instance that is identified by the unique symbol, e.g., `'cup-1'` or `'cup-4'`.

The details on how the probabilistic inference is carried out are given in [7, 29]. Our contribution lies in coupling a probabilistic inference system with an anchoring system, which enables the conjoined system to probabilistically reason on low-level sensor data. This is for example not the case in [19], where they used AR-tags to observe objects in the world.

V. Evaluation and Results

Evaluating a real-world operating anchoring framework, with several interacting components as described in Section IV, is undoubtedly a challenging task. Noisy sensor readings and erroneous attribute measurements are inevitably present and will propagate through the components of the processing pipeline. The evaluation presented in this section is, therefore, limited to: *1)* the performance of the suggested anchoring matching approach (introduced in Section V-A), and *2)* the integrated combined anchoring and reasoning system (presented in Section V-B).

A. Learning the Anchoring Matching Function

The evaluation presented in this section has a two-folded purpose: *1)* collect annotated ground truth data about objects in dynamic scenarios, and *2)* learning to determine which of the the two anchoring functionalities *acquire* or *re-acquire* (cf. Section IV-C and Figure 2 – № 2), to initiate based on the matching distances given of the initial anchoring matching function (given by Eqs. (1) to (5), as described in Section IV-C1).

1) Data Collection: A benefit of using perceptual anchoring is that the percepts pointed to by the anchor are the most recent and adequate perceptual representation of an object. For the evaluation presented in this paper, we have exploited the updated and maintained representations, found in anchors, in order to collect human annotated ground truth data. This data collection was conducted through a *human-annotation interface* that was queued with segmented perceptual sensor data given by the perceptual pre-processing pipeline, presented in Section IV-A. By utilizing this interface, all data about unknown candidate objects, together with the perceptual data of possible matching anchored objects, could be presented and visualized for the human user. Hence, the human was able to provide feedback about the action that the human counterpart would consider as the appropriate anchoring action for each presented candidate object (i.e., *acquire* a new anchor for a queued object, or *re-acquire* an existing anchor). The procedure for collecting our ground truth data is described and exemplified in Figure 3.

Behind the scene of proposed human-annotation interface, exemplified in Figure 3, the data that in reality was collected and stored was matching similarity scores, provided by Eqs. (1) to (5). Together with each set of matching similarity scores (as result of comparing the attributes of an unknown candidate object against the attributes of an existing anchored object), was further an annotated label of 1 stored if the user considered an existing object as a matching object, or 0 otherwise. Worth noting is that the collected data purely represent that the human user was considering as the most appropriate action for each presented scene. Hence, we were also able to gather samples of objects in, for example, ambiguous situations where an identical (but physically different) instance of an object was introduced while the similar counterpart was still observed. Furthermore, given such ambiguous situations with a number of possible matching anchored objects that could match a selected candidate object, as depicted in Figure 3 – № 5-6, we assumed that there could only exist *one* true match (labeled 1), while the reaming candidates were non-matching candidates (labeled 0). As a result, we were able to collect several samples for each human action.

2) Experimental Evaluation: With the use of the human-annotation interface, as described in the previous section, we were able to collect a dataset of a total of 5400 samples [6]. A dataset that we, subsequently, have used for this particular evaluation in order to train the anchoring system to initiate proper anchoring functionality for different situations. During the data collection, several typical problematic anchoring scenarios, e.g., scenarios there new ambiguous objects are introduced in the scene, scenarios with partly occluded objects, scenarios where existing objects were disappearing and reappearing in the scene, etc., were executed in order to cover a broad range of different situations. Moreover, the data collection was conducted on several occasions for the purpose of capturing changes in the environmental conditions, e.g., changes in light conditions.

[6]The collected data set is available under http://reground.cs.kuleuven.be and the *human-annotation interface* is available under bitbucket.org/reground/anchoring.

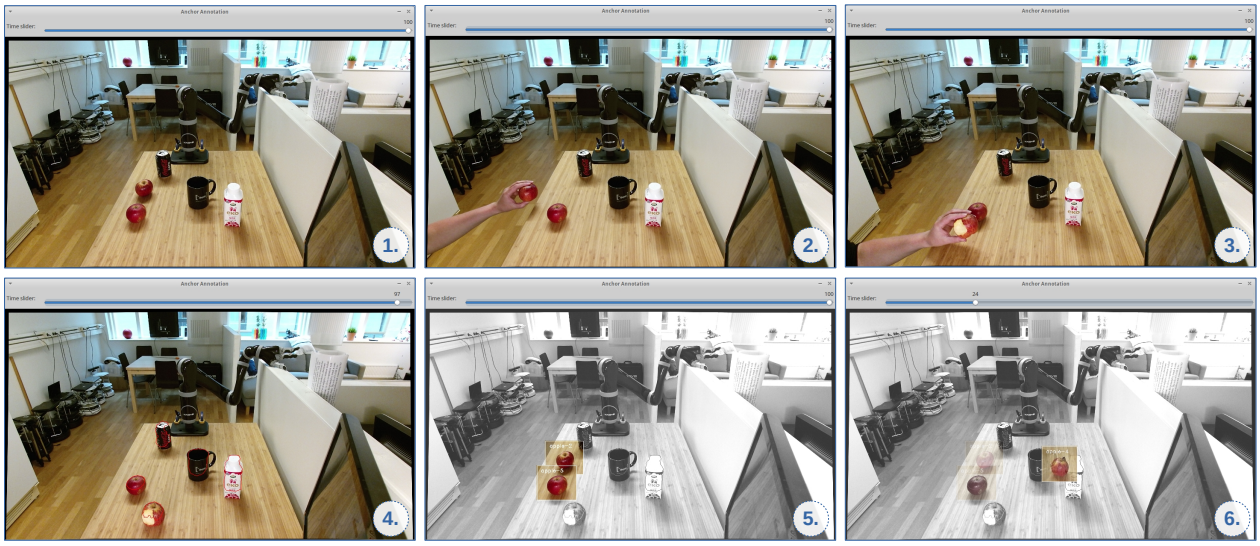

Fig. 3. A depiction of our *human-annotation interface* that was used in order to collect ground truth data of anchored objects. In conjunction with changes in the scene, as illustrated by № 1-3, the human user has the possibility to *freeze* the execution of the framework and providing feedback about what he/she would consider as the appropriate anchoring action for a candidate objects. Once the execution is frozen, the human user can select segmented candidate objects, e.g., the moved `'apple'` as illustrated in № 4, after which the framework is responding by displaying an updated representation of a number of already anchored objects, shown in № 5, which best (attribute-wise) corresponds to the selected object. The human user can then provide positive feedback about a matching anchored object (by selecting the representation of the matching anchored object), or negative feedback (simply by clicking anywhere else on the screen). Also, to covering the time aspect, and to suggest possible matching anchored objects that have not been perceived recently, we have further added a time slider, illustrated in the top part of № 6. Through this time slider can the user adjust the time factor k for the purpose of selecting the best matching candidate object that was last observed at a time $t - k$.

Given the collected data, which was comprised of sets of similarity scores together with corresponding labels (with a label of 1 for a matching set of similarity scores, or a label of 0 for a non-matching set), our approach for learning how to correctly anchoring objects, and thereby learn to invoke correct anchoring functionality (*acquire* or *re-acquire*), was through the evaluation of different classification algorithms. More specifically, for this evaluation we have tested and trained following classification algorithms:

- Support Vector Machine (SVM)[38], with ν-Support Vectors (trained with $\nu = 0.1$), and with a *Histogram intersection* kernel function
- Multi Layer Perceptron (MLP), with *back-propagation* training, two hidden layers and a layer configuration, according to: $x - 10 - 15 - 2$
- k-Nearest Neighbor (k-NN), trained and tested with $k = 3$
- Normal Bayes Classifier (Bayes)[39]

Collected dataset was randomly divided $70/30$ into training/test samples, giving us a total of 3780 *training* samples and 1620 *testing* samples. Resulting average classification accuracy of each trained approach is listed in Figure 4.

Given the result, presented in Figure 4, it is seen that the best *average classification accuracy* of 96.4% was achieved by the use of the SVM classifier. The highest average *F1 score* (for a *true match*) of 94.4% was, likewise, achieved with the same SVM classifier. By the results seen in Figure 4, it should, however, also be noted that the differences in accuracy between the MLP classifier and the SVM classifier are close to insignificant (only 0.2%). Nevertheless, the best trained resulting SVM model was formally integrated as a part of the initial *matching*

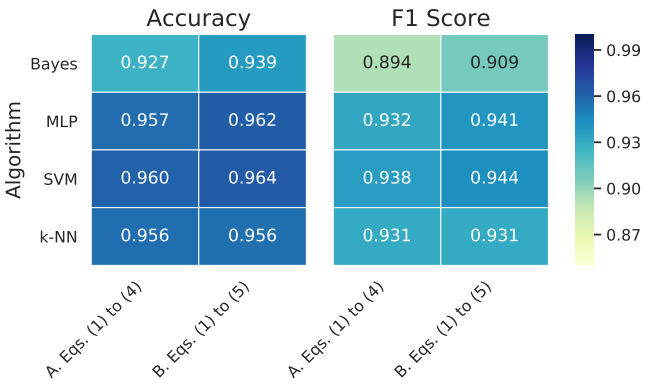

Fig. 4. Resulting average *classification accuracy* together with *F1 score* for each used model for our approach to learn the anchoring functionalities.

function of the anchoring system such that the predicted result of the SVM model was used to determine if an unknown candidate object was matching an existing anchor (i.e., if the object should be *re-acquired* as an existing matching anchor), or if no current anchors were matching the candidate object (i.e., if a new anchor should be *acquired* for the object). Integrated classification approach was, subsequently, used for the remaining experiments presented in this paper.

By comparing the results between omitting (*column A*) or considering (*column B*) the time difference as an additional feature, mapped to a time distance according to Eq. (5), it is also evident that the time t, in fact, is a relevant factor for the concept of anchoring. The intuition behind including this additional time feature in the evaluations presented in

this section was to capture time-dependent changes in the environment while learning how to anchor objects, e.g., the position of an object can only change with a limited velocity between sequential frames. Through examining the results for the best resulting SVM models, it is seen that our intuition was correct and that we achieved 0.4% better classification accuracy and 0.6% better F1 score, as a result of including the time difference as a feature. Worth noted, it can further be seen by the results in Figure 4, that the k-NN classifier was the only classifier that did not benefit from increasing the dimensionality of the input data by including the time difference as a feature.

Finally, it should be noted that the integrated classification approach can, in some cases, returning several matching candidate anchors (i.e., a candidate object can be *re-acquired* as more than one existing matching anchor). It is, therefore, important to globally consider all possible candidates for all observed objects in each frame in order to determine the best matching candidate anchor for each observed object. For the work presented in this paper, we used an SVM classifier with continuous output values such that the globally best matching candidate anchor was determined in a *winner takes all* manner.

B. Tracking of Occluded Objects

Despite the accuracy of the anchoring system, presented in previously Section V-A, there are scenarios where the pure anchoring system fails to correctly *acquire* (or *re-acquire*) an object, e.g., when an object is occluded and moved by another object. For a changing and adaptable system to handle the world modeling of such scenarios, the system must further incorporate *model-based object tracking* in order to maintain objects that are not perceived by the input sensors. In this section, we will exemplify how our approach of integrating DDC into the anchoring system (cf. Section IV-D and Figure 2 – № 2&3) can handle such scenarios with occluded objects, and as a subsequent result further improve the anchoring accuracy.

1) Proof of Concept: To demonstrate how the anchoring system benefits from the feedback of the inference system (and consequently how the inference system benefits from the same integration), we plot the particles in form of point positions (representing the belief of the world in the inference system, as described in Section IV-D), concurrently with the output of the anchoring system, as exemplified in Figure 5. The mean position in 3-D space of the particles for each object that was not directly perceived at time t, e.g., objects occluded by another object, was subsequently fed back to the reinstated *track* functionality of the *anchoring system* such that the position of an anchor (even though the anchored object was not observed), was updated to the most probable position according to the *inference system*.

Comparing the resulting anchored objects, seen in Figure 5, it is evident that there is a significant difference in resulting anchors. In the case where only the *anchoring system* was used (Figure 5 – *2nd row from top*), it can be seen that the initial `'apple-1'` object (seen in Figure 5 – № 1) is lost while the object is occluded and moved by the `'skin-1'` object. Consequently, when the apple object reappears in the scene, the anchoring system cannot determine if the object is a new `'apple'` or the previously anchored `'apple-1'`, and as a result *acquire* an new anchor `'apple-3'` (seen in Figure 5 – № 2). However, in the case where both the *anchoring system* and the *inference system* are used (Figure 5 – *bottom row*), and where the position of the *tracked* `'apple-1'` object (seen in Figure 5 – № 3) is fed back to anchoring system while the object is moved, it can seen that the apple object instead is correctly *re-acquired* as `'apple-1'` once the object reappears in the scene (seen in Figure 5 – № 4). Note that rather than using a dedicated classifier for recognizing different human body parts, we have, instead, fine-tuned our GoogLeNet model to recognize human `'skin'` objects as one of the object categories.

2) Exemplifying Scenarios: Given the exemplified proof-of-concept (presented in the previous Section V-B1), we will in this section further demonstrate a number of scenarios where our suggested integrated system excels (compared to an anchoring approach that exclusively is based on perceptual observations of objects):

1) *Simple occlusion* – we start with two objects (among other objects) that are both visible. We then hide the smaller one of the objects behind the bigger one. The occluded object does not produce any sensor data. We can, however, reason about it. Then the smaller object reappears in the scene, we should be able to associate the reappearing object with the one from before (same anchor with high probability).
2) *Moving an occluded object* – we now want to *track* an object for which no sensor data is available. We start again with two objects that are both visible. We then hide the smaller object underneath the bigger object, move the bigger object and, subsequently, reveal the smaller object. We should be able to associate the reappearing object with the one from before.
3) *Moving occluded objects with unexpected revealing* – similar scenario as before only this time we start out with also having an unknown object hidden which the observer initially does not know about. We now hide again the (visible) smaller object underneath the bigger object, move the bigger object, but this time reveal the initially unknown hidden object. The system should recognize this as a new object. Then the other (initially visible) smaller object is revealed, the system should recognize this object as the previously anchored object.
4) *Shell game* – we start with three identical containers and a smaller object. We then hide the smaller object underneath one of the three containers and start shuffling the containers around. We should now be able to ask the system under which of the three containers the hidden objects is located.

In Figure 6, we exemplify our results of stated scenarios with a number of screen-shots during the execution of each scenario.[7]. The scenarios were performed in near real-time on a laptop Intel(R) i7 CPU 2.60GHz with 16 GB memory and

[7]Full videos are available under http://reground.cs.kuleuven.be/

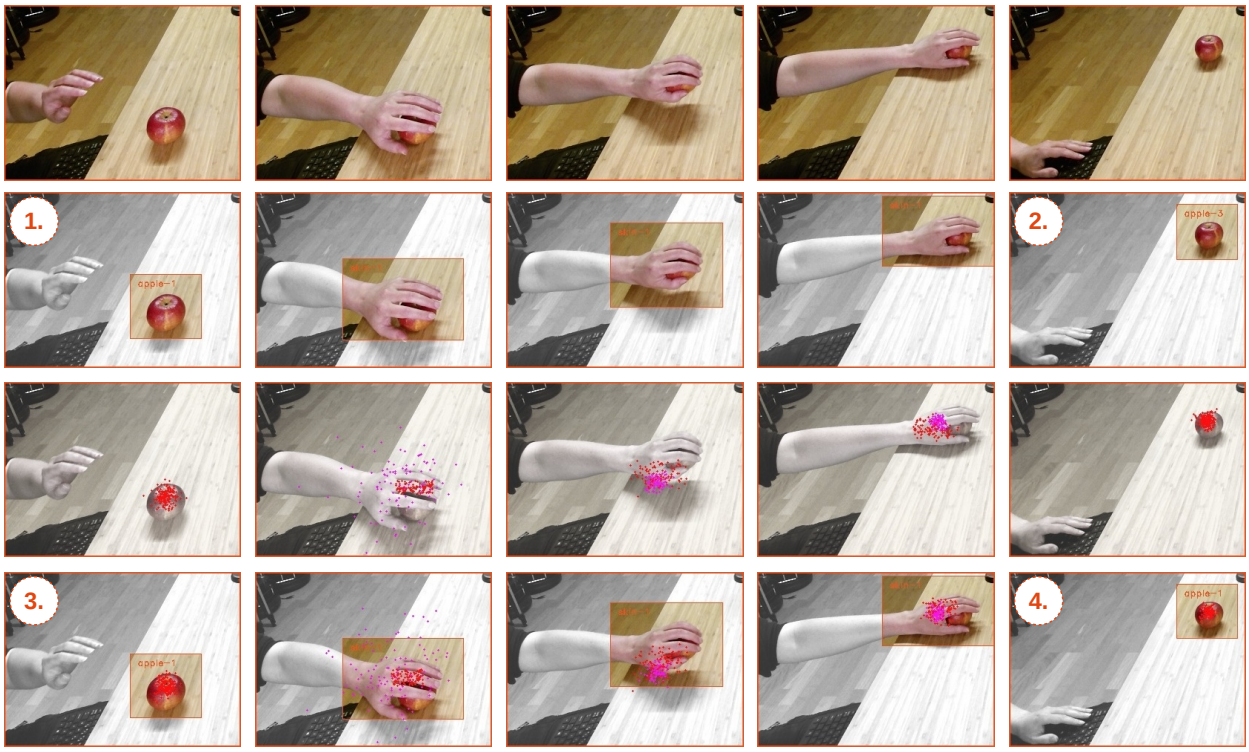

Fig. 5. A depiction of how suggested system benefits of combined object anchoring and probabilistic object tracking. *Rows in order from the top: 1st)* representing screen-shots of a scenario where a human hand is occluding an apple while the apple is moved, *2nd)* corresponding resulting anchored objects while *only* using the *anchoring system* (note that the original `'apple-1'` object is lost while it is occluded and moved by the `'skin-1'` object, and a new `'apple-3'` object is, therefore, *acquired* in the end of the scenario), *3rd)* plotted particles given by the *inference system* during execution of suggested integrated approach, and *4th)* corresponding resulting anchored objects of the *anchoring system* supported by the feed back of the *inference system* (note that in this case is the position of `'apple-1'` object tracked while it is occluded and moved by the `'skin-1'` object, and the `'apple-1'` object is, accordingly, *re-acquired* in the end of the scenario).

an NVIDIA Quadro M1000M. This constitutes a promising feature of our approach as we do not need access to high-performance machines to deploy our system.

In the first example with *simple occlusion* (Figure 6 – *1st row from top*), it can be seen that as soon as the `'cup'` occludes the smaller `'ball'` object, the object is immediately tracked and maintained by the probabilistic reasoner (seen in Figure 6 – № 1). Opposite, once the `'ball'` objects reappear in the scene, it is no longer any need to probabilistically track the object, and the object is, once again, maintained through anchoring (seen in Figure 6 – № 2).

Through the second example (Figure 6 – *2nd row from top*), it is illustrated how the combined system handles *movements during occlusions*. In this example, a `'glove'` object (a human hand) is occluding while moving an `'apple'` object. As soon as the `'apple'` object is occluded by the `'glove'`, the `'apple'` object is tracked and maintained though probabilistic reasoning (seen in Figure 6 – № 3). The tracked position of the occluded object is continuously fed back to the anchoring system (through the newly instated anchoring *track* functionality), and the `'apple'` object is, consequently, *re-acquired* as the same `'apple-1'` once the object reappears in the scene (seen in Figure 6 – № 4).

In the third example (Figure 6 – *3rd row from top*), we demonstrate how our combined systems truly works in symbiosis. In this case, a similar scenario of *moving an occluded object* is exemplified where a `'ball'` (`'ball-1'`) is occluded while moved by a `'glove'`. However, another unknown `'ball'` is initially also hidden underneath the `'glove'` (in the human hand). This hidden `'ball'` is later introduced in the scene during the execution of the scenario (seen in Figure 6 – № 5). Nevertheless, since the second `'ball'` is different in appearance (compared to `'ball-1'`), this newly introduced object is correctly *acquired* as an new `'ball-2'` object, while the first `'ball-1'` object correctly remains tracked, and is subsequently *re-acquired* as the same `'ball-1'` object once reappearing in the scene (seen in Figure 6 – № 6).

Finally, in the fourth example (Figure 6 – *4th-6th row from top*), we reconnect with our initial motivation statement (outlined in Section I), through presented screen-shots captured during execution of the *shell game* scenario. In this example, a smaller `'block'` object (`'block-3'`) is hidden underneath one of three identical larger `'block'` objects (`'block-1'`), seen in Figure 6 – № 7. All the larger `'block'` objects are, subsequently, moved around and shuffled. Nevertheless, during all movements is the hidden `'block-3'` object tracked though the relation with the occluding counterpart (`'block-1'`), i.e., the inference system is repeatedly speculating about the position of the hidden `'block-3'`, and the tracked position of is continuously fed back to the anchoring system. Consequently,



Fig. 6. Examples of screen-shots captured during the execution of stated scenarios. Visual perceived anchored objects are symbolized with the unique anchor id (e.g., ‘ball-2’), while occluded hidden objects are depicted by plotted particles that represent possible positions of the occluded object in the inference system. *Rows in order from the top: 1st)* example of *simple occlusion* where a ‘ball’ is hidden behind a ‘cup’, *2nd)* depicts the *movement* of an *occluded object* where a ‘glove’ (or human hand) is occluding while moving an ‘apple’, *3rd)* similar example of *moving an occluded object* where a ‘glove’ is occluding while moving a ‘ball’ (‘ball-1’), but in this case is also another ‘ball’ object (‘ball-2’) introduced during the execution of the scenario, *4-6th)* illustrate a *shell game* scenario where a smaller object (‘block-3’) is hidden under one of three identical containers (‘block-2’), and where the containers, subsequently, are shuffled around.

once the hidden object is revealed and reappear in the scene (seen in Figure 6 – № 8), the object is correctly *re-acquired* as the same ‘block-3’ object.

VI. Conclusions and Future Work

In this paper, we have presented how we are able to improve the overall anchoring process by introducing a post-anchoring high-level probabilistic reasoning procedure with the purpose of predicting the state of objects that are not directly perceived through the perceptual sensor data, e.g., in case of object occlusions. To retain the integrated framework in a coherent cognitive state, we have suggested a loosely coupled integration between proposed inference system and anchoring system, while a tight feedback loop is preserved in order to maintain consented tracked positions of objects. We have presented the proof-of-concept of how this integrated

framework is used to model and manage a consistent semantic world model of perceived objects in dynamic scenarios. We have further introduced a novel anchoring matching approach based on classification of humanly annotated ground truth data of real-world objects for determining whether a perceived object has previously been observed (or not), and, subsequently, invoke correct anchoring functionality (*acquire* or *re-acquire*), in order to correctly anchor perceived objects. Through the presented results, we have shown that our learned anchoring matching approach is able to accurately anchoring objects and maintaining consistent representations of objects with an accuracy of 96.4%.

In the scope of the ReGROUND project (cf. Section I), a possible future direction is to exploit how anchored objects and their spatial relationship, tracked over time, facilitate the learning of both the function of objects, as well as object affordances – similar to previously presented works on learning objects affordances from visual data [40, 41, 42]. However, previously presented works have commonly assumed an approach based on the tracking of human hand actions, e.g., exploiting the spatial-temporal relationships between objects and human hand actions to learn the function of objects [40]. For the approach presented in this paper, we only consider the tracking of object instances (where a human `'hand'` object might constitute one such object instance), and where a relational particle filter approach is utilized for high-level tracking. Hence, it is also possible to further infer both the function of objects and object affordances by employing probabilistic logic programming.

The integration of high-level object tracking into the anchoring framework is also a subject for further investigation. As of now, only uni-modal probability distribution can be handled by the anchoring system. This means that the rich and intricate probability distributions that can be expressed in DDC, for example multi-modal distribution of the position of unobserved objects, can not be passed on to the anchoring system and be handled in a probabilistic fashion. Allowing this would render our approach truly probabilistic and equally allow us to keep track of multiple hypotheses.

ACKNOWLEDGMENTS

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, the Research Foundation - Flanders and the Swedish Research Council (Vetenskapsrådet). The work is also supported by the Swedish Research Council (Vetenskapsrådet) under the grant number: 2016-05321. The authors would also like to thank Prof. Alessandro Saffiotti for his valuable comments and discussions during the work.

REFERENCES

[1] J. Elfring, S. van den Dries, M. van de Molengraft, and M. Steinbuch, "Semantic world modeling using probabilistic multiple hypothesis anchoring," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 95–105, 2013.

[2] S. Coradeschi, A. Loutfi, and B. Wrede, "A short review of symbol grounding in robotic and intelligent systems," *KI-Künstliche Intelligenz*, vol. 27, no. 2, pp. 129–136, 2013.

[3] S. Coradeschi and A. Saffiotti, "Anchoring symbols to sensor data: preliminary report," in *Proc. of the 17th AAAI Conf.* Menlo Park, CA: AAAI Press, 2000, pp. 129–135. [Online]. Available: http://www.aass.oru.se/~asaffio/

[4] ——, "An introduction to the anchoring problem," *Robotics and Autonomous Systems*, vol. 43, no. 2-3, pp. 85–96, 2003. [Online]. Available: http://www.aass.oru.se/Agora/RAS02/

[5] A. Loutfi, S. Coradeschi, and A. Saffiotti, "Maintaining coherent perceptual information using anchoring," in *Proc. of the 19th IJCAI Conf.*, Edinburgh, UK, 2005, pp. 1477–1482.

[6] A. Persson, M. Länkvist, and A. Loutfi, "Learning actions to improve the perceptual anchoring of objects," *Frontiers in Robotics and AI*, vol. 3, p. 76, 2017. [Online]. Available: http://journal.frontiersin.org/article/10.3389/frobt.2016.00076

[7] D. Nitti, T. De Laet, and L. De Raedt, "Probabilistic logic programming for hybrid relational domains," *Mach. Learn.*, vol. 103, no. 3, pp. 407–449, Jun. 2016.

[8] D. Fierens, G. Van den Broeck, J. Renkens, D. Shterionov, B. Gutmann, I. Thon, G. Janssens, and L. De Raedt, "Inference and learning in probabilistic logic programs using weighted boolean formulas," *Theory and Practice of Logic Programming*, vol. 15, no. 3, pp. 358–401, 2015.

[9] L. Antanas, J. Davis, L. D. Raedt, A. Loutfi, A. Persson, A. Saffiotti, D. Yuret, O. Arkan Can, E. Unal, and P. Zuidberg Dos Martires, "Relational symbol grounding through affordance learning: An overview of the reground project," in *International Workshop on Grounding Language Understanding (GLU)*. Stockholm, Sweden: Satellite of INTERSPEECH, 2017.

[10] K. LeBlanc, "Cooperative anchoring : sharing information about objects in multi-robot systems," Ph.D. dissertation, rebro University, School of Science and Technology, 2010.

[11] N. Blodow, D. Jain, Z. C. Marton, and M. Beetz, "Perception and probabilistic anchoring for dynamic world state logging," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*, Dec 2010, pp. 160–166.

[12] D. Reid, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[13] R. Bellman and R. Corporation, *Dynamic Programming*, ser. Rand Corporation research study. Princeton University Press, 1957.

[14] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Data association for semantic world modeling from partial views," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1064–1082, 2015.

[15] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE*

Transactions on Automatic Control, vol. 54, no. 3, pp. 481–497, 2009.

[16] C. Manfredotti, "Modeling and inference with relational dynamic bayesian networks," in *Canadian Conference on Artificial Intelligence*. Springer, 2009, pp. 287–290.

[17] L. Mösenlechner and M. Beetz, "Using physics-and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior." in *ICAPS*, 2009.

[18] M. Tenorth and M. Beetz, "Knowrob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[19] D. Nitti, T. De Laet, and L. De Raedt, "Relational object tracking and learning," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 935–942.

[20] K. LeBlanc and A. Saffiotti, "Cooperative anchoring in heterogeneous multi-robot systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Pasadena, CA, 2008, pp. 3308–3314. [Online]. Available: http://www.aass.oru.se/~asaffio/

[21] A. Loutfi, "Odour recognition using electronic noses in robotic and intelligent systems," Ph.D. dissertation, Örebro University, Örebro, Sweden, 2006.

[22] A. Loutfi and S. Coradeschi, "Smell, think and act: A cognitive robot discriminating odours," *Autonomous Robots*, vol. 20, pp. 239–249, 2006.

[23] A. Loutfi, S. Coradeschi, M. Daoutis, and J. Melchert, "Using knowledge representation for perceptual anchoring in a robotic system," *Int. Journal on Artificial Intelligence Tools*, vol. 17, pp. 925–944, 2008.

[24] M. Daoutis, S. Coradeschi, and A. Loutfi, "Cooperative knowledge based perceptual anchoring," *International Journal on Artificial Intelligence Tools*, vol. 21, no. 3, 2012.

[25] A. Chella, S. Coradeschi, M. Frixione, and A. Saffiotti, "Perceptual anchoring via conceptual spaces," in *Proceedings of the AAAI-04 workshop on anchoring symbols to sensor data*, 2004, pp. 40–45.

[26] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[27] L. Sterling and E. Y. Shapiro, *The art of Prolog: advanced programming techniques*. MIT press, 1994.

[28] B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. De Raedt, "The magic of logical inference in probabilistic programming," *Theory and Practice of Logic Programming*, vol. 11, no. 4-5, pp. 663–680, 2011.

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

[30] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[31] A. Khoreva, R. Benenson, J. H. Hosang, M. Hein, and B. Schiele, "Simple does it: Weakly supervised instance and semantic segmentation." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[32] T. Wiedemeyer, "IAI Kinect2," https://github.com/code-iai/iai_kinect2, Institute for Artificial Intelligence, University Bremen, 2014 – 2015, accessed June 12, 2015.

[33] S. Holzer, R. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 2684–2689.

[34] A. Trevor, S. Gedikli, R. Rusu, and H. Christensen, "Efficient organized point cloud segmentation with connected components," in *3rd Workshop on Semantic Perception Mapping and Exploration (SPME), Karlsruhe, Germany*, 2013.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[38] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[39] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, 2013.

[40] H. Kjellström, J. Romero, and D. Kragić, "Visual object-action recognition: Inferring object affordances from human demonstration," *Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 81–90, 2011.

[41] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[42] H. S. Koppula and A. Saxena, "Physically grounded spatio-temporal object affordances," in *European Conference on Computer Vision*. Springer, 2014, pp. 831–847.