Project no.:	**IST-FP6-IP-027657**
Project full title:	**Perception, Action & Cognition through Learning of Object-Action Complexes**
Project Acronym:	**PACO-PLUS**
Deliverable no.:	**D3.1.4**
Title of the deliverable:	**Advances of human upper body and human grasp action representation**

Contractual Date of Delivery to the CEC:	**31st January 2008**
Actual Date of Delivery to the CEC:	**31st January 2008**
Organisation name of lead contractor for this deliverable:	**Aalborg University (AAU)**
Author(s): Volker Krüger, Hedvig Kjellström, Danica Kragic	
Participants(s): AAU, KTH	
Work package contributing to the deliverable:	**WP3.1**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**50**
Start date of project:	1st Feb. 2006 **Duration:** 48 month

Projectco-funded by the European Commission within the Sixth Framework Programme (2002-2006)
Dissemination Level

PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This report contains a single scientific publication that is relevant for the deliverable D3.1.4. This deliverable is concerned with deriving a representation of action on a higher abstraction level. The enclosed summary outlines the context of this work.

Keyword list: action and grasp recognition, computer vision, robotics, AI, stochastic grammars

Introduction

Deliverable D3.1.4, *Advances of human upper body and human grasp action representation*, is concerned with two aims:

1. to become able to identify human upper body movements and actions which may include objects. Examples of such movements are humans pointing at objects or humans grasping objects.

2. to become able to synthesize actions based on those seen before.

These two objectives have been the base line for workpackage WP3 since the start of the project.

While deliverable D3.2.3 focuses on a representation of grasps and arm actions based on which recognition can be performed, this deliverable D.3.1.4 investigates a higher abstraction of arm movements and grasps. In other words, D3.2.3. is concerned with the signal level of the action and grasp representation while deliverable D3.1.4 is concerned with the symbolic representation of action.

The use of a symbolic representation of action has several advantages:

1. the symbolic representation becomes in general independent from the embodiment and we become able to use terms like *reach out*, *grasp object*, *move object* while the actual definition of what *grasp object* or *reach out* means is dealt with on a different level, e.g., with the techniques described in D3.2.3.

2. It has been pointed out at several occasions (see appended paper) that the symbolic level improves recognition done on the signal level. The signal level is able to provide a statistical likelihood of a visual observation being a certain action. Since actions never appear in isolation but in sequences and even larger contexts the symbolic level is able to improve and refine the belief coming from the visual input. In statistical terms, with the symbolic level we become able to exploit the higher order Markov properties of actions while on the signal level, the possibilities of using higher order Markov properties are limited (1st order Markov properties are used in most cases)[1].

3. the symbolic level allows planning of actions based on the known actions using, e.g., a classic STRIPS planner.

One major difficulty in using a higher abstraction for recognition is to tie the signal level to the symbol level. Given the representation of signals, e.g., a visual representations of human grasps and arm movements, the problem is how to identify the necessary symbols that capture the "meaning" of the underlying signal information. The term *meaning* is put in quotes because the meaning does not arise per se but is usually provided by a human supervisor when tying portions of the signals to symbols such as *reach out* or *grasp* (as done, e.g., in [2]).

In this deliverable, we have developed and tested a statistical learning approach that allows a) to replace a human supervisor through a non-supervised learning approach and b) to automatically generate a stochastic context free grammar based on which recognition (parsing) and synthesis can be done.

[1]we omit most references here as all necessary references are found in the appended paper

Based on a set of observed actions, the approach is able to identify correlations between these movements which then constitute a symbol. The Kullback-Leibler divergence is used as the criteria to decide when two movements parts constitute the same or different symbols.

The learning approach has been applied on a scenario that is typical for this project and that has been used already within the earlier work by Vicente et al. [2] (see also deliverable D3.2.1). An interesting observation is that based on the statistics of the observed actions, our approach recovers primitives that show a strong correlation to those defined by hand in [2].

The appended paper [1] will be submitted to the Int. Journal on *Advanced Robotics*.

Attached Documents

[1] Sanmohan, V. Krueger, D. Kragic, and H. Krellstroem. Primitive based action representation and recognition. *Advanced Robotics*, 2009. to be submitted.

References

[2] I. S. Vicente, V. Kyrki, and D. Kragic. Action recognition and understanding through motor primitives. *Advanced Robotics*, 21:1687–1707, 2007.

Primitive based action representation and recognition

Sanmohan[†], Volker Krüger[†], Danica Kragic[‡], Hedvig Kjellström[‡]

***Abstract*—We investigate modeling and recognition of arm manipulation actions at different levels of complexity. Motivated by the good recognition results using manually segmented primitives we automate the manual process of action primitive detection and present a sequential learning algorithm that allows a non-supervised detection of the action primitives. We also generate an action grammar based on these primitives. Here, we assume no prior knowledge on primitives but look for correlating segments across various sequences. All actions are then modeled within a single HMMs whose structure is learned incrementally as new data is observed.**

***Index Terms*—Imitation learning, high level event and activity understanding, generative and discriminative models .**

I. Introduction

A standard approach in teaching tasks such as robot arm movements to a robot, is to store complete arm movement that are to be executed by the robot. For complex tasks an entire library of specific arm movement may be required. Humanoid robots, that are required to interact with humans need to have the ability to a) recognize human movements in order to react to them and/or learn to perform tasks from human demonstration and b) cope possibly with complex human environments by way of adapting their movements [1], [2], [3], [4], [5], [6].

One way for humanoid robots to recognize a human movement is to find in the library the movement model that explains the observed movement best. However, for large libraries this becomes very unrobust in terms of recognition rate and very ineffective in terms of computation time. Concerning the ability to adapt the movements, the robot is limited to choose a predefined movement from its library.

An alternative to storing a predefined and complete movement library is to break down the library movements into their common pieces and to represent the movements instead in a grammatical manner, with the common movement *primitives* as the grammar symbols. In linguistics [7] and computer vision [8] this has been proven to an effective approach for recognition, both in terms of recognition performance and computational efficiency. Similar results were recently shown in robotics [9] where the Vicente *et al.* compared the use of a movement library with a more grammatical-like representation.

A grammatical representation of actions has the further advantage of allowing to re-combine the grammatical symbols through planning strategies [10] and thus allowing the robot to adapt to new situations and even develop new movements based on new re-combinations of the available set of available movement primitives, an issue that was also discussed in [9].

A common problem in using a grammatical representation is the definition of a proper set of primitives. All above mentioned works [7], [8], [9] rely on hand-selecting the primitives. In this paper, we present a systematic approach for finding the primitives for the robotics scenario presented in [9] automatically.

In Sec. II we give a summary of the work by Vicente *et al* [9] which could be considered as the starting point of our work. In [9], motion trajectories are segmented in a supervised manner and the resulting primitives are modeled with an hidden Markov model (HMM). The observed superiority of recognition rates [7], [8], [9] and the ability to model unknown actions with primitives motivates the search for an unsupervised method for trajectory segmentation and modeling. This is in detail explained in Sec. III. We base our approach

†Computer Vision and Machine Intelligence Lab, Copenhagen Inst. of Technology, Aalborg University. Ballerup, Denmark. email: {san,vok}@cvmi.aau.dk

‡ Royal Institute of Technology, Computational Vision and Active Perception lab, Centre for Autonomous Systems, S-100 44 Stockholm, Sweden. email:{danik,hedvig}@nada.kth.se

on hidden HMMs, and we use a model merging technique to build a model and segment trajectories into primitives. In this way, we arrive at a single HMM which is similar to the one in the supervised learning scenario in [9]. We further learn a stochastic context free grammar for the primitives we have found.

II. Previous work

Our work is an enhancement to the work done in [9]. Hence we start with a brief overview of [9] so that we could compare the works easily.

The results of [9] are a study of modeling and understanding of manipulation actions performed by humans on a table top scenario. Five actions are considered: a) pick up an object from a table, b) rotate an object on a table, c) push an object forward, d) push an object to the side, and e) move an object to the side by picking it up.

Each action is performed in 12 different conditions: Objects placed on two different heights and two different locations on the table, and the demonstrator stand in three different locations (0, 30, 60 degrees). All the actions are demonstrated by 10 different people.

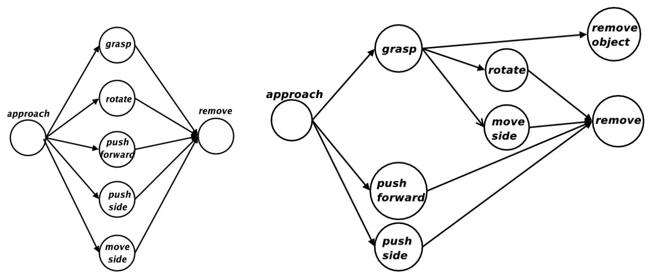

Fig. 1: (left): HMM model I with actions as primitives; (right): HMM II with composite actions.

Four sensors are attached to each person and their positions in 3D coordinates are measured. The sensors are located on: a) chest, b) back of hand, c) thumb, and d) index finger. The measurements from chest sensor is used to provide a reference to the demonstrator position while the sensor at the back of the hand is used as a reference for the thumb and index finger. The raw measurements are then preprocessed and the following 12 measurements are used for experiments:

- position of the hand relative to the chest
- position of the index finger and the thumb relative to the hand
- velocity of the hand

HMM	pf	ps	r	g	m
pf	87.50	4.17	0.00	4.17	4.17
ps	8.33	48.33	2.50	3.33	37.5
r	0.83	2.5	95	1.67	0.00
g	5.83	10	9.17	52.5	22.5
m	1.67	24.17	4.17	2.5	67.5

TABLE I: Approach 1. Actions as primitives. Results of using HMM model I. Correct results are given on the diagonal.

Primitives are manually extracted from the data and two different HMM structures are considered for modeling the actions which are shown in Fig. 1 and their results are compared. In the first model the primitives are *grasp*(g), *rotate*(r), *push forward*(ps), *push side*(ps), *move side*(m), *approach* and *remove*. In the second model, the grasping part of *rotate* and *move side* primitives were considered as separate primitives. SVMs are used to recognize the primitives and the outcome of SVMs are then fed into the HMM used for modeling the actions. Using the outcome of SVMs as observations, the HMM parameters are learned through standard Baum-Welch algorithm.

The results of using Model I is presented in Tab.I. The entries on the diagonal shows the correct recognition rates. In this model individual primitives did not yield good results due to their high overlap.

In HMM model II , the common parts are kept as a separate primitive as shown in Fig. 1 on the right. This way of representation will give the primitives a semantic meaning as well. One new state, *remove with object*, was introduced to show that the end state is different from other cases . Each person is holding the object at the end only for the *grasp* action.

Tab. II presents the recognition results by the HMM where the numbers on the diagonal give the correct recognition rate. The results of recognizing *grasp* and *move* have increased significantly.

HMM	pf	ps	r	g	m
pf	85	7.5	5	0.83	1.67
ps	9.17	47.5	4.17	2.5	36.67
r	0	0	92.5	0	7.5
g	4.17	7.5	10.83	72.5	5
m	1.67	10	6.67	0	81.67

TABLE II: Hmm classification results using Model II

A. *Discussion*

We could see the work of [9] as an extensive study on the modeling of the manipulation actions, which have the characteristic of being very similar to each other. The most important findings of their experiments could be stated as: a) sequences of simple semantic primitives can be used in describing actions, and b) actions learned as sequences of primitives from other demonstrators can be combined with knowledge of personal primitives to recognize new actions.

III. Automatic Segmentation of Primitives

From the discussions above we can see that an efficient model could be made if we have the primitives at hand. Thus, it is desired to have a mechanism to detect the primitives automatically from action sequences. We note from the previous section that we had two types of primitives: primitives that were unique to an action and primitives that were common to more than one action. Thus our hypothesis is that if we segment action sequences into parts that are common across more than one action and parts that are unique to each of the actions, we will arrive at a set of action primitives that can be used for recognition as discussed in the previous section. We therefore formalize our problem of primitive segmentation as follows:

We define two sets of primitives. One set contains parts that are unique to one *type* of action and another set that contains parts that are common to more than one *type* of action. Two sequences are of the same *type* if they do not differ significantly, e.g., two different walking paths. Hence we attempt to segment sequences into parts that are common across sequences types and parts that are not shared. Then, each sequence will be a combination of these segments. We also want to generate rules that govern the interaction among the primitives. Keeping this in mind we state our objectives as:

1) Let $\mathcal{L} = \{X_1, X_2, \cdots, X_m\}$ be a set of data sequences where each X_i is of the form $x_1^i x_2^i \cdots, x_{T_i}^i$ and $x_j^i \in \mathbb{R}^n$. Let these observations be generated from a finite set of sources (or states) $\mathcal{S} = \{s_1, s_2, \cdots s_r\}$. Let $S_i = s_1^i s_2^i \cdots, s_{T_i}^i$ be the state sequence associated with X_i. Find a partition $\mathcal{S}'$ of the set of states $\mathcal{S}$ where $\mathcal{S}' = \mathcal{A} \cup \mathcal{B}$ such that $\mathcal{A} = \{a_1, a_2, \cdots, a_k\}$ and $\mathcal{B} = \{b_1, b_2, \cdots, b_l\}$ are sets of state subsequences of X_i's and each of the a_i's appear in more than one state sequence and each of the b_j's appear in exactly one of the state sequence. The set $\mathcal{A}$ corresponds to common actions and the set $\mathcal{B}$ correspond to unique parts.
2) Generate a grammar with elements of $\mathcal{S}'$ as symbols which will generate primitive sequences that match with the data sequences.

A. *Outline of our Approach*

We approach the problem by building a model that will generate the first data sequence that we encounter and check if the upcoming data sequences could have been generated from the constructed model. If not, modify the model to accommodate the newly observed data sequence. We continue this until we are able to a create a single model that is capable of generating all the data sequences. In our case, we make a single hidden Markov model that will generate all the data sequences (explained in Sec. III-B-III-B3). Then by examining the sequence of states the observation sequences are going through, the common states are identified. States(or sequence of states) common to sequences are separated out to form $\mathcal{A}$ and the remaining contiguous states make the set $\mathcal{B}$ which were defined in Sec. III. Creation of the sets $\mathcal{A}$ and $\mathcal{B}$ is explained in Sec. III-C.

B. *Modeling the Observation Sequences.*

1) Modeling the First Sequence: Let X_1 be the first sequence with data points $x_1^1 x_2^1 \cdots x_{T_1}^1$. Since we have just one data sequence to start with, we generate a few more spurious sequences of the same type by adding Gaussian noise to it. Then we choose (μ_i^1, σ_i^1), $i = 1, 2, ...k1$ so that parts of the data sequence are from $\mathcal{N}(\mu_i^1, \Sigma_i^1)$ in that order. The value of $k1$ is such that $\mathcal{N}(\mu_i^1, \Sigma_i^1)$, $i = 1, 2, ...k1$ will cover the whole data. This value is not chosen before hand and varies with the variation and length of the data.

The next step is to make an HMM $\lambda_1 = (A1, B1, \pi 1)$ with $k1$ states where $k1$ is the number of Gaussians needed to cover X_1. We let $A1$ to be a left-right transition matrix and $B1_j(x) = \mathcal{N}(x, \mu_j^1, \Sigma_j^1)$. All the states at this stage get a label 1 to indicate that they are part of sequence type 1. We require this information to link final primitives with different types of sequences and also for generating a grammar for primitives.

2) Modeling the rest of the data: Let $n-1$ be the number of types of data sequences we have seen so far. Let X_c be the next data sequence to be processed. Calculate $P(X_c|\lambda_M)$ where λ_M is the current model at hand. If we get a high value for $P(X_c|\lambda_M)$ it indicates that λ_M models sequences of type X_c well, and so we proceed to the next data sequence. A low value for $P(X_c|\lambda_M)$ indicates that the current model is not good enough to model the data sequences of type X_c and hence we make a new HMM λ_c for X_c as described in Sec. III-B1. The newly constructed HMM λ_c will be used to modify λ_M so that the updated λ_M will be able to generate data sequences of type X_c. The modification procedure of λ_M using λ_c is described in Sec. III-B3. We increase the number of types of data sequences by one at this stage. All the states in X_c will be labeled n.

We might get a high value for $P(X_k|\lambda_M)$ for a new data sequence which has no unique part of its own but is part of several different types of data sequences we have seen so far. We resolve this by making use of the state labeling we have performed during the modeling. Whenever we get a high value for $P(L_k|\lambda_M)$ we look at the Viterbi path of the data sequence and examine the labels of the state sequence. If it is a new type then then there will be two states whose labels have empty intersection. In that case we increase the number of types of data sequences by one and append the new type number to each of the states it is passing through.

3) Merging of similar states: This section explains the most important part of our method: modifying the existing model to generate a newly observed type of data. We do this by adding new states or by modifying existing states.

Suppose we want to merge λ_c into λ_M where λ_M is the current model so that $P(X_k|\lambda_M)$ is high if $P(X_k|\lambda_c)$. We do this by adding states to λ_M from λ_c or by merging states of λ_M with states of λ_c. Let $C_c = \{s_{c1}, s_{c2}, \cdots, s_{ck}\}$ and $C_M = \{s_{M1}, s_{M2}, \cdots, s_{Ml}\}$ be the set of states of λ_c and λ_M respectively. Then the state set of the modified λ_M will be $C_M \cup D_1$ where $D_1 \subseteq C_c$. Each of the states s_{ci} in λ_c affects λ_M in one of the following ways:

1) If $d(s_{ci}, s_{Mj}) < \theta$, for some $p \in \{1, 2, \cdots l\}$, then s_{ci} and s_{Mj} will be merged into a single state. Here d is a distance measure and θ is a threshold value. The output probability distribution associated with s_{Mj} is modified to be a combination of the existing distribution and ${b^k}_{s_{ci}}(x)$. Thus ${b^M}_{Mj}(x)$ is a mixture of Gaussians. We append n to the label of the state s_{Mj}. All transitions to s_{ci} are redirected to s_{Mj} and all transitions from s_{ci} will now be from s_{Mj}.
2) If $d(s_{ci}, s_{Mj}) > \theta$ f.a. j, a new state is added to λ_M. i.e. $s_{ci} \in D_1$. Let s_{ci} be the r^{th} state to be added from λ_c. Then, s_{ci} will become the $(Ml+r)^{th}$ state of λ_M. The output probability distribution associated with this new state in λ_M will be the same as it was in λ_c. Hence ${b^M}_{Ml+r}(x) = \mathcal{N}(x, \mu_{s_{c_i}}, \Sigma_{s_{c_i}})$. Initial and transition probabilities of λ_M are adjusted to accommodate this new state. The newly added state will keep its label n.

We use Kullback-Leibler Divergence to calculate the distance between states. The K-L divergence from $\mathcal{N}(x, \mu_0, \Sigma_0)$ to $\mathcal{N}(x, \mu_1, \Sigma_1)$ has a closed form solution given by :

$$D_{KL}(Q||P) = \frac{1}{2}\left(\log\frac{|\Sigma_1|}{|\Sigma_0|} + \mathrm{tr}(\Sigma_1^{-1}\Sigma_0)\right) + \frac{1}{2}\left((\mu_1-\mu_0)^T\Sigma_1^{-1}(\mu_1-\mu_0) - n\right) \quad (1)$$

Here n is the dimension of the space spanned by the random variable x.

Now we elaborate more on the addition and merging of states into the combined model. Our aim is to make the new model compatible with the newly observed type of data sequences. Since the states are probability distributions, if we see that two probability distributions corresponding to different states are very close we do not need to keep them apart. Keeping these two states together will help us to model the observations generated from two distributions by a single one. We use (1) to compute the similarity measure of two states. We can observe that (1) will not handle mixture of Gaussians. We still use this equation to evaluate component wise distances in mixtures and check if any of the components are close to the distribution we are testing. We justify this criteria since our aim is to find out if a new state is to be embedded into another state or not.

C. Finding Primitives

Primitive searching starts when we have processed all the available data sequences. Now using Viterbi algorithm on the final merged model λ_M, the hidden states associated with each of the sequences are generated. Let $P_1, P_2, \cdots P_r$ be different Viterbi paths at this stage. Since we want the common states that are contiguous across state sequences, it is similar to finding the longest common substring(LCS) problem. We take all paths with non-empty intersection and find the largest common substring a_k for them. Then, a_k is added to $\mathcal{A}$ and is replaced with an empty string in all the occurrences of a_k in $P_i,\ \ i = 1, 2, \cdots r$. We continue to look for largest common substings until we get an empty string as the common substring for any two paths. Thus, we end up with new paths $P'_1, P'_2, \cdots P'_r$ where each P'_i consists of one or more segments with empty string as the separator[1]. These remaining segments in each P'_i are unique to P_i. Each of them are also primitives and form the members of the set $\mathcal{B}$. Our objective was to find these two sets $\mathcal{A}$ and $\mathcal{B}$ as was stated in Sec. III.

We also note that the computational complexity of calculating the longest common subsequence between two sequences of length T_i and T_j is $O(T_i + T_j)$. Hence primitive finding is solvable in linear time.

D. Generating the grammar for primitives

Let $\mathcal{S}' = \{c_1, c_2, \cdots c_p\}$ be the set of primitives available to us. We wish to generate rules of the

[1]The segmentation is caused by the gaps produced by the removal of elements of $\mathcal{A}$.

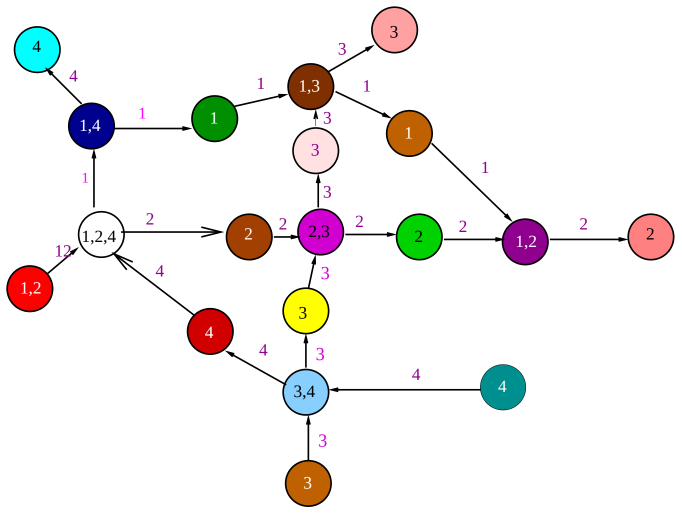

Fig. 2: Directed graph for finding the grammar

form $P(c_i \rightarrow c_j)$ which will give the likelihood of occurrence of the primitive c_j followed by primitive c_i. We do this by constructing a directed graph G which encodes the relations between the primitives. Using G we will derive a formal grammar for the elements in $\mathcal{S}'$.

Let n be the number of types of data that we have processed. Then, each of the states in our final HMM λ_M will have labels from a subset of $\{1, 2, \cdots, n\}$, see Fig. 2. By way of definition each of the states that belong to a primitive c_i will have the same label set l^{c_i}. Let $\mathcal{L} = \{l_1, l_2 \cdots, l_p\}\ \ p \geq n$ be the set of different type of labels received by the primitives. Let $G = (V, E)$ be a directed graph where $V = \mathcal{S}'$ and $e_{ij} = (c_i, c_j) \in E$ if there is a path $P_k = \cdots c_i c_j \cdots$ for some k. We have given the directed graph constructed for our test data described in Sec. III-E2 in Fig. 2.

We proceed to derive an SCFG from the directed graph G we have constructed. Let $N = \mathcal{S}'$ be the set of terminals. To each vertex c_i with an outgoing edge with label $l^{e_{ij}}$, associate a corresponding non-terminal $A^{l^{e_{ij}}}_{c_i}$. Let $\mathcal{N} = S \cup \{A^{l^{e_{ij}}}_{c_i}\}$ be the set of all non-terminals where S is the start symbol. For each primitive c_i that occurs at the start of a sequence and connecting to c_j define the rule

$$S \longrightarrow c_i A^{l^{c_i}}_{c_j} \quad . \tag{2}$$

To each of the internal nodes c_j with an incoming edge e_{ij} connecting from c_i and an outgoing edge e_{jk} connecting to c_k define the rule

$$A^{l^{c_i} \cap l^{c_j}}_{c_i} \longrightarrow c_j A^{l^{c_j} \cap l^{c_k}}_{c_k} \quad . \tag{3}$$

For each leaf node c_j with an incoming edge e_{ij} connecting from c_i and no outgoing edge define the rule

$$A_{c_j}^{l^{c_i} \cap l^{c_j}} \longrightarrow \epsilon \ . \tag{4}$$

The symbol ϵ denotes an empty string. We assign equal probabilities to each of the expansions of a nonterminal symbol except for the expansion to an empty string which occurs with probability 1. Thus

$$P(A_{c_i}^{l_{ij}} \longrightarrow c_j A_{c_j}^{l_{jk}}) \ = \ \frac{1}{|c_i^{(o)}|} \ \text{ if } |c_i^{(o)}| > 0 \ . \tag{5}$$

$$P(A_{c_i}^{l^{e_{ij}}} \longrightarrow \epsilon) \ = \ 1 \ \text{ if } |c_i^{(o)}| = 0 \ . \tag{6}$$

where $|c_i^{(o)}|$ represents the number of outgoing edges from c_i and $l_{mn} = l^{c_m} \cap l^{c_n}$. Let $\mathcal{R}$ be the collection of all rules given in (2), (3), and (4). For each $r \in \mathcal{R}$ associate a probability $P(r)$ using (5) and (6). Then $(\mathcal{N}, \mathcal{S}', S, \mathcal{R}, P(.))$ is the stochastic grammar that models our primitives.

One might wonder why the HMM λ_M is not enough to describe the grammatical structure of the observations and why the SCFG is necessary. The HMM λ_M would have been sufficient for a single observation type. However for several observation types as in final λ_M, regular grammars, as modeled by HMMs are usually too limited to model the different observation types so that different observation types can be confused.

E. *Experiments*

We have run three experiments: In the first experiment we generate a simple data set with very simple cross-shaped paths. The second experiment is motivated by the surveillance scenario of Stauffer and Grimson [11] and shows a complex set of paths as found outside our building. The third experiment is motivated by the work of Vincente and Kragic [9] on the recognition of human arm movements.

1) Testing on Simulated Data: We illustrate the result of testing our method on a set of two sequences generated with mouse clicks. The original data set for testing is shown in Fig. 3. We have two paths one from A to B and the other from C to D. These paths intersect in the middle at E. If we were to remove the points around E we will the have segments AE, EB, CE and ED. We extracted these segments with the above mentioned procedure. The result covering with Gaussians is shown in Fig. 4. When the model merging took place, the overlapping states in the middle were merged into one. The result is shown in Fig. 5. The primitives that we get are colored. As one can see in Fig. 5, primitive b is a common primitive and belongs to our set A and primitives a, c, d and e belong to our set B.

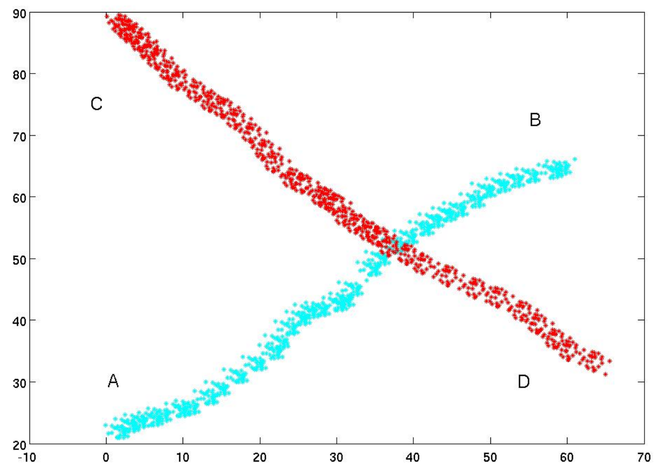

Fig. 3: Simple simulated 2d data sequences.

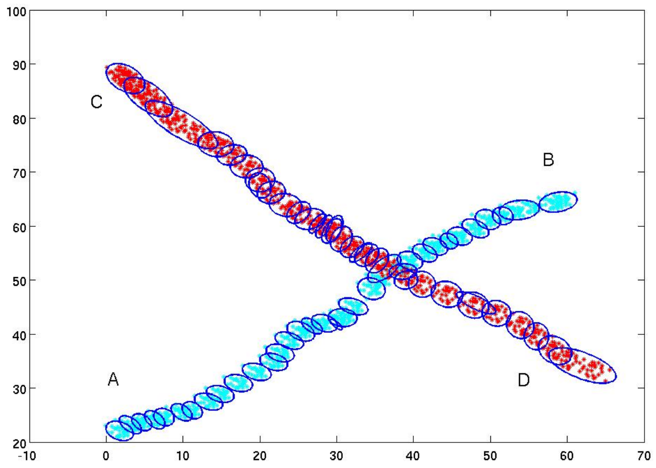

Fig. 4: This figure shows the automatically generated states for the simulated data.

2) 2D-Trajectory Data : The second experiment was done on a surveillance-type data inspired by [11]. The paths represent typical walking paths outside of our building. In this data there are four different types of trajectories with heavy overlap, see Fig. 6. We can also observe that the data is quite noisy. Fig. 7 shows the result of covering with Gaussians.

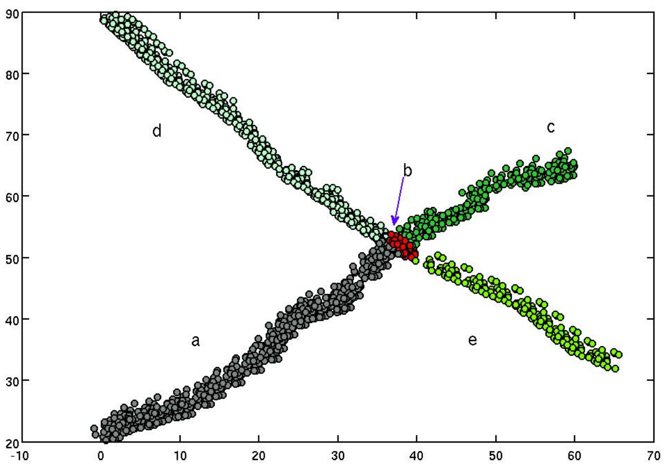

Fig. 5: The finally detected primitives with different colors. Primitive b is a common primitive and belongs to set A, primitives a, c, d and e belong to set B.

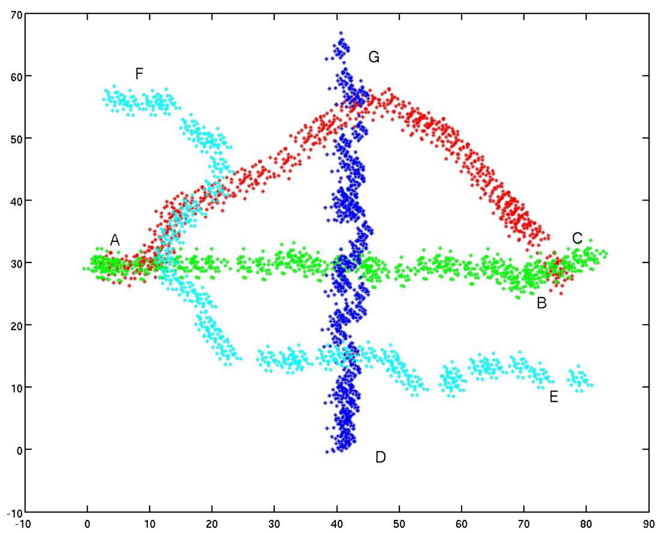

Fig. 6: Trajectories from tracking data. Each type is colored differently. Starting points are A, A, D and E for red, green, blue and cyan trajectories respectively. Only a part of the whole data is shown.

The result of primitive segmentation is shown in Fig. 8 on the right. Different primitives are colored differently and we have named the primitives with different letters. As one can see, our approach results in primitives that coincide roughly with our intuition. Furthermore, our approach is very robust even with such noisy observations and lot of overlaps.

It should also be noted at this point that this kind

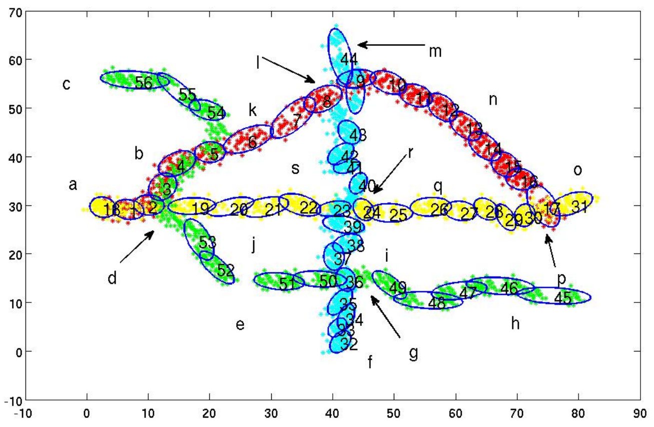

Fig. 7: The results of the covering procedure with the Gaussian mixtures. The numbers shown are the state numbers in the final model.

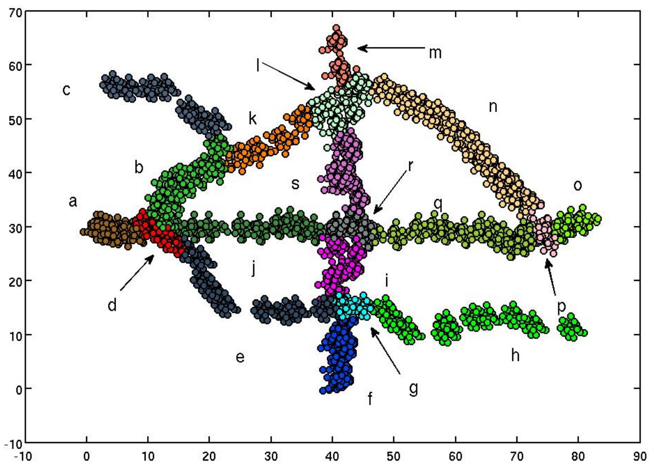

Fig. 8: This figure shows the detected primitives. Each primitive is colored differently and is denoted by a letter.

of merging will not make the intersection arbitrarily large. Merging is done only when there is a good overlap. Also for each new type of sequences, there cannot be more than one Gaussian that gets merged into the same state.

3) Hand gesture data: Finally, we have tested our approach on the dataset described in Sec. II without annotation. Thus we use only the trajectory information for the sensors attached to the hand. The original data is available on line [12]. We expect to extract a set of primitives so that each of these sequences can be expressed as a combination of these primitives. Since each of these sequences started and ended at the same position, we expect

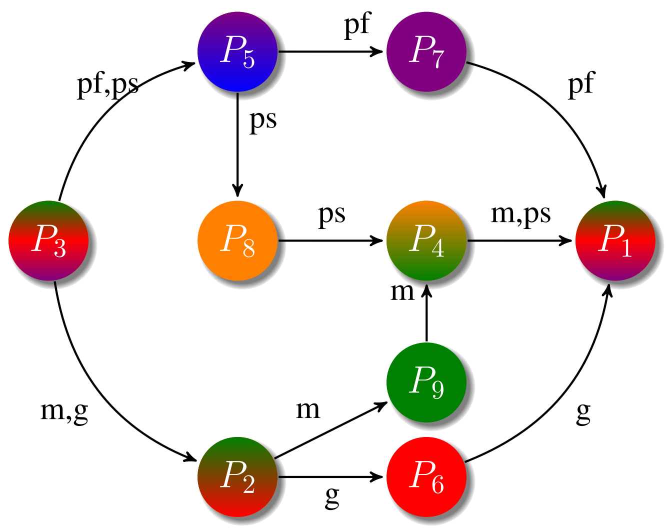

Fig. 9: The temporal order for primitives of hand gesture data. Node number corresponds to different primitives. Multi-colored nodes belong to more than one action. All actions start with P_3 and end with $P1$.

the primitives that represent the starting and end positions of actions will be the same across all the actions.

By applying the techniques described in Sec. III to the hand gesture data, we ended up with 9 primitives. The temporal order of primitives for actions for different actions are shown in Fig. 9. One can compare this with Fig. 1 and see that they are very closely related. For an easy comparison we plot the result of converting a grasp action sequence into a sequence of extracted primitives along with ground truth data in Fig. 10. We can infer from the figures Fig. 9 and Fig. 10 that P_3 and P_2 together constitute *approach* primitive, P_6 refers to *grasp* primitive and P_6 corresponds to *remove* primitive. Similar comparison could be made with other actions using the comparison diagram given in Fig. 11.

Using these primitives, an SCFG was built as described in Sec. III-D. This grammar is used as an input to the Natural Language Toolkit (NLTK, *http://nltk.sourceforge.net*) which is used to parse the sequence of primitives.

Results of primitive segmentation for *push sideways, push forward, move,* and *grasp* actions are shown in the tables III, IV, V and VI respectively. The numbers given in the tables

Person	Primitives for Push Aside action				
Person 1	3	2	9	4	1
Person 2	3	5	8	4	1
Person 3	3	5	8	4	1
Person 4	3	5	8	4	1
Person 5	3	5	8	4	1
Person 6	3	5	8	4	1
Person 7	3	5	8	4	1
Person 8	3	5	8	4	1
Person 9	3	2	9	4	1
Person 10	3	2	9	4	1

TABLE III: Primitive segmentation and recognition results for Push aside action. Sequences that are identified incorrectly are marked with yellow color.

represent the primitive numbers shown in Fig. 9 . The sequences that are identified correctly are marked with Aqua color and the sequences that are not classified correctly are marked with yellow color. We can see that all the correctly identified sequences start and end with the same primitive as expected. In Tab:VI, Person 1 and Person 4 are marked with a lighter color to indicate that they differ in end and start primitive respectively from the correct primitive sequence. This might be due to the variation in the starting and end position in the sequence. We could still see that the primitive sequence is correct for them.

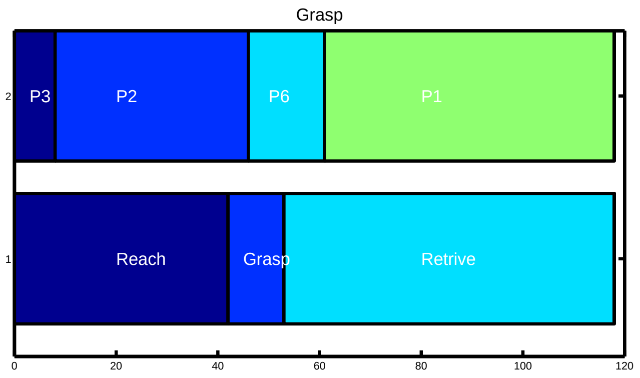

Fig. 10: Comparing automatic segmentation with manually segmented primitives for one grasp sequence. Using the above diagram with Fig. 9, we can infer that P_3 and P_2 together constitute *approach* primitive, P_6 refers to *grasp* primitive and P_1 corresponds to *remove* primitive.

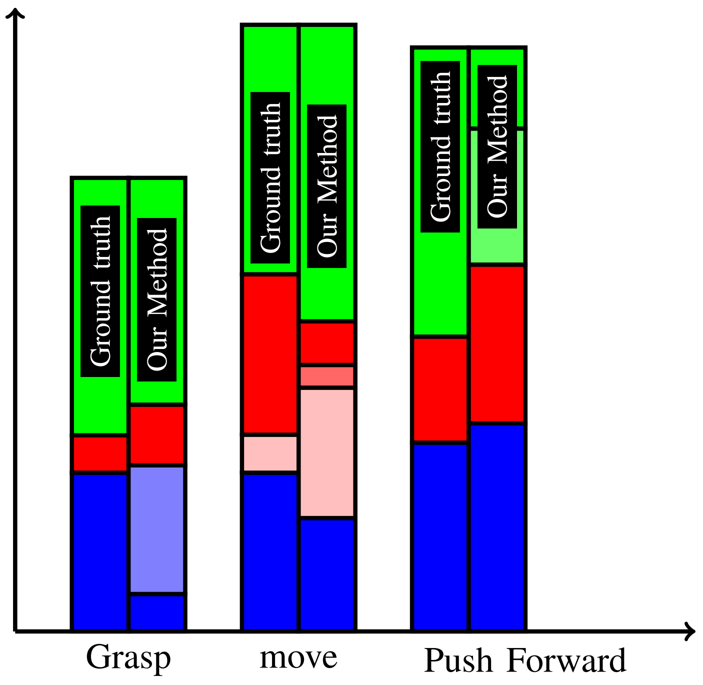

Fig. 11: Comparing primitive segmentation with ground truth data

Person	Primitives for Push Forward action				
Person 1	3	5	7		1
Person 2	3	5	7		1
Person 3	3	5	7		1
Person 4	3	5	7		1
Person 5	3	5	7		1
Person 6	3	5	8	4	1
Person 7	3	5	7		1
Person 8	3	5	7		1
Person 9	3	5	8	4	1
Person 10	3	5	8	4	1

TABLE IV: Primitive segmentation and recognition results for Push Forward action. Sequences that are identified incorrectly are marked with yellow color.

IV. Conclusions

We have presented and tested an approach for automatically computing a set of primitives and the corresponding stochastic context free grammar from a set of training observations. Our stochastic regular grammar is closely related to the usual HMMs. One important difference between common HMMs and a stochastic grammar with primitives is that with usual HMMs, each trajectory (action, arm movement, etc.) has its own, distinct HMM. This means that the set of HMMs for the given trajectories are not able to reveal any commonalities between them. In case of our arm movements, this means that one is not able to deduce that some actions share the grasp movement part. Using the primitives and the grammar, this is different. Here, common primitives are shared across the different actions which results into a somewhat symbolic representation of the actions. Indeed, using the primitives, we are able to do the recognition in the space of the primitives or symbols, rather than in the signal space directly, as it would be the case when using distinct HMMs. Using this symbolic representation would even allow to use AI techniques for, e.g., planning or plan recognition. Another important aspect of our approach is that we can modify our model to include a new action without requiring the storage of previous actions for it.

Person	Primitives for Move action				
Person 1	3	2	9	4	1
Person 2	3	5	8	4	1
Person 3	3	2	9	4	1
Person 4	3	2	9	4	1
Person 5	3	2	9	4	1
Person 6	3	5	8	4	1
Person 7	3	2	9	4	1
Person 8	3	2	9	4	1
Person 9	3	2	9	4	1
Person 10	3	2	9	4	1

TABLE V: Primitive segmentation and recognition results for Move Object action. Sequences that are identified incorrectly are marked with yellow color.

Our work is segmenting an action into smaller meaningful segments and hence different from [13] where the authors aim at segmenting actions like walk and run from each other. Many authors point at the huge task of learning parameters and the size of training data for an HMM when the number of states are increasing. But in our method, transition, initial and observation probabilities for all states are assigned during our merging phase and hence the use of the EM algorithm [14] is not required. Thus our method is scalable to the number of states. Our approach of using states have a close connection to [15] but our method is superior in preserving the temporal order and hence in recognition.

Person	Primitives for Grasp action				
Person 1	3	2	6		
Person 2	3	2	6		1
Person 3	3	5	7	6	1
Person 4		2	6		1
Person 5	3	2	6		1
Person 6	3	2	6		1
Person 7	3	2	9	4	1
Person 8	3	2	6		1
Person 9	3	2	6	7	1
Person 10	3	2	6		1

TABLE VI: Primitive segmentation and recognition results for Grasp Object action. Sequences that are identified incorrectly are marked with yellow color. Peson 1 and Person 4 differ in their starting and ending primitive from the correct primitive sequence. This could be explained as the variation in the starting and ending positions for those sequences.

It is interesting to note that stochastic grammars are closely related to Belief networks [16] where the hierarchical structure coincides with the production rules of the grammar. We will further investigate this relation ship in future work.

In future work, we will also evaluate the performance of normal and abnormal path detection using our primitives and grammars.

References

[1] S. Schaal, “Is imitation learning the route to humanoid robots?,” *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] K. Ogawara, S. Iba, T. Tanuki, H. Kimura, and K. Ikeuchi, “Recognition of human task by attention point analysis,” *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3, pp. 2121–2126, 2000.

[3] Y. Kuniyoshi, M. Inaba, and H. Inoue, “Learning by watching: extracting reusable task knowledge from visual observation of human performance,” *Robotics and Automation, IEEE Transactions on*, vol. 10, pp. 799–822, Dec 1994.

[4] S. Ekvall, *Robot task learning from human demonstration*. PhD thesis, KTH, Stockholm, Sweden, 2007.

[5] A. Billard, S. Calinon, and R. Dillmann, “Robot programming by demonstration,” in *Springer Handbook of Robotics*, pp. 1371–1394, 2008.

[6] S. Ekvall and D. Kragic, “Grasp recognition for programming by demonstration tasks,” in *IEEE Int. Conf. on Robotics and Automation, ICRA ’05*, pp. 748–753, 2005.

[7] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[8] Y. Ivanov and A. Bobick, “Recognition of visual activities and interactions by stochastic parsing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 852–872, 2000.

[9] I. S. Vicente, V. Kyrki, and D. Kragic, “Action recognition and understanding through motor primitives,” *Advanced Robotics*, vol. 21, pp. 1687–1707, 2007.

[10] V. Krueger, D. Kragic, A. Ude, and C. Geib, “Meaning of action,” *Int. Journal on Advanced Robotics, Special issue on Imitative Robotics, T. Inamura and G. Metta (eds.)*, 2007.

[11] C. Stauffer and W. Grimson, “Learning Patterns of Activity Using Real-Time Tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, 2000.

[12] http://www.nada.kth.se/~danik/gesture_database/.

[13] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, “Segmenting motion capture data into distinct behaviors,” in *GI ’04: Proceedings of Graphics Interface*, pp. 185–194, Canadian Human-Computer Communications Society, 2004.

[14] A. Dempster, , M. Laird, and D. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[15] A. Bobick and A. Wilson, “A state-based approach to the representation and recognition of gesture,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 12, pp. 1325–1337, Dec 1997.

[16] M. I. Jordan, ed., *Learning in graphical models*. MIT Press, 1998.