Discovering Action Primitive Granularity from Human Motion for Human-Robot Collaboration

Elena Corina Grigore
Yale University, Department of Computer Science
New Haven, CT, 06511
Email: elena.corina.grigore@yale.edu

Brian Scassellati
Yale University, Department of Computer Science
New Haven, CT, 06511
Email: brian.scassellati@yale.edu

***Abstract*—Developing robots capable of making sense of their environment requires the ability to learn from observations. An important paradigm that allows for robots to both imitate humans and gain an understanding of the tasks people perform is that of action primitive discovery. Action primitives have been used as a representation of the main building blocks that compose motion. Automatic primitive discovery is an active area of research, with existing methods that can provide viable solutions for learning primitives from demonstrations. However, when we learn primitives directly from raw data, we need a mechanism to determine those primitives that are appropriate for the task at hand: is *brushing one's teeth* a suitable primitive or are the actions of *grabbing the toothbrush*, *adding toothpaste onto it*, and *executing the brushing motion* better suited? It is this level of granularity that is important for determining well-suited primitives for applications. Existing methods for learning primitives do not provide a solution for discovering their granularity. Rather, these techniques stop at arbitrarily chosen levels, and often use clear, repetitive actions in order to easily label the primitives. Our contribution provides a framework for discovering the appropriate granularity level of learned primitives for a task. We apply our framework to action primitives learned from a set of motion capture data obtained from human demonstrations that includes hand and object motions. This helps find a well-suited granularity level for our task, avoiding the use of low levels that don't capture the necessary core pattern in the actions, or high levels that miss important differences between actions. Our results show that this framework is able to discover the best suited primitive granularity level for a specific application.**

I. Introduction

Robotic systems deployed in industry today work in isolation from humans, and perform precise, repetitive tasks based on well-defined plans and known task structures. A great deal of robotics research has focused on how to develop adaptive systems, capable of offering supportive behaviors to a person during a task [3]. To accomplish this, the robot needs to understand the state of the environment and how it is changing. For complex tasks, state changes encompass both the movements of the person, and the motions of the objects manipulated by the human. To this end, there has been a great deal of attention on learning from observations of a person performing a particular task. Demonstrations can be leveraged to both teach the robot how to imitate certain human motions we wish to replicate, and to allow the robot to learn about the structure of the task and its progression.

To handle the complexity of human motion, researchers have formalized it to be composed of basic units of action, called action or motor primitives. Primitives can be sequenced or combined to generate more complex behavior [30]. A rich area of research focuses on learning action primitives from raw data, in order to avoid the need for hand-crafting, and to allow for scalability. The goal is to use these learned primitives in high-level planners that typically assume that a library of primitives is available for them to build upon [10, 32].

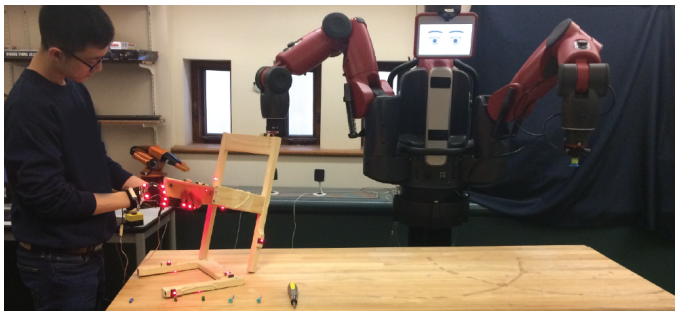

Fig. 1: Sample assembly during data collection. The motion capture system records users' hand movements (via gloves fitted with sensors), and the trajectories of the chair components.

Obtaining clear and useful primitives necessary for existing planners is difficult and depends on the task at hand. If our task is to observe a person's general morning routine, we might benefit from primitives like *brush teeth*, *get dressed*, etc. However, what if we instead wish to have a robot learn how to imitate the person's physical movements during this routine? In that case, primitives like *grab toothbrush*, *add toothpaste*, *execute brushing motion* might be better suited, depending on the robot's physical capabilities. Other situations also show the need to investigate this level of granularity. For example, using primitives with low granularity such as *press gas pedal* or *release gas pedal* when driving might prove to be useful since the motion of pressing the pedal has an easily identifiable signature. Using a low level for a primitive like *move cup to mouth* when observing a person drinking, on the other hand, might cause problems due to the variability in the human's motion when executing this action.

Our work aims to find a solution to the problem of discovering an appropriate level of granularity of primitives for the task at hand. Existing methods of learning primitives from observations do not tackle this problem. Such techniques often aim to learn primitives that are clear and easy to

label, such as *running*, *jogging* or other repetitive motions, or investigate robot motion, which is inherently different from human movement. In order to make use of existing techniques for learning primitives and use these as inputs for higher-level planners that have already been developed, discovering the appropriate level of granularity is of great importance.

In this paper, we present a framework for primitive granularity discovery, and implement it for a human-robot collaboration (HRC) scenario involving the complex task of assembling an IKEA chair. We record a set of 28 participants performing our task, using a motion capture system to acquire both hand motions and object movements. Fig. 1 shows a participant putting together the chair during our data collection phase. We learn primitives in the data sets via a beta process hidden Markov model (BP-HMM) [13]. Of the learned primitives, we set out to discover the appropriate level of granularity for object *pick up*, a widely used primitive in robotics. We use task-specific factors in order to establish reasonable minimum and maximum levels of granularity, and use these levels to test what classification performance we obtain when distinguishing them from the rest of the learned primitives. We then aim to find the lowest granularity level with high classification performance. Our insight comes from the observation that high granularity levels can provide us with good classification performance when distinguished from the rest of the primitives, yet they can hide multiple subclasses of primitives that might be relevant to the considered task. Of course, when we reach a level that is too low, the classification performance will drop significantly, and thus this does not represent an appropriate level. Our results show that the presented framework successfully finds a well-suited granularity level for the *pick up* primitive within the context of our HRC scenario.

II. Related Work

Our work focuses on discovering the appropriate level of granularity for learned primitives from observations of human and object motion data. In this section, we first give a brief overview of the types of primitives high-level planners expect as their inputs. We then present relevant work in the area of action primitive discovery from motion data.

A. Primitives Expected by High-Level Planners

High-level planners can work with different levels of abstraction. Most planners assume as their input collections of primitives and do not concern themselves with the acquisition of these primitives. Furthermore, such planners do not specify the granularity level expected for the primitives they take in, and include information about this level only indirectly, through the handcrafted specification of the inputs. Different types of representations expected by planners include: action templates represented with preconditions and effects (e.g., "put-block-on-table(x y)" describes the action of moving block x from the top of block y to the table) [34], primitive skills based on actions that the agent can perform (i.e. specifying a start condition, a set of effects, and a set of executable actions) [26], or primitives with manually specified inputs and outputs (e.g., unscrew takes in a stud as input and returns a nut as output) [24]. Other work investigated symbolic descriptions for use in low-level environments for planning [17]. Although this technique goes beyond using handcrafted primitives, it also requires careful specifications of preconditions and effects of atomic components.

B. Primitive Action Detection

Understanding the motions that make up different tasks requires finding patterns in the data to facilitate the identification and characterization of actions. Thus, there has been considerable effort in the area of learning how to decompose motion data into atomic parts called motion or action primitives.

1) Methods and Approaches: One class of widely used techniques for the purpose of primitive detection is the Hidden Markov Model (HMM). An example of the use of this method is work that learns primitives from observations of human motions by first segmenting the motion via an HMM [19], and then applying another HMM over the found segments to learn the primitives as clusters [19]. Another example uses Gaussian mixture models together with HMMs in order to represent the primitives and the transitions between them, respectively [5]. Yet other work employed parametrized HMMs over clusters of segments detected based on changes in object state rather than looking at the full body motion [18].

Other techniques employed for primitive detection include using Orthogonal Matching Pursuit on basis functions that model time series data [20], propagation networks that work on activities represented as partially ordered intervals [31], spatio-temporal non-linear dimension reduction [15], and automatic methods for segmenting long sequences of data into segments that usefully identify different primitives [2]. Non-negative matrix factorization has also been utilized to perform the primitive detection part and then use the learned representation to predict future motions [11], as well as to predict linguistic labels given as input for every motion data observation [22]. Another method leveraged by work in this area is that of grammar-based motif discovery in order to identify trajectories corresponding to the reusable primitive actions in a narrated human demonstration [25].

2) Building on Action Primitives: Work that explores ways of clustering or detecting higher abstractions based on learned primitives include contributions from different fields. In communities like computer vision, much of the focus is placed on building up from learned primitives to intuitive behaviors that do not necessarily need to feed into higher-level planners, but are directly used by various applications, such as camera surveillance [23], or detection of actions, gestures and expressions [35]. This research makes contributions to building abstractions on top of learned primitives, but does not face the problem of needing to bridge the gap between such abstractions and inputs needed for higher-level planners. Hence, this work tackles the problem of finding primitive granularity levels only indirectly.

In robotics, there are a number of contributions that do look at what abstractions can be built onto learned primitives to

tackle tasks at higher levels. This work is more relevant to our aim of finding the granularity of learned primitives. In this area, Jenkins and Mataric [14] address defining behavior vocabularies of human motion data for movements that include punching, dancing, handwaving, and so on. Kulić et al. [19] look at mapping actions from full body human motion demonstrations of behaviors like raising arms and bending down to similar actions that a humanoid robot can perform. Other work focuses on learning about the structure of a task in a bottom-up approach that starts with automatic segmentation of learning from demonstration (LfD) data and continuing to build useful abstractions on top [16, 9, 6, 27].

The work of Jenkins and Mataric [14], and Kulić et al. [19] differs in nature from our work in that we consider different types of primitives that occur over the progression of a task and also take into account object trajectories, while the cited contributions consider actions that are repetitive in nature or look at the full body posture of a person. The LfD contributions cited above are more similar in nature to our attempts, but they consider robot motions, whereas we investigate human motion data in the present paper. Motions generated by humans and robots are inherently different. Thus, even when utilizing the same approaches and techniques for primitive discovery on both types of data, the learned human and robot primitives fit into the task differently.

Furthermore, our contribution focuses on a framework for discovering primitive granularity levels that are well-suited for the considered task. It delves into identifying appropriate granularity levels to an extent that the work cited above does not explore. Our work brings attention to the possible hidden structure that might exist in choosing an appropriate granularity level, even when obtaining good performance on classification tasks for the desired primitives.

III. Framework for Discovering Primitive Granularity Levels

In order to use high-level planners that expect well-defined primitives as their inputs, it is important to discover those primitives that (i) help capture the core patterns in the actions part of the task, and (ii) constitute the main building blocks of the observed motion without hiding further subclasses within. If the level we employ cannot capture the main differences we observe in the data, our reliable identification of primitives can be compromised. This is bound to happen with the choice of too low (i.e. fine-grained) of a granularity level, when this level is not sufficient to capture the common basic actions present in the task at hand. For example, if we choose a low primitive level such as *move cup to mouth*, this might prove problematic due to the different ways in which a person can move their hand with the cup to their mouth. On the other hand, if we choose too high (i.e. too coarse) of a level, this can end up obscuring deeper structure present in the primitives. This can prevent us from generalizing well from the data and from reusing learned primitives in an effective way when we build higher-level behaviors. An illustrative example here is *brush teeth*, where we might benefit from learning lower-level primitives like *grab toothbrush*, *add toothpaste*, and *execute brushing motion*. These primitives can then be used for higher-level actions, such as *brush teeth* itself, but also for an action like *clean tongue*, that might require the same *grab toothbrush* primitive. Our framework provides a method for discovering precisely this appropriate level of granularity for the considered application or scenario.

Framework 1 Discovering learned primitive granularity levels

1: **Data collection and preparation:**
2: Collect raw data from scenario relevant to the considered task, from which primitives are to be learned. Data should be collected in conditions that simulate as best as possible the desired scenario.
3: Divide data set into training and test sets, depending on desired method of validation.
4: **Learn action primitives from raw data:**
5: Identify an appropriate algorithm for learning primitives from the raw data.
6: Apply algorithm from 5 to training set.
7: Apply algorithm from 5 to test set, using the learned parameters from 6.
8: **Classification experiments:**
9: **Classification method:**
10: Identify an appropriate method for classifying time series composed of the learned primitives.
11: Apply method from 10 to training set.
12: **Primitive choice:**
13: Choose what primitive(s) to investigate.
14: Identify factors that engender different granularity levels of primitives found at step 13. These factors rely heavily on the task.
15: **Establish granularity level interval:**
16: Identify the lowest reasonable level, and apply method from 10 using the learned parameters from 11 to test whether good performance is obtained.
17: Identify the highest reasonable level, and apply method from 10 using the learned parameters from 11 to test whether good performance is obtained.
18: **Find final granularity level:**
19: Decrease level and test with lower and lower levels starting from the highest, until a significant decrease in classification performance occurs.
20: Choose the lowest level with high classification performance as well-suited for the task: (i) high performance for high levels can hide different subtypes of primitives relevant to the task; (ii) when the significant decrease in performance occurs, it likely indicates that the level with poor performance is too low, and the one with high performance is appropriate.

We introduce our framework for primitive granularity discovery in Framework 1. This outlines the general methodology to be applied for determining appropriate levels of granularity

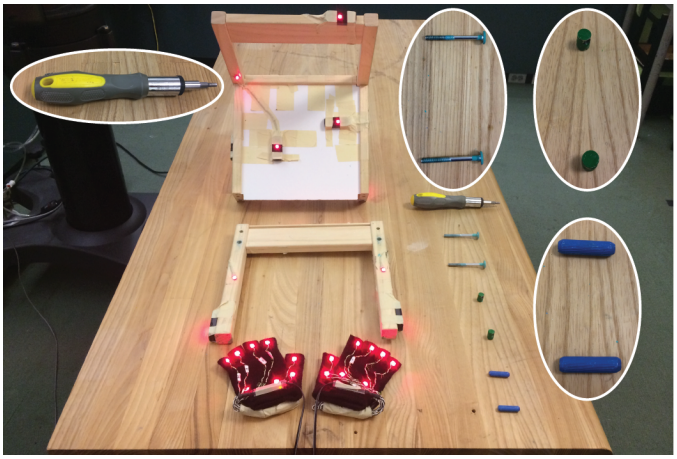

Fig. 2: The components part of the assembly task: a frame, a main component, two pegs (blue), two nuts (green), two screws (cyan), and a screwdriver (yellow). We also include the gloves participants wear for tracking hand movements.

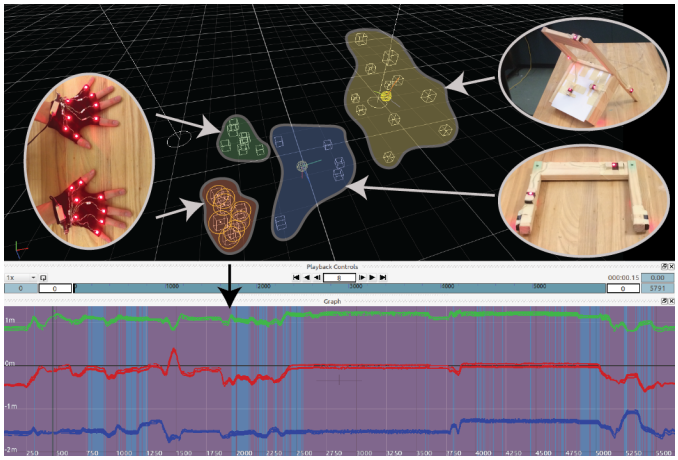

Fig. 3: Screen shot of the motion capture system software highlighting: (i) green—the group of eight individual markers mounted on the left hand glove, (ii) orange—the group of eight individual markers mounted on the right hand glove, (iii) blue—-the individual markers mounted on the frame, together with the center of the frame rigid body, and (iv) yellow—the individual markers mounted on the main component, together with the center of the main component rigid body. We show the individual markers visible at the time of the screen capture, with the two rigid body centers still being tracked even with marker occlusions. We also include the (x, y, z) coordinates (red, green, and blue, respectively) of the right hand markers recorded during an assembly session. The x, y, and z signals show the values for all eight markers, highlighting the high correlation present for markers on the same glove.

for a considered task. In Section IV, we introduce the experimental setup we employed for our framework, and in Section V, we present the implementation of each framework step.

IV. Experimental Setup

In this paper, we consider the task of assembling part of an IKEA chair. We use a children's chair for ease of tracking and moving objects around in the workspace, but we choose one that requires complex assembly, including several small components and the need to use a tool. We consider part of the assembly task, namely attaching the front frame onto the rest of the chair. This part contains similar steps as those required for the rest of the assembly. It is composed of the necessary and sufficient actions for showcasing all the different types of objects and movements that are part of the full chair building process. Fig. 2 shows all the components of the task: the front frame (which we refer to as the frame herein), the rest of the chair (which we refer to as the main component herein), two pegs, two nuts, two screws, and a screwdriver. To complete the task, participants need to: (i) place both pegs in either the frame or the main component, (ii) place both nuts, in any order, in the sides of the main component, (iii) attach the frame onto the main component, (iv) place the screws into the two holes of the frame, and (v) twist the screws in.

The motion capture system we employ throughout our data recording sessions is PhaseSpace [28], which utilizes active LED markers as its tracking technology. To acquire motion capture data, we hand fashioned a pair of gloves specifically for participants to be able to manipulate all the different objects during this task. We used fitted, stretching material gloves without the tips of the fingers, which we mounted with eight sensors each for tracking. We also tracked the frame and the main component. Fig. 3 presents a screen shot of the motion capture system software, highlighting the gloves and chair components as tracked by PhaseSpace. Each glove is fitted with eight sensors, and thus the two groups of eight sensors represent the participant's left and right hand, respectively. For the chair components, we created rigid bodies. A rigid body represents a collection of markers that are fixed relative to each other. This allows for regarding the collection as one tracking target and making the tracking more robust to individual marker occlusions. Fig. 3 includes the two rigid bodies created for the frame and the main component, respectively. For each, we utilized the cartesian coordinates of the rigid body center, which allowed us to effectively track each of the two objects as single points in the workspace. We used redundant sensors placed on different planes, to make the tracking robust to occlusions occurring when participants would place the objects on the table in different positions, or cover some of the sensors. We did not create rigid bodies for the gloves, since these markers did not maintain a fixed position relative to each other when participants' hands moved.

We employ this experiment to emulate the complexities of assembly tasks humans engage in. We impose no restrictions on participants about how to perform the assembly. This provides us with a thorough understanding of what different levels of granularity we need to consider for primitives relevant to intricate HRC tasks. For such tasks, even primitives as intuitive sounding as *pick up* are not trivially defined. Given that we employ different types of objects, each with a different result on the motion data signal, it is not immediately apparent what level of granularity to use for such a primitive. In our scenario,

pick up can be applied to a peg, a nut, a screw, the screwdriver, the frame, or the main component. The granularity level here refers to what types of object *pick up* we label as positive examples of this class of primitive. A low granularity level for our task would involve distinguishing between different small objects, like pegs, nuts, screws, and the screwdriver, while a high level would involve treating pick ups for all the small and big objects as positive examples of this class.

V. Framework Implementation for HRC Scenario

In this section, we detail the implementation of the presented framework for the considered HRC scenario.

A. Data Collection and Preparation

We present the data collection phase that corresponds to lines 1–3 in Framework 1. To implement line 2, we recorded data from a total of 28 participants partly assembling a children's IKEA chair (mounting the frame onto the main component of the chair). We recorded each assembly as a different trial, with five-to-ten trials per participant, for a total of 158 trials across the data set. Trials varied in length, from $57''$ to $4'21''$, with an average length of $2'10''$. The motion capture system recorded at a rate of approximately $1.25\,\mathrm{Hz}$.

During each trial, we recorded motion capture data for each of the 16 markers mounted on the gloves (eight markers on each glove) and all the markers mounted on the rigid bodies. We considered (x, y, z) cartesian coordinates for the glove markers and the centers of the two rigid bodies. For each glove, we averaged over the eight values corresponding to the eight markers each was fitted with, resulting in an average (x, y, z) signal for each hand. We did so given the fact that the information in the signals from the different markers was redundant. This provided us with a good representation for each hand. Our choice is supported by a principal component analysis (PCA) performed on the signals of the eight markers for the right hand, and similarly, on those of the eight markers for the left hand. For each hand, we performed PCA three times, for the x, y, and z signals. We included the average of the eight markers for the dimension in question, resulting in nine total variables for each dimension. For each dimension, the PCA found that the principal component composed of all nine variables approximately equally weighted accounted for at least 89% of the variance. This motivates our choice of representing each dimension with the average of the eight markers for one hand, showing the nine variables within one dimension are highly correlated.

To implement line 3 of our framework, we divided our data into a training set and a test set. Out of our total of 158 time sequences, we leave out a total of 28, one per person. We apply the action primitive algorithm on the training set, and then apply the estimated parameters of the model to the 28 held-out sequences to mimic learning primitives in real-time, after the model parameters have been learned. The primitive classification is applied in a similar manner, training the classifier on the training set, and using the learned parameters to mimic classifying on real-time data. We note here that each held-out trial per participant was randomly selected, and that each was segmented in accordance with the description in Subsection V-C1. This resulted in approximately 25 windows per each of the 28 held-out sequences. The results that we present in Section VI constitute test results across the different 28 participants, supporting a strong generalization.

B. Learning Action Primitives from Raw Data

The approach we use to implement lines 4–7 from Framework 1 is based on the beta process hidden Markov model (BP-HMM) presented by Fox et al. [8] and Hughes et al. [13]. We used the implementation made available online by Hughes [12]. This method uses a non-parametric Bayesian approach based on a stochastic process—the Beta Process (BP) [33]—to learn behaviors (primitives) that are shared among several related time series. Such Bayesian nonparametric techniques provide us with the advantage of learning from sequential data that does not require knowing the number of hidden states in advance. We pick the BP-HMM algorithm presented herein (Framework 1, line 5) due to its capability of handling multiple times series and learning a set of primitives shared among them, with tractable analysis of hundreds of series. Below, we present the algorithm, as described by Hughes et al. [13].

In our case, a time series represents a recording of an assembly session. The i^{th} series is assigned a sparse binary vector $f_i = [f_{i1}, f_{i2}, ...]$, that indicates the presence or absence of each feature in the set of shared primitives. For N time series, matrix $F = [f_1; ...; f_i; ...; f_N]$ contains the binary features of all the time series, and is generated by the BP:

$$B|B_0, \gamma, \beta \sim BP(\beta, \gamma B_0), B = \sum_{k=1}^{\infty} b_k \delta_{\gamma_k} \tag{1}$$

Realization B of the BP contains potentially infinitely many features k. For each feature, $\theta_k \sim B_0$ represents the model parameters, and $b_k \in (0, 1)$ represents the feature's inclusion probability in the respective i^{th} series. The binary feature vector for the i^{th} series is obtained by independent Bernoully draws $f_{ik} \sim Ber(b_k)$. Parameter γ determines the Poissson(γ) distribution that represents the number of active features in series i. Parameter β dictates the frequency of sharing features between series.

The BP is then combined with an HMM to form the BP-HMM. The binary vector f_i determines a finite set of primitives available for the i^{th} series. A single primitive $z_{it} = k$ from the set $\{k : f_{ik} = 1\}$ is then assigned to each time step t, determining parameters θ_k that generate the data value x_{it} at that time step.

Given that only one primitive is active at each time step, we applied the BP-HMM algorithm on three different signals composing our task: one representing the person's left hand (i.e. the (x, y, z) signal averaged over the eight markers mounted on the left-hand glove), one representing the right hand (similar to the left hand), and one representing the tracked objects (to more easily scale up to tracking several objects throughout a task, we learn primitives based on the signals provided by all the task components). Thus, we obtained

TABLE I: Primitive Granularity Levels and Labels for Classification Experiments

	$granularity\ level = low$		$granularity\ level = intermediate$	$granularity\ level = high$	
	2 *classes*	3 *classes*	2 *classes*	2 *classes*	3 *classes*
label 1	screws	screws	small objects	small and big objects	small objects
label 2	—	nuts	—	—	big objects
label 0	rest	rest	rest	rest	rest

[1] We employ 2-class and 3-class experiments to analyze at what granularity level we can distinguish between primitives labeled differently.
[2] *Label* 1 and *label* 2 (when it exists), refer to primitives at different granularity levels. A bar (—) appears under 2-class experiments, and indicates a missing label.
[3] *Label* 0 always refers to the "rest" of the primitives, i.e. all primitives that have not been labeled with 1 or 2.
[4] "Small objects" include the pegs, the nuts, the screw, and the screwdriver. "Big objects" include the frame, and the main component.

three sets of primitives, corresponding to the left hand, right hand, and objects, respectively. Prior to applying the algorithm on the three different sets of time series, we normalized the data within each set to have mean zero and unit variance. To implement lines 6 and 7 from Framework 1, we applied the BP-HMM algorithm on the training set to learn the model parameters, and then used these parameters when applying the algorithm to the test set. The training and test sets are as defined in Subsection V-A.

C. Classification Experiments

In this subsection, we detail the implementation of Framework 1, lines 8–20.

1) Classification method: The classification method we employ for the time series consisting of learned primitives is the k-Nearest Neighbors (k-NN) algorithm [1], with dynamic time warping (DTW) [4] as the time series similarity metric. This constitutes our implementation of Framework 1, line 10. Although DTW is known to be computationally expensive, it is considered one of the best measures for use with time series across application domains [7] and has also been shown to be able to run quickly even on very large data sets [29]. Classifying time sequences represents a non-trivial task, and the choice of similarity metric in this context is particularly known to be an open research question [21]. Our choice of the current classification method is based on the fact that DTW has proven a successful metric for measuring time series similarity, especially when used together with the k-NN algorithm [29]. Our analyses have also shown that, for our learned primitive representation under the form of time series, other classification methods, such as support vector machines and multinomial logistic regression, have proved ineffective.

To apply the algorithm, we divided each time sequence into non-overlapping windows of fixed size 400 (approximately 5 seconds), starting from the beginning of each time series, and normalized all the data to have mean zero and unit variance. For cases when the series' length did not equally divide into the window length, we padded the sequence with the last encountered value. Since we applied the BP-HMM algorithm for primitive action discovery three times for each of the right hand, left hand, and the objects, we have three time series per trial, each represented as primitive sequences. For each window of size 400, we concatenate the three primitive sequences corresponding to the right hand (RH), left hand (LH), and objects (Obj), respectively. This results in a sequence of length 1200 per window, with format RH_LH_Obj. Next, we implement Framework 1, line 11. Since k-NN is a lazy algorithm (i.e. does not do any generalization using the training data), we create a set consisting of all the 1200-length windows obtained from segmenting the learned primitive training data set, ready to use for the testing phase.

2) Primitive choice: To implement Framework 1, line 13, we choose to explore *pick up*, since it represents one of the most prevalent primitives in robotics. In addition, *pick up* presents strong relevance to HRC scenarios such as the one we investigate in this paper. Both humans and robots need to frequently execute *pick up* actions during the assembly of the chair, and need to do so for different kinds of objects.

In order to represent the *pick up* primitive, we consider the factors that engender different types of pick ups for our task (Framework 1, line 14). We thus turn our attention towards the types of objects part of the chair assembly. As described in Section IV, we have four different types of small objects: pegs, nuts, screws, and a screwdriver, and two bigger objects: the frame and the main component. Looking at our objects, a low granularity would involve differentiating between *peg pick up*, *nut pick up*, *screw pick up*, and *screwdriver pick up*. This means treating each type of *pick up* as a separate class. A high level of granularity would involve simply distinguishing that an object is being picked up, whether small or large. These represent our established lowest and highest reasonable levels of granularity, as appears in Framework 1, lines 15–17. An intermediate level would be placed in between, distinguishing between *small object pick up* and *big object pick up*.

3) Find final granularity level: As stated in Framework 1, lines 16 to 17, we start with the low and high granularity levels, and use classification performance to gauge if they represent reasonable minimum and maximum levels for the task at hand. We then implement Framework 1, line 19. We work down from the highest level and test an intermediate value. We do so in order to ensure we are not fooled by choosing too high of a level, which might hide multiple types of primitives.

In order to find the lowest level with high performance, per Framework 1, line 20, we perform three main classification experiments and two additional ones. The three main experiments consist of a binary classification task. Each classification aims to distinguish between the primitives at the considered level of granularity and the rest of the primitives present in the data set. We segment each time sequence into windows

of 400 time steps. We then manually label the windows that include the considered *pick up* primitives as the positive class (labeled as 1), and the rest of the windows as the negative class (labeled as 0). We perform two additional experiments to better interpret the results at the intermediate and high levels, and to test exactly what types of *pick up* primitives we can distinguish between. These experiments are performed using three classes, with windows that include the first type of *pick up* primitive labeled as 1, the second type labeled as 2, and the rest labeled as 0. The different levels are presented in Table I.

When applying the k-NN algorithm to the classification experiments described above, we employed a k value of 3 for the binary classification tasks and a value of 5 for the 3-class tasks. For the 3-class tasks, ties can appear when two class labels occur an equal number of times. In cases of ties, we decrease k until no tie is present.

VI. Results

In this section, we present the results of implementing our framework for the considered HRC scenario.

A. BP-HMM Algorithm Results

Fig. 4 shows an example of the learned primitives for a time sequences representing a single trial. The results highlight two sets of primitives: right hand (resulted in 55 global primitives), and objects (resulted in 86 global primitives). We did not visualize the primitives resulted for the left hand so as not to overload the graph. For the left hand, we obtained 51 global primitives. The fact that we obtained a similar number of primitives for the right and left hand indicates that the representation was able to pick up on similar types of movements a person can perform with the right and left hand. We posit that the number of primitives detected for the right hand is slightly higher than that detected for the left hand due to the fact that most people used their right hand far more often than they did their left hand (picking up the small objects placed to their right in the workspace, and even picking up bigger objects that were placed to the left of the small items).

B. Granularity Discovery Results

We evaluate the performance of our classification experiments by looking at **precision**, **recall** and **F1 score**. Table II presents the results of our experiments, and provides the definitions of these metrics. The table shows the results for the low, intermediate, and high granularity levels, including the 2-class and, where present, for the 3-class experiments.

Since we train our classifiers with unbalanced classes (i.e. positive labels occur much less frequently than negative ones), we do not place too much emphasis on the high scores for the negative labels. We present them in the table for completeness. Our first, 2-class, low granularity level classification task provided poor results, with an **F1 score** of 0.31. This task represents training our classifier with only screw pick ups as positive labels. This suggests this granularity level is too low for our task, since we have many similar small-sized objects. Our intermediate level classifier provides us with good

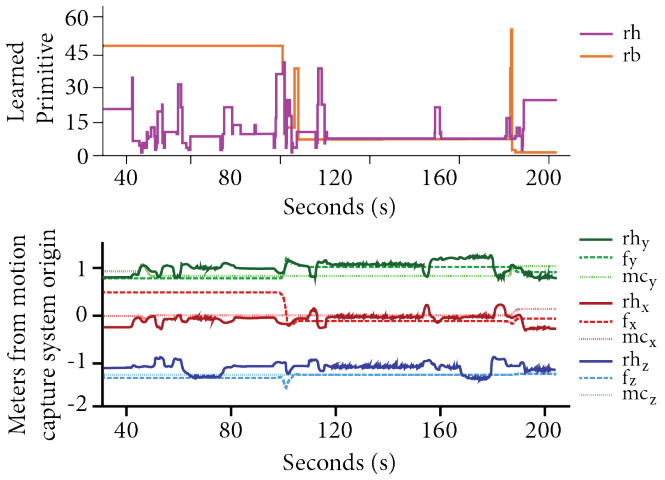

Fig. 4: A time series from an assembly session including the raw motion capture data in the bottom subfigure, and the learned primitives in the upper subfigure. Here, we omitted the left hand signals for ease of visualization. The bottom subfigure presents the (x, y, z) coordinates for each of the: (i) right hand (*rh*), (ii) frame (*f*), and (iii) main component (*mc*). The x, y, and z values are represented in red, green, and blue, respectively. The coordinates represent displacement in meters from the motion capture system origin, set up in the center of the data collection room. The upper subfigure presents the learned primitives for the: (i) right hand (*rh*) in magenta, and (ii) the two rigid bodies (*rb*) in orange. We applied the primitive learning algorithm on the f and mc data together, resulting in a single set of primitives for the two rigid bodies.

performance, with an **F1 score** of 0.82. This level represents pick ups for similar-sized objects (i.e. pegs, nuts, screws, and screwdriver). The high granularity classifier also provides us with a good **F1 score**, of 0.81. However, since we employed a binary classification scheme, a similar result to that of our intermediate classifier might signify several subclasses are hiding under the positive label for the more general *pick up* action. In this high granularity case, we train the classifier with both small- and big-sized objects labeled as positive examples.

Since we expect our intermediate classifier to perform significantly worse if a high granularity would be appropriate for this task, we believe the high level classifier is now hiding two subclasses, namely *pick up* for small objects and *pick up* for big objects. To test this, we perform two additional classification tasks, employing three classes instead of two. The low level 3-class task labels screws with 1, nuts with 2, and the rest with 0. The high level 3-class task labels small objects with 1, big objects with 2, and the rest with 0. We obtain poor performance for the former (**F1** scores of 0.40 and 0.46 for classes screws and nuts, respectively) and good performance for the latter (**F1** scores of 0.78 and 0.82 for classes small objects and big objects, respectively). This indicates that, in the case of the low level classifier, we cannot differentiate between different types of small objects, and so the classifier cannot distinguish between labels 1 and 2. In the case of the high level classifier, we can indeed differentiate

TABLE II: Results of Classification Experiments

	$granularity\ level = low$		$granularity\ level = intermediate$	$granularity\ level = high$	
	2 *classes*	3 *classes*	2 *classes*	2 *classes*	3 *classes*
Precision					
label 1	0.33	0.50	0.81	0.81	0.78
label 2	—	0.40	—	—	0.82
label 0	0.93	0.95	0.94	0.93	0.98
Recall					
label 1	0.29	0.40	0.84	0.81	0.88
label 2	—	0.43	—	—	0.75
label 0	0.94	1	0.93	0.81	0.95
F1 score					
label 1	0.31	0.40	0.82	0.81	0.78
label 2	—	0.46	—	—	0.82
label 0	0.93	0.98	0.93	0.93	0.97

[1] **Precision** is defined in the standard way, as $true\_positives/(true\_positives + false\_positives)$
[2] **Recall** is defined in the standard way, as $true\_positives/(true\_positives + false\_negatives)$
[3] **F1 score** is defined in the standard way, as $2 \times (\mathbf{precision} \times \mathbf{recall})/(\mathbf{precision} + \mathbf{recall})$
[4] A bar (—) appears under 2-class experiments, and indicates an absent value for the missing label.

between the pick ups for small objects and those for big ones.

Our results show strong support for the intermediate level classifier having the appropriate level of granularity for the *pick up* primitive in our scenario. This highlights the importance of evaluating different levels of primitive granularity in order to find the best suited one for the task at hand.

C. Participant Classification Results

As a complement to the classification tasks performed on windows resulted from segmenting the data sequences, we also performed a classification task per person in order to show the primitive action representation is also useful at a higher level. We applied the same k-NN with DTW algorithm described in Subsection V-C1, but this time by using the full sequences. We concatenated the sequences in a similar manner as above, with format RH_LH_Obj, where RH, LH, and Obj.

We divided the data in the same fashion, leaving out one time series per participant (ending up with 28 series in the test set) and aiming to classify participants correctly. Our algorithm classifies 21 out of 28 participants correctly, giving us a percentage accuracy of 75%. We do not present usual **precision**, **recall**, and **F1 score** metrics for this classifier since they are ill-defined for classes expecting a single example, which is missing for incorrectly classified participants.

VII. Conclusions

In this paper, we presented a framework for discovering the appropriate level of primitive granularity for the task at hand. We focused on an HRC scenario, for which learning from human demonstrations is important to develop adaptive robots. We recorded a motion capture data set of human and object motions during an IKEA chair assembly task, and used a BP-HMM algorithm to learn primitives. We presented our framework for discovering a well-suited level of granularity for the considered task, and applied it to *pick up*, a prevalent primitive in robotics, relevant to our HRC application.

The results show that our framework is able to successfully discover the appropriate granularity level for the considered task. We highlight that even when obtaining high classification performance, high granularities are not necessarily best suited for the task, as we uncover hidden structure in the data at this level. In our scenario, this occurs because it is not immediately obvious that picking up different object types should be regarded as different primitives. When we label different-sized object *pick up* primitives as all pertaining to the positive class of examples, we are mistakenly lead to believe this is a well-suited level. Such a decision can be made based on intuitive descriptions found in the expected inputs for high-level planners. However, our results show that simply following the expected structure of such inputs without ensuring that the level is actually well-suited for the task, can result in overlooking important differences in such primitives.

Our framework represents a contribution towards providing existing methods in robotics with the types of inputs they expect to function as desired. We would like to acknowledge limitations of the presented work, and consider automating the process of finding reasonable minimum and maximum granularity levels, as well as that of exploring the intermediate levels an important future direction. Another issue to notice is that the task-specific factors we use have impact on the results of the framework. This makes the choice of such factors for a different task non-trivial, yet represents an important way of injecting domain knowledge into the framework. Finally, another worthwhile improvement is investigating how primitives learned from human motion relate to robot primitives for the same action. For example, the *pick up* primitive might have a granularity level better suited for the task if it more closely resembles the type of *pick up* the robot can perform.

Acknowledgments

The authors would like to thank Alessandro Roncone and Olivier Mangin for their thoughtful suggestions and feedback. This work is funded by a subcontract with Rensselaer Polytechnic Institute by the Office of Naval Research, under Science and Technology: Apprentice Agents, and by NSF Expedition in Computing 1139078.

References

[1] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[2] Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K Hodgins, and Nancy S Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004*, pages 185–194. Canadian Human-Computer Communications Society, 2004.

[3] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human–robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66, 2008.

[4] Donald J Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[5] Jesse Butterfield, Sarah Osentoski, Graylin Jay, and Odest Chadwicke Jenkins. Learning from demonstration using a multi-valued function regressor for time-series data. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 328–333. IEEE, 2010.

[6] Silvia Chiappa and Jan R Peters. Movement extraction by detecting dynamics switches and repetitions. In *Advances in neural information processing systems*, pages 388–396, 2010.

[7] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[8] Emily B Fox, Michael C Hughes, Erik B Sudderth, Michael I Jordan, et al. Joint modeling of multiple time series via the beta process with application to motion capture segmentation. *The Annals of Applied Statistics*, 8(3):1281–1313, 2014.

[9] Daniel H Grollman and Odest Chadwicke Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 261–266. IEEE, 2010.

[10] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5469–5476. IEEE, 2016.

[11] Sven Hellbach, Julian P Eggert, Edgar Körner, and Horst-Michael Gross. Basis decomposition of motion trajectories using spatio-temporal nmf. In *International Conference on Artificial Neural Networks*, pages 804–814. Springer, 2009.

[12] Michael Hughes. Nonparametric bayesian inference for sequential data., 2012. Available at http://michaelchughes.github.io/NPBayesHMM/.

[13] Michael C Hughes, Emily Fox, and Erik B Sudderth. Effective split-merge monte carlo methods for nonparametric models of sequential data. In *Advances in Neural Information Processing Systems*, pages 1295–1303, 2012.

[14] Odest Chadwicke Jenkins and Maja J Mataric. Automated modularization of human motion into actions and behaviors. 2002.

[15] Odest Chadwicke Jenkins and Maja J Mataric. Deriving action and behavior primitives from human motion data. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2551–2556. IEEE, 2002.

[16] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andre S Barreto. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in neural information processing systems*, pages 1162–1170, 2010.

[17] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[18] Volker Krüger, Dennis Herzog, Sanmohan Baby, Aleš Ude, and Danica Kragic. Learning actions from observations. *IEEE Robotics Automation Magazine*, 17(2):3—43, 2010.

[19] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, page 0278364911426178, 2011.

[20] Yi Li, Cornelia Fermuller, Yiannis Aloimonos, and Hui Ji. Learning shift-invariant sparse representation of actions. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2630–2637. IEEE, 2010.

[21] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[22] Olivier Mangin and Pierre-Yves Oudeyer. Learning to recognize parallel combinations of human motion primitives with linguistic descriptions using non-negative matrix factorization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3268–3275. IEEE, 2012.

[23] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001.

[24] Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L Sidner, and Daniel Miller. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 205–212. ACM, 2015.

[25] Anahita Mohseni-Kabir, Victoria Wu, Sonia Chernova, and Charles Rich. What's in a primitive? Identifying reusable motion trajectories in narrated demonstrations. In *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*,

pages 267–272. IEEE, 2016.

[26] Negin Nejati, Pat Langley, and Tolga Konik. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning*, pages 665–672. ACM, 2006.

[27] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[28] PhaseSpace. Phasespace motion capture, 2013. Available at http://www.phasespace.com/.

[29] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.

[30] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[31] Yifan Shi, Yan Huang, David Minnen, Aaron Bobick, and Irfan Essa. Propagation networks for recognition of partially ordered sequential action. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–862. IEEE, 2004.

[32] Austin Tate. Generating project networks. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 2*, pages 888–893. Morgan Kaufmann Publishers Inc., 1977.

[33] Romain Thibaux and Michael I Jordan. Hierarchical Beta Processes and the Indian Buffet Process. In *AISTATS*, volume 2, pages 564–571, 2007.

[34] Qiang Yang. Formalizing planning knowledge for hierarchical planning. *Computational intelligence*, 6(1):12–24, 1990.

[35] Yang Yang, Imran Saleemi, and Mubarak Shah. Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1635–1648, 2013.