frontiers in Robotics and AI

TECHNOLOGY REPORT
published: 30 January 2017
doi: 10.3389/frobt.2016.00076

Learning Actions to Improve the Perceptual Anchoring of Objects

Andreas Persson, Martin Längkvist and Amy Loutfi*

Department of Science and Technology, Center for Applied Autonomous Sensor Systems (AASS), Örebro University, Örebro, Sweden

In this paper, we examine how to ground symbols referring to objects in perceptual data from a robot system by examining object entities and their changes over time. In particular, we approach the challenge by (1) tracking and maintaining object entities over time; and (2) utilizing an artificial neural network to learn the coupling between words referring to actions and movement patterns of tracked object entities. For this purpose, we propose a framework that relies on the notations presented in perceptual anchoring. We further present a practical extension of the notation such that our framework can track and maintain the history of detected object entities. Our approach is evaluated using everyday objects typically found in a home environment. Our object classification module has the possibility to detect and classify over several 100 object categories. We demonstrate how the framework creates and maintains, both in space and time, representations of objects such as "`spoon`" and "`coffee mug`." These representations are later used for training of different sequential learning algorithms to learn movement actions such as "`pour`" and "`stir`." We finally exemplify how learned movements actions, combined with commonsense knowledge, further can be used to improve the anchoring process *per se*.

Keywords: perceptual anchoring, symbol grounding, action learning, sequential learning algorithms, commonsense knowledge, object classification, object tracking

OPEN ACCESS

Edited by:
Katerina Pastra, Cognitive Systems Research Institute, Greece

Reviewed by:
Tom Ziemke, University of Skövde; Linköping University, Sweden
Nicos Angelopoulos, Wellcome Trust Sanger Institute (WT), UK

****Correspondence:***
Andreas Persson
andreas.persson@oru.se

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

Received: *05 April 2016*
Accepted: *08 December 2016*
Published: *30 January 2017*

Citation:
Persson A, Längkvist 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

1. INTRODUCTION

In cognitive robotics, the task planning ability of a robot depends on symbol grounding or subsets of it, since the robot ultimately requires that the symbols that it uses are anchored in the physical world. For example, anchoring is needed to execute commands such as "lift the crate" or "put the cup on the table." In this context, anchoring has been defined as a special case of physical symbol grounding, which concerns physical objects (Coradeschi and Saffiotti, 2003). In particular, anchoring considers how to maintain a consistent identity of an object into a structure that (1) describes an object, its properties, and attributes both symbolically and perceptually; and (2) maintains over time these associations. Therefore, perceptual anchoring requires constructing a consistent relation between the perceptual data and the symbols that refer to the same object.

Today much of the work in perceptual anchoring in robotics still resides on anchoring the individual symbols that refer to single objects and their properties. This approach, while useful for object recognition, provides only part of a solution in a full human–robot interaction (HRI) context where humans could instruct and communicate with robots. For humans, recent research suggests that indeed action learning and word learning should not be treated as two separate processes, but rather as two highly integrated processes. Thus, the Gibson theory of affordances has inspired many works within robotics to learn affordances *via* self-exploration using manipulation actions,

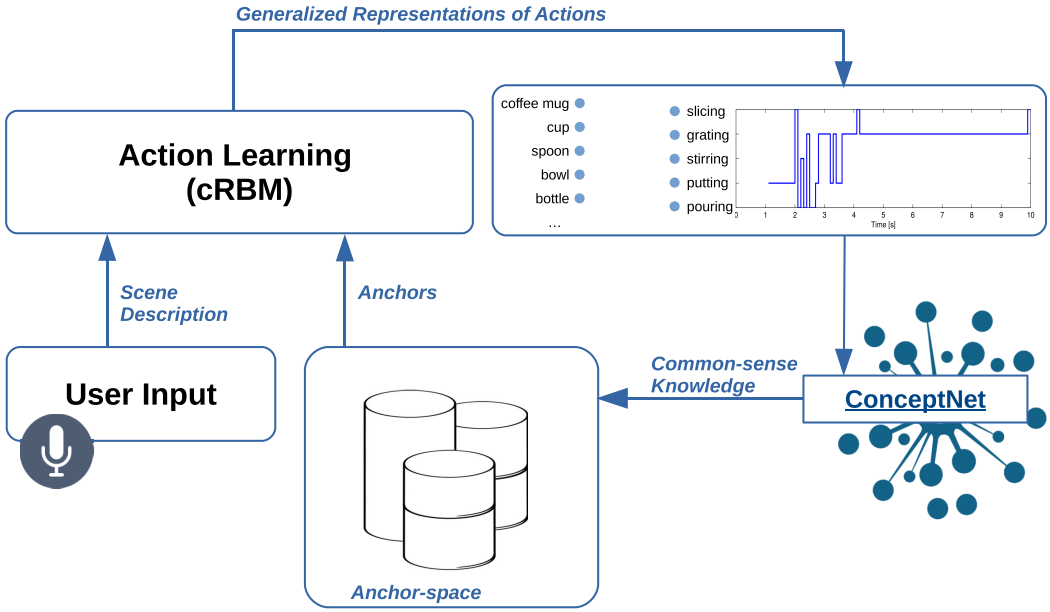

FIGURE 1 | The primary objective of the work presented in this paper is to use the historical trace of anchored objects, which are maintained in an anchor space, combined with user-provided scene descriptions to learn generalized representations of object actions. Learned action representations are subsequently used as feedback to the anchoring system. This feedback loop further utilizes the commonsense knowledge of ConceptNet (Speer and Havasi, 2012) to query and verify that an anchored object indeed affords an action.

such as in work on the iCub robot (Marocco et al., 2010; Browatzki et al., 2012). But worth noting is that further research also suggests that action learning and label learning can also be observed through observations of other performing tasks (Mareschal and Johnson, 2003). Following along the lines of Hahn and Gershkoff-Stowe (2010), this paper examines how robotic systems can learn the relation between naming an object and actions performed on the objects. To achieve this, we will rely on notions presented in perceptual anchoring and extend these notions to utilize the historical trace (track) of how objects (or anchors) change over time in a given scene. Based on the position of objects in time, and with respect to each other, a learning framework based on the sequential learning algorithms generates a generalized representation of various actions with a mapping to its symbolic terms (e.g., "`put`," "`stir`," "`pour`"). By using these grounded actions, we demonstrate how we are able to improve the anchoring process *per se* and, hence, improve the symbol–percept correspondence of objects. This improvement is achieved by forming an early identification of an action in a novel scene, and then through subsequent utilization of the symbolic description stored in anchors, we query a semantic network to verify whether the identified objects indeed afford those actions. The results of the query are then used to resolve both errors and uncertainties that are inherent in the spatial-temporal object tracking and specifically in the object matching component of anchoring. An overview of the work that we present in this paper is depicted in **Figure 1**.

2. RELATED WORK

The symbol grounding problem (SGP) has been defined by Harnad (1990). Symbol grounding addresses the problem of grounding the meaning of symbol tokens in anything different than other symbols, e.g., sensor data perceived by a mobile robot. Grounding symbolic knowledge about actions and objects to a perceived model of the physical world has been addressed in many works over the past decades (for a review of symbol grounding and symbol grounding in relation to perceptual anchoring, in particular, see the report by Coradeschi et al. (2013)). Nonetheless, a few notable contributions in grounding of actions are the works presented by Lemaignan et al. (2012) and Stramandinoli et al. (2011, 2012). In the study by Lemaignan et al. (2012), the authors presented a grounded shared model of the world that is used for both human–robot verbal and non-verbal interactions. The model is realistic in the sense that 3D models of objects are created and maintained in such a way that spatial relations between objects can be used to reasoning upon various concepts, relations, and actions in human–robot interaction scenarios. In the study by Stramandinoli et al. (2011, 2012), the authors presented a cognitive model of high-order concepts that are grounded by using basic concepts and actions that are directly grounded in robot sensorimotor experiences. The suggested model is learned using a recurrent neural network (RNN), which is trained using the temporal sequences of robot action primitives to learn higher-order concepts. The approach that we present in this paper is similar in the use of an artificial neural network to learn action concepts. However, we will approach the problem with the utilization of the temporal sequences of object movements (rather than the temporal sequences of robot action primitives).

A particular case of symbol grounding that addresses the problem of creating and maintaining the connection between perceptual sensor data and symbolic knowledge has been denoted as the anchoring problem (Coradeschi and Saffiotti, 2003). Since

the initial formal definition of perceptual anchoring, presented by (Coradeschi and Saffiotti, 2000), the definition has undergone several extensions and refinements. In the work presented by Loutfi et al. (2005), the authors introduced an extension to bottom-up anchoring together with multiple and non-vision-based modalities. The first connection to a knowledge representation and reasoning mechanism was presented by Loutfi et al. (2008), thereby the properties of anchored objects could be reasoned upon. An extension to consider large-scale knowledge bases, such as a Cyc knowledge base (Lenat, 1995), together with commonsense reasoning was later presented by Daoutis et al. (2012). Another notable work is the introduction of probabilistic multiple hypothesis anchoring by Elfring et al. (2013), where multiple hypothesis tracking-based data association is used to maintain changes in anchored objects. As an alternative to traditional anchoring, an earlier work on perception and probabilistic anchoring was presented by Blodow et al. (2010). The study by Blodow et al. (2010) was also the first reported work where the history of objects was considered to use a Markov logic network for probabilistic data association. However, the history of objects was maintained as computationally complex scene instances, and the approach was, therefore, not intended for tracking objects in real time, but rather for tracking objects in object kidnapping scenarios, i.e., an object disappears from the scene and later reappears in a different location. Both tracking objects and maintaining coherent representations of perceived objects are important aspects of anchoring. However, no previously reported work on anchoring has taken into consideration the history of a tracked object *per se* to learn additional information.

The problem of recognizing object actions, which we address in this paper, is typically addressed in the literature through the use of hand and object tracking. The approach of combined hand and object tracking to recognize kitchen activities is presented in the study by Lei et al. (2012), where the object recognition is utilized by an SVM classifier and the (hand) action recognition is based on the PCA features from 3D hand trajectories and bag-of-words of snippets of trajectory gradients. However, learning to classify and recognize the perceived action is by itself a challenging task. Aksoy et al. (2011) address this problem through maintaining a representation of relations between objects at decisive time points during manipulation tasks. The authors suggest a graphical representation constructed from tracked image segments so that topological transitions in the graph can be stored (and learned) in a transition matrix called the semantic event chain (SEC). Thus, they learn the semantics of object–action relations through observation. The main difference with our work is that we do not put the focus on manipulation but rather on spatial-temporal changes of the objects. Our motivation is to improve the learning of object labels, rather than the classification of specific human motions. As an alternative to learning complex actions (or activities), Tenorth and Beetz (2013) introduced a complementary knowledge processing system (KnowRob). The authors introduced the use of commonsense information on a larger scale (such as information from the Web) to reason upon a perceived scene and inferring possible actions (or action primitives) for robot manipulation tasks. Although we focus on spatial-temporal changes of the objects, we follow a similar fashion and leverage from preexisting semantic information that is available in large knowledge sources such as the ConceptNet semantic network (Speer and Havasi, 2012) and the Caffe deep learning framework (Jia et al., 2014) to resolve ambiguities in the action recognition and object classification processes, respectively.

Object affordance is further a topic that has increased in popularity in the research fields of both semantic object recognition (Sun et al., 2013) and computer vision (Kjellström et al., 2011). In the work presented by Sun et al. (2013), the authors utilized object attributes (e.g., color, shape) learned from RGB-D data to identifying objects based on natural language queries that contained appearances and name attributes. A probabilistic approach is used in the learning of affordances as presented in the study by Kjellström et al. (2011), where the authors utilize the idea that it is possible to exploit spatial-temporal relationships between objects and human hand actions to learn the function of objects. A similar approach to learning affordances is presented by Koppula et al. (2013) and Koppula and Saxena (2014), where they have used RGB-D data to track the relations between objects and human activities to jointly model the human activities and object affordances with the use of Markov random fields. However, in the context of perceptual anchoring, we need to approach the problem differently as we approach the problem from the structure of anchors (rather than human hand activities) to obtain the relation between objects. For that purpose, we introduce the idea of using commonsense knowledge to explore whether a particular object affords an action.

3. METHOD: ANCHORING FRAMEWORK

In this section, we describe our anchoring framework that is used to maintain a consistent notion of objects, both perceptually and symbolically. We further modify the framework to include a learning layer that uses the information about how object changes over time, and with respect to each other, to learn actions. Perceptual anchoring, as defined by Loutfi et al. (2008), extends the original framework to allow for a bottom-up anchoring process and consists of a number of components, 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. A percept π_i is a structured collection of measurements assumed to originate from the same physical object; an attribute ϕ_i is a measurable property of percepts with values in the domain $D(\phi_i)$. Let $D(\Phi) = \bigcup_{\phi \in \Phi} D(\phi)$.
- A *predicate grounding relation* $g \subseteq \mathcal{P} \times \Phi \times D(\Phi)$, which embodies the correspondence between (unary) predicates and values of measurable attributes. The relation g maps a certain predicate to compatible attribute values.

An anchor is an internal data structure α_t^x, indexed by time t and identified by a unique individual symbol x (e.g. "`coffee-mug-1`," "`orange-4`"), which encapsulates and maintains the correspondences between percepts and symbols that refer to the same physical object. Hence, perceptual anchoring

(shortly referred to as *anchoring* throughout the remaining sections of this paper) is also a formally structured procedure to create the coupling between machine-interpreted sensor data and human-readable semantic symbols. Since perceptual anchoring was motivated by the need for robotic planning systems to plan and execute actions involving objects, an anchor has traditionally denoted the latest perceptual update of information. No previously reported work on anchoring has maintained a history of anchors, as their perceptual and symbolic information changed over time. In this study, we argue that the history of an anchor is a necessary prerequisite to enable learning upon interactions with and between objects.

Before describing an extension of anchoring that allows the maintenance of anchors' current and previous information, we will introduce our perceptual processing pipeline in the following section, which enables the extraction and creation of anchors.

3.1. Object Detection, Tracking, and Classification

An efficient perceptual processing pipeline is a necessity for retrieving and storing measurements over time of an object (which further is a requirement for anchoring that object to its symbolic representation). Indeed, an anchoring framework must address object recognition and object tracking. Recall that anchoring not only identifies objects but also identifies and tracks specific instances of objects, both symbolically and perceptually. For performing the lower level object detection, tracking, and classification, the system setup is based on publicly available core libraries and systems, including the Point Cloud Library[1] (PCL), the Open Computer Vision library[2] (OpenCV), and the Robot Operating System[3] (ROS). An overview of the perceptual processing pipeline is presented in **Figure 2**.

The initial step of our processing pipeline is an object segmentation procedure, which is performed with the purpose of detecting objects of interest. The object segmentation method is based on point cloud data, which is given as input data by an RGB-D sensor. Moreover, for efficiency, the segmentation relies 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). The segmentation procedure can be described briefly using the following steps:

- Estimate 3D surface normals based on integral images (Holzer et al., 2012). This function uses the algorithm for calculating average 3D gradients over six integral images, where the horizontal and vertical 3D gradients are used to compute the normal as the cross-product between two gradients.
- Planar segmentation based on the calculated surface normals, where the largest segmented plan is selected as the ground plane.
- Object segmentation through clustering of the remaining points (points that are not part of the detected planar surface). This segmentation uses a connected component segmentation, presented by Trevor et al. (2013), where a Euclidean comparison function is used to connect the components that constitute the cloud cluster of an individual object.

The resulting output of the object segmentation is subsequently m number of point cloud clusters (where m varies between frames). An example of segmented objects with boundary points projected to corresponding 2D imagery is seen in **Figure 3** (left). For consistency with the definition of anchoring, we will here denote segmented clusters as percepts $\{\pi_1^{cloud}, \pi_2^{cloud}, \ldots \pi_m^{cloud}\}$,

[1]http://pointclouds.org/.
[2]http://opencv.org/.
[3]http://www.ros.org/.

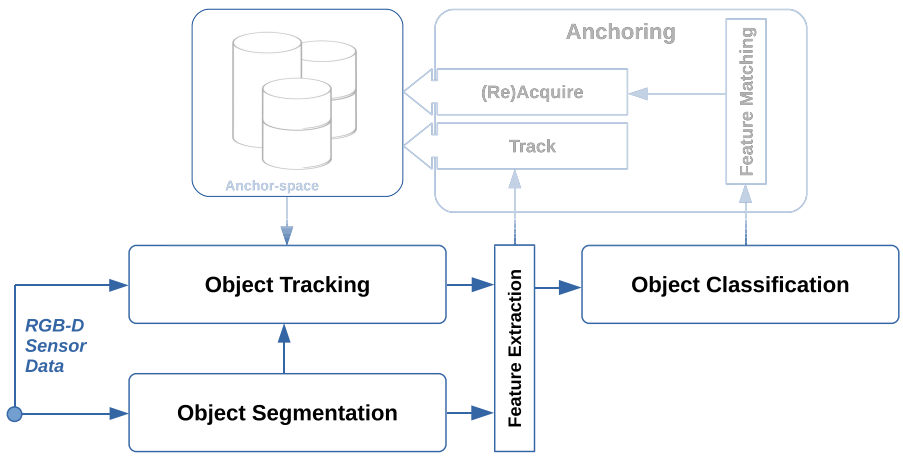

FIGURE 2 | An overview of the computational steps in the perceptual processing pipeline. Sensor input RGB-D data are initially fed to an object segmentation procedure, which attempts to detect objects in the scene. In addition to object detection, an object tracking procedure is tracking the movements of detected objects. Detected objects are then forwarded to an object classification procedure before objects are processed and anchored by an anchoring management system.

FIGURE 3 | *Left:* segmented objects represented by contours of segmented point clouds (projected from 3D point clouds to 2D imagery); and *right:* classified objects together with symbolic categories and probabilities as a result of processing segmented objects through a deep convolutional network (DCN) classifier.

which each corresponds to the spatial 3D point cloud data of an individual object.

The same input RGB-D stream of sensor data is also used for tracking the movements of objects. This object tracking procedure is dependent on the same segmented percepts $\{\pi_1^{cloud}, \pi_2^{cloud}, \ldots \pi_m^{cloud}\}$ to initiate the tracking. More specifically, the object tracking procedure utilizes a particle filter-based tracking algorithm, which is included in the Point Cloud Library (Rusu and Cousins, 2011). A drawback of such particle filter algorithm is that the algorithm is designed for tracking only one object of interest at a time. A pool of particle filters (one filter for each detected object) has therefore been employed in our system setup. The integration between the object tracking procedure and our anchoring approach is further addressed in Section 3.2.

Next, segmented or tracked percepts are forwarded further down the pipeline for feature extraction and category classification with the goal of symbolically associating a category label to each object. The first step of this process is to extract a 3D bounding box around each percept $\pi^{cloud}_{i|i=1,2\ldots m} \in \{\pi_1^{cloud}, \pi_2^{cloud}, \ldots \pi_m^{cloud}\}$, where each extracted bounding box is projected to a corresponding 2D visual imagery. Also, we will here denote visual data as percepts $\{\pi_1^{image}, \pi_2^{image}, \ldots \pi_m^{image}\}$, which each corresponds to the visual 2D imagery data of a segmented object. Furthermore, a position attribute, $\phi_i^{pos} \in \mathbb{R}^3$, is measured as a point at the center of each segmented percept π_i^{cloud}.

Finally, a symbol category classification procedure is initiated in addition to the feature extraction process. For this classification, we exploit recent advancements in deep learning through the Caffe deep learning framework (Jia et al., 2014). This deep convolutional network (DCN) makes use of the 1 K ILSVRC-2012 model, developed and learned by Krizhevsky et al. (2012), which uses the DeCAF implementation (Donahue et al., 2013). Hence, the output of the classifier is 1 of the 1,000 object categories (of which the ILSVRC-2012 model was trained with), together with the corresponding predicted category probability. In the context of anchoring, we assume that all trained object categories (e.g., "`coffee mug`," "`wooden spoon`," "`orange`," "`tomato`") forms the set of possible predicate symbols $\mathcal{P}$. The input for the Caffe framework is the segmented visual percepts π_i^{image}. The resulting object categories with predicted category probabilities are denoted by $p_i^{obj} \in \mathcal{P}$ and ϕ_i^{obj}, respectively. To exemplify, the *predicate grounding relations* for the attributes that have been measured through the Caffe framework for the segmented objects illustrated in **Figure 3** (right) are here denoted: "`coffee mug`" $\times \phi_1^{obj} \times 0.4352$ and "`orange`" $\times \phi_2^{obj} \times 0.4383$.

3.2. Object Anchoring

The advantage of using an anchoring system over a traditional object tracking system is that both semantic and perceptual information are maintained. Moreover, the goal for our suggested framework is to use generalized representations of actions to ground a symbolic description of an action directly to an observed movement of an object. We will then utilize the symbolic descriptions of anchors together with learned actions in the interest of performing high-level queries about object affordances. This latter step is described in Section 4.4.

The traditional anchoring definition has assumed unary perceptual-symbol correspondences. This definition will not permit anchoring of object movements over time as anchors are updated for every new perceived matching instance of an object. To circumvent this problem, we will in this section extend the anchoring definition and introduce two types of attributes: (1) *static attributes* ϕ, which are unary within the anchor (according to the traditional definition) and which combined identifies an anchor; (2) *volatile attributes* ϕ_t, which are individually indexed by time t, and which are maintained in sets of attribute instances φ, such that $\phi_t \in \varphi$.

To exemplify and reconnecting to Section 3.1, an attribute of type ϕ^{obj} is considered as a *static attribute* (since this attribute classifies the object), while an attribute of type ϕ^{pos} is considered as a *volatile attribute* (since this attribute is part of a movement history of an anchor) and which is therefore maintained in a set of 3D points (each index by time t), such that $\phi_t^{pos} \in \varphi^{action}$. Other types of volatile attributes for objects could be $\phi_t^{gas} \in \varphi^{smell}$ in the case of olfactory concepts as described in the study by Loutfi et al. (2005), or in the case of color of certain objects $\phi_t^{color} \in \varphi^{state}$ for representing fruit as discussed in the study by Loutfi et al. (2008). In this paper, we only consider action (φ^{action}) as a set of position attributes (ϕ_t^{pos}).

The entry point into the anchoring management system, illustrated in **Figure 4**, is *via* the matching procedure. This match compares the classification attribute ϕ_i^{obj} and symbol p_i^{obj} of a candidate object against the classification attribute and symbol of a previously stored anchor α^x according to:

$$d_x^{obj}(\phi_i^{obj}, \phi_x^{obj}) = \begin{cases} \frac{1}{\exp\left(\frac{|\phi_i^{obj} - \phi_x^{obj}|}{\phi_i^{obj} \times \phi_x^{obj}}\right)} & \textbf{if } p_i^{obj} \equiv p_x^{obj} \\ 0 & \textbf{else} \end{cases} \quad (1)$$

Second, and because of the separation between the *object tracking* pipeline and *object classification* pipeline, this match further compares the L^2 distance (in a three-dimensional spatial space) between a position attribute ϕ_i^{pos} of a candidate object and the last observed position $\phi_{t-k}^{pos} \in \varphi_x^{action}$ of a previously stored anchor α^x. Inspired by the work presented by Blodow et al. (2010), this distance is then mapped to a normalized similarity distances according to:

$$d_x^{pos}(\phi_i^{pos}, \phi_{t-k,x}^{pos}) = \frac{1}{\exp(L^2(\phi_i^{pos}, \phi_{t-k,x}^{pos}))}, \quad (2)$$

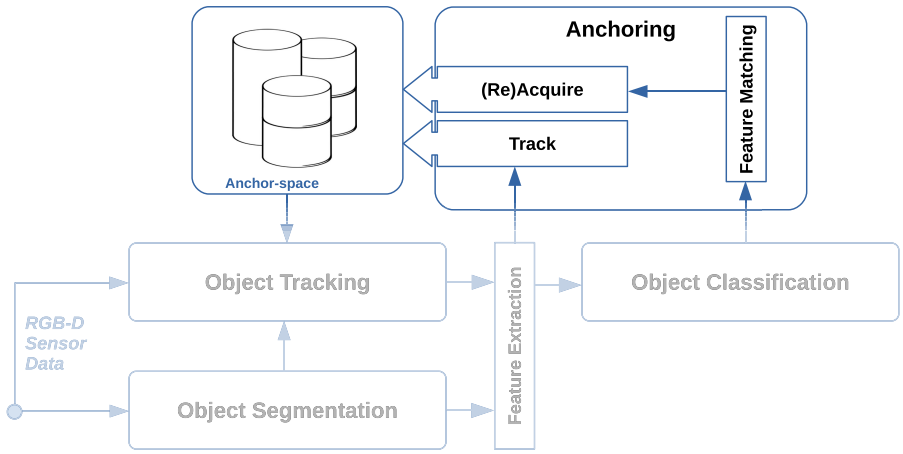

FIGURE 4 | An overview of the anchoring management system. Based on the result of an initial matching procedure, anchors, in an *anchor-space*, are created and maintained through two core functionalities: *re-acquire* and *acquire*. The system further utilizes a *track* functionality, which for this work is integrated with the object tracking procedure on the perceptual level.

where an identifier x is unique for an anchor as whole and therefore also unique for the all the percepts, attributes, and symbols that are encapsulated in the anchor. Hence, the identifier x is, in this context, used to index a particular attribute or predicate symbol that belongs to an anchor α^x. The matching procedure, according to Eqs. 1 and 2, is repeated for all previously stored anchors $\alpha^{x|x\in\mathcal{X}}$, and based on the result of the matching procedure, the *anchor-space* is maintained according to two core functionalities:

- **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^x_t, 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.

The distinction between an *acquire* of a new anchor or the *re-acquire* of an existing anchor is decided by two threshold values, which for this work were set to $th_{obj} = 0.3$ and $th_{pos} = 0.9$, i.e., if $d_x^{obj} > th_{obj}$ or $d_x^{pos} > th_{pos}$ for an existing anchor α^x according to Eqs. 1 and 2, then the *re-acquire* functionality is initiated. The thresholds represent the degree of matching between a candidate object and an existing anchor. Otherwise, a new anchor α^x is created through the *acquire* functionality. Where the structure of an anchor is expressed as $\alpha^x_t = \{(\phi^{obj}, p^{obj}), \{\phi^{pos}_t, \phi^{pos}_{t-1}, \ldots \phi^{pos}_{t-k}\} \in \varphi^{action}\}$. Hence, given the perceived object as illustrated in **Figure 3**, the content of an anchor can (simplified) be depicted as $\alpha_t^{\text{`orange}-4'} = \{(\text{`orange'}, 0.4383), \{(x,y,z)_t, (x,y,z)_{t-1}, \ldots (x,y,z)_{t-k}\}\}$.

Inspired by the work on probabilistic multiple hypothesis anchoring presented by Elfring et al. (2013), we further utilize a *track* functionality that is integrated with the object tracking procedure, described in Section 3.1:

- **Track** – takes an anchor α^x defined for time $t-k$ and appends volatile attributes ϕ_t, index by time t, to corresponding sets φ of an anchor α^x.

This integration is facilitated by sharing the unique identifier x, for each anchor α^x, such that the identifier is provided as an associated identifier for each particle filter for the purpose of to directly track an anchor on the perceptual level. Hence, a measured position attribute ϕ_x^{pos}, of a tracked segmented percept ϕ_x^{cloud}, is directly forwarded to the *track* functionality of the anchoring system. The anchoring frame rate for our suggested approach is, therefore, equivalent to the rate at which suggested particle filter (or pool of particle filters) can process the input stream of RGB-D data. Worth noting is that the *track* and the *re-acquire* functionalities have been reported as an integrated functionality in previous work on anchoring (Loutfi et al., 2005).

Finally, we stress that the proposed method presented in this paper concerns segmented objects projected to image regions (for object classification) and position attributes (for tracking object movements), i.e., attributes that are essential for learning an action of an object in a described scene. However, the suggested framework can with ease be extended to cover additional attributes, such as color and shape, which can be used to complement a scene description based on adjectives described by the user input. For example, in a scene description of "stirring with a brown spoon," a color attribute can be used as a complement to identify an object instance of interest in the scene.

3.3. Learning Actions from Anchors

The resulting *anchor-space*, maintained by the anchoring management system presented in the previous Section 3.2, is subsequently used as input for a learning system, as illustrated in **Figure 1**. By storing information of specific object entities (anchors), our aim

is to learn a general representation of an action that is grounded in sensor data. For this work, three algorithms are evaluated for learning actions based on object movements using the maintained anchors: a hidden Markov model (HMM) (Rabiner and Juang, 1986), a conditional restricted Boltzmann machine (cRBM) (Taylor et al., 2007), and a recurrent neural network (Hochreiter and Schmidhuber, 1997).

3.3.1. Hidden Markov Model

The hidden Markov model (HMM) (Rabiner and Juang, 1986) is a commonly used algorithm for modeling sequential data. The HMM consists of a transition matrix, T, and an emission matrix, E. The transition matrix T_{ij} defines the probability of going from hidden state i to j and has a size $S \times S$, where S is the total number of states. The emission matrix E_{ij} defines the probability of observing omission j while being in hidden state i and has a size of $S \times O$ where O is the number of observations.

The HMM requires a discrete state space. In this study, we set the number of states equal to the number of actions to be classified. For the observations, we use a Gaussian Mixture Model (GMM) to transform the input vector of raw sensor data from each time sample to a discreet value for the observation. Therefore, the number of observations is defined by the number of Gaussian components in the GMM.

3.3.2. Conditional Restricted Boltzmann Machine

The conditional Restricted Boltzmann Machine (cRBM) (Taylor et al., 2007) is a generative probabilistic model for modeling structured sequential data. They have been previously used for tasks such as modeling full-body human motion (Taylor et al., 2007), learning motion primitives from demonstration (Kulić and Nakamura, 2011), style-content separation and motion style interpolation (Chiu and Marsella, 2011), and recently modeling robot walking motion under varying circumstances (Luo et al., 2014).

The structure of a cRBM is similar to a restricted Boltzmann machine (RBM) except that it has dynamic bias vectors, A_k and B_k, that depend on the previously visible layer, see **Figure 5A**. The number of previously visible layers that affect the currently hidden layer is called the model order and determines the model's temporal memory. For consistency with the original cRBM definition, visible variables are here denoted **v**, while hidden variables are denoted **h**.

The energy function for a given sequence of visible layers and the hidden layer in a cRBM is defined as:

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{h}^T(t)\mathbf{W}\mathbf{v}(t) - \mathbf{b}^T\mathbf{h}(t) - \mathbf{c}^T\mathbf{v}(t) - \sum_{k=1}^{n}\mathbf{v}^T(t-k)A_k\mathbf{v}^T(t) - \sum_{k=1}^{n}\mathbf{v}^T(t-k)\,B_k\mathbf{h}^T(t), \tag{3}$$

where A_k is the autoregressive connection between the visible layer at time $t-k$ and the current visible layer, and B_k is the connection between the visible layer at time $t-k$ to the currently hidden layer. Furthermore, b_j and c_i are the bias vectors for the hidden and visible layers, respectively.

The joint distribution is defined as $P(\mathbf{v},\mathbf{h}) = \frac{1}{z}\exp^{E(\mathbf{v},\mathbf{h})}$, where z is the partition function. The probabilities for going up or down a layer are as follows:

$$P(h_j|\mathbf{v}) = \sigma\left(b_j + \sum_i W_{ij}v_i + \sum_k\sum_i B_{ijk}v_i(t-k)\right), \tag{4}$$

$$P(v_i|\mathbf{h}) = \sigma\left(c_i + \sum_j W_{ij}h_j + \sum_k\sum_i A_{ijk}v_i(t-k)\right), \tag{5}$$

where $\sigma(\dot{})$ denotes a logistic sigmoid function.

One advantage of the cRBM over the hidden Markov model (HMM) is the use of a distributed hidden state that more efficiently models multiple underlying influences, whereas the HMM uses a discrete K-state multinomial for all previous observations (Taylor, 2009). Other advantages of cRBMs that have been reported are that the observation distribution is an undirected, bipartite graph that provides simple and efficient inference and that cRBMs can be used as building blocks for deep networks (Taylor, 2009).

3.3.3. Recurrent Neural Network

Recurrent neural networks (RNN) (Hochreiter and Schmidhuber, 1997) are an increasingly popular family of algorithms for modeling long-term temporal sequences (Längkvist et al., 2014). They have been used for speech recognition (Graves et al., 2013; Graves and Jaitly, 2014), language translation (Sutskever et al., 2014), and

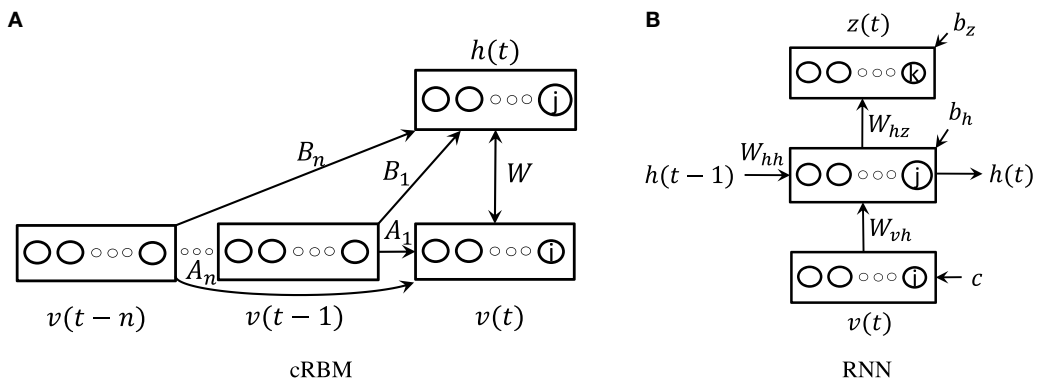

FIGURE 5 | Scheme of (A) a cRBM model and (B) an RNN model with a simple cell.

text generation (Sutskever et al., 2011). A depiction of an RNN with a simple cell can be seen in **Figure 5B**. The equations for the hidden layer, h, and the output layer, z, for an RNN with a simple cell are given by:

$$h(t) = \sigma\left(W_{vh}v(t) + W_{hh}h(t-1) + b_h\right). \quad (6)$$

$$z(t) = \sigma\left(W_{hz}h(t) + b_z\right). \quad (7)$$

However, it is difficult to learn long-term time-dependencies using this approach due to vanishing and exploding gradients that may occur during learning (Pascanu et al., 2013). The long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) cell has been proposed to solve this problem by introducing gates that allow the model to control the amount of influence the currently visible layer and previously hidden layers have on the currently hidden layer, allowing the model to "forget" previous input.

The equations for an RNN with an LSTM cell are given by:

$$i(t) = \sigma\left(W_{ih}h(t-1) + W_{ix}v(t) + b_i\right), \quad (8)$$

$$o(t) = \sigma(W_{oh}h(t-1) + W_{ox}v(t) + b_o), \quad (9)$$

$$f(t) = \sigma(W_{fh}h(t-1) + W_{fx}v(t) + b_f), \quad (10)$$

$$g(t) = \tanh\left(W_{gh}h(t-1) + W_{gx}v(t) + b_g\right), \quad (11)$$

$$c(t) = f(t) \odot c(t-1) + i(t) \odot g(t), \quad (12)$$

$$h(t) = o(t) \odot \tanh(c(t)), \quad (13)$$

$$z(t) = \text{softmax}(h(t)), \quad (14)$$

where $i(t)$, $o(t)$, and $f(t)$ are the input gate, output gate, and forget gate, respectively; $g(t)$ and $c(t)$ are the cell input and cell activation, respectively; $\sigma()$ is the logistic sigmoid function; and $x \odot y$ is the element-wise product of x and y.

The main difference between a cRBM and an RNN is that an RNN is recurrent in the sense that the hidden layer depends on the previous hidden layer instead of a fixed number of previously visible layers like in the cRBM. This means that there is no model parameter for the model order in an RNN that needs to be set, and this allows RNNs to model sequential data of varying lengths (Sutskever et al., 2014). In this study, we will use the RNN with an LSTM cell and will be referred to as LSTM for the remainder of this paper.

4. RESULTS: LEARNING MOVEMENT ACTIONS

In this section, we describe the experiments conducted to evaluate suggested learning approach for learning anchored movement patterns. Evaluating a complex system that operates in the real world is not a trivialized task. The anchoring framework presented in this paper is a distributed system that consists of several individual components, which all can generate and propagate erroneous results. Therefore, this evaluation is limited to the resulting anchors with the history of object movements (produced as the output of the anchoring management system described in Section 3.2) and the potentiality to learn and classify object movement actions through suggested learning system (described in Section 3.3), in particular.

4.1. Experimental Setup

For this evaluation, three different sequential algorithms are used to model five different classes of movement actions ("`pouring`," "`stirring`," "`putting`," "`grating`," and "`slicing`"). For each class, approximately 5–10 scenarios were observed, i.e., all perceptual data for the scene was anchored while a user input sentence that described the movement action in the scenario was provided, e.g., "I am slicing the lemon with the knife." All anchored data were subsequently manually processed, where anchored objects together with movement patterns of objects (based on the correspondence between the classified semantic category labels and the vocabulary of the user input sentence) were extracted. It is worth noting that not all recorded scenarios did capture the movement patterns of the objects of interest in this case, and such scenarios were therefore omitted from this evaluation. This problem was mainly a result of the human intervention (objects were occluded by the human user that conducted the recorded scene or that objects were too much affected by motion blur). Nonetheless, the total extracted data that were used for this evaluation consisted of 32 recorded scenarios, which contained anchored information about two objects (of interest) in each scenario (for a total of 8 "`pouring`," 5 "`putting`," 6 "`stirring`," 4 "`grating`," and 9 "`slicing`" movement actions).

All anchored data were acquired with the use of a Q.bo mobile robot[4] with a 2.6 GHz Intel(R) i3-2120T CPU and 4GB of RAM, which is equipped with an Asus Xtion Pro live RGB-D sensor.

4.1.1. Anchoring Frame Rate

A crucial aspect that affects the capability to learn action is the rate at which suggested anchoring system is capable of detecting and tracking objects. The benefit of using a segmentation procedure based on organized point cloud data (presented in Section 3.1) compared to a traditional RANSAC-based planar segmentation is that no downsampling of the data is necessary to achieve performance (Trevor et al., 2013). However, in our suggested anchoring approach, the tracking of the position of objects is further utilized by a particle filter-based tracking algorithm on the perceptual level (presented in Section 3.2). The two parameters that can influence the frame rate of our approach is, therefore, the number of particles and the size of the pool of filters (which is consistent with the number of detected objects to track). For verifying the performance of our system, the frame rate was measured with respect to the number of used particle filters (i.e., the number of tracked objects). The same tracking algorithm supported by OpenMP multiprocessing (also included in the Point Cloud Library (Rusu and Cousins, 2011)) was further used for comparison.

The results, seen in **Figure 6**, show an object segmentation procedure at a stable frame rate of 1 Hz. For previously reported works by Elfring et al. (2013) and Blodow et al. (2010), which both presented probabilistic methods that are based on the measurements of segmented objects, are the segmentation frame rate hence the maximum rate. We will, therefore, consider this frame rate as the baseline for our approach. In **Figure 6**, it can be seen that the support of OpenMP enables twice as many particles per object at the same frame rate compared to tracking without

[4]http://thecorpora.com/index.php/home.

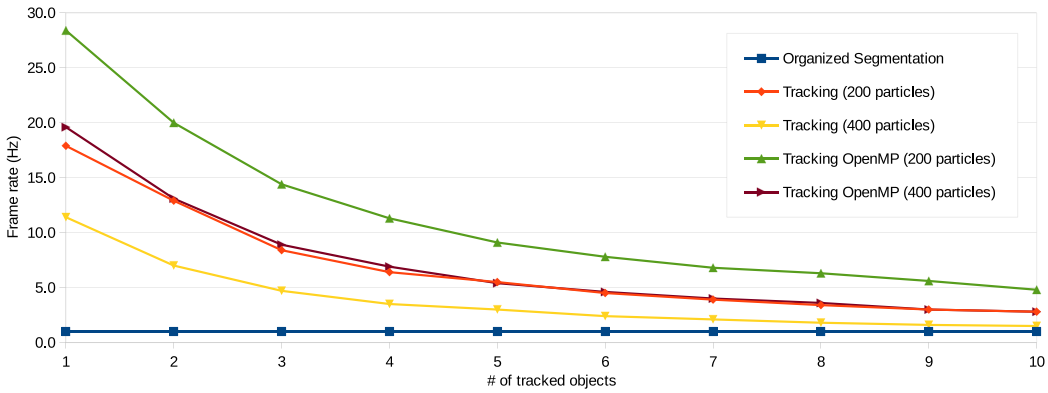

FIGURE 6 | The frame rate (Hz) for suggested segmentation procedure and particle filter-based tracking procedure with respect to the number of tracked objects.

multiprocessing support. For our anchoring track functionality, we have therefore used an OpenMP supported approach with 400 particles for each filter. This combination gives an adequate trade-off between prediction and performance, which provides, for our system setup, a frame rate >5 Hz for up to 5 tracked objects in the scene.

4.1.2. Preprocessing of Data

For the purpose of extracting additional information, the data were subsequently preprocessed and augmented. The velocity of each object movement was calculated by taking the derivative of the position. The velocity was also saturated at −0.1 and 0.1 to avoid large values due to noise. The frame rates, presented in **Figure 6**, are portrayed as average rates. In reality, the anchoring management system provides samples of varying intervals. The data were, therefore, resampled and interpolated to a sampling rate of 10 Hz. Each action was moved to start at origin to remove any movement offset. Data augmentation was achieved by rotating each action in a full circle at 10° intervals, giving a total of 1,152 recorded movement samples. All signals were normalized by subtracting the mean of the signal and then dividing by the standard deviation of the signal. The training, validation, and testing set was randomly divided from the total 1,152 movement sample readings with an 80/10/10% split. The training set was then doubled by switching the two anchors so that the first anchor becomes the second anchor and the second anchor becomes the first anchor. This was performed to eliminate the restriction during running time that the anchors have to be a specific object for each action.

4.1.3. Learning Movement Actions

Three different learning models are used for learning to distinguish between the five different actions classes: HMM, cRBM, and LSTM. The models are first optimized on the validation set and then evaluated by calculating the classification accuracy on the test set. All three models use a respective MATLAB implementation.

For the HMM, the hidden states are set as the five action classes and the discreet observations are modeled as the output component from a Gaussian Mixture Model (GMM). The transition matrix and emission matrix are learned using the standard MATLAB HMM implementation from the *Statistics and Machine Learning Toolbox*.

The model parameters of the cRBM are trained using contrastive divergence (Hinton, 2002) with a decaying learning rate starting at 10^{-4} until the validation error has not improved in the last 10 epochs. Further parameters were set similar to the default values provided in the open-source MATLAB implementation of cRBM (Taylor et al., 2007) that was used in this work, i.e., a mini-batch size of 100, a weight decay of 0.002, a momentum of 0.9, and a desired sparse activation of 0.1 with a penalty cost of 0.3.

The LSTM network is trained using a MATLAB implementation called LightNet (Ye et al., 2016). Training of the model parameters are done using backpropagation through time (BPTT) (Graves and Schmidhuber, 2005) using adaptive learning rate and *ADAM* (Kingma and Ba, 2014) for the stochastic gradient-based optimization.

The optimal choice for the different model parameters for the three algorithms is evaluated in Section 4.2.2.

4.2. Parameter Selection

In this section, we present the selection of all parameters together with each parameter's intermediate resulting influence on the evaluated models.

4.2.1. Selection of Input Data

The position was tracked for each object and given in 3D coordinates. The velocity and acceleration of each object were extracted by calculating the first and second derivatives for each position. As with most high-dimensional multivariate time series data, some of these signals may contain a high amount of noise or even be redundant. Therefore, as an initial experiment, the classification accuracy was calculated using only the position, velocity, acceleration, or a combination of them, as input data. From the results, presented in **Table 1**, it can be seen that using only the position gives a better result than using only the velocity or the acceleration for the HMM and cRBM. Using only the velocity gave a better result for the LSTM compared to only using position or acceleration. The best result for all three models was a combination of both the position and the velocity (even better performance than when

acceleration was also added). The remaining experiments in this section will, therefore, use the position and the velocity as input data.

4.2.2. Optimization of Model Parameters

The most influential user parameter for all three models is the model complexity. This parameter adjusts the model capacity of the algorithm and has a significant influence on the classification accuracy and the training time. The classification accuracy as a function of the model complexity for the three models can be seen in **Figure 7**.

For the HMM model, the model complexity is defined by the number of outcomes for the HMM, i.e., the number of components in the Gaussian Mixture Model. The number of GMM components was chosen as [10, 30, 50, 70, 150, 200], and it can be seen that the highest accuracy on the validation set was achieved with 150 components.

The model complexity for the cRBM is defined by the number of hidden units and the model order. Different model orders of [5, 10, 15, 20] were first evaluated with a fixed number of hidden units of 500. The best result was achieved with a model order of 15. However, a comparable outcome was achieved with a model order of 10 with a significantly lower training time (40 min with a model order of 10 compared to 60 min with a model order of 15). Therefore, the model order is chosen as 10. The number of hidden units for the cRBM was evaluated with [50, 100, 200, 500, 1,000]. In **Figure 7**, a greater number of hidden units used for the cRBM results in a better performance. However, the gain of using 1,000 over 500 hidden units is rather small compared to the increase in training time (40 min with 500 units compared to 3.5 h for 1,000 units), which makes the choice of 500 units a good trade-off between accuracy and training time.

For the LSTM, the number of hidden units was chosen between 10 and 100 with an increase of 10. The best result, regarding classification accuracy and training time, was achieved with using 50 hidden units. It is also worth noting that the worst result of using only 10 hidden units achieved a similar result as the best result for the cRBM of using 1,000 units.

TABLE 1 | A comparison of the classification accuracy (%) on the test set between three different models when the position, velocity, acceleration, or a combination of them, for the two objects that are involved in an action scenario, are used as input.

Input data (#signals)	HMM	cRBM	LSTM
Pos (6)	61.4	70.2	81.7
Vel (6)	59.1	54.2	86.1
Acc (6)	22.9	37.8	57.9
Pos + Vel (12)	63.0	80.3	96.3
Pos + Vel + Acc (18)	60.1	67.8	93.1

4.3. Classification Results

In this section, we present the classification results of our evaluation. Input for this classification was all position and (extracted) velocity reading at each sample. The predicted classification label at each sample for two actions ("`pour`" and "`stir`") are shown in **Figure 8**. The first 10 samples for the cRBM are not classified since they are used to initialize the cRBM, which has a model order of 10. It can be seen that all three models have misclassifications at the beginning of the action. The LSTM is fastest to correctly classify the action, followed by the cRBM and finally the HMM.

Figure 9A shows the average amount of correctly classified motion samples of all samples in the test set, where the results are given as a function of the percentage of the action, i.e., percentage of the progress of the entire action. The movement has the highest probability to be correctly classified around 20, 30, and 50% of the action for the LSTM, cRBM, and HMM, respectively. The LSTM has the best accuracy of the three models independent of the location of the action.

Figure 9B show an overview of where in time the misclassifications for the LSTM are for each action in the test set. The y-axis shows all the 116 actions in the test set, and the time is shown in the x-axis. Black areas indicate the misclassifications for the LSTM. Most misclassifications are at the beginning of the motion, and

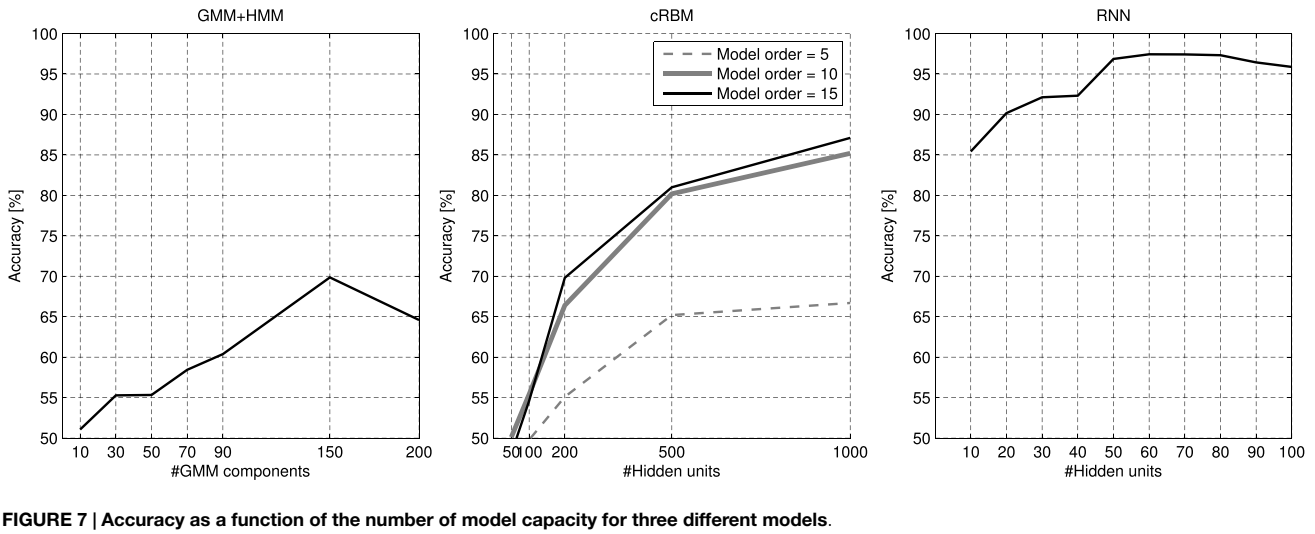

FIGURE 7 | Accuracy as a function of the number of model capacity for three different models.

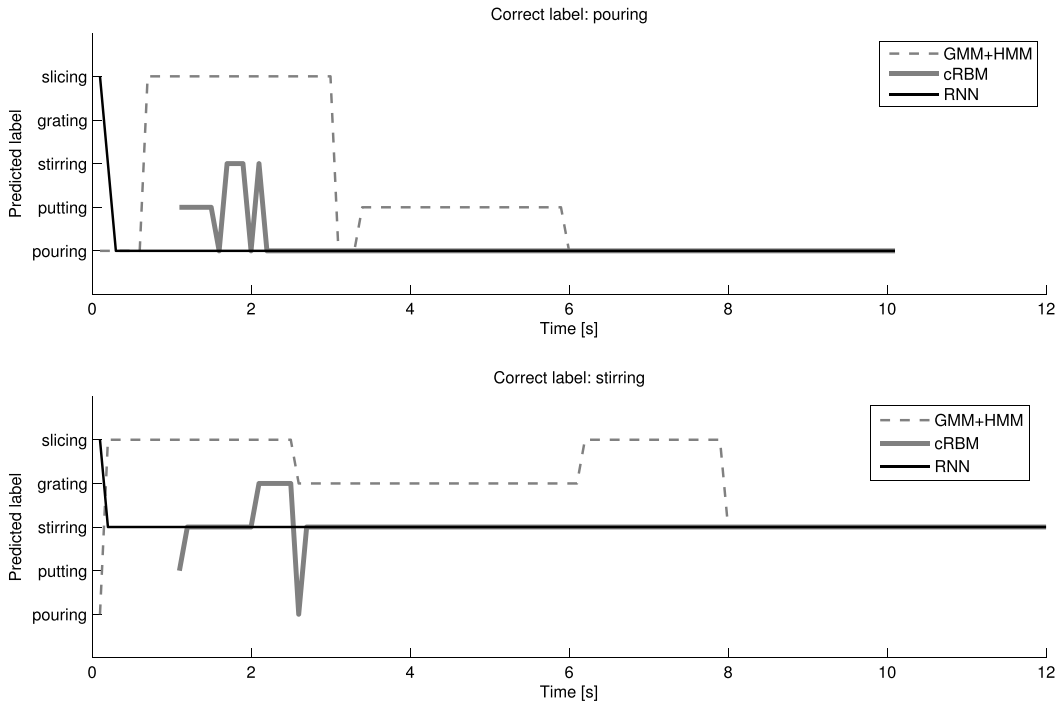

FIGURE 8 | Predicted labels at each time sample for three different models for one action of "pour" (*top figure*) and one action of "stir" (*bottom figure*) in the test set.

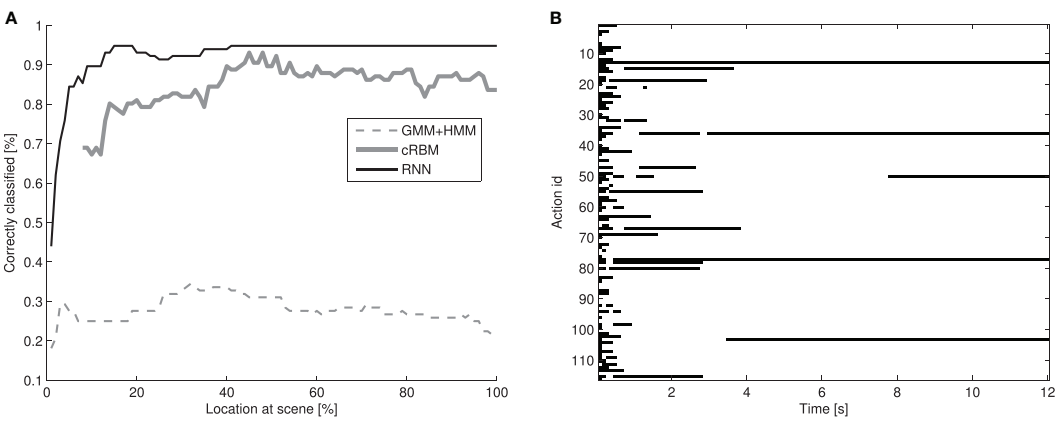

FIGURE 9 | (A) The average classification accuracy for three models at each percentage of all movement actions in the test set. **(B)** Misclassifications (black) for the LSTM for each action in the test set (y-axis) over time (x-axis).

most actions get the correct classification after a few seconds, with a couple of exceptions.

Most misclassified samples can be removed by a postprocess of classifying the full action as the average of all predicted samples in that action. The confusion matrix for the average action classification for the cRBM and LSTM can be seen in **Table 2**. Averaging the result of the HMM gave worse results since the majority of the samples are misclassified. The cRBM is capable of correctly classifying all "put" and "slice" actions. The LSTM has some confusion classifying "put" actions as "put" but instead correctly classifies all the actions "pour," "stir," "grate," and "slice." The overall accuracies when action averaging is used for the cRBM and LSTM are 91.38 (106/116) and 94.83% (110/116), respectively.

These results show that representational learning algorithms, such as cRBM and LSTM, are capable of learning and classifying action sequences from raw object tracking sensor data and outperforms traditional sequential methods, such as an HMM. The results of the HMM could be improved by discretizing the observations into more higher level movements that last longer than a single sample point. However, defining and categorizing such

TABLE 2 | Confusion matrices for the average classification for the (A) cRBM and (B) LSTM.

True label	%	Predicted label				
		Pour	Put	Stir	Grate	Slice
(A) cRBM	Pour	22	0	1	0	0
	Put	0	20	0	0	0
	Stir	2	5	12	0	0
	Grate	0	1	0	17	0
	Slice	0	0	0	0	36
(B) LSTM	Pour	23	0	0	0	0
	Put	0	14	6	0	0
	Stir	0	0	19	0	0
	Grate	0	0	0	18	0
	Slice	0	0	0	0	36

movements would require human manual effort and expertise and would have to be re-evaluated if new motions were added. The advantage of the cRBM and the LSTM is that the representations are learned directly from the raw data and adding new motions only require a retraining of the algorithm. However, comparing the two representational learning algorithms, and when action averaging is used, the cRBM can achieve comparable results to the LSTM. Nonetheless, if a fast classification is desired, then the LSTM is preferable. The advantage of the LSTM is that there is no fixed model order that the user needs to set, and output is produced at every time step, while the cRBM needs to use the first samples to initialize the model. The memory of the LSTM is also not limited to a specific model order like the cRBM and is, therefore, capable of modeling longer and varying temporal sequences. A disadvantage of the LSTM is that the training procedure is more complicated since the algorithm uses back-propagation through time and the LSTM cell is more computationally heavy, which results in a longer training time (90 min for the chosen LSTM structure compared to 40 min for the chosen cRBM structure).

4.4. Using Action Learning to Improve Anchoring

Once a generalized representation of an action is learned, the representation can then feedback into the anchoring system as illustrated in **Figure 1**. To motivate the need for this feedback loop, we refer to the object classification procedure in Section 3.1. Recall that the result from the object classification was a category name with an associated probability of a match against the Caffe framework, which has been trained with 1,000 of object categories. Herein lies the trade-off of using large-scale sources of information for object classification. On the one hand, we can move away from toy scenarios for anchoring and cope with many objects. On the other hand, the object classification task becomes increasingly difficult.

To illustrate, consider both **Figures 10A,B**. These figures illustrate two frames during a "`pouring`" and a "`stirring`" scenario, respectively. Due to uncertainties in the sensor data, occlusions by the human, and changes in environmental conditions, throughout the each scene, the objects to be anchored vary in number and type as illustrated by the results of the of the Caffe framework throughout each seen, seen in **Figures 10C,D**. Nonetheless, the anchoring management system manage to handle these kinds of uncertainties to some extent as illustrated by **Figures 10E,F**, which shows the results of the anchoring management system taking into account the output from the functionalities (*acquire*, *re-acquire*, and *track*) and subsequent threshold values. Recall that the anchoring management system both track objects on the sensor level (through the *track* functionality) and maintains the most accurate representation of anchored objects on a higher level (through the *acquire* and *re-acquire* functionalities). Hence, the resulting classification accuracy, seen in **Figures 10E,F**, will increase as the system perceives better representations of observed objects. Moreover, to compensate for "false-positive" anchors, as the result of glitches and swift movements, our anchoring management system has further adopted the idea of *deletion of anchors*, initially presented by Coradeschi and Loutfi (2008). Hence, a new anchor is acquired with a lifespan (set to 2.5 *s* for this work). This lifespan is then decreased for every time *t* that the anchor is either not tracked or reacquired until the lifespan has reached a zero and the anchor is deleted.

Nonetheless, errors are present in the output from the anchoring system. In the scenario exemplified in **Figure 10A**, the RBG-D sensor was not able to detect the transparent middle part of the "`beer bottle`" object, and the anchoring system has, therefore, created two anchors for the "`beer bottle`" object. In the scenario exemplified in **Figure 10B**, the human intervention is interfering with the scene to the extent that the *track* of the original "`wooden spoon`" object is lost and a new and "`nematode`" object are *acquired* instead. Our approach here is to resolve these errors by (1) using the action learning system to determine which new action is being performed on the objects and (2) leveraging from the symbolic representation of actions and anchors to query whether the particular anchors afford those actions. On the basis of the results, we can then refine the anchoring and matching process to take into account the set φ^{action} of *volatile attributes* to resolve ambiguities in the object classification.

To exemplify, consider the different combinations of anchored objects and the predicted classification label of the perceived action in each scenario, seen in **Figures 11A,B**, together with the probability certainties of the top predicted classes shown in **Figures 11C,D**. By the results for the "`pouring`" action, seen **Figure 11A**, it is seen that the correct action label is conclusively predicted if the anchor that is labeled "`coffee mug`" is combined with either of the "`beer`" bottle anchors. Furthermore, the probability certainties of the classified action labels, seen in **Figure 11C**, shows a high certainty for each combination of "`beer bottle`" + "`coffee mug`" for being a "`pouring`" action. It is also shown that the probability for both combinations of "`beer bottle`" + "`coffee mug`" is increasing during the action and is the highest at the end of the action, which is not the case a combination of "`beer bottle`" + "`beer bottle`," which fails to correctly classify the action and has a varying probability certainty during the action and is the lowest at the end of the action. Hence, we can, in this case, assume that it in fact is the same "`beer bottle`" (rather than two separate objects involved in two separate actions at the same time and approximately the same locations in 3D spatial space). However, by examining the

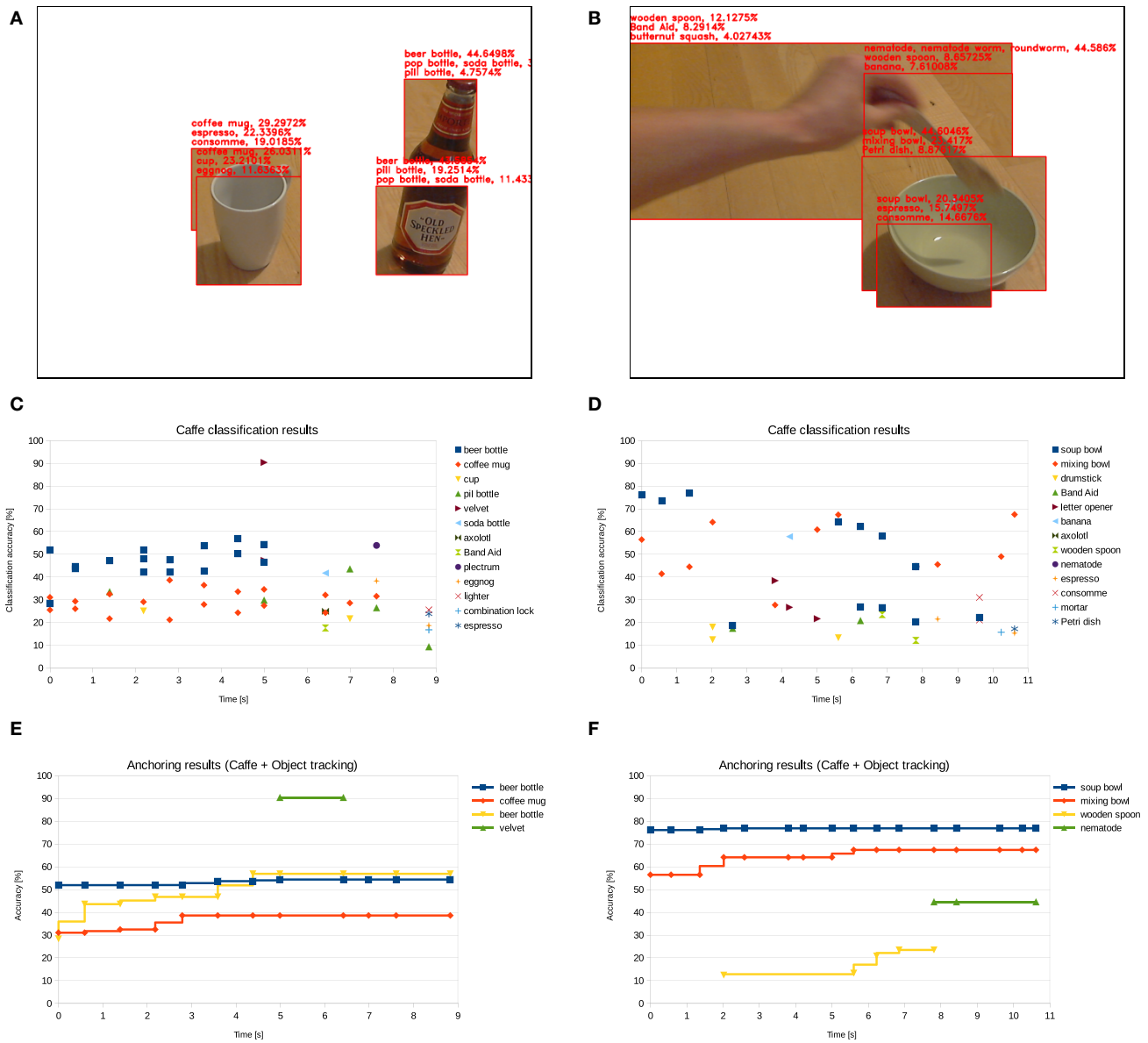

FIGURE 10 | Example frames of a "pouring" action (A) and a "stirring" action (B). Classification results over the whole observed "pouring" action (C) and "stirring" action (D), given by the Caffe framework. Corresponding resulting anchors for each "pouring" action **(E)** and "stirring" action **(F)**, respectively.

results for the "stirring" action, seen **Figures 11B,D**, it is not initially evident that the correct action label is classified. None of the individual combinations of "wooden spoon" + "soup bowl" or "nematode" + "soup bowl" are classified as a "stirring" action. This is not a surprising result since the anchoring system loses *track* of the "wooden spoon," as seen in **Figure 10F**. The use of only "soup bowl" + "nematode" fails to correctly classify the action since the "nematode" anchor did not appear until $t = 8\,s$ and did not see the "wooden spoon" being "put" into the "soup bowl" and therefore predicts the action as "slicing." However, if we instead consider the full movement trajectory of a combination of both perceived objects ("wooden spoon" + "nematode"), the probability certainty for the "soup bowl" + ("wooden spoon" + "nematode") has two spikes: the first when the spoon is "put" into the bowl and the second when the spoon starts to stir and at that time changes the predicted label to "stirring." However, the question remains, which object is involved in the action: the "wooden spoon" or the "nematode"?

To resolve this ambiguity, we have to rely on the symbolic representation and other sources of information. More specifically, to query whether a particular anchor affords an action, the ConceptNet 5[5] semantic network is utilized (Speer and Havasi, 2012). The ConceptNet network consists of commonsense relational knowledge between concept (word or phrases). Hence, this

[5]http://conceptnet5.media.mit.edu/.

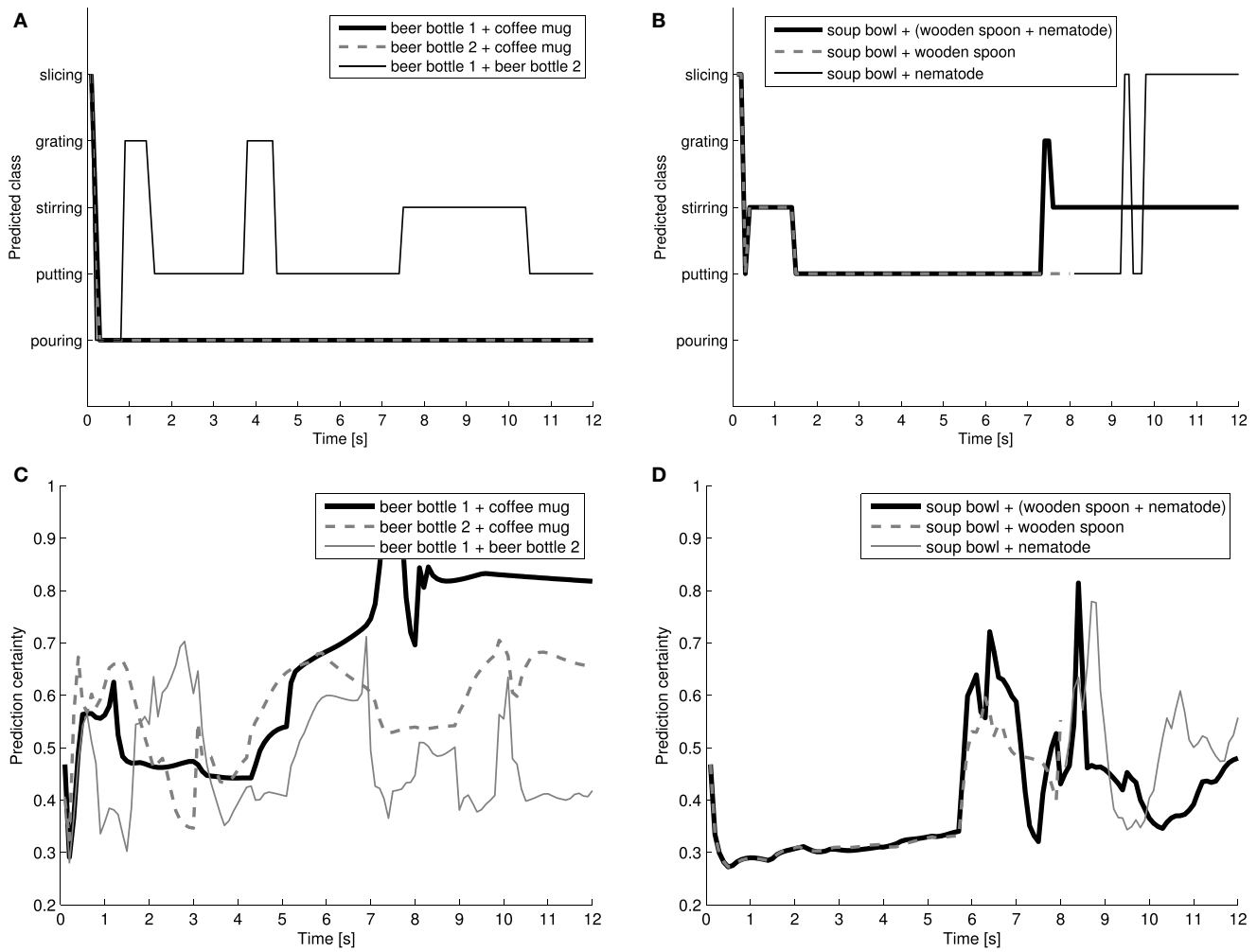

FIGURE 11 | Predicted labels from different combinations of anchors for the action of (A) "`pouring`" and (B) "`stirring`." The corresponding probability of the top predicted class is showed in **(C,D)** for pouring and stirring, respectively.

network is also suitable for queries of affordances. Now, as we start to perceive an action and start to predict an action label, as the scenario progress over time, the predicted action label can simultaneously be used for query the ConceptNet semantic network. The result of this prediction is not only a grounded symbolic name for a novel action, but the same prediction is also an estimation of possible object candidates that afford that action. The result of such query is presented in **Table 3**, where it is evident that "`wooden spoon`" affords a "`stirring`" action. This affordance can be further confirmed through query the ConceptNet for the association between the concept "`soup bowl`" and "`wooden spoon`," which results in a similarity score of 46.28%. The same is not true for a "`nematode`," which has a zero similarity if associated with a "`soup bowl`," and therefore, we can favor the "`wooden spoon`" by the knowledge that it affords a "`stirring`" action.

TABLE 3 | The top 10 responses for a query of the concept "stir" in ConceptNet.

Concept	Relation
wooden_spoon	[a wooden spoon] is for [stirring]
stirrer	[stirrer] is related to [stir]
pot	[a pot] is for [stirring]
spoon	You can use [a spoon] for [stirring]
bestir	The word [bestir] etymologically comes from the word [stir]
storm	The word [storm] etymologically comes from the word [stir]
band	[Stir] is an instance of [band]
champagne_whisk	[champagne whisk] is related to [stir]
elt	[elt] is related to [stir]
lather	[lather] is related to [stir]

5. CONCLUSION AND FUTURE WORK

In this paper, we have outlined our work on the topic of learning actions from representations of object entities that are tracked and maintained, in both space and time. We leverage from notions within perceptual anchoring to maintain and use percept-symbol information. We have both presented an extension of the perceptual anchoring problem that accounts for anchoring of an attribute history (such as positions of an object over time) and introduced a framework that can handle the stream of sensor data that is

required for tracking the movements of objects. We have presented an evaluation of our work, where a probabilistic generative model is used to learn high-level action concepts from anchored objects. We have performed a signal and model parameter selection and shown that a good trade-off between the performance of the classification accuracy and the training time could be achieved by using a subset of the signals and a lesser amount of hidden units and model order. Finally, we have presented how the resulting model of learned action can be used to improve anchoring. For this purpose, we have exemplified how a predicted action can be utilized, with the extension of commonsense semantic knowledge, to estimate which objects afford the predicted action.

A drawback of the framework used for the presented work was that the data acquisition (through the anchoring system) and the action learning (through the learning system) were conducted on (physically) separate systems, where the data had to be processed manually in between the systems. In future work, we will fully integrate both systems into one single system for the purpose of learning generalized representations of object actions in a larger scale and over a length of operation (and without the human intervention). The overall future goal is consequently to integrate such learned representations, formally, into the definition of perceptual anchoring such that a matching procedure can be used to establish both what types of actions that apply to an object and what kinds of effects an action will have on the surrounding environment. From the results, presented in Section 4.3, it is evident that the representational learning algorithms, such as cRBM and LSTM, are the most prominent candidates for such integration. An advantage of RNNs compared to cRBMs is that they do not have a model parameter for the model order, i.e., the number of previously visible layers that affects the currently hidden layer. However, an RNN can be difficult to train for learning long-term dynamics. The RNN with an LSTM cell was used in this work to facilitate this. As an alternative, a long-term recurrent convolutional network (LRCN) has recently been proposed (Donahue et al., 2015). LRCN has previously been used to generate descriptions of images and videos by using CNNs and RNNs for the visual and the sequence learning. The difference to our work is that we used the tracking of objects represented by time series as input instead of a single image or a sequence of images. However, the LRCN model is also integrated with the Caffe framework (Jia et al., 2014), which is used for the object classification procedure of our anchoring framework. This integration favors the LRCN model as a prominent candidate for further work.

The tracking functionality of the suggested approach (presented in Section 3.2) was adequate for table top scenarios with moving objects, which we have explored in this work. However, our anchoring approach will not be able to handle, with the same frame rate, more complex scenarios that entail tracking of multiple objects of a higher order, as seen in **Figure 6**. Multiobject tracking in the context of anchoring is, therefore, a topic that needs further attention in future work. For example, a possible future direction is to follow a recent trend of object tracking with the use of graphical processing unit (GPU), such as the utilization of a similar particle filter-based tracking approach for a GPU architecture, as presented by Choi and Christensen (2013). Another notable GPU-based alternative is the use of model-based combined pose detection and tracking as suggested by Pauwels et al. (2015), which uses the graphical and computational capability of a GPU for the purpose of combine dense motion and depth cues with sparse keypoint correspondences to maintain a scene model. Moreover, a common approach for learning both the interaction that involves objects (Oikonomidis et al., 2013; Kyriazis and Argyros, 2014) and object affordances (Koppula et al., 2013; Koppula and Saxena, 2014), from RGB-D data, is to utilize a tracking algorithm for tracking the hand movement of a human user. The learning method that we have suggested in this work is solely based on the tracked movements of object entities. The human hand movement can consequently even become a problem in some cases, as described in Section 4.1. However, a possible direction of future work could follow a similar philosophy of tracking human hand movements and thereby "anchoring" the human user in addition to the anchoring of objects.

A benefit of using the Caffe framework (Jia et al., 2014), with a pretrained model, is that such implementation requires minimum resources (implementation-wise), while still providing a significant possibility of object categories. However, all pretrained categories of a model might not be relevant for a particular domain, e.g., for a household domain. Hence, another aspect of further attention is to improve the visual categorization by extending the 1 K ILSVRC-2012 model (Krizhevsky et al., 2012; Donahue et al., 2013), with more categories and fine-tuning the model according to categories that are more relevant for a domestic robot. A similar extension of the Caffe framework is addressed together with the introduction of the open-vocabulary object retrieval system, presented by Guadarrama et al. (2014). However, contrary to the open-vocabulary work, which is approaching the object retrieval problem *top-down* (i.e., semantic symbols from language queries were projected to objects in the currently perceived scene), the work presented in this paper is approaching the problem *bottom-up*. Another important aspect covered in work on the open-vocabulary system is the distinction between category-level semantics (e.g., "a coffee mug") and instance-level semantics (e.g., "my coffee mug"). The difference between category- and instance-level semantics is likewise an important aspect of perceptual anchoring, and a similar categorization is, therefore, an aspect that will be further studied in future work.

AUTHOR CONTRIBUTIONS

AP has outlined the extension of the framework to include volatile properties, performed experiments, and implementation of the framework, ML has developed the learning algorithms and performed the experimental validation with AP, and AL has developed the notions in perceptual anchoring and the ideas in the paper together with the other authors.

ACKNOWLEDGMENTS

This work has been supported by the Chist-Era ReGround project, and by the Swedish Research Council (Vetenskapsrådet) under grant number: 2016-05321.

REFERENCES

Aksoy, E. E., Abramov, A., Dörr, J., Ning, K., Dellen, B., and Wörgötter, F. (2011). Learning the semantics of object–action relations by observation. *Int. J. Robot. Res.* 30, 1229–1249.

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

Browatzki, B., Tikhanoff, V., Metta, G., Bülthoff, H. H., and Wallraven, C. (2012). "Active object recognition on a humanoid robot," in *Robotics and Automation (ICRA), 2012 IEEE International Conference*, (IEEE), 2021–2028.

Chiu, C.-C., and Marsella, S. (2011). "A style controller for generating virtual human behaviors," in *The 10th International Conference on Autonomous Agents and Multiagent Systems*, (Taipei, Taiwan: International Foundation for Autonomous Agents and Multiagent Systems), 3.

Choi, C., and Christensen, H. I. (2013). "RGB-D object tracking: a particle filter approach on GPU," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference*, (Tokyo), 1084–1091.

Coradeschi, S., and Loutfi, A. (2008). "Chapter 15: A review of past and future trends in perceptual anchoring," in *Tools in Artificial Intelligence*, ed. P. Fritzche (I-Tech Eduacation and Publishing).

Coradeschi, S., Loutfi, A., and Wrede, B. (2013). A short review of symbol grounding in robotic and intelligent systems. *KI-Künstliche Intelligenz* 27, 129–136. doi:10.1007/s13218-013-0247-2

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.

Coradeschi, S., and Saffiotti, A. (2003). An introduction to the anchoring problem. *Robotics and Autonomous Systems* 43, 85–96. Special issue on perceptual anchoring.

Daoutis, M., Coradeschi, S., and Loutfi, A. (2012). Cooperative knowledge based perceptual anchoring. *Int. J. Artif. Intell. Tools* 21:1250012. doi:10.1142/S0218213012500121

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., et al. (2015). "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., et al. (2013). DeCAF: a deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*.

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

Graves, A., and Jaitly, N. (2014). "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, Vol. 14, Beijing, China, 1764–1772.

Graves, A., Mohamed, A., and Hinton, G. (2013). "Speech recognition with deep recurrent neural networks," in *The 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, (IEEE), 6645–6649.

Graves, A., and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Netw.* 18, 602–610. doi:10.1016/j.neunet.2005.06.042

Guadarrama, S., Rodner, E., Saenko, K., Zhang, N., Farrell, R., Donahue, J., et al. (2014). "Open-vocabulary object retrieval," in *Robotics: Science and Systems*, eds D. Fox, L. E. Kavraki, and H. Kurniawati.

Hahn, E. R., and Gershkoff-Stowe, L. (2010). Children and adults learn actions for objects more readily than labels. *Lang. Learn. Dev.* 6, 283–308. doi:10.1080/15475441003635315

Harnad, S. (1990). The symbol grounding problem. *Physica D* 42, 335–346. doi:10.1016/0167-2789(90)90087-6

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comput.* 14, 1771–1800. doi:10.1162/089976602760128018

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi:10.1162/neco.1997.9.8.1735

Holzer, S., Rusu, R. B., Dixon, M., Gedikli, S., and Navab, N. (2012). "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*, (Algarve, Portugal), 2684–2689.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., et al. (2014). Caffe: convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Kingma, D. P., and Ba, J. (2014). Adam: a method for stochastic optimization. *CoRR*. abs/1412.6980. Available at: http://arxiv.org/abs/1412.6980

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

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 *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, (Cham, Switzerland: Springer International Publishing), 831–847.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, eds F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc), 1097–1105.

Kulić, D., and Nakamura, Y. (2011). "Incremental learning of full body motion primitives" in *From Motor Learning to Interaction Learning in Robots*, eds O. Sigaud and J. Peters (Berlin, Heidelberg: Springer Berlin Heidelberg), 383–406.

Kyriazis, N., and Argyros, A. A. (2014). "Scalable 3D tracking of multiple interacting objects," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference*, (IEEE), 3430–3437.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognit. Lett.* 42, 11–24. doi:10.1016/j.patrec.2014.01.008

Lei, J., Ren, X., and Fox, D. (2012). "Fine-grained kitchen activity recognition using RGB-D," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp '12*, (New York, NY: ACM), 208–211.

Lemaignan, S., Ros, R., Sisbot, E. A., Alami, R., and Beetz, M. (2012). Grounding the interaction: anchoring situated discourse in everyday human-robot interaction. *Int. J. Soc. Robot.* 4, 181–199. doi:10.1007/s12369-011-0123-x

Lenat, D. B. (1995). Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM* 38, 33–38. doi:10.1145/219717.219745

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 *Proceedings of the 19th IJCAI Conference*, (Edinburgh, UK), 1477–1482.

Luo, D., Han, X., Wang, Y., and Wu, X. (2014). "Modelling and generalizing achieved robot skills with temporal restricted Boltzmann machines," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference*, (IEEE), 835–840.

Mareschal, D., and Johnson, M. H. (2003). The what and where of object representations in infancy. *Cognition* 88, 259–276. doi:10.1016/S0010-0277(03)00039-8

Marocco, D., Cangelosi, A., Fischer, K., and Belpaeme, T. (2010). Grounding action words in the sensorimotor interaction with the world: experiments with a simulated icub humanoid robot. *Front. Neurorobot.* 4:7. doi:10.3389/fnbot.2010.00007

Oikonomidis, I., Kyriazis, N., and Argyros, A. (2013). Tracking the articulated motion of human hands in 3D. *ERCIM News* 2013, 95.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML* 28, 1310–1318.

Pauwels, K., Rubio, L., and Ros, E. (2015). Real-time pose detection and tracking of hundreds of objects. *IEEE Trans. Circuits Syst. Video Technol.* 26:2200–2214. doi:10.1109/TCSVT.2015.2430652

Rabiner, L., and Juang, B. (1986). An introduction to hidden markov models. *IEEE ASSP Mag.* 3, 4–16. doi:10.1109/MASSP.1986.1165342

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

Speer, R., and Havasi, C. (2012). "Representing general relational knowledge in conceptnet 5," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, (Istanbul, Turkey), 3679–3686.

Stramandinoli, F., Cangelosi, A., and Marocco, D. (2011). "Towards the grounding of abstract words: a neural network model for cognitive robots," in *Neural Networks (IJCNN), The 2011 International Joint Conference*, (IEEE), 467–474.

Stramandinoli, F., Marocco, D., and Cangelosi, A. (2012). The grounding of higher order concepts in action and language: a cognitive robotics model. *Neural Netw.* 32, 165–173. doi:10.1016/j.neunet.2012.02.012

Sun, Y., Bo, L., and Fox, D. (2013). "Attribute based object identification," in *Robotics and Automation (ICRA), 2013 IEEE International Conference*, (IEEE), 2096–2103.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, (Washington, USA), 1017–1024.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Advances in Neural Information Processing Systems, NIPS'14*, (Cambridge, MA, USA: MIT Press), 3104–3112.

Taylor, G. W. (2009). *Composable, Distributed-State Models for High-Dimensional Time Series*. Ph.D. thesis, Department of Computer Science, University of Toronto.

Taylor, G. W., Hinton, G. E., and Roweis, S. T. (2007). "Modeling human motion using binary latent variables," in *Advances in Neural Information Processing Systems*, Vol. 19, eds B. Schölkopf, J. Platt, and T. Hoffman (Vancouver, Canada: MIT Press), 1345–1352.

Tenorth, M., and Beetz, M. (2013). Knowrob: a knowledge processing infrastructure for cognition-enabled robots. *Int. J. Robot. Res.* 32, 566–590. doi:10.1177/0278364913481635

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

Ye, C., Zhao, C., Yang, Y., Fermüller, C., and Aloimonos, Y. (2016). "Lightnet: a versatile, standalone matlab-based environment for deep learning," in *Proceedings of the 2016 ACM on Multimedia Conference*, (New York, NY: ACM), 1156–1159.

Conflict of Interest Statement: 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.