Primitive Based Action Representation and Recognition

Sanmohan and Volker Krüger

Computer Vision and Machine Intelligence Lab,
Copenhagen Institute of Technology, 2750 Ballerup, Denmark
`{san,vok}@cvmi.aau.dk`

Abstract. There has been a recent interest in segmenting action sequences into meaningful parts (action primitives) and to model actions on a higher level based on these action primitives. Unlike previous works where action primitives are defined a-priori and search is made for them later, we present a sequential and statistical learning algorithm for automatic detection of the action primitives and the action grammar based on these primitives. We model a set of actions using a single HMM whose structure is learned incrementally as we observe new types. Actions are modeled with sufficient number of Gaussians which would become the states of an HMM for an action. For different actions we find the states that are common in the actions which are then treated as an action primitive.

1 Introduction

Similar to phonemes being the building blocks of human language there is biological evidence that human action execution and understanding is also based on a set primitives [2]. But the notion of primitives for action does not only appear in neuro-biological papers. Also in the vision community, many authors have discussed that it makes sense to define a hierarchy of different action complexities such as *movements*, *activities* and *actions* [3]. In terms of Bobick's notations, *movements* are action primitive, out of which activities and actions are composed.

Many authors use this kind of hierarchy as observed in the review by Moeslund et al [9]. One way to use such a hierarchy is to define a set of action primitives in connection with a stochastic grammar that uses the primitives as its alphabet. There are many advantages of using primitives: (1) The use of primitives and grammars is often more intuitive for the human which simplifies verification of the learning results by an expert (2) *Parsing* primitives for recognition instead of using the signal directly leads to a better robustness under noise [10][14] (3) AI provides powerful techniques for higher level processing such as planning and plan recognition based on primitives and parsing. In some cases, it is reasonable to define the set of primitives and grammars by hand. In other cases, however, one would wish to compute the primitives and the stochastic grammar automatically based on a set of training observations. Examples for this can be found in surveillance, robotics, and DNA sequencing.

In this paper, we present an HMM-based approach to learn primitives and the corresponding stochastic grammar based on a set of training observations. Our approach is able to learn on-line and is able to refine the representation when newly incoming data supports it. We test our approach on a typical surveillance scenario similar to [12] and on the data used in [14] for human arm movements.

A number of authors represent action in a hierarchical manner. Staffer and Grimson [12] compute for a surveillance scenario a set of action primitives based on co-occurrences of observations. This work is used to motivate the surveillance setup of one of our experiments.

In [11] Robertson and Reid present a full surveillance system that allows high-level behavior recognition based on simple actions. Their system seems to require human interaction in the definition of the primitive actions such as *walking*, *running*, *standing*, *dithering* and the qualitative positions (*nearside-pavement*, *road*, *driveway*, *etc*). This is, what we would like to automate.

In [4] actions are recognized by computing the cost through states an action pass through. The states are found by k-means clustering on the prototype curve that best fits sample points according to a least square criterion. Hong et al [8] built a Finite State Machine for recognition by building individual FSM s for each gesture. Fod et al. [5] uses a segmentation approach using zero velocity crossing. Primitives are then found by clustering in the projected space using PCA.

The idea of segmenting actions into atomic parts and then modeling the temporal order using Stochastic Context Free Grammar is found in [7]. In [6], signs of first and second derivatives are used to segment action sequences. These works require the storage of all training data if one wishes to modify the model to accommodate a new action. Our approach eliminate this requirement and thus make it suitable for imitation learning.

Our idea of merging of several HMMs to get a more complex and general model is found in [13]. We propose a merging strategy for continuous HMMs. New models can be introduced and merged online.

1.1 Problem Statement

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 not shared and parts that are common across sequences types. 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 $\mathcal{L} = \{X_1, X_2, \dots, X_m\}$ be a set of data sequences where each X_i is of the form $x_1^i x_2^i \dots, x_{T_i}^i$ and $x_j^i \in \mathbb{R}^n$. Let these observations be generated from a finite set of sources (or states) $\mathcal{S} = \{s_1, s_2, \dots, s_r\}$. Let $S_i = s_1^i s_2^i \dots, s_{T_i}^i$ 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 $\mathcal{A} = \{a_1, a_2, \dots, a_k\}$ and $\mathcal{B} = \{b_1, b_2, \dots, b_l\}$ 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.

2 Modeling the Observation Sequences

We take the first sequence of observations X_1 with data points $x_1^1 x_2^1 \dots x_{T_1}^1$ and 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) , $i = 1, 2, \dots, k^1$ so that parts of the data sequence are from $\mathcal{N}(\mu_i^1, \Sigma_i^1)$ in that order. The value of k^1 is such that $\mathcal{N}(\mu_i^1, \Sigma_i^1)$, $i = 1, 2, \dots, k^1$ 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 $\lambda_1 = (A^1, B^1, \pi^1)$ with k^1 states. We let A^1 to be a left-right transition matrix and $B_j^1(x) = \mathcal{N}(x, \mu_j^1, \Sigma_j^1)$. All the states at this stage get a label 1 to indicate that they are part of sequence type 1. This model will now be modified recursively.

Now we will modify this model by adding new states to it or by modifying the current output probabilities of states so that the modified model λ_M will be able to generate new types of data with high probability. 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. 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 the beginning and the states are labeled n . The newly constructed HMM λ_c will be merged to λ_M so that the updated λ_M will be able to generate data sequences of type X_c .

Suppose we want to merge λ_c into λ_M so that $P(X_k | \lambda_M)$ is high if $P(X_k | \lambda_c)$ is high. Let $C_c = \{s_{c1}, s_{c2}, \dots, s_{ck}\}$ and $C_M = \{s_{M1}, s_{M2}, \dots, s_{Ml}\}$ 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 $p \in \{1, 2, \dots, l\}$, 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 $b_{s_{ci}}^k(x)$. Thus $b_{Mj}^M(x)$ 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} . The basic idea behind merging is that we do not need two different states which describe the same part of the data.
2. If $d(s_{ci}, s_{Mj}) > \theta$, $\forall j$, 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 :

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

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

2.1 Finding Primitives

When all sequences have been processed, we apply Viterbi algorithm on the final merged model λ_M , and find the hidden states associated with each of the sequences. Let $P_1, P_2, \dots, P_r$ 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 , $i = 1, 2, \dots, r$.

We continue to look for largest common substrings until we get an empty string as the common substring for any two paths. Thus we end up with new paths $P'_1, P'_2, \dots, P'_r$ where each P'_i consists of one or more segments with empty string as the separator. 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. 1.1.

3 Generating the Grammar for Primitives

Let $\mathcal{S}' = \{c_1, c_2, \dots, c_p\}$ be the set of primitives available to us. We wish to generate rules of the 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 $\{1, 2, \dots, n\}$, see Fig.1. 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 $\mathcal{L} = \{l_1, l_2, \dots, l_p\}$ $p \geq n$ 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 $P_k = \dots c_i c_j \dots$ for

Fig. 1. The figure on the left shows the directed graph for finding the grammar for the simulated data explained in experiments section. Right figure: 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 P_1 . Here g=grasp, m=move object, pf=push forward and ps=push sideways.

some k . We have given the directed graph constructed for our test data in Fig. 1.

We proceed to derive a precise Stochastic Context Free Grammar (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 $A_{c_i}^{l^{e_{ij}}}$. Let $\mathcal{N} = S \cup \{A_{c_i}^{l^{e_{ij}}}\}$ 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 $S \rightarrow c_i A_{c_j}^{l^{e_{ij}}}$. 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 $A_{c_i}^{l^{e_{ij}} \cap l^{e_{jk}}} \rightarrow c_j A_{c_k}^{l^{e_{jk}}}$. For each leaf node c_j with an incoming edge e_{ij} connecting from c_i and no outgoing edge define the rule $A_{c_j}^{l^{e_{ij}} \cap l^{e_{jk}}} \rightarrow \epsilon$. 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 $P(A_{c_i}^{l^{e_{ij}}} \rightarrow c_j A_{c_j}^{l^{e_{jk}}}) = \frac{1}{|c_i^{(o)}|}$ if $|c_i^{(o)}| > 0$ and $P(A_{c_i}^{l^{e_{ij}}} \rightarrow \epsilon) = 1$ otherwise..

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 above. For each $r \in \mathcal{R}$ associate a probability $P(r)$ as given in the construction of rules. Then $(\mathcal{N}, \mathcal{S}', S, \mathcal{R}, P(\cdot))$ 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.

Fig. 2. The top left figure shows the simulated 2d data sequences. The ellipses represent the Gaussians. The top right figure shows the finally detected primitives with different colors. Primitive b is a common primitive and belongs to set $\mathcal{A}$, primitives a,c,d,e belong to set $\mathcal{B}$. The bottom left figure shows trajectories from tracking data. Each type is colored differently. Only a part of the whole data is shown. The bottom right figure shows the detected primitives. Each primitive is colored differently.

4 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 [12] and shows a complex set of paths as found outside our building. The third experiment is motivated by the work of Vincente and Kragic [14] on the recognition of human arm movements.

4.1 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. 2 at top left. We have two paths which intersect in the middle. If we were to remove the intersecting points we will get four segments. We extracted these segments with the above mentioned procedure. When the model merging took place, the overlapping states in the middle were merged into one. The result is shown in Fig. 2 at top right. The primitives that we get are colored. As one can see in Fig. 2, primitive b is a common primitive and belongs to our set $\mathcal{A}$, primitives a,c,d,e belong to our set $\mathcal{B}$.

Fig. 3. Comparing automatic segmentation with manually segmented primitives for one grasp sequence. Using the above diagram with the right figure in Fig.1, 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.

4.2 2D-Trajectory Data

The second experiment was done on a surveillance-type data inspired by [12]. 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. 2 bottom left. We can also observe that the data is quite noisy.

The result of primitive segmentation is shown in Fig. 2 on the bottom 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.

Hand Gesture Data. Finally, we have tested our approach on the dataset provided by Vincente and Kragic [14]. In this data set, several volunteers performed a set of simple arm movements such as *reach for object*, *grasp object*, *push object*, *move object*, and *rotate object*. Each action is performed in 12 different conditions: two different heights, two different locations on the table, and having the demonstrator stand in three different locations (0,30, 60 degrees). Furthermore all actions are demonstrated by 10 different people. The movements are measured using magnetic sensors placed on: chest, back of hand, thumb, and index finger. In [14], the segmentation was done manually and their experiments showed that the recognition performance of human arm actions is increased when one uses action primitives. Using their dataset, our approach is able to provide the primitives and the grammar automatically. We consider the 3-d trajectories

Table 1. Primitive segmentation and recognition results for Push aside and Push Forward action. Sequences that are identified incorrectly are marked with yellow color.

Person	Push Aside				
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

Person	Push Forward				
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

for the first four actions listed above along with a scaled velocity component. Since each of these sequences started and ended at the same position, we expect 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.2 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.1. We also compare our segmentation with the segmentation in [14]. We plot the result of converting a grasp action sequence into a sequence of extracted primitives along with ground truth data in Fig.3. We can infer from the figures Fig.1 and Fig.3 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 also.

Using these primitives, an SCFG was built as described in Sec.3. 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.

Table 2. Primitive segmentation and recognition results for Move Object and Grasp actions. Sequences that are identified incorrectly are marked with yellow color.

Person	Move				
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

Person	Grasp				
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

Results of primitive segmentation for *push sideways*, *push forward*, *move*, and *grasp* actions are shown in the tables 1 and 2. The numbers given in the tables represent the primitive numbers shown in Fig. 1. 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.2, 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.

5 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 become 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.

Our work is segmenting an action into smaller meaningful segments and hence different from [1] 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 EM algorithm is not required. Thus our method is scalable to the number of states.

It is interesting to note that stochastic grammars are closely related to Belief networks 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. Barbič, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J.K., Pollard, N.S.: Segmenting motion capture data into distinct behaviors. In: *GI 2004: Proceedings of Graphics Interface 2004*, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, pp. 185–194. Canadian Human-Computer Communications Society (2004)
2. Bizzi, E., Giszter, S.F., Loeb, E., Mussa-Ivaldi, F.A., Saltiel, P.: Modular organization of motor behavior in the frog's spinal cord. *Trends Neurosci.* 18(10), 442–446 (1995)
3. Bobick, A.: Movement, Activity, and Action: The Role of Knowledge in the Perception of Motion. *Philosophical Trans. Royal Soc. London* 352, 1257–1265 (1997)
4. Bobick, A.F., Wilson, A.D.: A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(12), 1325–1337 (1997)
5. Fod, A., Matarić, M.J., Jenkins, O.C.: Automated derivation of primitives for movement classification. *Autonomous Robots* 12(1), 39–54 (2002)
6. Guerra-Filho, G., Aloimonos, Y.: A sensory-motor language for human activity understanding. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*, December 4–6, 2006, pp. 69–75 (2006)
7. Fermüller, C., Guerra-Filho, G., Aloimonos, Y.: Discovering a language for human activity. In: *AAAI 2005 Fall Symposium on Anticipatory Cognitive Embodied Systems*, Washington, DC, pp. 70–77 (2005)
8. Hong, P., Turk, M., Huang, T.: Gesture modeling and recognition using finite state machines (2000)
9. Moeslund, T., Hilton, A., Krueger, V.: A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding* 104(2–3), 90–127 (2006)
10. Rabiner, L.R., Juang, B.H.: *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs (1993)
11. Robertson, N., Reid, I.: Behaviour Understanding in Video: A Combined Method. In: *International Conference on Computer Vision*, Beijing, China, October 15–21 (2005)
12. Stauffer, C., Grimson, W.E.L.: Learning Patterns of Activity Using Real-Time Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8), 747–757 (2000)
13. Stolcke, A., Omohundro, S.M.: Best-first model merging for hidden Markov model induction. Technical Report TR-94-003, 1947 Center Street, Berkeley, CA (1994)
14. Vicente, I.S., Kyrki, V., Kragic, D.: Action recognition and understanding through motor primitives. *Advanced Robotics* 21, 1687–1707 (2007)