Primitive Based Action Representation and recognition

Baby, Sanmohan

Publication date:
2010

Document Version
Tidlig version også kaldet pre-print

Link to publication from Aalborg University

Citation for published version (APA):
Baby, S. (2010). *Primitive Based Action Representation and recognition*. Computer Vision and Media Technology Laboratory (CVMT), Aalborg University.

Primitive Based Action Representation and Recognition

A PhD dissertation

by

Sanmohan
Computer Vision and Machine Intelligence Lab
Aalborg University Copenhagen, Denmark
Email:san@cvmi.aau.dk

Abstract

The presented work is aimed at designing a system that will model and recognize actions and its interaction with objects. Such a system is aimed at facilitating robot task learning. Activity modeling and recognition is very important for its potential applications in surveillance, human-machine interface, entertainment, biomechanics etc. Recent developments in neuroscience suggest that all actions are a compositions of smaller units called primitives.

Current works based on primitives for action recognition uses a supervised framework for specifying the primitives. We propose a method to extract primitives automatically. These primitives are to be used to generate actions based on certain rules for combining. These rules are expressed as a stochastic context free grammar. A model merging approach is adopted to learn a Hidden Markov Model to fit the observed data sequences. The states of the HMM approximates local properties of a long sequence.

Observation sequences are used to learn a model expressing the data in a structured way. Based on the learned model, recurring parts in the sequences are identified. Primitives that make up the observation sequences are identified as the recurring and unique parts appearing in the sequences. Extracted primitives are used to make a primitive graph from which a grammar for the observed primitives are derived.

This method is further extended to include object context. We observe that human actions and objects can be seen as being intertwined: we can interpret actions from the way the body parts are moving, but as well from how their effect on the involved object. While human movements can look vastly different even under minor changes in location, orientation and scale, the use of the object can provide a strong invariant for the detection of motion primitives. Movements that produce the same state change in the object state space are classified to be instances of the same action primitive. This allows us to define action primitives as sets of movements where the movements of each primitive are connected through the object state change they induce.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a longer time perspective, it is envisioned that robots will move into our home and offices. It would be necessary to have robot learning mechanisms that would enable the robots to adapt and operate in a dynamic environment because it would be impossible to pre-program a robot with all possible world states that it might encounter. It would be more desirable to teach the robot through examples. Then probably users can make a demonstration of a task and the robot can learn to do it. For example we can think of a table setting scenario of [3] where the robot has to learn to recognize the plates, glasses and other objects. Then it has to learn to grasp them in a robust manner, and transport them to the correct location on the table. The robot should understand, from examples, that glasses can go on top of plates but plates cannot go on top of glasses etc. It is also to be noted that the objects could be in a different location each time. Therefore it is not enough just to imitate the motion trajectory.

Robots can be taught to perform tasks in several ways [4, 5, 6, 7, 8, 3, 4, 9, 10, 11, 12]. According to [13], there are two diametrically opposite ways to teach robots: *tell* the robot in detail what it has to do or give the robots some learning strategy and let the robot figure out what the appropriate action is. The former strategy was common in the beginning and the robots were pre-programmed to operate in a specific and highly controlled environment and perform some pre-specified tasks for which controls were specified [14]. Such an approach is not suitable when the robot is required to learn a new task or when it needs to adapt to changing environment. Moreover, it is difficult to program complex tasks in detail and specify exhaustively all new situations the robot might encounter [13]. An example of this is the Honda robot [15, 16] that can walk, climb stairs and manipulate objects. It took nearly 10 years program the robot with these capabilities. On the other hand, learning strategies are meant to prepare robots to deal with new situations.

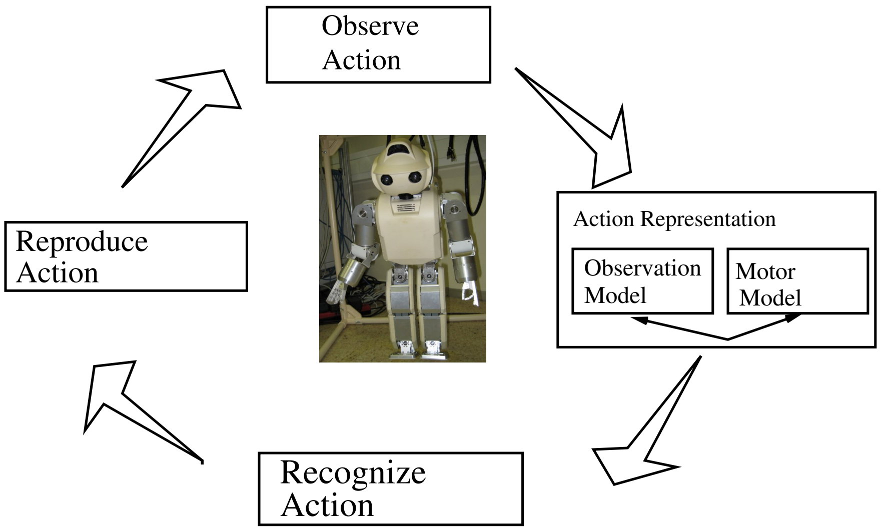


Figure 1.1: Imitation learning process. An internal representation is formed from observed actions for future recognition and reproduction. In this work, we deal with representing and recognizing actions.

The learning techniques such as reinforcement learning [17, 18, 7] and genetic algorithms [19] used so far have the capacity to learn anything theoretically but in practice their learning power is limited [13, 20]. A combination of programming and learning strategies can be found in [21] where a robot is programmed with a set of basis behaviors and is expected to learn to use these behaviors. This approach does not scale well in modeling higher-level behaviors [22].

As we pointed out earlier, robots are to become common in our daily lives. In such a scenario, a user friendly interaction and teaching method are required to improve the performance of the robot and deal with unforeseen circumstances. Imitation learning or learning from observation is seen as a natural solution to this problem [23, 24, 25]. In the imitation learning approach, the robot learns by observing an expert performing some task. Thus it is a simple channel of communication between the robot and an expert. The advantages of such an approach have been pointed out by several authors in the past [26, 27, 28, 29, 9, 30, 1]. Bakker and Kuniyoshi [13] points out adaptation, efficient communication, compatibility with other learning systems, efficient learning and good company as some of the reasons in endowing robots with the ability to imitate. This way even a person without the knowledge of complex robot programming can "program" the

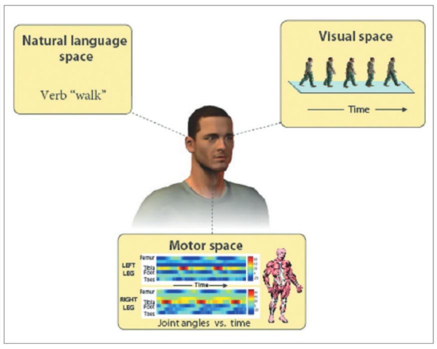


Figure 1.2: Representation of actions in different spaces [31].

robot. The user does not need to perform a complex robot programming to teach the robot.

Imitation learning scenario comes with a lot of challenges. Imitation depends on several perceptual, cognitive and motor capabilities. The imitating system should have perceptual capabilities to detect motion and object, motor capabilities to imitate the perceived action and cognitive capabilities to determine what to imitate, how to imitate etc. Other challenges such as evaluating the success of imitation, dealing with changing environment during the execution motion are also to be dealt with. By defining a metric of imitation, one can find an optimal controller to imitate by minimizing this metric [32, 33]. The concept of a metric of evaluation is illustrated in Fig. 1.3. Considering only the displacement of objects, one can evaluate the similarity of the effect of the demonstrated task and the effect of the imitated task using *relative displacement, absolute displacement* or *relative position*. Examples of situations where each of these metrics become useful can be provided [34]. The imitator might need help from a teacher or several repetition of the same task with different configurations to choose the correct metric of evaluation.

In imitation learning, robots are to learn from demonstrated examples, form an internal representation of the observed action, use the representation to recognize actions, and finally reproduce the observed actions. This process is illustrated in Fig. 1.1. Action representation plays a key role in this cycle of imitation learning. Encoding actions using primitives has been advocated as an efficient way to represent actions [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 29]. Primitives are units of behavior above the level of motor or

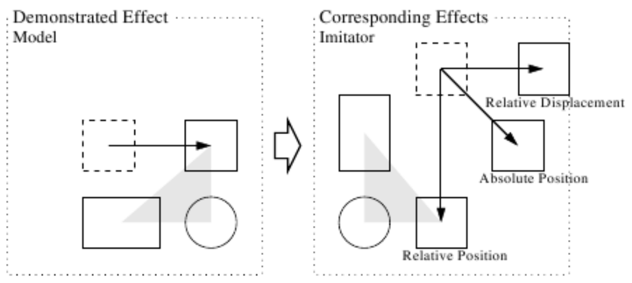


Figure 1.3: Illustration of the use of metric of imitation [34] . Depending on metric of imitation, dissimilar imitative behaviour can result from dissimilar configuration of objects.

muscle commands [41]. The concept of primitives can be motivated from several perspectives such as human movement hierarchies, human perception of actions and studies from neuro-science. We elaborate on each of these below.

The first motivation for primitives comes from the different levels/layers of hierarchy that can be used to analyze and interpret human movements. Bobick defined one such hierarchy [45]:

- **Movement:** A motion which is characterized by a definite space-time trajectory in some configuration space. For a given viewing condition, the appearance of movements is consistent. This is a basic motion that can be detected using low level processing of features.
- **Activity:** Activity describes motion consisting of a sequence of movements. Activities do not refer to external elements.
- **Action:** Actions include the context of the motion and can be considered as the highest level of abstraction. According to Bobick, actions are the boundary where perception meets cognition. This is because different instances of the same action can have different interpretation depending on the context or object being manipulated. To recognize an action, the observed motion has to be linked to its context.

In this thesis, our taxonomy to denote movement hierarchies will be slightly different from the one we have seen above and is given in Fig. 1.4. At the lowest level we have *primitives*. Several primitives put together can

Simple Action: An action sequence with only one type of movement in it. Eg. Pushing an object towards right.
Complex Action: An action sequence that contains more than one type of movement. Eg. Pushing object right followed by pushing object up.
Primitive: A segment of data consisting of one more points from data. The points in a primitive are adjacent in the time axis. Every action sequence can be written as a concatenation of primitives.

Figure 1.4: Meaning of some terminology used in this thesis.

generate a *simple action*. A simple action sequence will contain only one meaningful action such as *move the object from A to B*. Combinations of simple actions make a *complex action*.

We take the table setting scenario to explain the meaning and interpretation of these terms. *Picking up a knife* for arranging the table could be considered as a movement, see Fig. 1.5. *Picking up a knife* and *Placing the knife on the table* are sequences of two movements and make an activity. Now by associating the placing of the knife with the plate, we can infer that the action being done is 'arranging the table'. If the knife is picked up and placed in a dish washer then the action is 'cleaning up the table'. So the same sequence of activities describe two actions with the change of context. The knowledge required to recognize motion at different levels of hierarchy is different. Movements appear differently for different viewing conditions and can be recognized by analyzing the variation of the features over time. To recognize an activity, component movements and their statistical relations are required. To recognize action, the context of the motion is to be known. Primitives are analogous to the movement of Bobick, but need not be the same always. Sometimes several primitives are needed to make up a movement [36].

Primitives can also be motivated from the way we perceive motion. According to [31], there are at least three different spaces in which humans perceive an action as illustrated in Fig. 1.2. The *visual space* representation allows one to recognize an observed action. The *motor space* representation enables one to perform a task. The *natural language space* representation

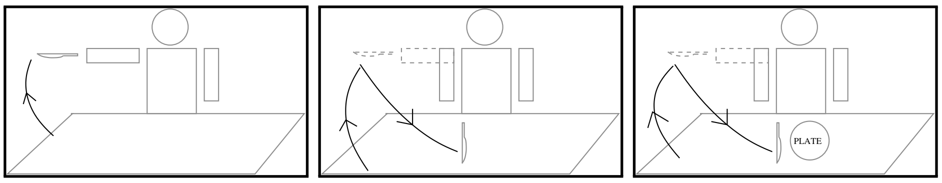


Figure 1.5: Illustration of the action hierarchy of Bobick[45]. *Picking up a knife* is a movement. *Picking up a knife* and *Placing knife on the table* are sequences of two movements and make an activity. Associating this activity with the plate on the table(which gives the context), we know the action being done is 'arranging the table'.

allows one to communicate about the action. A unifying frame work connecting these spaces will facilitate the building of intelligent systems that learn through imitation and *understand* human actions [31]. Primitive based representation of actions will help us to map the lower level signals to higher level natural language.

The final motivation for primitives comes from experiments in neuroscience. There are experimental evidences that suggest that the same part of brain is involved when an action is performed or when the same action is observed [46, 47, 48, 49, 50, 51] in animals. Similar results have been observed in humans as well. Xu et al. [52] have found that auditory and visual stimuli of the same action activate a common, left-lateralized network of inferior frontal and posterior temporal regions in which symbolic gestures and spoken words may be mapped onto common, corresponding conceptual representations.

Having seen the importance of primitives, the next challenge is to segment actions into a set of primitives. It is common to define the primitives by hand [53, 2, 41]. Hand seleceted primitives have the advantage that they are appealing and intuitive to humans and meaningful. But it would be nice to have learning mechanisms where these primitives could be learned from data. Primitives detected automatically using such learning methods do not always match with that of human defined primitives [18, 36].

Primitives vary depending on the application domain and the type of actions that are considered. In general action recognition scenarios, whole body motions are considered in a large amount of works. Overviews of such approaches are given in [54, 55, 56, 57]. In the robotics field, the interest mostly lies in gestures that are mainly used for human robot interaction. A

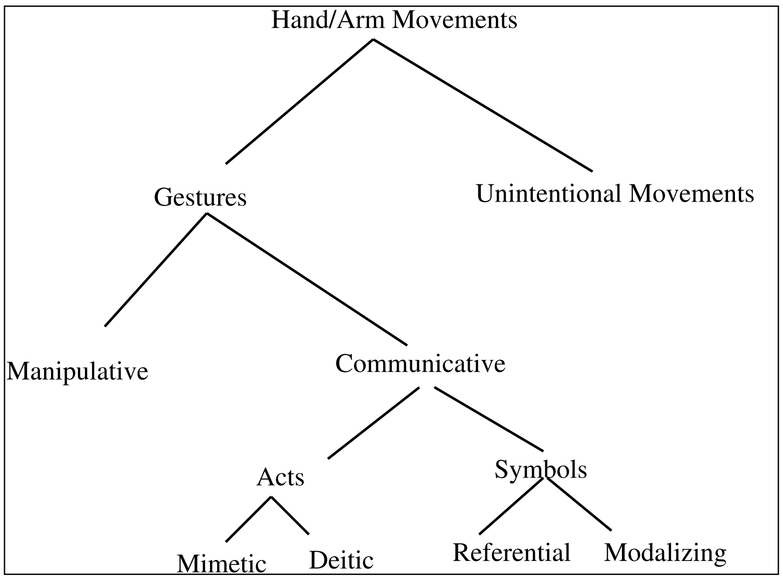


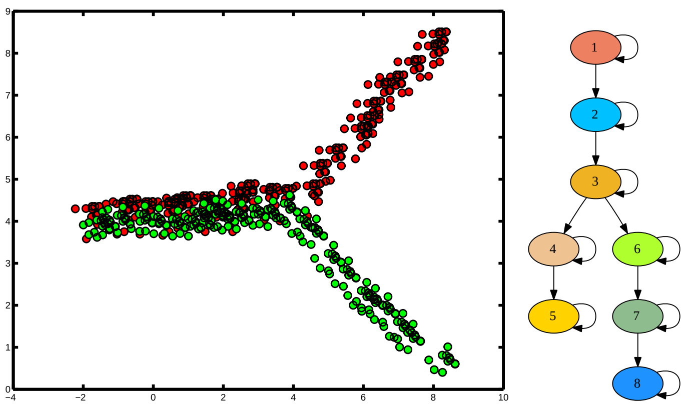
Figure 1.6: Taxonomy of hand gestures. Gestures used for manipulating objects are separated from the gestures for communication purposes.

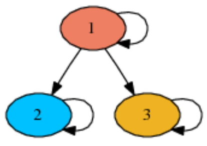
taxonomy of such gestures are given by [58] and is shown in Fig. 1.6. Manipulative gestures are the ones used to act on objects. We consider mainly manipulation actions in this thesis. It is common to segment manipulative gestures into three phases [58]:

- preparation
- nucleus(stroke [59])
- retraction

In [60] and [61, 62] hand gesture recognition is performed using different phases of actions. In [61, 62], the retraction part is treated as a part of the nucleus part. Our actions in this work consist of a *reach* part, *manipulation* part and a *retrieve* part.

We consider a scenario similar to the one studied in [2]. In their work, several manipulation actions were performed on an object in a table top. Each of the manipulation actions were defined as a primitive. We attempt to segment actions into primitives automatically for a similar scenario. At the same time, we would like to find out how these primitives are related. This will give us the knowledge about how these primitives are combined to generate complex actions. We find the relation among primitives as a





$$\begin{aligned} S &\rightarrow P1A_2^{12} && (0.5) \\ &\rightarrow P1A_3^{12} && (0.5) \\ A_2^{12} &\rightarrow P2 && (1.0) \\ A_3^{12} &\rightarrow P3 && (1.0) \\ P1 &\rightarrow p1 && (1.0) \\ P2 &\rightarrow p2 && (1.0) \\ P3 &\rightarrow p3 && (1.0) \end{aligned}$$

Figure 1.7: Overview of our modeling approach. Sequence of points are converted to sequence of states. Sequence of states are then converted to sequence of primitives. From the sequence of primitives we get a grammar for actions.

stochastic grammar. Our approach is tested with a publicly available data set used in [2] and our own data set [63]. Our approach is based on the assumption that similar actions have a common underlying hidden state sequence. Therefore actions are first expressed as a sequence of hidden states. Primitives are detected from these sequences by finding common state subsequences. From the temporal continuity of primitives, a grammar for the primitives can then be found. This is schematically represented in Fig. 1.7.

To summarize, we learn the primitives from observed data automatically. Actions are decomposed into primitives and a grammar that defines the rules to combine these primitives are extracted automatically. The main contributions of this work are:

1. Automatic extraction of primitives from observed data. Described in Ch. 4.
2. Use of object context in learning primitives. Described in Ch. 5.
3. Extraction of grammar for primitives. Described in Ch. 4 and Ch. 5.

4. Incremental learning of primitives and grammar. Described in Ch. 5.

5. Detection of learned primitives in novel sequences. Described in Ch. 5.

Most of the results presented in this thesis have been published in peer reviewed journals/conferences and are mentioned below:

1. **Learning Actions from Observations**, Volker Krueger, Dennis Herzog, Sanmohan, Danica Kragic and Ales Ude, IEEE Robotics and Automation Magazine, 17(2),30-43, June 2010.

2. **Unsupervised Learning of Action Primitives**, Sanmohan, Volker Krueger and Danica Kragic, Humanoids 2010, Accepted.

3. **Primitive based action representation and recognition**. Sanmohan, Volker Krueger, Danica Kragic and Hedvig Kjellström, Advanced Robotics, Accepted.

4. **Parametric Primitives for Hand Gesture Recognition**, Sanmohan and Volker Krueger, International Conference on Intelligent Control, Robotics, and Automation,639-642, Venice, Italy, October 28, 2009 - October 30, 2009.

5. **Primitive Based Action Representation and Recognition**, Sanmohan and Volker Krueger, Scandinavian Conference on Image Analysis, 2009, 31-40.

6. **Automatic primitive finding for action modeling**, Sanmohan and Volker Krueger, Themis 2008, British Machine Vision Conference Workshop Proceedings 2008, 35-43.

7. **Action Primitives Using Object Context**, Sanmohan, Volker Krueger and Danica Kragic, Advanced Robotics journal submission, under review.

The contents of Ch. 4 have been accepted by the Advanced Robotics journal(item 3 in the list above) and is currently scheduled to appear in Advanced Robotics Vol. 25 which will be published in April 2011. The contents of Ch. 5 will be published in the Proceedings of the Humanoids 2010 conference(item 2 in the list above). The complete results of Ch. 5 has been submitted to the Advanced Robotics journal and is under review.

CHAPTER 2

Related Work

Contents

In this chapter we review some of the important works related to this thesis. We consider mainly works that are related to learning primitives. Primitive based learning approaches have gained much attention in imitation learning but are not limited only to this field. Primitive learning is closely connected to several hierarchical action modeling approaches in computer vision and other related areas. Segmenting actions into primitives could be seen as imposing some kind of structure on the data. Identifying hidden structure in the data, one can represent actions at different levels of a hierarchy.

2.1 Learning without Primitives/Non-hierarchical Methods.

In this section we review some of the action recognition methods that do not resort to higher level modeling of data. Approaches in this category fall into two types: generative and discriminative methods. In generative models, a model that can generate data is constructed. In discriminative methods a decision boundary that best separates different types actions is sought. We are more interested in generative models and hence omit a discussion on discriminative methods.

Several authors have used HMMs for learning actions from demonstrated examples [64, 65, 66]. Tso et. al [65] have used HMMs to represent and reproduce Cartesian trajectories. They used the trajectory with the highest likelihood in the training set for reproducing a task. Yang et. al [67] have used HMMs for teleoperation of a robot gripper in the joint space or in the task space using positions or velocities of the gripper. Their approach depends heavily on prior knowledge of the task to be accomplished. They have considered only one action, namely exchanging an Orbit Replaceable Unit, in their experiments.

Generative models such as HMMs fit a data to the model using a training set and optimize the parameters[67, 68, 69, 70, 71]. Using the learned model, novel sequences can be classified. By sampling from generative models, one can generate sequences and thus this type of approach is suitable for synthesis. Some issues with HMM approaches are parameter specification and optimization. The performance of the HMM model depends a great deal on the number of states and state transitions. Some authors find the best number of states by experimenting with the data. Some others have set the parameters using an information theoretic criteria such as minimum description length [72]. In [73], a heuristic approach to determine the topology of HMM is described. From an initial model with high number of states and full transitions, a final model is arrived by successively removing one state at each iteration. For a fixed HMM topology, the the parameters of an HMM can be optimized using the Baum-Welch re-estimation procedure [69] which is an instance of the Expectation-maximization algorithm [74]to find the Maximum Likelihood estimates with missing data. But a poor initialization of EM algorithm can result in convergence of the parameters at a local maximum.

2.2 Learning with Primitives/Hierarchical Methods.

Several authors have represented actions in a hierarchical manner [45, 75, 76]. Most works require the manual modeling of atomic movements/primitives. The contribution of our work is that we perform this segmentation automatically. Expressing actions as a combination of several smaller meaningful parts takes its inspiration from neuro-biological evidences that perceiving an action and executing the same action are connected in the brain [46, 47, 48, 49, 50, 51]. Finding a basis set of action primitives is being considered as an important factor in developing intelligent systems with cognitive capabilities. In [31], signs of first and second derivatives was used to segment data into primitives. To each segment, its state, time duration and angular displacement were recorded. Explicit time duration in the modeling of primitives implies that we have well time-aligned data. This is not true in most cases due to variability in execution of a task by various subjects, measurement errors etc. In [36] primitives were found by thresholding angular velocities. In their work human movement data was collected using sensors attached to hand (4 sensors : center of the upper arm, center of lower arm, above writst, phalanx of the middle finger). The recorded 3D coordinates were then converted to joint angles corresponding to the 4 DOF of human arm. The four dimensional data of joint trajectories were segmented into several segments by thresholding the magnitude of angular velocity and the resulting segments for each of the joints were interpolated with 100 elements in order to apply principal component analysis. Elements of each joint were concatenated to form 400 dimensional vectors and PCA was applied to reduce the dimension to 11 by choosing the most significant 11 eigen vectors and projecting the input data. Reproduction of the original movement was performed by projecting points back to the input space from the latent space. To apply this method, strong assumptions must be made about the segmentation of the data, and the duration of the primitives. Velocity thresholding gets more complicated as the number of joints increases due to spurious zero velocity crossings. We provide a higher level abstraction of primitives using a stochastic context-free grammar which is not possible with the approach in [36]. Primitives found using joint angles are useful when the primitives are to be performed but are not portable across agents with different embodiments. This means that the primitives for the same action would be different in another robot with different degrees of freedom.

In [77] human motions are represented as a binary tree. Actions are recognized by finding the optimal node transitions in the tree. In their approach, initial observations are mapped to a set of virtual markers on a normalized virtual subject. Initially all the observations belong to the root node. Samples belonging to a node are segmented to two parts and are assigned to two leaf nodes. All the samples belong to a node are projected to its principal component. Then a threshold value is used to segment samples into two parts. Optimal threshold is selected using a minimum-error thresholding technique [78]. The process is repeated until a node with fewer than a predetermined number of frames is created. Length of all the branches are made to be the same by attaching a copy of the leaf node to the shorter branches. The final binary tree is used to recognize human motion, state estimation and prediction and robot motion planning. This approach requires time aligned sequences of same length to apply PCA. To extend this approach for sequential learning is not straight forward. Our states in the final HMM and the nodes in the last layer of [77] are comparable to some extent. The observations that belong to one state are neighbours in time in our case. In [77], frames belonging to one node are close in the 1D projected space. Takano and Nakamura [39] have also approached the problem of finding motion primitives using HMMs. They have modeled each actions via a discrete hidden Markov model. In their approach primitives are assumed to be known whereas our approach learns the primitives from the data.

In [18] subgoals are detected from trajectories by detecting regions that the agent visits frequently on successful trajectories but not on unsuccessful trajectories. A successful trajectory is defined as a path in which the agent succeeds to attain the end state ignoring the number of steps it took. This approach is tested in a grid world with an agent equipped with four primitive movements: *up, down, right, left*. Frequently visited states were identified using diverse density approach and the frequent states were selected as subgoals. Using these subgoals, the agent is able to reduce the number of it needs to reach the final goal. This approach is suitable for a discretized world of states. In our approach, the states are to be learned from continuous input signals. In [37] a dynamic systems frame work for primitive detection using trained models is presented. They assumes that the movement segments are known apriori. This approach is tested with only a specific scenario. Another work where primitives are defined by hand is [53]. They use primitives such as lines, circles, arcs in 2D.

In [79] several HMMs are used to model individual primitives. HMMs work in parallel on the input stream. Output is fed into a stochastic grammar which will remove uncertainties in the primitive detection. In this work each point in the input signal is converted to a primitive by taking maximum over all possible starting points. The point wise conversion of primitive is corrected by an SCFG which is specified by hand. In our case, we do not convert single points. We convert a sequence to a sequence of primitives. Uncertainties are handled by the model and the grammar over primitives.

Hayes and Demiris [80] taught a robot to traverse a maze by following a teacher. The robot closely followed a teacher and whenever there was a significant change in the movement direction of the teacher, the robot noted the environment and type of movement executed. At the end,the robot ends up with several symbolic rules. On left hand side(LHS) of such rules were the description of environment in which actions occurred and the right hand side(RHS) was the action executed(turn right, turn left or move straight). For example if the robot observes that the teacher rotated 90 degrees and then moved forward when there was a wall in the front and on the right, it will generate and store the rule Walls sensed: FRONT, RIGHT$\rightarrow$ Rotate 90 degrees and Move forward. When the robot is navigating the maze by itself it constantly matches the perceived environment with the LHS of the stored rules and reproduce the corresponding action on the RHS. Thus when it senses that there are walls in the front and on the right, it will Rotate 90 degrees and Move forward. The success of this approach is due to the limited number of simple actions involved.

2.3 Learning with Object Context

Many actions of interest involves manipulation of objects. Depending on the data used, works on primitive segmentation fall into three categories: a) actions involve objects. But the object state is not considered [2]. b) actions do not involve objects [36] c) only object information is considered [41]. The actions considered in [36] do not involve objects and are very simple. In this work, we do not require pre-segmented information [81, 2] to learn the primitives.

Kuniyoshi et al. [1] presented a method where the robot learns reusable task plans by observing a human performing simple assembly task in a tabletop environment. The movements of the human are classified as actions known to the robot such as *pick, move, place etc.* When the task is completed, the robot is able to reproduce the sequence of actions. Move-

Value	Speed	Direction
:Staying	$v < V_L$	—
:Moving Up	$V_L < V_H$	$A_U < a$
:Moving Down	$V_L < V_H$	$a < A_D$
:Moving Laterally	$V_L < V_H$	$A_D < a < A_U$
:Moving Fast	$V_H < v$	—

Table 2.1: Symbol conversion table used in [1]. V_L=Lower threshold for speed, V_H=Higher threshold for speed, A_D=Threshold angle for downward, A_U=Threshold angle for upward.

ments are characterized by motion features of the hand and the relative location of the hand with respect to the objects. For example moving down is specified as when the speed is between a lower and upper limit and the angle is less than the threshold for downward motion. In this case the robot is only required to recognize actions that are already known to it. Mapping from observed actions to the motor level is done using symbolic labels. Then conversion is done using a table as shown in Tab. Fig. 2.1. Once the robot recognizes the label, it is mapped to a pre-programmed robot action sequence.

In [82], different objects are manipulated and the order of manipulated objects are used to predict the next probable action. In their work, objects were labeled with IC tags. Each action sequence was immediately converted to a symbolic form using the object label(cup, teabag , pot etc), location of the object(cupboard,cabinet, medicine box) and the human action(taken out, stored). For example, taking out spoon from cupboard was recorded as spoon-a0. Prefix span algorithm [82] was then used to extract multiple frequency patterns. Here the transformation of data from the raw signal to symbols is treated as a simple problem. Such a simple conversion approach is suitable in most real world application scenarios. We use a probabilistic approach to convert the raw signal to symbols.

In [61] a graphical model is used to model actions involving objects. They considered actions such as drinking, spraying, Answering phone, Making a call, Pouring and Lighting. They segmented sequences into reaching part and manipulation part and each part was modeled separately. Their approach involves time duration in modeling primitives. Moore et. al [83] used different layers in recognizing actions involving objects. When an object is contacted by a person, actions are compared to the pre-trained actions associated with that object. These actions are modeled by HMM. These methods require labeled data for training individual action models. Our learning approach does not require labeled data.

A way of learning constrains in the object space(task space) using supervised clustering techniques is presented in [8]. Key-frames are detected using zero-velocity crossing . Tasks are represented as a sequence of one or more elementary actions: reach-and-grasp(obj), transfer-and-release(obj). Using this representation assembling different workspace objects was learned. What is interesting and challenging is learning the elementary actions without human assistance.

2.4 Discussion

For primitive based action modeling the main task is in learning the primitives from data rather than specifying it manually. As we pointed out earlier, the manual specification of primitives are intuitive in most scenarios, but since they are different for different contexts, the primitives are constrained to one scenario. When different actions are modeled by individual models, the same parts will be represented multiple times. The common elements across different actions will not be detected. This is one cause of confusion among different actions. Another feature that one require for an intelligent system is a mechanism to increase its capabilities when needed. Most of the works mentioned above do not have a sequential learning possibility. Our work is aimed at contributing to these important aspects of learning. We detect common elements across different actions, learn the primitives from data and derive the grammar governing the primitives. Moreover our approach is sequential and more actions can be incorporated into the model without requiring the presence of past observed data.

Chapter 3

Mathematical Preliminaries

Contents

In this chapter we give brief outline some of the techniques we have used in the later chapters. This chapter is also intended to give the reader familiarity with some of the notations that is to follow later. In the chapters to follow, we will be using an HMM based approach to model data and learn primitives. Out approach is different from traditional HMM approaches where individual HMM parameters for different types of actions are optimized using training data. Baum-Welch algorithm is normally employed to optimize the parameters. In our approach, the parameters learned from data automatically and the need for Baum-Welch estimation is avoided.

3.1 Hidden Markov Models

An HMM is a stochastic finite state machine, consisting of a set of states and corresponding transitions between states. It is a popular method for modeling stochastic sequences with an underlying finite-state structure. An HMM is characterized by the following:

- N, the number of states in the model. The individual states are denoted by $S_1, S_2, ...S_n$ and the state at time t as q_t.
- M, the number of distinct observation symbols per state.

- The state transition probability distribution $A = (a_{ij})$ where

$$a_{ij} = p(q_{t+1} = S_j | q_t = S_i), \quad 1 \leq i, j \leq N \ .$$

- The observation symbol probability distribution in state j, $B = b_j(k)$ where

$$b_j(k) = p(v_k \text{ at } t | q_t = S_j), \quad 1 \leq j \leq N, 1 \leq k \leq M \ .$$

- The initial state distribution $\pi = \{\pi_j\}$ where

$$\pi_j = p(q_1 = S_j), \quad 1 \leq i \leq N \ .$$

The compact notation $\lambda = (A, B, \pi)$ is used to denote an HMM. The probability density matrices A, B, and π are determined by training. The most commonly used method is the Baum-Welch method[69], which is an iterative process that finds the local maximum given some starting values of A, B, and π.

3.2 Model Merging of Discrete HMMs

`Best-first model merging´ is a general technique for dynamically choosing the structure of a neural or related architecture while avoiding over-fitting [84]. It is applicable to both learning and recognition tasks and often generalizes significantly better than fixed structures. An application of model merging in the probabilistic modeling of language could be found in [85]. An initial HMM that directly encodes the training data is built at first. Successively more general models are produced by merging HMM states. A Bayesian posterior probability criterion is used to determine which states to merge and when to stop generalizing.
The idea is to find the model M that maximizes the posterior probability $P(M|X)$. From Bayes' law we have

$$P(M|X) = \frac{P(M)P(X|M)}{P(X)} \ . \tag{3.1}$$

Since the data X is fixed, M_{MAP} maximizes $P(M)P(X|M)$. This could be seen as a generalization of Maximum-likelihood estimation method, as

the prior $P(M)$ is combined with the usual likelihood term $P(X|M)$.
The global prior for a model M is given by

$$P(M) = P(M_G) \prod_q \underbrace{P(M_S^{(q)}|M_G)}_{\text{structure prior}} \underbrace{P(\theta_M^{(q)}|M_G, M_S^{(q)})}_{\text{parameter prior}} \quad . \tag{3.2}$$

Here the product is calculated over the states q of the HMM. $P(M_G)$ is a prior for global aspects of the model structure. The global factor $P(M_G)$ is assumed to be unbiased, i.e we do not prefer any particular model, and therefore ignored in the maximization. It is the parameter prior and structure prior of the model that determines which model to select.
The parameters of a state q with n_t^q transitions and n_e^q emissions contribute a factor,

$$P(\theta_M^{(q)}|M_G, M_S^{(q)}) = \frac{1}{B(\alpha_t, ..., \alpha_t)} \prod_i \theta_{q^i}^{\alpha_t - 1} \frac{1}{B(\alpha_e, ..., \alpha_e)} \prod_j \theta_{q^j}^{\alpha_e - 1} \quad . \tag{3.3}$$

Here α_t and α_e are the prior weights for transitions and emissions, θ_{q^i} is the transition probabilities from state q to state i, θ_{q^j} is the probability of observing j th symbol at state q.
The structural prior for a state q is computed as,

$$P(M_S^{(q)}|M_G) = p_t^{n_t}(1 - p_t)^{|Q| - n_t^Q} pe^{n_e^q}(1 - p_t)^{|\sum| - n_e^q} \quad . \tag{3.4}$$

Here $|Q|$ is the total number of states and $|\sum|$ is the total number of observation symbols and $B(.)$ is the beta function.
Merging is done until the model posterior is less than the previous step. An example for merging is shown in figure 3.1. The process starts from an HMM that models the two observed sequences *ab* and *abab*. Then different states are merged together to end up with the final model. We extend the merging concept to continuous HMMs in this work.

3.3 Model Merging of HMMs with Continuous Densities

In Sec. 3.2 we saw that two states were merged if that increased the likelihood of the model. This approach is not directly applicable in the continuous

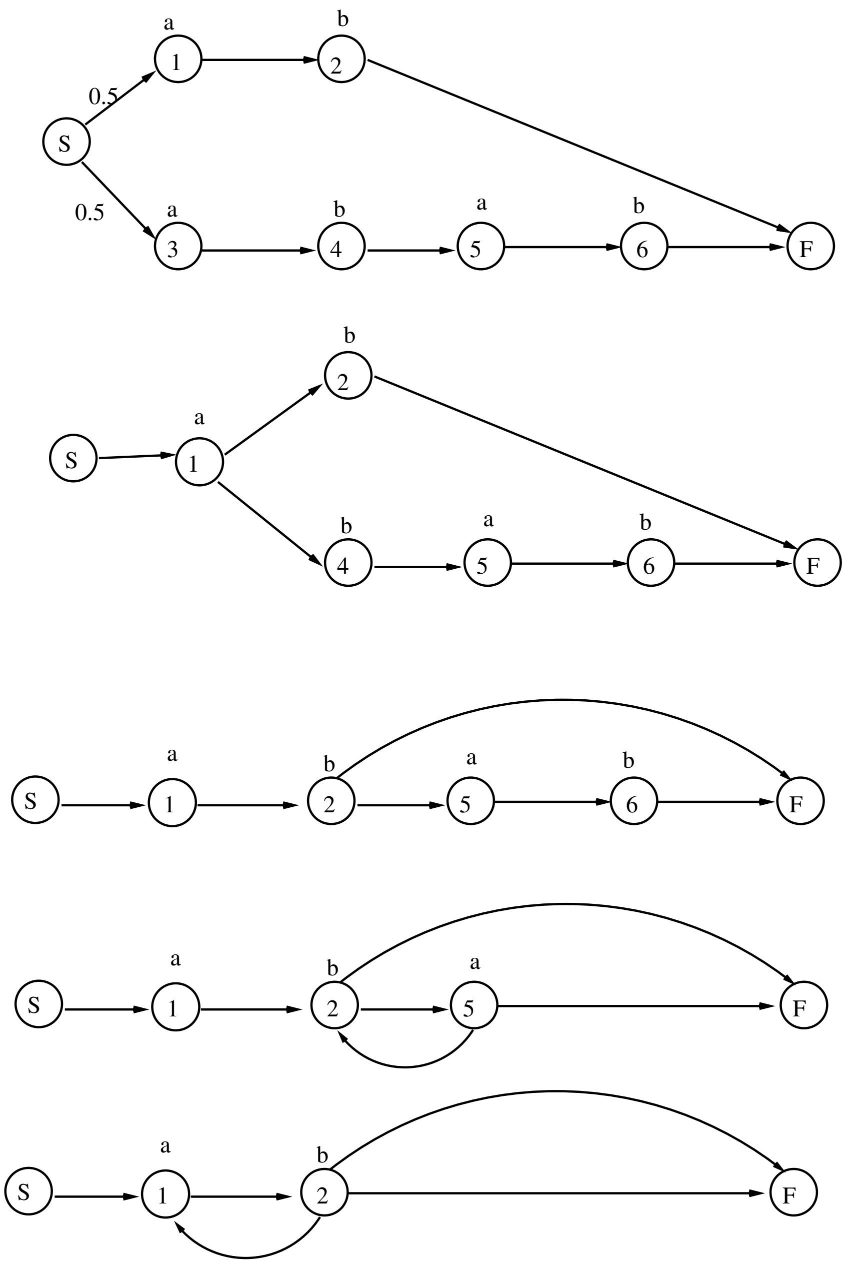


Figure 3.1: Model merging in discrete HMM.

scenario. We propose a different criteria for merging states in a continuous HMM. We will be using this approach to arrive at our HMM model for the data used in the later chapters. The structure of the HMM will be inferred from data and this is one of the main difference in our approach from traditional approaches where a structure is specified before hand.

Let λ_A and λ_C be to HMMs that are to be merged together. Let us denote the merged HMM by λ_M. Initially λ_M is a copy of λ_A. Then more states are added to it or existing states are modified after comparing the states of λ_M and λ_C. Let $S_c = \{s_1, s_2, \ldots, s_c\}$ and $S_M = \{s'_1, s'_2, \ldots, s'_M\}$ be the set of states of λ_C and λ_M, respectively. Then, the state set of the modified λ_M will be $S_M \cup D_1$ where $D_1 \subseteq S_c$. Each of the states s_i in λ_C affects λ_M in one of the following ways:

1. If $d(s_i, s'_j) < \theta$, for some $s_{\in} S_c$ and some $s'_j \in S_M$, then s_i and s'_j will be merged into a single state. Here, d is a distance measure and θ is a threshold value. We have used two different methods to modify the output probability distribution of the state s'_j. In Ch. 4 the output probability distribution associated with state s'_j in λ_M is modified to be a combination of the existing distribution and $b_{cs_i}(x)$. Thus $b_{Ms'_j}(x)$ is a mixture of Gaussians. In Ch. 5 the output probability distribution associated with state s'_j in λ_M is calculated using Eq. 5.3. All transitions to state s_i in λ_c are redirected to state s'_j in λ_M, and all transitions from state s_i in λ_C will now be from state s'_j in λ_M.

2. If for a state $s_i \in S_c$, $d(s_i, s'_j) > \theta, \quad \forall s'_j \in S_M$, a new state is added to λ_M. Let s_i be the r^{th} state to be added from λ_C. Then, s_i will become the $(M+r)^{th}$ 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 $b_{Ms'_{M+r}}(x) = \mathcal{N}(x; \mu_{s_i}, \Sigma_{s_i})$. Initial and transition probabilities of λ_M are adjusted to accommodate this new state. The newly added state will keep its label n.

When two states are merged together to become a single state, transitions to and from these states are to be adjusted accordingly. Consider the scenario in Fig. 3.2 where two HMMs are depicted. Imagine that the distance between the states S_{12} and S_{22} is very small and they are to be merged together. After the merging, the state S_{22} is removed and all transitions to S_{22} are redirected to S_{12}. All transitions from S_{22} are adjusted to be from S_{12}.

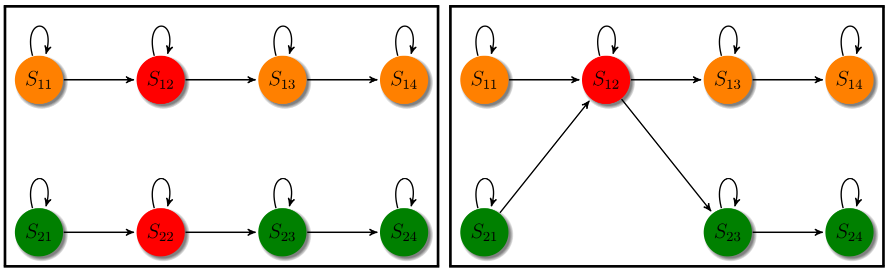


Figure 3.2: Model merging of continuous HMMs. The distance between the states S_{12} and S_{22} is very small and they are to be merged together. After the merging, the state S_{22} is removed and all transitions to S_{22} are redirected to S_{12}. All transitions from S_{22} are adjusted to be from S_{12}.

3.4 Stochastic Context Free Grammar

We use Stochastic Context Free Grammar for specifying the interaction between various primitives we find. A stochastic context free grammar(SCFG) M consists of

1. a set of nonterminal symbols N
2. a set of terminal symbols Σ
3. a start nonterminal $S \in N$
4. a set of productions or rules R
5. production probabilities $P(r)$ for all $r \in R$

The productions are of the form $X \to \lambda$ where $X \in N$ and $\lambda \in (N \cup \Sigma)^*$
A sentential form of M is a string ν of nonterminals and terminals, such that either $\nu = S$ or there is a sentential form μ from which ν can be produced by replacing one nonterminal according to a production of M, i.e., $\mu = \mu_1 X \mu_2, \nu = \mu_1 \lambda \mu_2$, and $X \to \lambda \in R$. A derivation in M is a sequence of sentential forms beginning with S each derived by a single rule application from its predecessor, such that at each step the replaced nonterminal X is always the left-most nonterminal in the sentential form. We write a derivation as $S \Rightarrow \nu_2 \Rightarrow \ldots \nu_k$. The probability of a derivation is defined by

1. $P(S)$=1
2. $P(S \Rightarrow \nu_2 \Rightarrow \ldots \nu_k) = P(S \Rightarrow \nu_2 \Rightarrow \ldots \nu_{k-1}) P(X \to \lambda)$

where $X \rightarrow \lambda$ is the production used in the step $\nu_{k-1} \Rightarrow \ldots \nu_k$.
The probability of a string x in M is

$$P(x|M) = \sum_{S \Rightarrow \cdots \Rightarrow x} P(S \Rightarrow \cdots \Rightarrow x)$$

Here the summation is over all the derivations that will end in x. The most likely derivation for a given string is the Viterbi parse. Viterbi parses are computed by dynamic programming. A chart is filled bottom-up, computing for each substring of a sentence and each nonterminal the partial parse with highest probability. Once the chart is completed, the maximum probability parse can be traced back from the root entry.

Pure statistical methods fail when the true characteristics of underlying pattern were structures(or shapes). Usual methods transform data into a feature space which might destroy the underlying structure of the data. We can employ syntactic techniques like SCFG by converting our data into symbols and construct grammar to learn the relation between symbols. Another scenario where SCFG becomes useful is illustrated in Sec. 5.4.

3.5 Kullback-Leibler Divergence

KullbackLeibler divergence is a measure of the difference between two probability distributions f and g. For continuous distributions it is defined as:

$$D_{KL}(f||g) = \int f(x) \log \frac{f(x)}{g(x)} \ dx \ . \tag{3.5}$$

K-L divergence is related to the Shannon entropy $H(f) = -\int f(x) \log f(x)$ and cross entropy $H(f, g) = \int f(x) \log \frac{1}{g(x)}$ as

$$D_{KL}(f||g) = H(f, g) - H(f) \ . \tag{3.6}$$

In this thesis we will be using the KullbackLeibler divergence between two multivariate normal distributions. Let f=$\mathcal{N}(x; \mu_1, \Sigma_1)$ and g=$\mathcal{N}(x; \mu_2, \Sigma_2)$ be two multivariate normal distributions. Then the K-L divergence from

$\mathcal{N}(x;\mu_1,\Sigma_1)$ to $\mathcal{N}(x;\mu_2,\Sigma_2)$ can be calculated as follows.

$$\begin{aligned} D_{KL}(\mathcal{N}(x;\mu_1,\Sigma_1)||\mathcal{N}(x;\mu_2,\Sigma_2)) &= \int \mathcal{N}(x;\mu_1,\Sigma_1)\log\frac{\mathcal{N}(x;\mu_1,\Sigma_1)}{\mathcal{N}(x;\mu_2,\Sigma_2)}\ dx \\ &= H(f,g) - H(f)\ \ . \end{aligned} \tag{3.7}$$

$$\begin{aligned} H(f,g) &= \int f(x)\log\left(\frac{(2\pi)^{d/2}|\Sigma_2|^{1/2}}{e^{-\frac{1}{2}(x-\mu_2)^t\Sigma_2^{-1}(x-\mu_2)}}\right)\ dx \\ &= (2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}\int f(x)(x-\mu_2)^t\Sigma_2^{-1}(x-\mu_2)\ dx \\ &= \log(2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}E((x-\mu_2)^t\Sigma_2^{-1}(x-\mu_2))\ dx \\ &= \log(2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}E(\mathrm{tr}[\Sigma_2^{-1}(x-\mu_2)(x-\mu_2)^t])\ dx \\ &= \log(2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}\mathrm{tr}[\Sigma_2^{-1}E((x-\mu_2)(x-\mu_2)^t)]) \\ &= \log(2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}\mathrm{tr}[\Sigma_2^{-1}E((x-\mu_1)+(\mu_1-\mu_2)) \\ &((x-\mu_1)+(\mu_1-\mu_2))^t)]) \\ &= \log(2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}\mathrm{tr}[\Sigma_2^{-1}E((x-\mu_1)(x-\mu_1)^t) \\ &+ \Sigma_2^{-1}E((x-\mu_1)(\mu_1-\mu_2)^t) + \Sigma_2^{-1}E((\mu_1-\mu_2)(x-\mu_1)^t) \\ &+ \Sigma_2^{-1}E((\mu_1-\mu_2)(\mu_1-\mu_2)^t)] \\ &= \log(2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}\mathrm{tr}[\Sigma_2^{-1}\Sigma_1 + 0 + 0 + \Sigma_2^{-1}(\mu_1-\mu_2)(\mu_1-\mu_2)^t] \\ &= \log(2\pi)^{d/2}|\Sigma_2|^{1/2} + \frac{1}{2}\mathrm{tr}[\Sigma_2^{-1}\Sigma_1] + (\mu_1-\mu_2)^t\Sigma_2^{-1}(\mu_1-\mu_2)\ \ . \end{aligned} \tag{3.8}$$

We have used a trick from linear algebra: $a^tAa = \mathrm{tr}(Aaa^t)$

$$\begin{aligned} H(f) &= -\int f(x)\log f(x)dx \\ &= \frac{1}{2}\log(2\pi)^d|\Sigma_1| + \frac{1}{2}\int f(x)(x-\mu_1)^t\Sigma_1^{-1}(x-\mu_1)\ dx \\ &= \frac{1}{2}\log(2\pi)^d|\Sigma_1| + E((x-\mu_1)^t\Sigma_1^{-1}(x-\mu_1))\ dx \\ &= \frac{1}{2}\log(2\pi)^d|\Sigma_1| + \frac{1}{2}E(\mathrm{tr}[\Sigma_1^{-1}(x-\mu_1)(x-\mu_1)^t]\ dx \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\log(2\pi)^d|\Sigma_1| + \frac{1}{2}\mathrm{tr}[\Sigma_1^{-1}E((x-\mu_1)(x-\mu_1)^t)] \\
&= \frac{1}{2}\log(2\pi)^d|\Sigma_1| + \frac{1}{2}\mathrm{tr}[\Sigma_1^{-1}\Sigma_1] \\
&= \frac{1}{2}\log(2\pi)^d|\Sigma_1| + \frac{1}{2}\mathrm{tr}[I] \\
&= \frac{1}{2}\log(2\pi)^d|\Sigma_1| + \frac{d}{2} \quad . \qquad (3.9)
\end{aligned}
$$

Using Eq. 3.8 and Eq. 3.9 in Eq. 3.7 we get:

$$
\begin{aligned}
D_{KL}(\mathcal{N}(x;\mu_1,\Sigma_1)||\mathcal{N}(x;\mu_2,\Sigma_2)) = \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1)\right) + \\
\frac{1}{2}\big((\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2) - d\big) \quad . \qquad (3.10)
\end{aligned}
$$

Chapter 4

Action Primitives

Contents

In Ch. 1, we have given motivations and advantages for the use of primitives in action representation especially in imitation scenario. Primitives can be thought of as building blocks for action representation similar to phonemes in human speech. Using primitives, one can form a unified frame work that is suitable for action representation, recognition, planning and synthesis [86]. In this chapter we desribe an automatic primitive learning method in the movement space from observed data. We consider a table top scenario where objects are being manipulated. Our approach is inspired from a previous work [2] where primitives were manually segmented and used for recognition. Our aim is to learn the primitives automatically from the data. We propose a sequential learning approach where the sequences are processed one by one and a model is learned incrementally. Input signals are then expressed as sequence of hidden states. Common contiguous segments are identified and segmented in the hidden state space. Using the segmented primitives, the grammar governing the primitives are derived automatically. The primitives extracted in this chapter depends on the location of the object and the location of the person performing the action. When there is a change in either of these two factors, the extracted primitives will change. A modified approach to overcome this limitation will be presented in Ch. 5. We start by giving a brief summary of the supervised approach [2].

4.1 Summary of the Supervised Approach

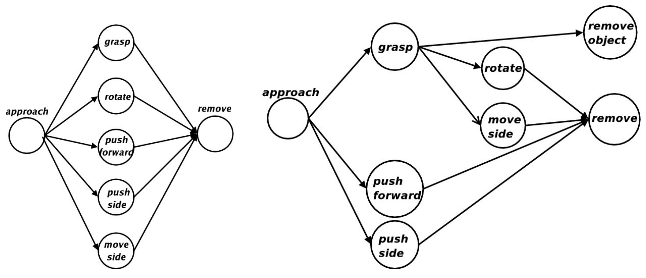


Figure 4.1: HMM models used in [2]. (Left): Structure of HMM model I with actions as primitives; (Right): Structure of HMM II with composite actions.

The work presented in [2] is a study of modeling and understanding of manipulation actions performed by humans in 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.

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 the chest sensor are 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

Primitives are manually extracted from the data and two different HMM structures are considered for modeling the actions which are shown in Fig. 4.1 and their results are compared. In the first model, Model *I* (Fig. 4.1, left), the primitives are *grasp*(g), *rotate*(r),*push forward*(ps), *push side*(ps), *move side*(m), *approach* and *remove*. In the second model, Model *II* (Fig. 4.1, right), 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* are presented in Tab.4.1 on the left. The entries on the diagonals show the rates of correctly recognized primitives. In this model individual primitives did not yield good results due to their high overlap.

In the HMM Model *II*, common parts are kept as separate primitives. This way different primitives become very dissimilar, and the intuitive selection of the primitives will give the primitives a semantic meaning as well, which is reflected by the use of verbs for describing the primitives. One new state, *remove object*, was introduced to show that this end state is different from other cases. Each person is holding the object at the end only for the *grasp* action.

Tab. 4.1 on the right presents the recognition results when using the Model *II* HMM. The numbers on the diagonal give again the rate of correctly classified primitives. The results of recognizing *grasp* and *move* have increased significantly.

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

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 4.1: Recognition results from [2] for primitives using different HMMs. (Left) Confusion matrix for the recognition rates using HMM model I. (Right) Confusion matrix for the recognition rates using HMM Model II. Rows represent predicted class and colums represent actual class. Correct results are given on the diagonal.

4.1.1 Discussion

The work of [2] should be seen as an extensive study on the modeling of the manipulation actions, taking into account common action in everyday settings having different meaning but being very similar to each other. The most important findings of the 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.

4.2 Automatic Segmentation of Primitives

From the discussions in Ch. 1 and Ch. 2, we can see that an efficient model for actions 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.In this chapter we describe our approach to finding primitives from observation sequences automatically. Sequences are processed one by one and are covered with Gaussian. Thus the Gaussians model the local characteristics of the data. The Gaussians will then become the hidden states of an HMM for that sequences. States for different sequences are merged together if they are found to be statistically near. Finally we arrive at a single HMM, that can model all the sequences that have been procesed. Using the final HMM, all sequences can be expressed as sequence of hidden states. By identifying common and uniques states in the sequences we extract the primitives. A grammar that defines the precedence rules for each of the primitives are then extracted automatically. The method is then tested with motion capture data and is compared with the results from a previous work on the same data. We conclude the chapter with a discussion on the advantages and disadvantages of our approach. These steps are explained in detail in the follwing sections of this Chapter.

Our approach is based on the idea of identifying common and unique parts in sequences. Identifying re-occcuring parts in sequences and re using them in modeling the data, we can reduce the complexity in representing the data. 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 modeling and recognizing actions.

We define two sets of primitives. One set contains parts that are unique to one *type* of action and another set 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 according to some predefined metrics. 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 grammatical rules that govern the order of the primitives in time. Keeping this in mind we state our objectives as:

1. Let $\mathcal{L} = \{X_1, X_2, \ldots, X_m\}$ be a set of data sequences where each X_i is of the form $x_1^i x_2^i \ldots, x_{T_i}^i$ and $x_j^i \in \mathbb{R}^n$. Let these observations be generated from a finite set of sources (or states) $\mathcal{S} = \{s_1, s_2, \ldots s_r\}$. Let $S_i = s_1^i s_2^i \ldots, s_{T_i}^i$ be the state sequence associated with X_i. Find a partition $\mathcal{S}'$ of the set of states $\mathcal{S}$ where $\mathcal{S}' = \mathcal{U} \cup \mathcal{V}$ such that $\mathcal{U} = \{a_1, a_2, \ldots, a_k\}$ and $\mathcal{V} = \{b_1, b_2, \ldots, b_l\}$ 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 $\mathcal{U}$ corresponds to common actions and the set $\mathcal{V}$ 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.

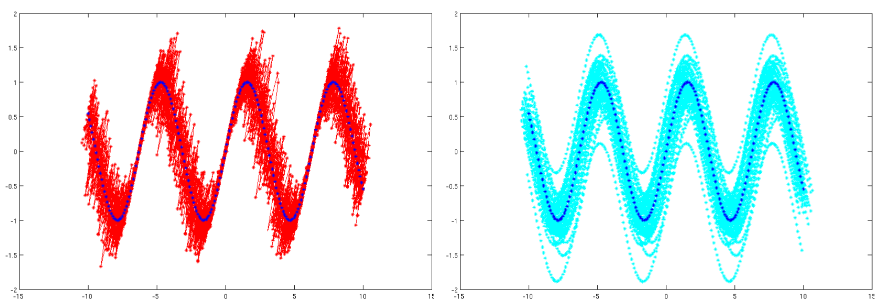


Figure 4.2: Left: Result of sampling from $\mathcal{N}(\mathbf{x_i}, \mathbf{\Sigma_l})$. Right: Result of sampling from $\mathcal{N}(x_{il}, \sigma_l)$. Blue curve shows the original curve. Sample sequences on the right figure are appropriate to model the variation in the original sequence.

4.2.1 Model Learning from Observation Sequences

Now we desribe our learning approach in more detail. An initial HMM is modified incrementally by adding more states to it or by modifying existing states. We start the process from the first observed sequence X_1.

4.2.1.1 Modeling the First Sequence

Let X_1 be the first sequence with data points $\mathbf{x}_1^1\mathbf{x}_2^1 \ldots \mathbf{x}_{T_1}^1$. Since we have just one data sequence to start with, we generate additional sequences as described below.

Each $\mathbf{x_i}$ in X_1 is of the form $\begin{pmatrix} x_{1i} & x_{2i} & \cdots & x_{di} \end{pmatrix}^t$. Since our model building approach is incremental, we want to incorporate some prior knowledge on expected variations on observed sequences each time a new sequence is processed. Let X_k be a repetition of the same motion X_1 . Even though points in X_k and X_1 are different we can expect that each point $\mathbf{y_j}$ in X_k will be close to some point $\mathbf{x_i}$ in P1 and $y_{jl} = x_{il} + \eta_l$ for some η_l. If we assume that σ_l is the expected variance for the l th dimension then y_{jl} can be considered as a sample from $\mathcal{N}(x_{il}, \sigma_l)$. By sampling enough data points from $\mathcal{N}(x_{il}, \sigma_l)$ for each i and l we will be able to make an estimation of the density of the data similar to X_1. On the other hand, if we assume that $\mathbf{y_j} = \mathbf{x_i} + \eta_\mathbf{i}$, and sample from $\mathcal{N}(\mathbf{x_i}, \mathbf{\Sigma_l})$ where $\mathbf{\Sigma_l}$ is a diagonal matrix with η_is in the main diagonal, we will be getting a poor approximation to the true distribution. The difference is illustrated in Fig. 4.2. Approximating the original distribution by sampling from $\mathcal{N}(\mathbf{x_i}, \mathbf{\Sigma_l})$ will require correct information about the variance of each of the dimensions of the data. But if we approximate the original distribution by sampling from $\mathcal{N}(x_{il}, \sigma_l)$, we only need a an approximate guess of the variance of the data. The generated samples gives the effect of observing additional sequences of the same type that are aligned along the time axis. Therefore, for each sequence X_i, additional sequences are generated by sampling from $\mathcal{N}(x_{il}, \sigma_l)$. Then these sequences are segmented into approximately equal parts and each part is modeled by a Gaussian. Each of these Gaussians correspond to a hidden state from which the sequence was generated. Points belonging to each part is considered to be a state and is modeled by a multivariate Gaussian. After this step, we can consider the sequence X_i to be a sequence of states. In estimating the parameters of the Gaussians, we do not require the EM algorithm since the points from each segment are associated to a particular Gaussian. Hence we use the maximum likelihood estimates of the samples of each segment for the parameters of the Gaussian distribution modeling that segment. Let $(\mu_i^1, \Sigma_i^1),\ \ i = 1, 2, ...k_1$ be the estimates so that we have an ordered coverage of the data points. The value of k_1 is such that $\mathcal{N}(x; \mu_i^1, \Sigma_i^1),\ \ i = 1, 2, ...k_1$ 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-like model $\lambda_1 = (A_1, B_1, \pi_1)$ with k_1 states where k_1 is the number of Gaussians needed to cover X_1. We let A_1 to be a left-right transition matrix and $B_{1_j}(x) = \mathcal{N}(x; \mu_j^1, \Sigma_j^1)$. All the states at this stage receive 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.

4.2.2 Modeling the Rest of the Data

We have created the model $\lambda_1 = (A1, B1, \pi 1)$ that will generate the first type of data that we have input. 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. 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. 4.2.1.1. 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. 4.2.3. We increase the number of types of data sequences by one at this stage. All the states in X_c will be labeled n.

The reader might wonder at this stage what happens if a new data sequence with a large overlap with one of the types that we have processed earlier, and which differs significantly only in a rather small area, e.g., a significant difference in the beginning or in the end.

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.

4.2.3 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 the current model λ_M so that $P(X_k|\lambda_M)$ is high if $P(X_k|\lambda_c)$ is high. We do this by either adding states to λ_M from λ_c or by merging states of λ_M with states of λ_c. Let $S_c = \{s_1, s_2, \ldots, s_c\}$ and $S_M = \{s'_1, s'_2, \ldots, s'_M\}$ be the set of states of λ_c and λ_M, respectively. Then, the state set of the modified λ_M will be $S_M \cup D_1$ where $D_1 \subseteq S_c$. Each of the states s_i in λ_c affects λ_M in one of the following ways:

1. If $d(s_i, s'_j) < \theta$, for some $s_{\in} S_c$ and some $s'_j \in S_M$, then s_i and s'_j 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 state s'_j in λ_M is modified to be a combination of the existing distribution and $b_{cs_i}(x)$. Thus $b_{Ms'_j}(x)$ is a mixture of Gaussians. We append n to the label of the state s'_j in λ_M. All transitions to state s_i in λ_c are redirected to state s'_j in λ_M, and all transitions from state s_i in λ_c will now be from state s'_j in λ_M.

2. If for a state $s_i \in S_c$, $d(s_i, s'_j) > \theta, \quad \forall s'_j \in S_M$, a new state is added to λ_M. Let s_i be the r^{th} state to be added from λ_c. Then, s_i will become the $(M+r)^{th}$ 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 $b_{Ms'_{M+r}}(x) = \mathcal{N}(x; \mu_{s_i}, \Sigma_{s_i})$. 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 two states s_i, s_j. The K-L divergence from $\mathcal{N}(x; \mu_{s_i}, \Sigma_{s_i})$ to $\mathcal{N}(x; \mu_{s_j}, \Sigma_{s_j})$ has a closed form solution given by :

$$\begin{aligned} D_{KL}(\mathcal{N}(x; \mu_{s_i}, \Sigma_{s_i}) || \mathcal{N}(x; \mu_{s_j}, \Sigma_{s_j})) = \frac{1}{2}\left(\log \frac{|\Sigma_{s_j}|}{|\Sigma_{s_i}|} + \mathrm{tr}(\Sigma_{s_j}^{-1}\Sigma_{s_i})\right) + \\ \frac{1}{2}\left((\mu_{s_j} - \mu_{s_i})^T \Sigma_{s_j}^{-1} (\mu_{s_j} - \mu_{s_i}) - n\right) \end{aligned} \tag{4.1}$$

Here, n is the dimension of the space spanned by the random variable x. We use a symmetric version of the above distance given by

$$\frac{D_{KL}(\mathcal{N}(x; \mu_{s_i}, \Sigma_{s_i}) || \mathcal{N}(x; \mu_{s_j}, \Sigma_{s_j})) + D_{KL}(\mathcal{N}(x; \mu_{s_j}, \Sigma_{s_j})) || \mathcal{N}(x; \mu_{s_i}, \Sigma_{s_i})}{2}. \tag{4.2}$$

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 detect 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 (4.1) to compute the similarity between two states. We can observe that (4.1) 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. This can be justified because our aim is to find out if a new state is to be embedded into another state or not.

4.2.4 Finding Primitives

Primitive searching starts when we have processed all the available data sequences. Now using the Viterbi algorithm on the final merged model λ_M, the hidden states associated with each of the sequences are generated. Let $T_1, T_2, \ldots T_r$ be different Viterbi paths at this stage. The problem of finding the contiguous state sequences that are common across different T_i is similar to finding the longest common substring(LCS) problem [87]. The pseudocode for finding the longest common substring of two strings is given in Fig. 4.3. We take all paths with non-empty intersection and find the largest common substring a_k for them. Then, a_k is added to $\mathcal{U}$ and is replaced with a new empty string symbol ε in all the occurrences of a_k in $T_i,\ \ i = 1, 2, \ldots r$. We continue to look for largest common substrings until we get the symbols ε as the only common substring for any two paths. Thus, we end up with new paths $T'_1, T'_2, \ldots T'_r$ where each T'_i consists of one or more segments with ε as the separator. These remaining segments in each T'_i are unique to T_i. Each of them are also primitives and form the members of the set $\mathcal{V}$. Our objective was to find these two sets $\mathcal{U}$ and $\mathcal{V}$ as was stated in Sec. 4.2.

In Fig. 4.3, we have given the dynamic programming approach for finding longest common substring of two strings and it runs in $O(T_i * T_j)$ where T_i and T_j are the lengths of the sequences. We also note that using generalized suffix tree approach the computational complexity can be reduced to $O(T_i + T_j)$ time [87]. Hence primitive finding is solvable in linear time.

4.2.5 Generating the Grammar for Primitives

Let $\mathcal{S}' = \{p_1, p_2, \ldots p_p\}$ be the set of primitives available to us. We wish to generate rules of the form $P(p_i \rightarrow p_j)$ which will give the likelihood of occurrence of the primitive p_j followed by primitive p_i. We do this by constructing a directed graph G which encodes the relations between the primitives. Using G we will be able to derive a stochastic context-free grammar with the set $\mathcal{S}'$ as the alphabet.

Let $G = (\mathcal{S}', E)$ be the primitive graph where two nodes in $\mathcal{S}'$ are connected if they appear together in some sequence i.e, $e_{ij} = (p_i, p_j) \in E$ if

Algorithm 4.2.1: Pseudocode for finding the longest common substring of two strings.

```
Input: String S, T
Output: Longest Common Substring $ret$
m=length of string S;
n=length of string T;
L= array($0 \cdots m, 0 \cdots n$);
z= 0 ;
ret= {};
foreach  $i = 1 \cdots m$ do
    foreach $j = 1 \cdots n$ do
        if $S[i] = T[j]$ then
            if $i = 1$ or $j = 1$ then
                L$[i, j] = 1$
            else
                $L[i, j] = L[i - 1, j - 1] + 1$
            end
        end
        if $L[i, j] > z$ then
            $z = L[i, j]$;
            $ret = \{\}$
        end
        if $L[i, j] = z$ then
            $ret = ret \cup S[i - z + 1 \cdots i]$
        end
    end
end
```

Figure 4.3: Pseudocode for finding the longest common substring of two strings.

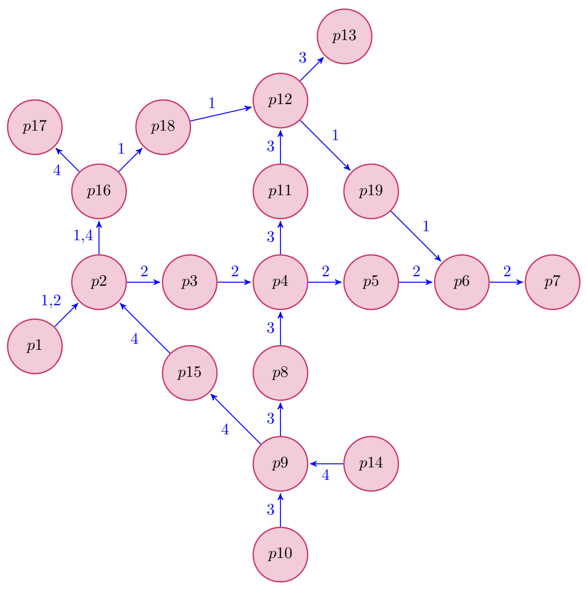


Figure 4.4: Directed graph for finding the grammar. This is the primitive graph for the data described in Sec. 4.2.8

there is a primitive path $P_k = \cdots p_i p_j \cdots$ for some k. Each primitive sequence will be a path in this primitive graph. Each primitive is then labeled with the class labels it belongs to. Let n be the number of movement classes that we have processed. Then each of the primitives will have labels from a subset of $\{1, 2, \cdots, n\}$, see Fig. 4.4. By way of definition each of the states that belong to a primitive p_i will have the same label set l^{p_i}. Let l^{p_i} be the label of node(primitive) p_i. Each of the edges $e_{ij} = (p_i, p_j)$ receives a label $l^{e_{ij}}$ which is the intersection of the labels of p_i and p_j. Thus $l^{e_{ij}} = l^{p_i} \cap l^{p_j}$. We wish to generate rules of the form $P(p_i \rightarrow p_j)$ which will give the likelihood of occurrence of the primitive p_j followed by primitive p_i. Using G we will derive a formal grammar for the elements in $\mathcal{S}'$. We have In Fig. 4.4 we give the directed graph constructed for our test data described in Sec. 4.2.8.

We proceed to derive a precise Stochastic Context Free Grammar (SCFG) from the directed graph G we have constructed. The set of primitives $\mathcal{S}'$ is the set of terminals. Let n_1, n_2 be the unique number of vertex and edge labels assigned. Then generate the nonterminals $B_1, B_2, \cdots B_{n_1}$ and $C_1, C_2, \cdots C_{n_2}$. Then there is a unique mapping from the set of edge and vertex labels to the set of nonterminals. Let $\mathcal{N} = S \cup \{B_i, C_j, i = 1, \cdots, n_1, j = 1, \cdots, n_2\}$ be the set of all non-terminals where S is the start symbol. For each primitive p_i that occurs at the start of a sequence and connecting to p_j define the rule

$$S \longrightarrow p_i B_m C_n \ . \tag{4.3}$$

Here B_m is the non-terminal associated with vertex p_i and C_n is the non-terminal associated with the edge (p_i, p_j). To each of the internal nodes p_j with an incoming edge e_{ij} connecting from p_i and an outgoing edge e_{jk} connecting to p_k define the rule

$$B_m C_n \longrightarrow p_j B_o C_p \ . \tag{4.4}$$

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

$$B_m C_n \longrightarrow p_i \ . \tag{4.5}$$

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

$$P(B_m C_n \longrightarrow p_j B_o C_p) = \frac{1}{|c_i^{(o)}|} \text{ if } |c_i^{(o)}| > 0 \ . \tag{4.6}$$

$$P(B_m C_n \longrightarrow p_i) = 1 \text{if } |c_i^{(o)}| = 0 \ . \tag{4.7}$$

where $|c_i^{(o)}|$ represents the number of outgoing edges from p_i. 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 $(\mathcal{N}, \mathcal{S}', S, \mathcal{R}, P(.))$ is the stochastic grammar that models our primitives. We have given the LHS of the rules using two non-terminals for better readability and understanding but they should be considered as a single non-terminal.

One might wonder why the HMM λ_F is not enough to describe the grammatical structure of the observations and why the SCFG is necessary. We illustrate the need for SCFG with an example. Consider three action classes with the following primitive sequences:

Class1: P1, P2, P5
Class2: P1, P2, P3
Class3: P1, P4, P5, P6

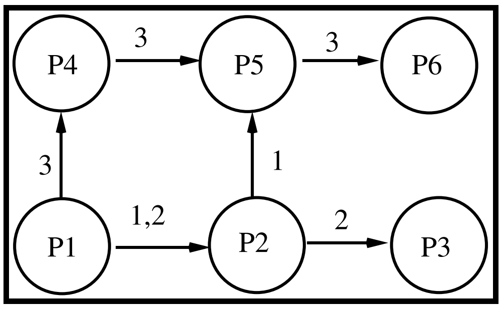


Figure 4.5: A primitive graph with three action classes. The three sequences { P1, P2, P3}, { P1, P2, P5, P6} and { P1, P4, P5, P6} belongs to Classes 1, 2 and 3 respectively. HMM approach is not sufficient to detect {P1, P2, P5, P6} to be an invalid sequence.

The primitive graph corresponding to these three classes is shown in Fig. 4.5. Now consider the primitive sequence P1, P2, P5, P6. This sequence does not belong to any of the given classes. But the sequence is valid according to the HMM structure and the sequence will have a high likelihood according to the HMM model. But if we apply the grammar rules learned from the primitive graph, the new sequence will be classified as an invalid sequence. For a theoretical treatment on parsing and advantages of context free grammar over regular grammar, we recommend [88] and [89].

4.2.6 Experimental Evaluation

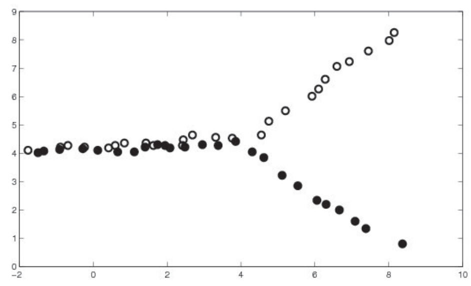
We have run four experiments: In the first experiment we have used a synthetic data set with two types of sequences. The second experiment is motivated by the surveillance scenario of Stauffer and Grimson [75] and shows a complex set of paths as found outside our building. The third experiment is motivated by the work of Vincente and Kragic [2] on the recognition of human arm movements. In the fourth experiment we learn the movement primitives for a chess game.

4.2.7 Testing on Simulated Data

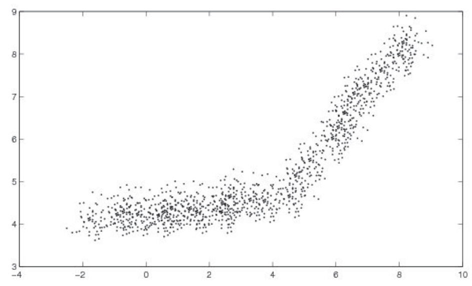
We illustrate the result of testing our method on a set of two sequences generated with mouse clicks. We have selected 2 simple sequences to illustrate the whole process. The two sequences share the initial portion as shown in Fig. 4.6(a). There is an intuitive segmentation with 3 parts for these two sequences: one segment containing the shared part and two separate segments for the unique parts. Our method extracts exactly these segments. The whole process is illustrated in Fig. 4.6. Sequence 1 with additional sequences generated by noise addition is shown in Fig. 4.6(b). Fig. 4.6(c) shows the result of covering these sequences with Gaussians. Covering of sequence 2 along with the first one is shown in Fig. 4.6(d). The sequences require 8 and 7 states respectively for covering. Hence the resulting individual HMMs will have 8 and 7 states respectively, see Fig. 4.6(e). Then the distances between the states of HMM1 and HMM2 are computed (Fig. 4.6(f)). Rows represent states of sequence 2 and columns represent states of sequence 1. One can notice the low values for the first four elements in the diagonal. Thus we have 4 pairs to merge: S_{21} with S_{11}, S_{22} with S_{12}, S_{23} with S_{13} and S_{24} with S_{14}. Merging is performed sequentially as shown in Fig. 4.6(g)-Fig. 4.6(j). When the model merging took place, the overlapping states were merged into one. The final HMM structure is shown in Fig. 4.6(j). The state sequences for the observed sequences are shown in Fig. 4.6(k) (Multiple occurrences are removed). Primitive segmentation will give us three primitives $p1, p2$ and $p3$, and is shown in Fig. 4.6(l) . Using the primitives, we can write the two sequences as primitive sequences: $(p1, p2)$ and $(p1, p3)$. Primitive tree in Fig. 4.6(m) shows the structure of primitives observed in the data. Using the primitive tree, a Stochastic Context Free Grammar is extracted by using the method illustrated in Sec. 4.2.5 and is shown in Fig. 4.6(n). The numbers in the brackets represent the probability of choosing the corresponding derivation. Finally the segmentation of original data using primitives is shown in Fig. 4.6(o).

4.2.8 2D-Trajectory Data

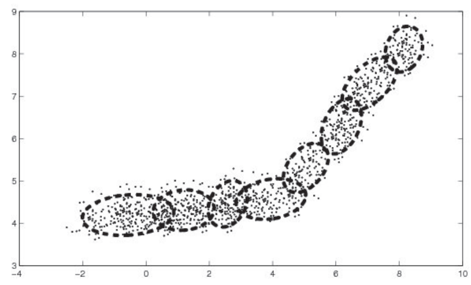
The second experiment was done on a surveillance-type data inspired by [75]. The paths represent typical walking paths outside of our building. In this data there are four different types of trajectories with heavy overlap, see Fig. 4.6(left). We can also observe that the data is quite noisy. Fig. 4.6(right) shows the result of covering with Gaussians. The result of primitive segmentation is shown in Fig. 4.7. Different primitives are colored differently and we have named the primitives with different letters. The detected common primitives are the junctions where different trajectories intersect. As one



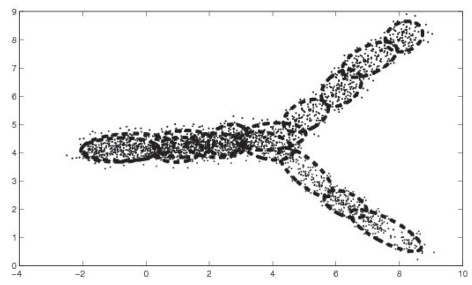
(a) Original Data



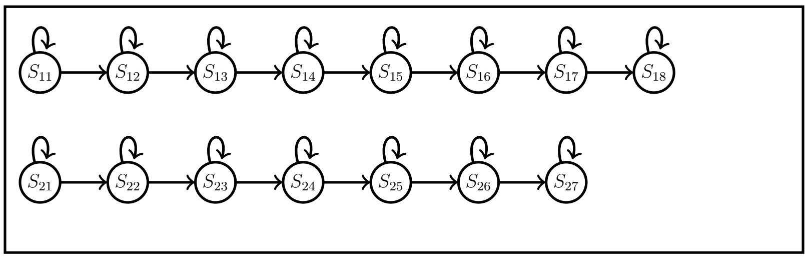
(b) Data with additional sequences



(c) Data Covered with Gaussians



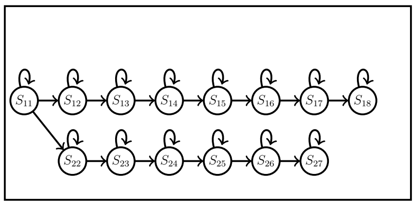
(d) Covering of sequence 2



(e) HMMs for the data

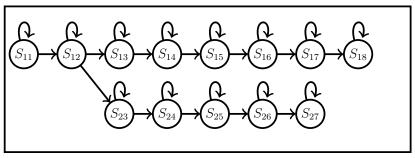
0.17	6.2	34.0	29.0	85.0	155.0	155.0	322.0
5.5	0.47	9.5	20.0	56.0	111.0	133.0	288.0
11.0	2.0	1.0	5.9	28.0	72.0	100.0	222.0
34.0	15.0	6.1	1.2	13.0	46.0	88.0	188.0
88.0	47.0	32.0	11.0	24.0	60.0	122.0	233.0
133.0	93.0	91.0	52.0	77.0	111.0	200.0	300.0
122.0	111.0	133.0	77.0	122.0	155.0	255.0	344.0

(f) Distance between states. Rows and columns represent states of HMM2 and HMM1 respectively

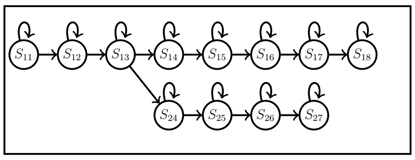


(g) Merging step 1. S_{21} is merged with S_{11}.

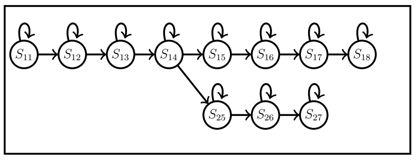
Figure 4.6: Illustration of the complete process with simulated data. Data was generated with mouse clicks.



(h) Merging step 2. S_{22} is merged with S_{12}.



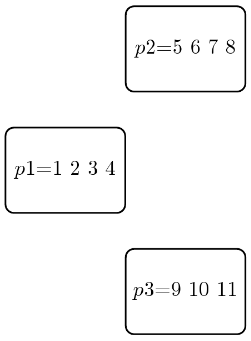
(i) Merging step 3. S_{23} is merged with S_{13}



(j) Merging step 4. S_{24} is merged with S_{14}

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 2 & 3 & 4 & 9 & 10 & 11 \end{pmatrix}$$

(k) State sequences of sequences



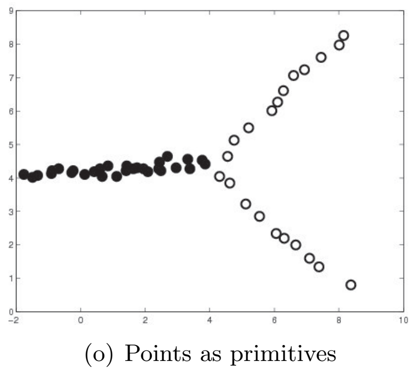
(l) Extracted primitives

$$\begin{tikzcd}
 & p2 \\
p1 \arrow[ur] \arrow[dr] & \\
 & p3
\end{tikzcd}$$

(m) Primitive graph

$$\begin{aligned}
S &\rightarrow P1A_2^{12} && (0.5)\\
&\rightarrow P1A_3^{12} && (0.5)\\
A_2^{12} &\rightarrow P2 && (1.0)\\
A_3^{12} &\rightarrow P3 && (1.0)\\
P1 &\rightarrow p1 && (1.0)\\
P2 &\rightarrow p2 && (1.0)\\
P3 &\rightarrow p3 && (1.0)
\end{aligned}$$

(n) Extracted grammar



(o) Points as primitives

Figure 4.6: Illustration of the complete process with simulated data. Data was generated with mouse clicks.

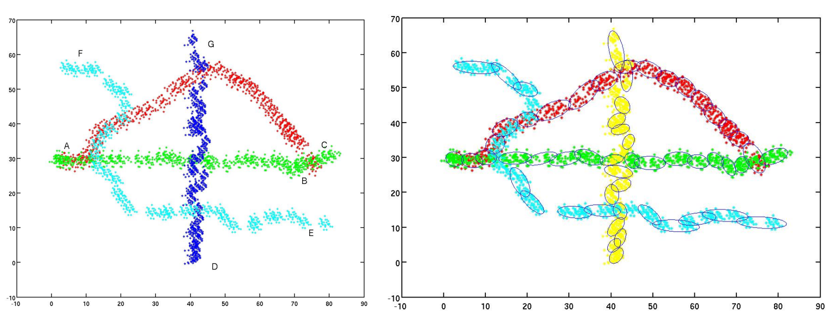


Figure 4.6: (Left)Trajectories from tracking data. Each type is colored differently. Values along the axes represent pixels. Only a part of the whole data is shown. (Right)The results of the covering procedure with the Gaussian mixtures. The numbers shown are the state numbers in the final model. Merged states are not shown. Hence some data points might appear to be unassigned.

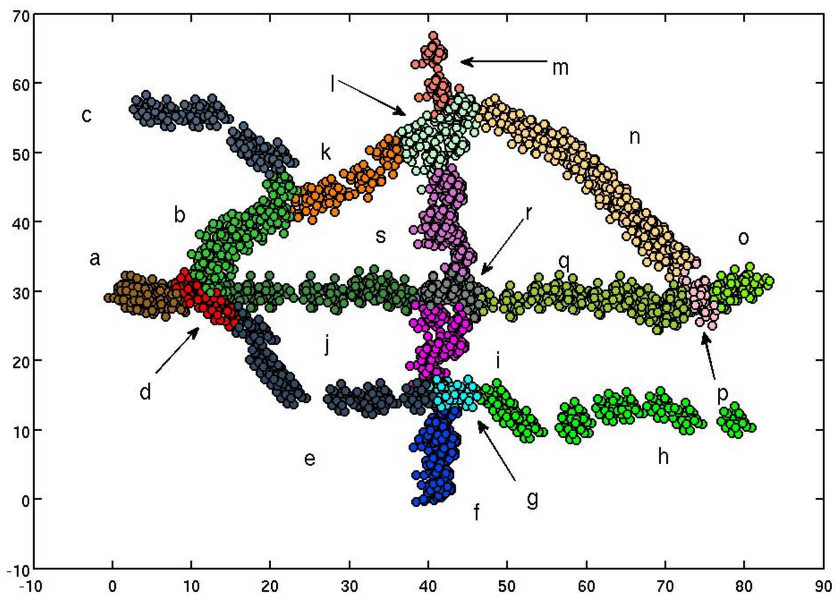


Figure 4.7: This figure shows the detected primitives. Each primitive is denoted by a letter.

can see, our approach results in primitives that are intuitive. 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 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.

4.2.9 Hand Gesture Data

We have tested our approach on the dataset described in Sec. 4.1 without annotation. Thus we use only the trajectory information for the sensors attached to the hand. The input to our system is the raw data from the sensors. We do not use the transformation of data described in Sec. 4.1. The original data is available online [90]. We expect to extract a set of primitives so that each of these sequences can be expressed as a combination of these primitives. In the following experiments, we have consider a subset of the data where each of the subjects perform actions from a fixed position to a fixed position on the table. 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. The result of applying our primitive segmentation for other positions of the subjects and the object will result in additional primitives that are disjoint from the subset we consider below.

By applying the techniques described in Sec. 4.2 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. 4.10. One can compare this with Fig. 4.1 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. 4.8. The ground truth was obtained by looking at each sequences and manually segmenting them. This particular sequence had 119 points in it. In the ground truth, *Reach* extends from t=1 to t=42, *Grasp* extends from t=43 to t=52 and *Retrive* extends from t=53 to t=119 . In our segmentation p_1 and p_2 combined extends from t=1 to t=44, p_3 extends from t=45 to t=61 and p_4 extends from t=62 to t=119. Thus we can infer from Fig. 4.10 and Fig. 4.8 that p_3 and p_2 together constitute the *approach* primitive, p_6 refers to the *grasp* primitive and p_6 corresponds to the *remove* primitive. Similar comparison could be made with other actions using the comparison diagram given in Fig. 4.9.

Using these primitives, a SCFG was learned as described in Sec. 4.2.5.

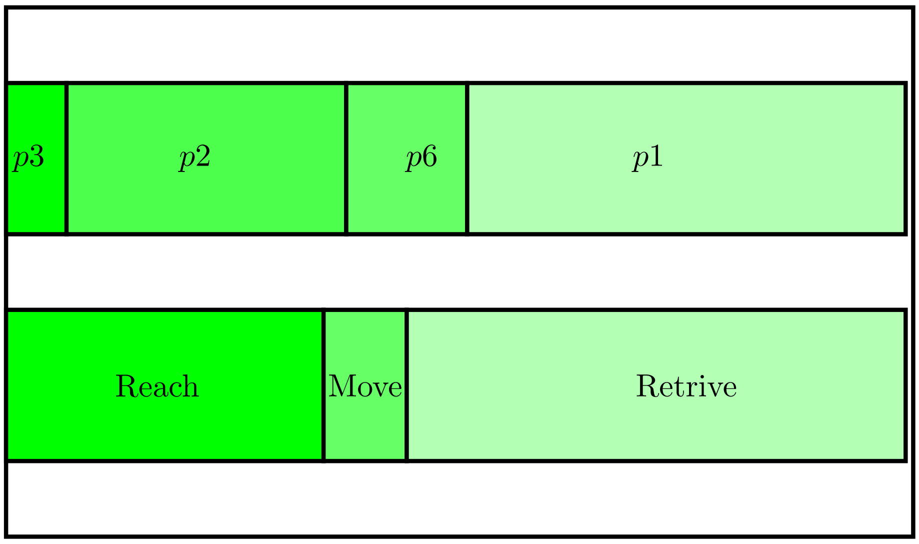


Figure 4.8: Comparing automatic segmentation with manually segmented primitives for one grasp sequence. The horizontal axis represents the length of the sequence. The plot compares a grasp sequence with length 119. Using the above diagram with Fig. 4.10, we can infer that p_3 and p_2 together constitute *approach* primitive, p_6 refers to *grasp* primitive and p_1 corresponds to *remove* primitive.

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. This grammar is used to test the validity of the primitive sequence for an unknown sequence. It can also be used to predict the future observation of a partially observed sequence.

Results of primitive segmentation for *push sideways, push forward, move,* and *grasp* actions are shown in the tables 4.2 and 4.3. The numbers given in the tables represent the primitive numbers shown in Fig. 4.10. The sequences that are identified correctly have a cyan background and the sequences that are not classified correctly have light gray background. We can see that all the correctly identified sequences start and end with the same primitive as expected. In Tab:4.3 on the right, Person 1 and Person 4 are marked with a dark 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.

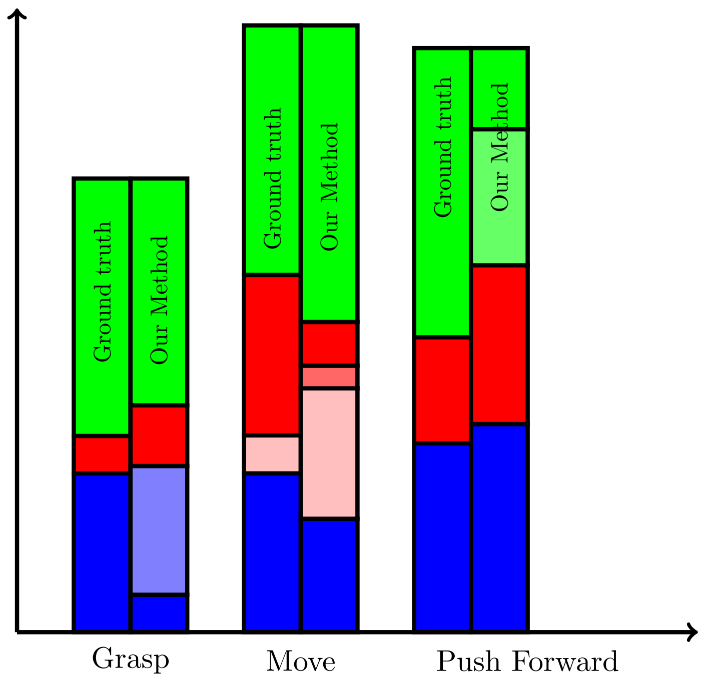


Figure 4.9: Comparing primitive segmentation with ground truth data. Average comparison results are shown. Height represents sequence length. Each of the segments in the bars represent a primitive.

4.2.10 Chess Movements Data

To further illustrate the application of our algorithm, we have tested our algorithm in a chess movements learning scenario. The aim of this experiment is to learn the different type of movements from the trajectory data. An object was placed on a chess board and was subjected to movements similar to that of chess pieces. We have recorded horizontal and vertical movements for the rook and queen at different lengths and the L-shape movements by the knight and diagonal movements for bishop. Some sample tracks are shown in Fig. 4.11. Only 8 out of the 12 knight moves were considered. In the collected dataset, we do not know how many types of movements are allowed, and what are the types of movements in there. Each of the recorded sequences were used learning the model. The extracted primitives and the resulting structure is shown in Fig. 4.12. In this figure p_1 represent moving

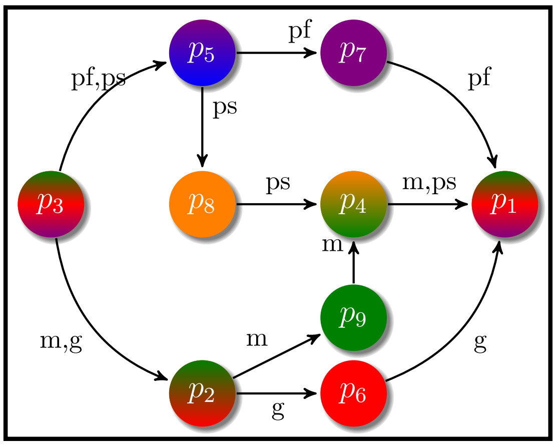


Figure 4.10: The temporal order for primitives of hand gesture data. Node number corresponds to different primitives. All actions start with p_3 and end with p_1 .

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

Table 4.2: Primitive segmentation and recognition results for Push aside and Push Forward action. Sequences that are identified incorrectly are marked in light gray.

one square to the right and p_1-p_2 represent moving two square to the right etc. L-shaped movements for the knight are expressed as a combination of primitives. For e.g., the sequence $p_1 - p_{25} - p_{11}$ is moving one square to the right and moving 2 square to the top is a knight move. Note that all of our primitives represent moving one square each. Paths along the diagonals represent moves for a bishop.

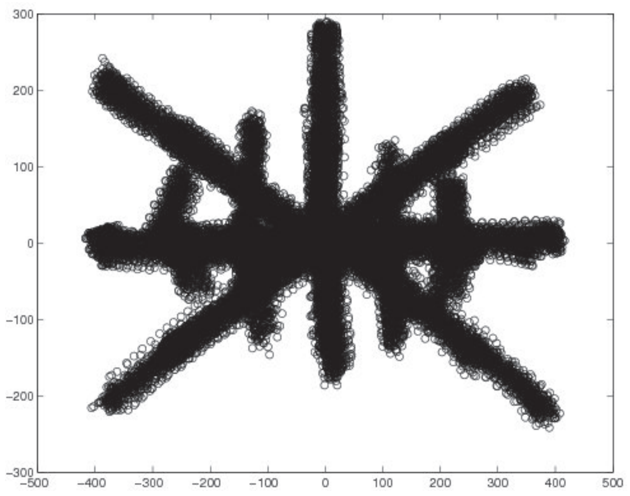


Figure 4.11: Some sample tracks for the chess data. Movements for rook, bishop and knight are shown.

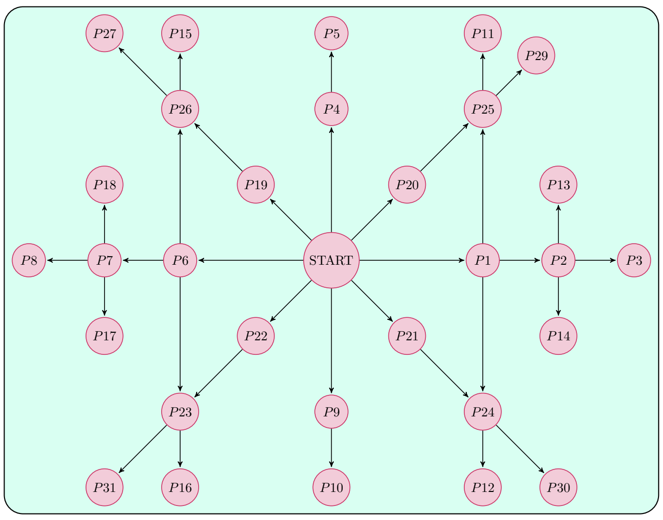


Figure 4.12: Extracted primitives for the chess data. Straight line paths represent moves for the rook and the queen. L-shaped paths represent moves for the knight. Paths along the diagonals represent diagonal moves for bishop.

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

Table 4.3: Primitive segmentation and recognition results for Move Object and Grasp actions. Sequences that are identified incorrectly are marked in light gray.

4.3 Discussion

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.

Our work is segmenting an action into smaller meaningful segments is hence different from [91] 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 [74] is not required. Thus our method is scalable to the number of states. Our approach of using states have a close connection to [92] but our method is superior in preserving the temporal order and thus it should also be superior in terms of recognition rates.

In [36] primitives are found by thresholding angular velocities. In this work 4 dimensional data of joint trajectories were segmented and the resulting segments for each of the joints were interpolated with 100 elements. Elements of each joint were concatenated to form 400 dimensional vectors and PCA was applied to reduce the dimension to 11. k-means clustering was then performed in the latent space to find the control points. Reproduction was performed by projecting points back to the input space. The use of PCA was unnecessary complication in this case since the input space was only 4 dimensional. Another disadvantage with this method is that strong assumptions must be made about the segmentation of the data, and the duration of the primitives. We have provided a higher level abstraction of primitives using a stochastic context-free grammar which is not possible with the approach in [36]. In [77] human motions are represented as a binary tree. Actions are recognized by finding the optimal node transitions in the tree. The binary tree construction approach in [77] is not suitable for sequential learning. Our states in the final HMM and the nodes in the last layer of [77] are comparable to some extent. The observations that belong to one state are neighbours in time in our case. In [77], frames belonging to one node are close in the 1D projected space. Takano and Nakamura [39] have also approached the problem of finding motion primitives using HMMs. They have modeled each actions via a discrete hidden Markov model. In their approach primitives are assumed to be known where as our approach learns the primitives from the data. In [18] subgoals are detected from trajectories by detecting regions that the agent visits frequently on successful trajectories but not on unsuccessful trajectories. This paper addresses a reinforcement learning scenario and appears to be quite different. They use a diverse density approach which requires positive and negative instances.

Though the approach has its merits, it also suffers from certain limitation/drawbacks. The approach works well if the subjects start the movements from a fixed position and move the object to a certain distance. Even if the subject repeats the same movement at the same scale, i.e. move object from location A to location B, but from a different position, we will end up with a different set of primitives. This is because we are looking at the hand trajectories which is going to look entirely different when the subject is moved. We propose a solution to this problem by looking at the object context in Ch. 5. Using object context, we are able to identify all movements from location A to location B to be the same, no matter where the subject starts the action from.

Chapter 5

Primitives with Object Context

Contents

In Ch. 4, the primitives were found by segmenting the hand trajectories. This method will yield a lot of primitives when the person changes his initial position or when the target position is changed. Moreover, the hand trajectories will not exhibit any structure in this case. All the actions we consider involve objects. If we look at the object trajectories, we get some kind of structure and trajectories corresponding to different actions will be separated. This is illustrated in Fig. 5.1. Therefore extracting primitives in the *object space* and then propagate the segmentation into the *action space* seems a better solution. In this chapter we will incorporate object context and extract primitives in an efficient way. We define *object space*

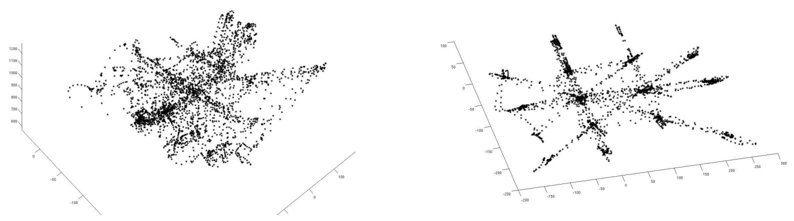


Figure 5.1: On the left, hand trajectories corresponding to actions performed at different locations are shown. On the right the object trajectories are shown. Clearly, the object trajectories exhibit better structure.

to be the space where the object movements are taking place and *action space/movement space* to be the space where the hands are moving. We use the object trajectories and find primitives as in Ch. 4. Primitive segmentation in the object space will induce a segmentation in the action space. This is outlined in Fig. 5.2. We have improved the approach in Ch. 4 in several ways here:

- Object information is used to segment primitives.
- Arc length is used to cover the data.
- When states are merged, they are not kept as mixtures as in Ch. 4. But they are merged to form a new state.
- Online primitive segmentation is proposed instead of the batch processing method in Ch. 4.
- Approximate primitive detection method is proposed.
- New large data set is used for extensive testing.

5.1 Modeling Object Action Interactions

We learn the primitives in the object state space by assuming that similar object trajectories have a common underlying hidden state sequence. Therefore object trajectories are first expressed as a sequence of hidden states. Primitives are detected from these sequences by finding common state subsequences. From the temporal continuity of primitives, a grammar for the primitives can then be found.

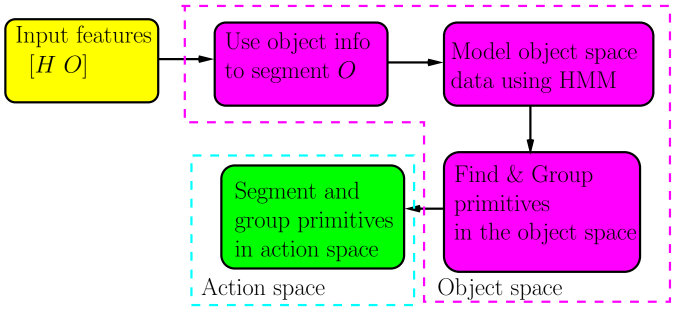


Figure 5.2: Overview of our approach. H denotes features from the action space and O denotes features for the object. Features for the object are first analyzed and segmented. This is then used to extract the primitives for the action space. Magenta boxes denote analysis in the object space and cyan box represents analysis in the action space.

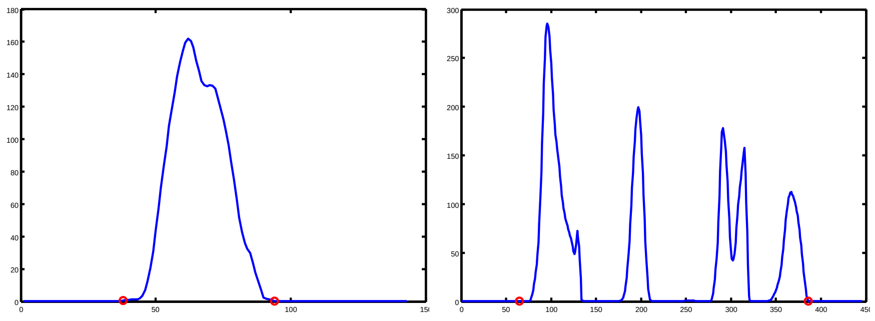


Figure 5.3: Thresholding the trajectories using a threshold. Time and the magnitude of the velocity at each time are plotted along x and y axis respectively. The first and last points above and below the threshold are chosen for segmenting the trajectories. On the left a sequence with only one manipulation action is shown. On the right a sequence with several actions in it are shown.

Let $[H_t^i \ O_t^i]$ represent the feature vector for i-th action sequence we are analyzing where H and O represent features for the arm and the object respectively. The subscript t is used for indexing time. Each of H_t and O_t are of the form $[P_t \ V_t]$ where P_t and V_t represent the position vector and velocity vector respectively. The increase and decrease of velocity of the object is then used to detect the starting and ending of the object movements. We choose t_1 and t_2 such that $|V_{t1}^O| > thresh$ and $|V_{t2}^O| < thresh$. The value of $thresh$ is chosen such that spurious movements due to measurement errors will be discarded. We choose t_1 such that $|V_t^O| < thresh$ for all $t < t_1$.

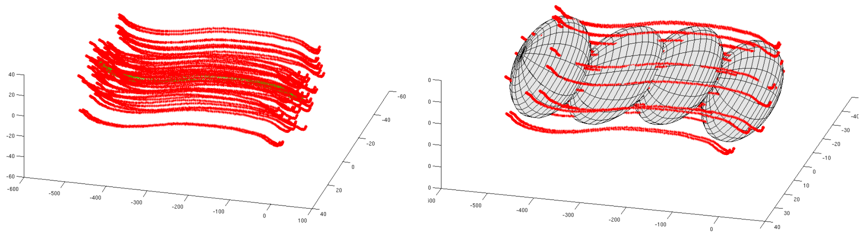


Figure 5.4: (Left:)A sequence with several additional sampled-sequences. (Right:) The result of applying Alg. 5.1.1.

Similarly t_2 is chosen such that $|V_t^O| < thresh$ for all $t > t_2$. Thus t_1 and t_2 are the first and last time instants where the magnitude of velocity is above the threshold. Thus if a sequence contains several manipulation actions in it with some rest in the middle, we will still be able to segment the whole movement part. Our aim is not segmenting each single manipulation part at this stage. This is illustrated in Fig. 5.3. On the left the start and end points for a single manipulation action is shown. On the right a complex action sequence with four manipulation actions is shown. In the second case, we are not attempting to separate each individual manipulation actions by thresholding. The first point above the threshold and the last point below the threshold will give us the start and end points of manipulation. We can segment the sequence of observations for the object as shown below:

$$\underbrace{O_1, O_2, \cdots}_{A}, \underbrace{O_{t_1}, \cdots O_{t_2}}_{B}, \underbrace{O_{t_2+1} \cdots O_T}_{C} \quad . \tag{5.1}$$

The observations in segment B denote the *act* part where the object state is changing while for the *approach* and *remove* parts A and C, no object state changes appear. The above segmentation in the object state space will induce a segmentation in the physical movement space:

$$\underbrace{H_1, H_2, \cdots}_{D}, \underbrace{H_{t_1}, \cdots H_{t_2}}_{E}, \underbrace{H_{t_2+1} \cdots H_T}_{F} \quad . \tag{5.2}$$

Since the actions can take place anywhere on the table and we want our primitives to be independent of the location of the object/subject we perform a transformation that will remove the location dependency of the features.

Applying the transformation $O_t - O_{t_1}$ for $t_1 < t < t_2$ we can imagine that the object starts to move from the origin in each of the sequences. From this point onwards all trajectories are assumed to begin at the origin.

We can now make use of the HMM model building approach described in Ch. 4 to model the observed sequences. At the end of the model building process, we will end up with a single HMM λ_F. Let $\mathbf{x_t^k}$ denote the object position at time t in k^{th} sequence. The sequence index k will be omitted where it is not necessary. We emphasize that our model building approach is based on the assumption that two similar sequences have a common underlying hidden sequence. By identifying this hidden sequence, we will be able to identify similar sequences. In Ch. 4 the sequences were segmented into approximately equal parts in an ad hoc manner. Here we propose to divide the sequences into parts according to the arc length of the trajectory. This is to avoid very small states with a lot of points that belong very close. Such a situation can arise when the object is not really moving but a small variation in object position is observed due to noise in the measurement.

The segmentaion and Gaussian estimation is explained in Alg. 5.1.1. We first calculate the spatial variation of the sequence P_i by calculating the arclength. The object trajectory P_i is then segmented into parts where each part has an arclength approximately equal to a predefined value α. Points belonging to each part is considered to be a state and is modeled by a multivariate Gaussian. After this step, we can consider the sequence P_i to be a sequence of states.

Algorithm 5.1.1: Algorithm for covering the data with Gaussians

```
Input: Observation sequence P=$\mathbf{x_1}, \cdots \mathbf{x_t}$, Threshold=$\alpha$,
       noise-level=$\sigma_l$
Output: Means $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ for a set of Gaussians covering the
        trajectory
Compute $arcLen$=arc length of the trajectory
Generate additional sequences by sampling from $\mathcal{N}(x_{il}, \sigma_l)$
Divide P into segments such that each part has an arc length
approximately $\alpha$
foreach segment $i$ = do
    Calculate $\boldsymbol{\mu}(i)$=mean(points in segment i)
    Calculate $\boldsymbol{\Sigma}(i)$=covariance(points in segment i)
end
```

After covering a sequence X_1 with Gaussians, we can think of the se-

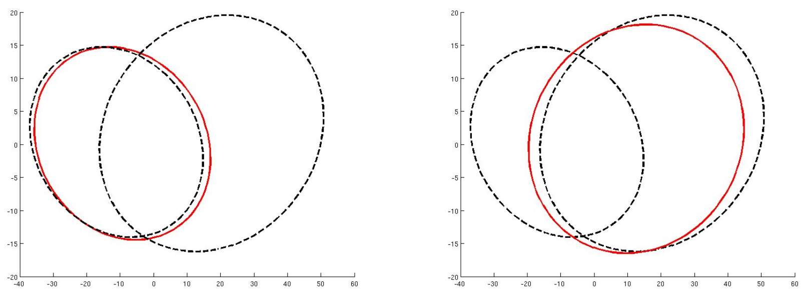


Figure 5.5: In this figure the dotted ellipse represents the covariances of two states. The solid line represents the updated covariance using Eq.5.3 and 5.4. (Left): When $\omega = 0.1$ the updated covariance is more like the first covariance (Right): When $\omega = 0.9$ the updated covariance is more close to the second covariance.

quence to be generated by the following process: State 1 is selected with probability 1 and a few samples are drawn from it. This is followed by selecting states $2, 3, \cdots$ in that order and samples drawn. The state transition probability can be assumed to be uniform. In another words, the sequence X_1 is a sample generated from $\sum_{i=1}^{N} w_i \mathcal{N}(\mathbf{x}, \boldsymbol{\mu_i}, \mathbf{\Sigma_l})$ where the Gaussian are selected in incremental order.

State comparisons and merging are done as in Ch. 4. In Ch. 4, when two states were merged together, the output probability associated with the new state was taken to be the mixture of the two distributions associated with the states. This means that if we train the model with a large number of similar sequences, there will be a large number of states to be merged together and hence the number of mixture components can be increased to a large number. Therefore, we follow a new approach that will avoid this situation. When two sequences are merged together the parameters of the merged state is updated using Eq. 5.3 and Eq. 5.4 [93]:

$$\mathbf{\Sigma^{-1}} = \omega \mathbf{\Sigma_0^{-1}} + (1-\omega)\mathbf{\Sigma_1^{-1}} \tag{5.3}$$

$$\boldsymbol{\mu} = \mathbf{\Sigma}(\omega \mathbf{\Sigma_0^{-1}} \boldsymbol{\mu_0} + (1-\omega)\mathbf{\Sigma_1^{-1}} \boldsymbol{\mu_1}) \tag{5.4}$$

Here ω is a parameter that controls the contribution from each of the states. This effect is illustrated in Fig. 5.5. A value close to 0 gives importance to first covariance and a value close to 1 gives importance to the other one. We have set $\omega = \frac{n}{n+1}$ where n is the number sequences used

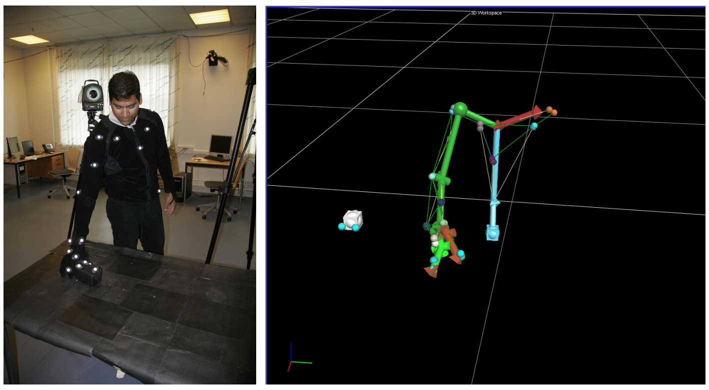


Figure 5.6: (Left:) The experimental setup for this paper. Markers are attached to both the person and the object. Object can be moved from any position to any other position on the table. (Right:) Object and arm template used in Vicon for recording the data.

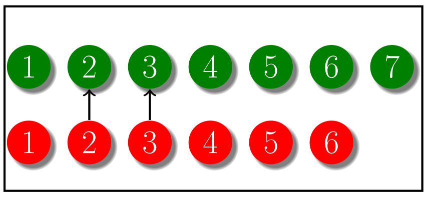
to create/modify a state. This prevents an outlier sequence to have a big influence on the final states.

5.2 Primitive Segmentation in the Object State Space

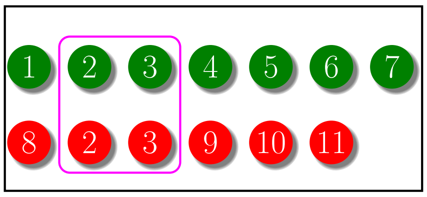
Primitive segmentation in the object state space takes place in parallel with the HMM building. Sequences that are similar will go through the same sequence of HMM states. These state sequences are then expressed as state changes by removing multiple occurrences as shown below.

$$\underbrace{s_1, s_1, \cdots s_1}, \underbrace{s_2, s_2, \cdots s_2}, \underbrace{\cdots\cdots} \underbrace{s_k, , s_k, \cdots s_k}$$
$$\Downarrow$$
$$s_1, s_2, \cdots s_k \quad .$$

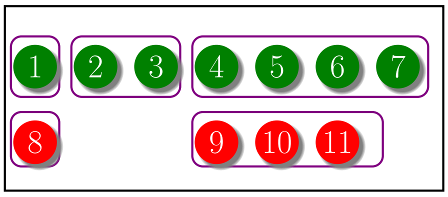
Treating the state sequences as strings, we can use the LCS algorithm given by Alg.4.2.1 shown in Fig. 4.3 and find the common state subsequences across various sequences as in Ch. 4. When we have processed only the first sequence P_1, the state sequence S_1 is a directed path. When the sequence P_2 is processed and some states are merged with λ_F, the merged states are the common parts of S_1 and S_2. This can be detected using the LCS algorithm. Removing each of the common subsequences from the state sequences, we



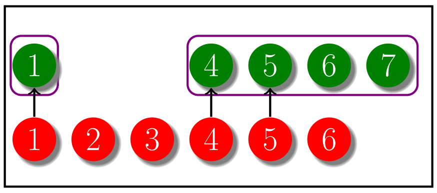
(a) Each row shows a sequence. States 2 and 3 in sequence 2 are to be merged with states 1,2 in sequence 1 respectively.



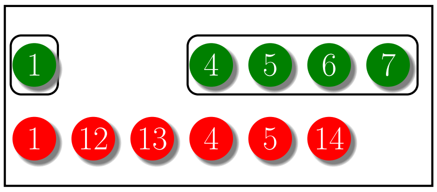
(b) State sequences after merging. States that are not merged in sequence two are given new state numbers.



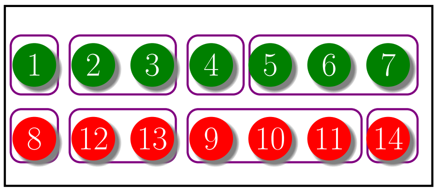
(c) Primitives after the segmentation algorithm.



(d) A new sequence is shown in row 2. States 1,4 and 5 of this sequence is to be merged with previously observed states 1,4 and 5 respectively.



(e) State sequence of sequence 3 after merging



(f) Updated set of primitives after observing sequence 3.

Figure 5.7: Illustration of primitive segmentation using LCS algorithm

can extract the remaining primitives that are unique. Then for each S_k for $k > 2$, the process is repeated with each of the primitives and S_k. This process is illustrated in Fig. 5.7. After this step, we will be able to represent the state sequences as as sequence of primitives. We refer to the iterative approach in this section as *online primitive detection* approach.

5.3 Segmentation and Grouping in the Movement Space

The modeling and segmentation method discussed so far is applied to the data in the object state space which was denoted by B in Eq. 5.1. Once

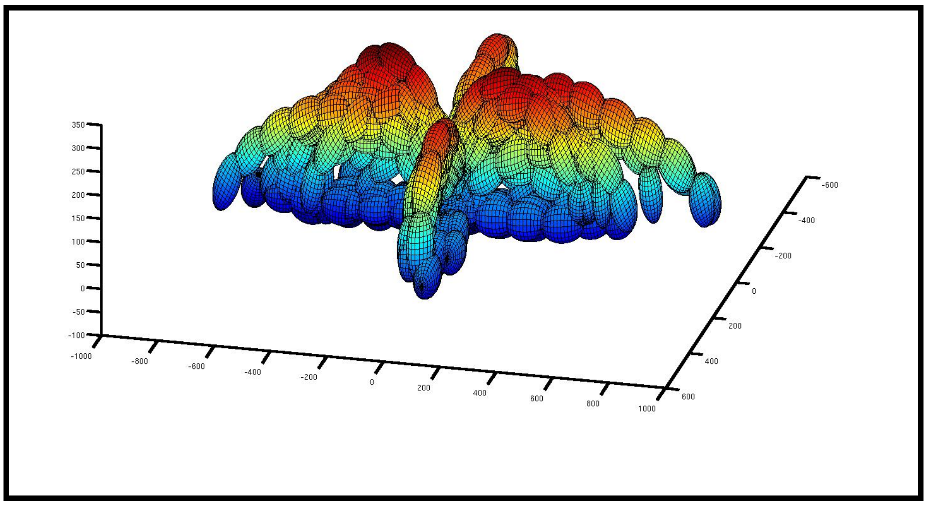


Figure 5.8: Data covered with Gaussians. Ellipsoids show the contours of Gaussians used to cover the data. Lengths of the trajectories indicate how much distance the object was moved.

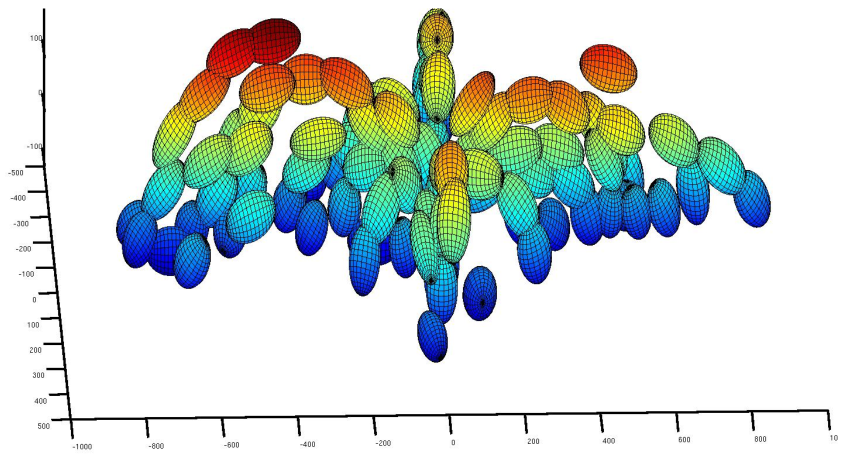


Figure 5.9: The final states in the object space after merging. Measurements along the axes are in mm.

we have primitives in the object state space we can find the corresponding segments in the movements space denoted as E in Eq. 5.2. The segment D represents the *approach object* part and the segment F represents the *retrieve hand* part. As both of these movements do not induce any object state change, they are each associated with a single movement primitive.

5.4 Generating the Grammar for Primitives

The grammar for the detected primitives can be generated as described in Sec. 4.2.5. This grammar rules can be used to verify the primitive sequences.

Algorithm 5.4.1: Algorithm for action classes using primitives. Two primitive paths belong to the same set if one is a subset of another.

```
Input: Set of different primitive sequences P={$P_1, \cdots P_n$}
Output: Set of unique primitive classes
C={$P_1$}
foreach $P_k$ in P, $k > 1$ do
    Compare $P_k$ with $C_i \in C$
    If $P_k \supset C_i$ for some $C_i \in C$ replace $C_i$ with $P_k$
    If $P_k \not\subseteq C_i$ for all $C_i \in C$ Add $P_k$ to C
end
```

Algorithm 5.4.2: Algorithm for action classes using primitives. Two primitive paths belong to the same set if they are equal.

```
Input: Set of different primitive sequences P={$P_1, \cdots P_n$}
Output: Set of unique primitive classes
C={$P_1$}
foreach $P_k$ in P, $k > 1$ do
    Compare $P_k$ with $C_i \in C$
    If $P_k = C_i$ for some $C_i \in C$ if $|P_k| > |C_i|$ replace $C_i$ with $P_k$
    If $P_k \neq C_i$ for all $C_i \in C$ Add $P_k$ to C
end
```

5.5 Recognition of Novel Sequences Using Primitive Paths

Once we have identified all the primitives in the object state space, we can represent each sequence as a primitive path. One way to identify different type of sequences is to find all unique primitive paths. Then two observed sequences are same if and only if they have the same primitive path. Another way to form clusters is to make two primitive paths(and consequently the observed sequences) in the same group if one is a subset of another. In that case pushing in one direction at different distances will belong together. These two types of grouping are given in Alg. 5.4.1 and Alg. 5.4.2. For our recognition purposes in the experiments in Sec. 5.6 , we have used Alg. 5.4.1. Before sequence classification could be done, primitives have to be identified in a novel sequence. This is explained in the next section.

5.5.1 Detecting Primitives in Unknown Sequences

Once we have built the model, we can segment an unknown sequence into sequence of known primitives in two ways. If the sequence contain only one manipulation action, we can compute the most probable path of the sequence given the model using the Viterbi algorithm and then find the primitives from the state sequence. We refer to this as the *exact method.*

But when we have to detect primitives from a long sequence with many manipulation actions, we need a different procedure. When we have a long sequence with many primitives, we cannot compute the state sequence of the observations directly. Since the aim of primitive detection is imitation, we only need to identify all the primitives in the sequence. An exact mapping of each of the points to a primitive is not required. Primitive detection in long sequences is done by first covering the sequence with Gaussians as described in Sec. 5.1. Then the new states are compared with the states of the final model using Kullback-Leibler divergence given in Eq. 4.1 and 4.2. Since we have modeled each of the manipulation primitive to start from the origin, when one primitive has ended, the states modeling the rest of the data will be different from the states in the final model. Hence when we encounter a state that is different from known states, the points from that time onwards are shifted to the origin and the process is repeated. We refer to this method as *approximate method.* In Fig. 5.12 we have shown the result of segmenting a complex sequence using our model and the comparison to ground truth is also shown. The ground truth segmentation is obtained by manually looking at the segmentation points in the complex sequence. The resulting segmentation is very close to the ground truth segmentation. The approximate method can be applied to simple sequences or complex sequences. Recognition results using these two methods are discussed in the experiments Sec. 5.6.

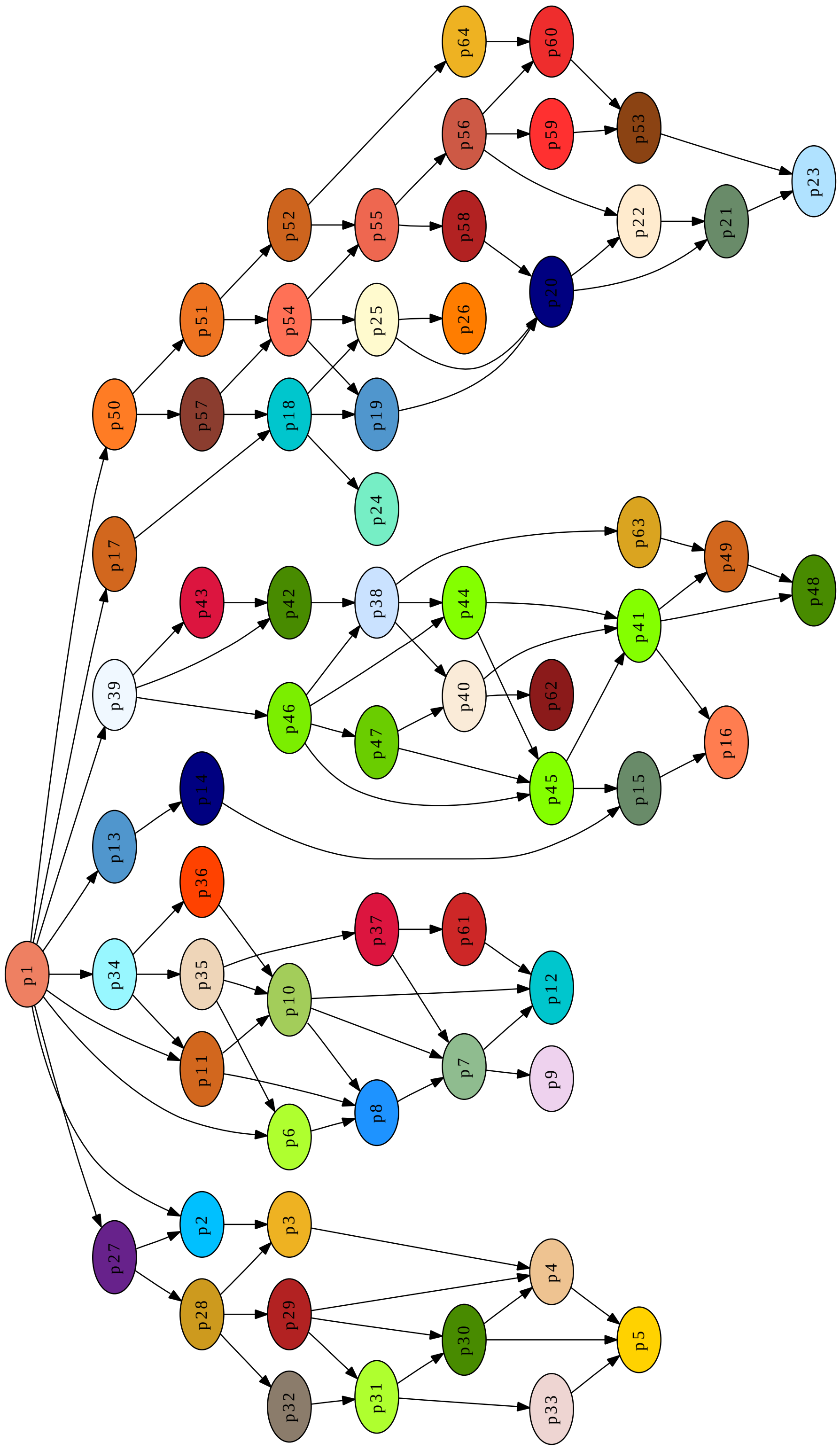


Figure 5.10: The result of our primitive extraction method in the object space.

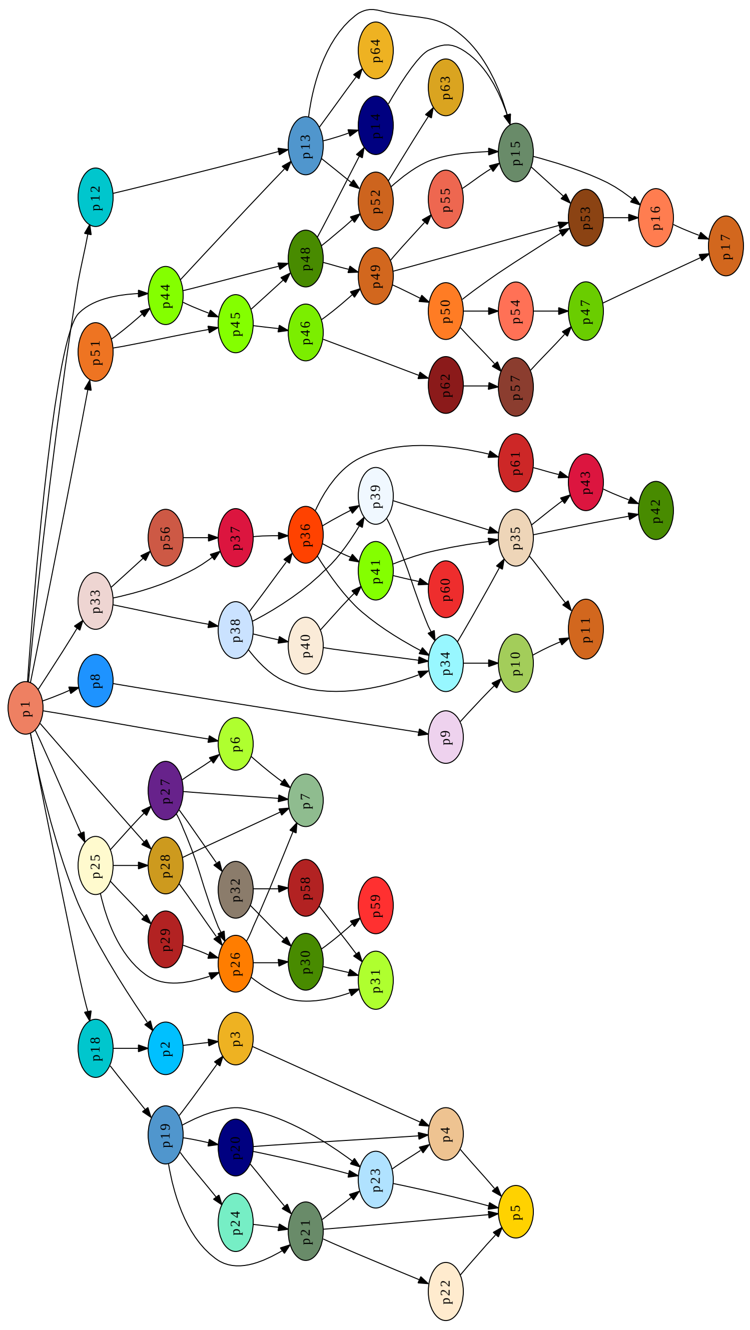


Figure 5.11: The result of our primitive extraction using the online method.

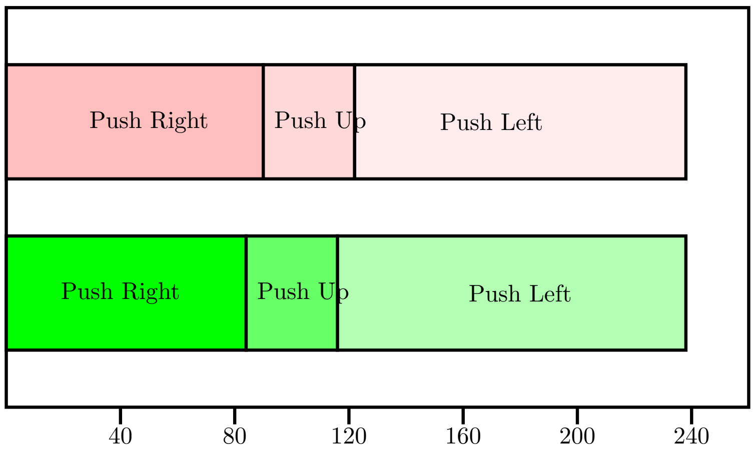


Figure 5.12: This figure shows the segmentation result for a complex sequence with three manipulation primitives. On the x-axis we show the length of the sequence. Ground truth segmentation is shown at the bottom and the segmentation using our approach is shown on top.

5.6 Experiments

We have collected a data set consisting of 13 subjects moving an object on a table. The data was recorded using a Vicon Nexus motion capture system [94]. A box equipped with 3 markers was used as the object to be manipulated. The subjects were equipped with 14 markers on their body. Among the 14 markers placed on each person, only the 3 markers placed on the hand were used in our experiments in analyzing the actions under consideration. The recording setup and the arm model used in Vicon are shown in Fig. 5.6. The system is able to record the 3-D position of the markers quite accurately. The three markers on the hand are used to calculate the position of the hand in our experiments. Only the hand location is used since it gives enough information for recognizing and imitating the demonstrated movements and this has been validated experimentally in [95]. The subjects were allowed to move the object from any position on the table to any other position on the table. The data $[H, O]$ used in these experiments are based on the 3-d trajectories of the hand, and the location and orientation of the object. The center of mass of the 3 markers on the hand was used to define the position vector H for the hand. The center of mass and the orientation of the object defined O. The distance moved varied each time of the repetition. Our aim is to learn all movement primitives and express all actions

Simple Actions		Complex Actions	
md	42	ml-pd	2
mf	37	pd-ml-pd	2
ml	46	pd-pl	3
mr	38	pf-pl	6
pd	49	pl-pd	5
pf	44	pl-pf	4
pl	64	pr-pd	4
pr	61	pf-pr	3

Table 5.1: Summary of the new data set. pr=push right, pl=push left, pf=push forward, pd=push down, mr=move right, ml=move left, mf=move forward, md=move down.

in terms of the primitives. The detected primitives will be used to identify actions in unknown sequences.

The new data set, available online [63], is an extension of the KTH data set [90] used in Ch. 4. In the KTH data set, object information is not included. Our new data set includes object location and orientation. We have also added more actions and complex actions. A summary of the recorded data set with the actions and the number of actions is shown in Tab. 5.1.

Each of the arm movements started from a rest position (*arm resting on the table*) and ended at the same position. The recorded data is segmented by thresholding object velocity as described in Sec. 5.2 to find the part where the object is being moved. The segmented data for the object movement part is then processed with Algorithm5.1.1 from Sec. 5.1 and the result is shown in Fig. 5.8.

To illustrate the effects of building the model with different type and number of sequences , we consider 3 kinds of experiments:

1. An experiment where the model is built with sequences that contain single manipulation actions only. This experiment is similar the one in Sec. 4.2.9. Our aim is learn primitives and primitives grammar from the data and then use it to detect primitives from novel sequences. This experiment is explained in Sec. 5.6.1.

2. An experiment where the model is built with sequences that contain single and multiple manipulation actions. Complex sequences are in-

cluded to see how the primitives change when complex sequences are included. The order in which complex sequences are introduced to the system do not affect the final set of primitives. This experiment is explained in Sec. 5.6.2.

3. An experiment to illustrate the sequential learning capability of our approach. This experiment illustrates how the model is updated sequentially while processing the sequences one by one. This experiment is explained in Sec. 5.6.3.

5.6.1 Model with Single Manipulation Action

For our first experiment we consider eight movements: *push object right, push object left, push object forward, push object downwards* , *move object right, move object left, move object forward* and *move object downwards.* The aim of the experiment is to extract the primitives for this data and test how good is the primitive representation of the data in recognizing novel sequences. We test the quality of our representation against the ground truth. The ground truth is the labeled information of whether a particular sequence is push right or push left etc.

Seventy percent of the sequences in our database is used to built an HMM as described in Sec. 5.1. The sequences are covered using Alg. 5.1.1. This give rise to a lot of states as shown in Fig. 5.8. States corresponding to only twenty percent of the data are shown in the figure for a clear visualization. The number of states are decreased after the merging process and are shown in Fig. 5.9. The sparsity structure of the transitions in the final HMM is visualized in Fig. 5.13. Using the constructed HMM, the observation sequences are converted to state sequences and the primitives are identified using the LCS algorithm described in Sec. 5.2. The primitive graph for the detected primitives using the batch processing method in Sec. 4.2.4 is as shown in Fig. 5.10. The primitive graph for the primitives found using the online method in this chapter is given in Fig. 5.11. These two graphs can be different. The difference comes from different state sequences during the model building and after the model building. Using the primitive graph, the SCFG for representing all the manipulation movements is derived and parts of the grammar is shown in Fig. 5.19. The full set of rules is given in Appendix A. It is shown that all movements start with primitive p1.

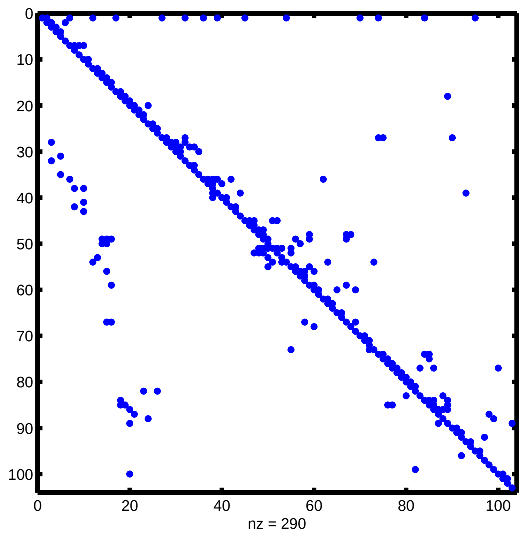


Figure 5.13: Sparsity structure of the final HMM transitions.

Since all states starts at the origin, a few of the initial points are assigned to state 1 in all the sequences. The parts D and F in Eq. 5.2 become the *approach* and *retrieve* parts respectively in the physical movement space. Note that the primitives in Fig.5.9 shows the primitives in the object space and also the induced primitives in the movement space. If we consider the primitive path $p1 \to p2 \to p3 \to p4 \to p5$ for Push down, the primitives $p2$, $p3$, $p4$, and $p5$ correspond to repetitions in our training data in which the object was pushed to right at 4 different distances.

Using this model, additional test sequences are segmented into different classes and is compared with the ground truth segmentation. Each ground truth group may be represented by one or more primitive paths. A new sequence is correctly identified if its primitive path is same as one of the primitive path for its ground truth. If the primitive path do not match with any of the known primitive paths, it is classified as an unknown sequence. Different primitive paths for our dataset are shown in Fig. 5.14 and Fig. 5.15. In Tab. 5.2, we give the results of our recognition test as a confusion matrix. Very few sequences are misclassified. Misclassification between push and move along the same direction can occur if the object was lifted up while pushing. The classification results when using primitives from the

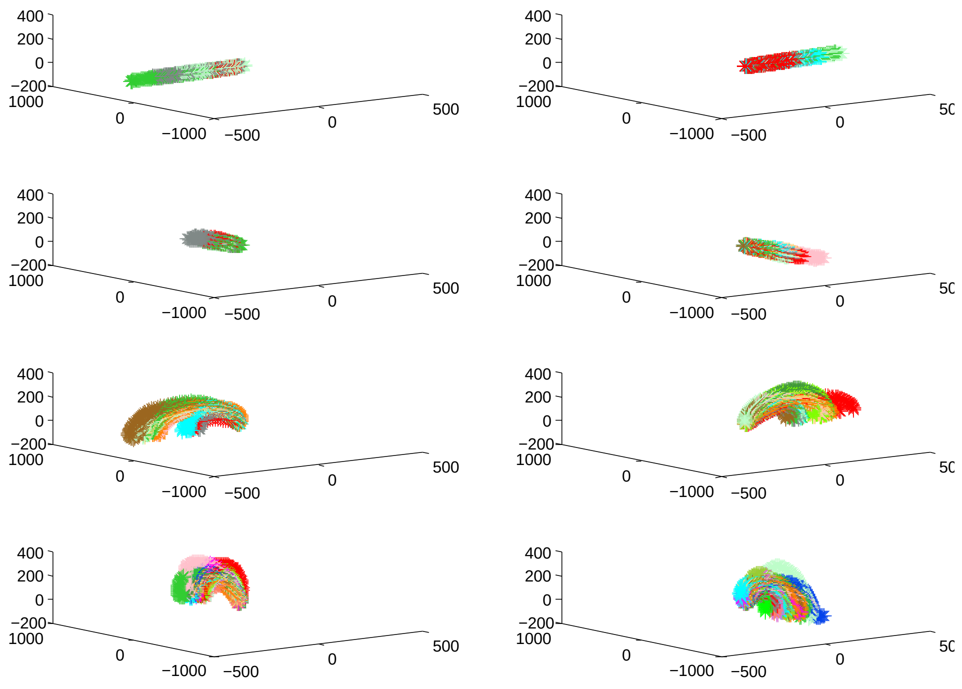


Figure 5.14: Different primitive paths for different actions. Different paths are given different colors.

online approach is shown in Tab. 5.4. We can also use the approximate method given in Sec. 5.5.1 to recognize the actions. The result of approximate method is shown in Tab. 5.3. The results of approximate and online methods are comparable to the results of the exact method. Primitives from single manipulation actions can be used to detect primitives from complex sequences. For this we have to use the approximate method described in Sec. 5.5.1. The result primitive segmentation on complex sequences is shown in Tab. 5.7. The effect of segmenting a complex sequence into its component primitives is shown in Fig. 5.12. As shown in the figure, there is a slight lag in detecting the end of each of the component primitives. This is because we need to observe some points to see that the trajectory is deviating from the current primitive. The results shows that the approximate method is quite good in detecting primitives.

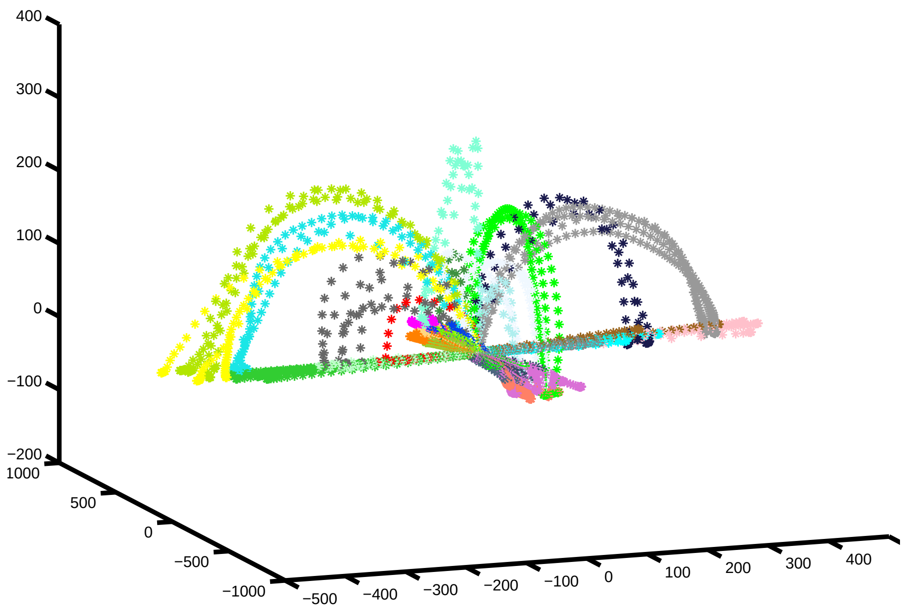


Figure 5.15: Primitive paths for different actions are shown. Sequences with the same primitive path are given the same color.

	pd	**pf**	**pl**	**pr**	**md**	**mf**	**ml**	**mr**	**U**
pd	15	0	0	0	0	0	0	0	0
pf	0	18	0	0	0	0	0	0	1
pl	0	0	30	0	0	0	0	0	0
pr	0	0	0	24	0	0	0	0	1
md	1	0	0	0	16	0	0	0	5
mf	0	3	0	0	0	10	0	0	1
ml	0	0	0	0	0	0	12	0	13
mr	0	0	0	0	0	0	0	13	2
U	0	0	0	0	0	0	0	0	0

Table 5.2: Primitive detection using exact inference. pr=push right, pl=push left, pf=push forward, pd=push down, mr=move right, ml=move left, mf=move forward, md=move down, U=unknown. True and detected labels are along rows and columns respectively.

	pd	**pf**	**pl**	**pr**	**md**	**mf**	**ml**	**mr**	**U**
pd	12	0	0	0	0	0	0	0	3
pf	3	14	0	0	0	0	0	0	2
pl	0	0	30	0	0	0	0	0	0
pr	0	0	0	23	0	0	0	0	2
md	5	0	0	0	17	0	0	0	0
mf	0	3	0	0	0	9	0	0	2
ml	1	0	0	0	0	0	18	0	6
mr	0	0	0	2	0	0	0	9	4
U	0	0	0	0	0	0	0	0	0

Table 5.3: Primitive detection using approximate method. pr=push right, pl=push left, pf=push forward, pd=push down, mr=move right, ml=move left, mf=move forward, md=move down, U=unknown . True and detected labels are along rows and columns respectively.

	pd	**pf**	**pl**	**pr**	**md**	**mf**	**ml**	**mr**	**U**
pd	15	0	0	0	0	0	0	0	0
pf	1	14	0	0	0	2	0	0	2
pl	0	0	30	0	0	0	0	0	0
pr	0	0	0	20	0	0	0	0	5
md	1	0	0	0	13	0	0	0	8
mf	0	0	0	0	0	8	0	0	6
ml	0	0	0	0	0	0	7	1	17
mr	0	0	0	0	0	0	0	7	8
U	0	0	0	0	0	0	0	0	0

Table 5.4: Confusion matrix for action recognition using the primitives from online method. pr=push right, pl=push left, pf=push forward, pd=push down, mr=move right, ml=move left, mf=move forward, md=move down, U=unknown. True and detected labels are along rows and columns respectively.

5.6.2 Model with Single and Multiple Manipulation Action

In this experiment we include all the action we have mentioned in the above experiment. In addition we include several sequences that contain more than one manipulation action. Two types of complex sequences are included: object Pushed Right followed by Push Up, object Pushed Left followed by Push Down. When complex sequences are included in the training set for building the model, new states are generated as necessary, see Fig. 5.16. For eg., if we have seen Push Up sequences before and observe a complex se-

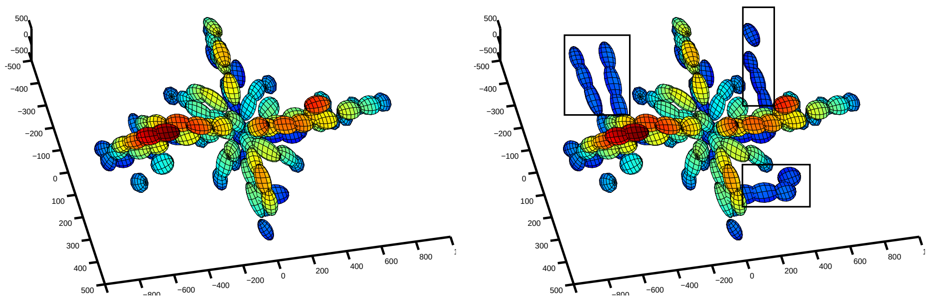


Figure 5.16: The result of including complex actions in the primitive learning phase. On the left states for the single action sequences are shown. When complex sequences are introduced, new states are added to the existing states as shown on the right.

quence *Push Right -Push Up*, then new states for *Push Up* will be generated. The result of the primitive segmentation after the model is built is shown in Fig. 5.17. As we can see, training with additional complex sequences result in additional primitives. A manipulation action followed by another one is treated as a new type of sequence and additional primitives are generated to support this type of sequences. The results of segmenting test sequences into primitives using the exact method is shown in Tab. 5.5. Detection results using the approximate method is given in Tab. 5.6. Recognition rates of both the methods are good. In the approximate method, the first sequence is identified correctly in most cases. If the ending of any component sequence is not detected correctly in any complex sequence, it will affect the recognition of all the following sequences. We note that the final primitives do not change depending on the order in which complex sequences are input to the system.

5.6.3 Primitives with Sequential Model Building

In this experiment we show the effect of our sequential model building and primitive detection. We illustrate the sequential building using a subset of the sequences from our dataset. First the model is built with several Push Right sequences and the resulting primitives are shown in Fig. 5.18(a). To update this model with additional sequences, we do not require the sequences we have already processed. The model is then updated using several Push Up sequences. After this step, the primitives are as shown in Fig. 5.18(b).

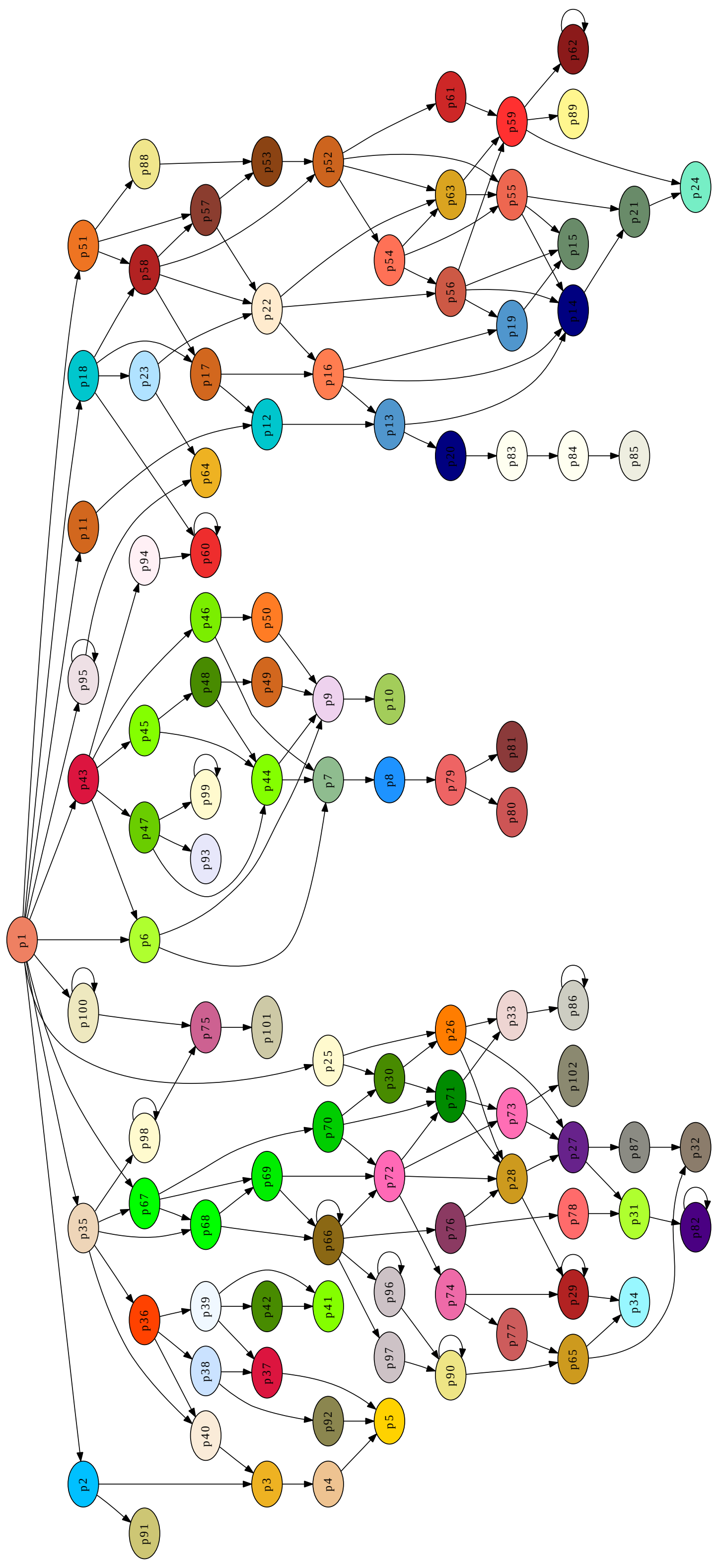


Figure 5.17: The result of our primitive extraction method when complex actions are included in the primitive learning phase.

	pd	**pf**	**pl**	**pr**	**md**	**mf**	**ml**	**mr**	**pf-pl**	**pl-pd**	**U**
pd	20	0	0	0	0	0	0	0	0	0	0
pf	0	15	0	0	0	3	0	0	0	0	0
pl	0	0	22	0	0	0	1	0	0	0	0
pr	0	0	0	22	0	0	0	0	0	0	0
md	1	0	0	0	15	0	0	0	0	0	2
mf	0	0	0	0	0	12	0	0	0	0	4
ml	0	0	0	0	0	0	10	0	0	0	7
mr	0	0	0	0	0	0	0	7	0	0	9
pf-pl	0	0	0	0	0	0	0	0	18	0	0
pl-pd	0	0	1	0	0	0	0	0	0	8	0
U	0	0	0	0	0	0	0	0	0	0	0

Table 5.5: Primitive detection results using exact method when complex sequences are used in primitive learning. pr=push right, pl=push left, pu=push up, pd=push down, U=unknown sequence

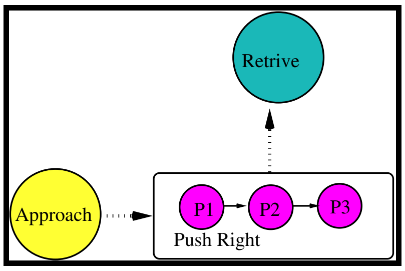
For the sequential updating we only require the current model parameters and the current set of primitives. The result of presenting the system with several Push Left sequences is shown in Fig. 5.18(c). The final result after adding Push down sequences to the model is same as the one shown in Fig. 5.18(d). The final model and the primitives are independent of the order in which different type of sequences are presented to the system.

5.6.4 Use of SCFG

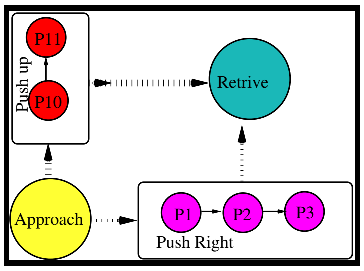
The detected primitive sequences are validated by the constructed SCFG. In this case we are assuming that the primitive detection is free of errors and hence the probabilities associated with the rules are not very much useful. But if there is a chance of uncertainty in the primitive detection part, the probabilities become useful, see [79, 96].

5.6.5 Effect of Parameters

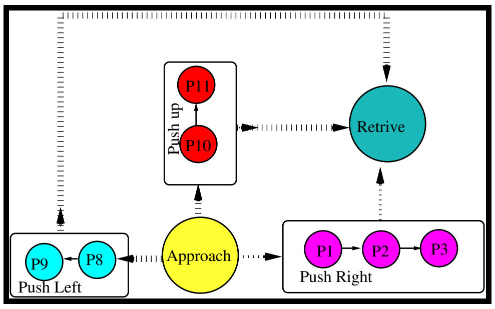
In this section we give a brief discussion on the effects of various parameters that we have used. In Algorithm 5.1.1 we have used a threshold parameter α. This parameter controls the size of the Gaussians for covering the data and can affect the final primitives we end up with. This effect is best explained with the chess scenario. If we set a high value for α such that each of the Gaussians cover 2 squares we will not know the difference between



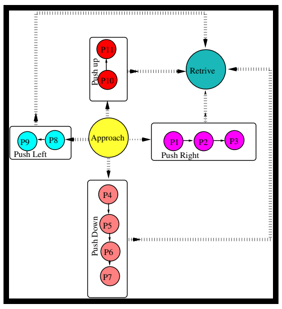
(a) Primitives when the model is learned with only push right movements.



(b) Primitives when push up movements are added to the model used to get Fig. 5.18(a).



(c) Primitives when push left movements are added to the model used to get Fig. 5.18(b).



(d) Primitives when push down movements are added to the model used to get Fig. 5.18(c).

Figure 5.18: Illustration of the sequential updating of the primitives. Primitives are added/refined as new data is observed. Data needed for learning the primitives in Fig. 5.18(b) are not used to get the updated primitives in this figure.

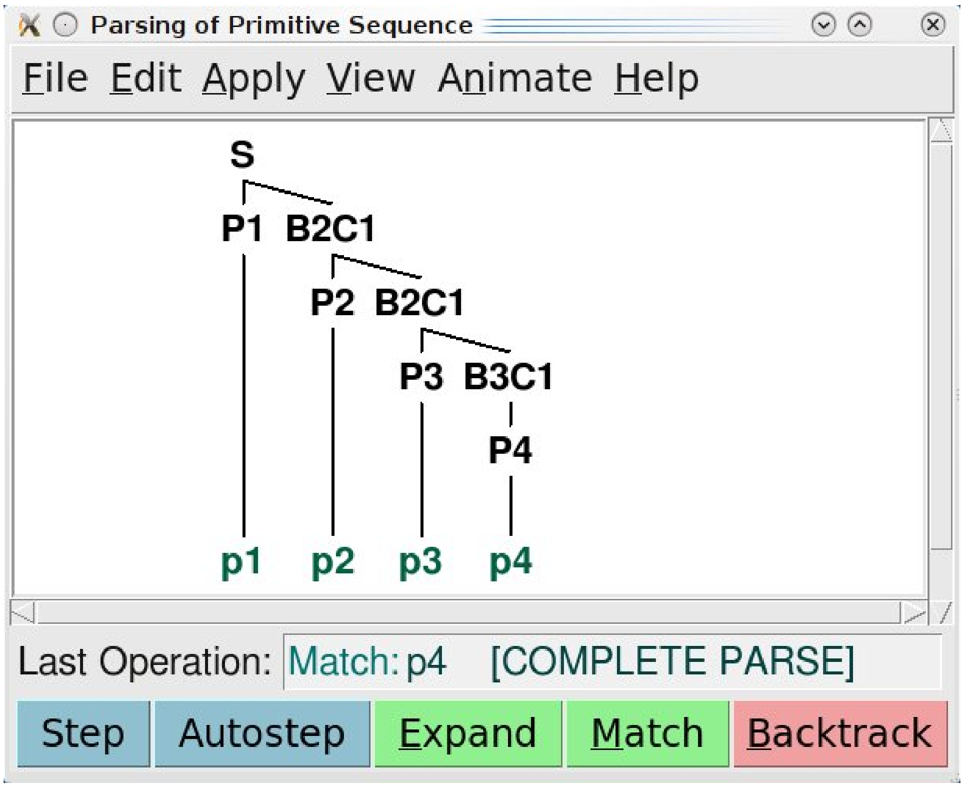
moving one square and two squares. Therefore the value should be chosen such that it will not exceed the smallest primitive we expect to find. At the same time if the value of α is too small, it will generate Gaussians that cover a very small area. We will have a high number of Gaussians and sequences from the same class will not generate same state sequences. This will generate additional primitives for describing the data. The effect of size of the Gaussians in the final result can be illustrated with simulated data that we have used in Sec. 4.2.8. Using small Gaussians results in a covering as shown in Fig. 4.6(c). Allowing large Gaussians will result in less number of Gaussians as shown in Fig. 5.21. In Fig. 5.21(a)-Fig. 5.21(c) the result of a particular covering is shown. Here, we have exactly one Gaussian to cover the shared region and 2 more Gaussians to share the rest of the data. The

```
S->p1 B2C1 [0.125]| p1 B6C3 [0.125]| p1 B11 [0.125]|
   p1 B13C11 [0.125]| p1 B17C14 [0.125]|
   p1 B27 [0.125]| p1 B34 [0.125]| p1 B39C45 [0.125]
B2C45->p2 [1]
B2C1->p2 B3C1 [0.5]
B3C1->p3 [0.5]
B2C1->p2 [0.5]
B3C1->p3 B4C2 [0.5]
B4C2->p4 B5C2 [0.5]
.
.
.
B25C73->p25 B20C73 [1]
B20C73->p20 [1]
B56C66->p56 B59C67 [0.33333]
B59C67->p59 B53C74 [1]
B53C74->p53 B23C74 [1]
B23C74->p23 [1]
B20C70->p20 B22C70 [0.5]
B22C70->p22 B21C75 [1]
B21C75->p21 [1]
B20->p20 [0.5]
B20C71->p20 [0.5]
B18C69->p18 B24C69 [0.33333]
B24C69->p24 [1]
```

Figure 5.19: grammar for actions

final result matches the one shown in Fig. 4.6(o). Further decrement in the number of Gaussians will fail to discover the structure in the data as shown in Fig. 5.21(d)-Fig. 5.21(f). In this case one sequence is covered with a single Gaussian. This sequence cannot be modeled with a single Gaussian. Thus the size of Gaussians should be such that they do not violate the assumption of normality for the underlying data. The Gaussians should be big enough so that in any repetition of the same sequence, it should pass through the same state sequence.

Another parameter of interest is θ which is the threshold for deciding



Last Operation: Match: p4 [COMPLETE PARSE]

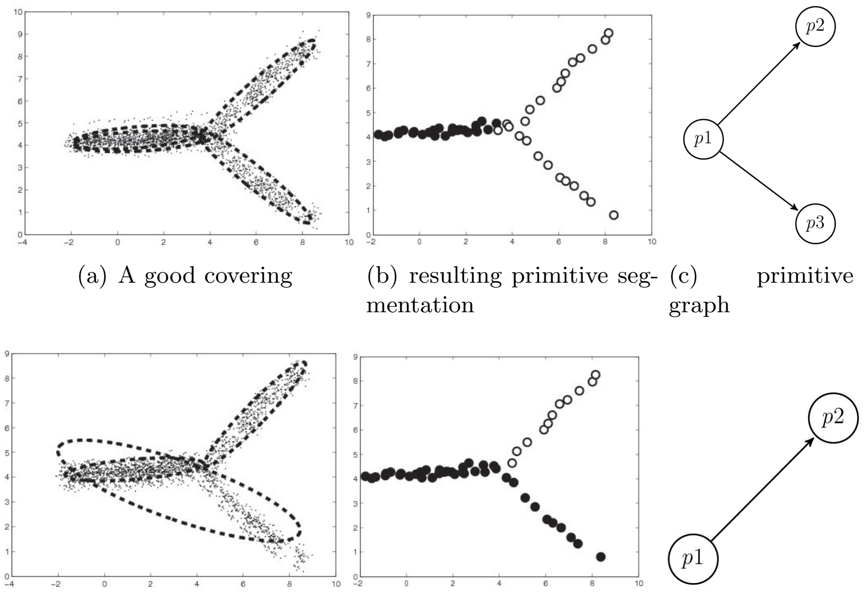
Figure 5.20: Primitive sequence parsing using NLTK

if two states should be merged or not. We chose this value to be half of average distance of adjacent pair states in a sequence. This ensures that states with good overlap are combined and represented by a single state.

5.6.6 Summary and Discussion

In chapter we have presented an improved primitive learning method using object information. Using the learned primitives, we are able to segment novel sequences into known sequence of primitives. Also novel sequences are classified using primitives. We are able to recover a primitive structure for the actions that is similar to the natural language description for the actions we have considered. Primitive based modeling of actions enables us to define a hierarchy of actions by converting continuous observations into discrete symbols. Several authors have represented actions in a hierarchical manner [45, 75, 76].

These works require the manual modeling of atomic movements/primitives.



(a) A good covering (b) resulting primitive segmentation (c) primitive graph

(d) A bad covering (e) resulting primitive segmentation (f) primitive graph

Figure 5.21: Illustration of the effect of parameters

The contribution of our work is that we perform this segmentation automatically.

The experimental results from [49] suggests that action perception and execution of motor primitives are connected through objects. There are also further studies from experimental psychology which confirms the role of objects in action understanding [47, 97]. In this paper we have exploited object information to learn action primitives.

Even though object detection and classification literature is quite large (for overview see [98]), there are not many attempts to combine it with action modeling [83, 61]. In [83] Hidden Markov models are combined with object context to classify hand actions. Image, object and action-based evidence was used to label and summarize activity and also to identify objects. They define a generalized class model to describe objects. Actions associated with each class were represented using trained HMMs. The states of such HMMs were connected to the regions through which the object moved for the particular action. Our approach learns such a model for modeling actions automatically. A graphical Bayesian model was used in [61] for modeling human-object interactions. Some of the conditional probabilities of this model was calculated using trained HMMs. These approaches require a good initial training of action models for later recognition even though a known structure is assumed. Our work goes beyond the state of the art in this area since it exploits object knowledge in the primitive learning process.

Our work relates to the recent work of [99] where a hierarchical tree structure is incrementally formed representing the motions learned by the robot. One of the issues raised is that each node representing a motion primitive may differ from those segmented in an off-line, supervised process. By integrating the object knowledge in the learning process, the resulting primitives are more similar to the ones generated in an off-line process.

	pd	pf	pl	pr	md	mf	ml	mr	pf-pl	pl-pd	pr-pd	mr-pd	U	pr-pd-pd	mf-pd-pd
pd	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pf	1	17	0	0	0	0	0	0	0	0	0	0	0	0	0
pl	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0
pr	0	0	0	21	0	0	0	0	0	0	1	0	0	0	0
md	3	0	0	0	9	0	0	0	0	0	0	0	6	0	0
mf	1	2	0	0	0	10	0	0	0	0	0	0	2	0	1
ml	0	0	0	0	0	0	9	0	0	0	0	0	8	0	0
mr	0	0	0	0	0	0	0	13	0	0	0	1	1	1	0
pf-pl	0	5	0	0	0	0	0	0	13	0	0	0	0	0	0
pl-pd	0	0	4	2	0	0	0	0	0	3	0	0	0	0	0
pr-pd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mr-pd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pr-pd-pd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mf-pd-pd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.6: Primitive detection results using the approximate method when complex sequences are used in primitive learning. pr=push right, pl=push left, pu=push up, pd=push down ,U=unknown sequence

	pd-pl	pf-pl	pf-pr	pl-pd	pl-pf	pr-pd	ml-pd	pd-ml-pd	pf-pf-pr	pr-pd-pd	ml-pd-pd	pd-ml-pd-pd	pl-pf-pr-pd
pd-pl	6	0	0	0	0	0	0	0	0	0	0	0	0
pf-pl	0	6	0	0	0	0	0	0	0	0	0	0	0
pf-pr	0	0	2	0	0	0	0	0	2	0	0	0	0
pl-pd	0	0	0	10	0	0	0	0	0	1	0	0	0
pl-pf	0	0	0	2	6	0	0	0	0	0	0	0	1
pr-pd	0	0	0	2	0	4	0	0	0	1	0	0	0
ml-pd	0	0	0	0	0	0	3	0	0	0	1	0	0
pd-ml-pd	0	0	0	0	0	0	0	3	0	0	0	2	0
pf-pf-pr	0	0	0	0	0	0	0	0	0	0	0	0	0
pr-pd-pd	0	0	0	0	0	0	0	0	0	0	0	0	0
ml-pd-pd	0	0	0	0	0	0	0	0	0	0	0	0	0
pd-ml-pd-pd	0	0	0	0	0	0	0	0	0	0	0	0	0
pl-pf-pr-pd	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.7: Confusion matrix for action recognition on complex sequences using the primitives from single sequences.

Bibliography

[1] 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, no. 6, pp. 799–822, Dec 1994. ix, 2, 15, 16

[2] I. S. Vicente, V. Kyrki, and D. Kragic, "Action recognition and understanding through motor primitives," *Advanced Robotics*, vol. 21, pp. 1687–1707, 2007. ix, 6, 7, 8, 15, 29, 30, 32, 42

[3] S. Ekvall and D. Kragic, "Robot learning from demonstration: A task-level planning approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 223–234, 2008. 1

[4] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004, robot Learning from Demonstration. 1

[5] M. Kaiser and R. Dillmann, "Building elementary robot skills from human demonstration," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, Apr 1996, pp. 2700–2705 vol.3. 1

[6] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: instructive demonstrations, generalization and practice," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems.* New York, NY, USA: ACM, 2003, pp. 241–248. 1

[7] P. Lima and G. Saridis, "Hierarchical reinforcement learning and decision making for intelligent machines," in *in Proceedings of 1994 IEEE Int. Conf. Robotics and Automation*, San Diego, CA, USA, 1994. 1, 2

[8] A. Chella, H. Dindo, and I. Infantino, "Learning high-level manipulative tasks through imitation," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, 6-8 2006, pp. 251–256. 1, 17

[9] A. Billard and M. J. Mataric, "Learning human arm movements by imitation: Evaluation of a biologically inspiredconnectionist architecture," *Robotics and Autonomous Systems*, vol. 37, no. 2-3, pp. 145–160, 2001. 1, 2

[10] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 409–413, 2006, the Social Mechanisms of Robot Programming from Demonstration. 1

[11] 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, no. 6, pp. 799–822, Dec 1994. 1

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

[13] P. Bakker and Y. Kuniyoshi, "Robot see, robot do: an overview of robot imitation," in *In AISB workshop on Learning in Robots and Animals*, 1996. 1, 2

[14] S. Calinon and A. Billard, "Learning of gestures by imitation in a humanoid robot," in *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions*, C. L. Nehaniv and K. Dautenhahn, Eds. Cambridge University Press, 2007. 1

[15] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of honda humanoid robot," in *IEEE International Conference on Robotics and Automation*, 1998, pp. 1321–1326. 1

[16] K. Hirai, "Current and future perspective of honda humamoid robot," in *International Conference on Intelligent Robots and Systems*, vol. 2, 7-11 1997, pp. 500–508. 1

[17] J. Peters and S. Schaal, "Reinforcement learning for parameterized motor primitives," in *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, July 2006, pp. 73–80. 2

[18] A. McGovern and A. G. Barto, "Automatic discovery of subgoals in reinforcement learning using diverse density," in *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, Williamstown, MA, USA, 2001, pp. 361–368. 2, 6, 14, 53

[19] Y. Davidor, *Genetic Algorithms and Robotics.* River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1991. 2

[20] R. A. Brooks and M. J. Mataric, "Real robots, real learning problems," in *Robot Learning*, J. H. Connel and S. Mahadevan, Eds. Kluwer Academic Press, 1993, pp. 193–213. 2

[21] P. Maes, "Behavior-based artificial intelligence," in *Proceedings of the second international conference on From animals to animats 2 : simulation of adaptive behavior.* Cambridge, MA, USA: MIT Press, 1993, pp. 2–10. 2

[22] J. K. Tsotsos, "Behaviorist intelligence and the scaling problem," *Artificial Intelligence*, vol. 75, no. 2, pp. 135–160, 1995. 2

[23] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems 9.* MIT Press, 1997, pp. 1040–1046. 2

[24] G. Hayes and J. Demiris, "A robot controller using learning by imitation," in *Proc. of the Intl. Symp. on Intelligent Robotic Systems*, 1994, pp. 198–204. 2

[25] K. Dautenhahn and C. L. Nehaniv, Eds., *Imitation in animals and artifacts.* Cambridge, MA, USA: MIT Press, 2002. 2

[26] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in cognitive sciences*, vol. 6, no. 11, pp. 481–487, 2002. 2

[27] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng, "Discovering Optimal Imitation Strategies," *Robotics and Autonomous Systems*, vol. 47, pp. 69–77, 2004. 2

[28] 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. 2

[29] J. Kober, B. Mohler, and J. Peters, “Learning perceptual coupling for motor primitives,” in *International Conference on Intelligent Robots and Systems*, 2008, pp. 834–839. 2, 3

[30] W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. van Schie, and H. Bekkering, “Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning,” *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 353–360, 2006, the Social Mechanisms of Robot Programming from Demonstration. 2

[31] G. Guerra-Filho and Y. Aloimonos, “A language for human action,” *Computer*, vol. 40, no. 5, pp. 42–51, 2007. 3, 5, 6, 13

[32] S. Calinon, *Robot Programming by Demonstration: a Probabilistic Approach.* CRC Press, 2009. 3

[33] C. L. Nehaniv and K. Dautenhahn, “Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications,” in *Interdisciplinary Approaches to Robot Learning*, J. Demiris and A. Birk, Eds., vol. 24. World Scientific Press, 2000, pp. 136–161. 3

[34] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders, “Evaluation of robot imitation attempts: comparison of the system’s and the human’s perspectives,” in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction.* ACM, 2006, pp. 134–141. 3, 4

[35] I. S. Vicente, V. Kyrki, D. Kragic, and M. Larsson, “Action recognition and understanding through motor primitives,” *Advanced Robotics*, vol. 21, no. 15, pp. 1687–1707, 2007. 3

[36] A. Fod, M. J. Matarić, and O. C. Jenkins, “Automated derivation of primitives for movement classification,” *Autonomous Robots*, vol. 12, no. 1, pp. 39–54, 2002. 3, 5, 6, 13, 15, 53

[37] D. Del Vecchio, R. M. Murray, and P. Perona, “Decomposition of human motion into dynamics-based primitives with application to drawing tasks,” *Automatica*, vol. 39, no. 12, pp. 2085–2098, 2003. 3, 14

[38] M. J. Mataric, *Sensory-motor primitives as a basis for imitation: linking perception to action and biology to robotics.* Cambridge, MA, USA: MIT Press, 2002. 3

[39] W. Takano and Y. Nakamura, "Integrating whole body motion primitives and natural language for humanoid robots," in *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, daejon, Korea(South), Dec. 2008, pp. 708–713. 3, 14, 53

[40] D. Kulic, H. Imagawa, and Y. Nakamura, "Online acquisition and visualization of motion primitives for humanoid robots," in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, 27 2009-Oct. 2 2009, pp. 1210–1215. 3

[41] D. C. Bentivegna and C. G. Atkeson, "Learning from observation using primitives," in *In IEEE International Conference on Robotics and Automation*, 2001, pp. 1988–1993. 3, 4, 6, 15

[42] M. Dogar, M. Cakmak, E. Ugur, and E. Sahin, "From primitive behaviors to goal-directed behavior using affordances," in *International Conference on Intelligent Robots and Systems*, 29 2007-Nov. 2 2007, pp. 729–734. 3

[43] Z. L. Husz, A. M. Wallace, and P. R. Green, "Human activity recognition with action primitives," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, Los Alamitos, CA, USA, 2007, pp. 330–335. 3

[44] O. C. Jenkins and M. J. Mataric, "Deriving action and behavior primitives from human motion," in *In International Conference on Intelligent Robots and Systems*, 2002, pp. 2551–2556. 3

[45] A. Bobick, "Movement, Activity, and Action: The Role of Knowledge in the Perception of Motion," *Philosophical Trans. Royal Soc. London*, vol. 352, pp. 1257–1265, 1997. 4, 6, 12, 80

[46] R. P. Woods, M. Brass, H. Bekkering, J. C. Mazziotta, M. Iacoboni, and G. Rizzolatti, "Coertical mechanisms of human imitation," *Science*, vol. 286, pp. 2526–2528, 1999. 6, 13

[47] K. Nelissen, G. Luppino, W. Vanduffel, G. Rizzolatti, and G. A. Orban, “Observing Others: Multiple Action Representation in the Frontal Lobe,” *Science*, vol. 310, no. 5746, pp. 332–336, 2005. 6, 13, 81

[48] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, “Premotor cortex and the recognition of motor actions,” *Cognitive Brain Research*, vol. 3, no. 2, pp. 131–141, March 1996. 6, 13

[49] V. Gallese, L. Fadiga, L. Fogassi, and G. Rizzolatti, “Action recognition in the premotor cortex,” *Brain*, vol. 119, no. 2, pp. 593–609, 1996. 6, 13, 81

[50] G. Rizzolatti, L. Fogassi, and V. Gallese, “Neurophysiological Mechanisms Underlying the Understanding and Imitation of Action,” *Nature Reviews*, vol. 2, pp. 661–670, Sept. 2001. 6, 13

[51] ——, “Parietal cortex: from sight to action,” *Current Opinion in Neurobiology*, vol. 7, pp. 562–567, 1997. 6, 13

[52] X. Jiang, P. J. Gannon, K. Emmorey, J. F. Smith, and A. R. Braun, “Symbolic gestures and spoken language are processed by a common neural system,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 49, pp. 20 664–20 669, 2009. 6

[53] O. C. Jenkins, M. J. Mataric, and S. Weber, “Primitive-based movement classification for humanoid imitation,” 2000. 6, 14

[54] D. M. Gavrila, “The visual analysis of human movement: a survey,” *Comput. Vis. Image Underst.*, vol. 73, no. 1, pp. 82–98, 1999. 6

[55] G. R. E. R. M. Lavee, “Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 5, pp. 489–504, 2009. 6

[56] T. Moeslund, A. Hilton, and V. Krueger, “A survey of advances in vision-based human motion capture and analysis,” *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 90–127, 2006. 6

[57] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. U. Chellappa, “Machine recognition of human activities: A survey,” *IEEE Transac-*

tions on Circuits and Systems for Video Technology, vol. 18, no. 11, pp. 1473–1488, Nov. 2008. 6

[58] V. Pavlovic, R. Sharma, and T. Huang, "Visual interpretation of hand gestures for human-computer interaction: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, jul 1997. 7

[59] D. McNeill and E. Levy, *Conceptual representations in language activity and gesture*, ser. Speech, Place and Action : Studies in Deixis and Related Topics. Wiley, 1982. 7

[60] M. W. Alex and R. Psarrou, "Data driven gesture model acquisition using minimum description length," in *In Proc. British Machine Vison Conference*, 2001. 7

[61] A. Gupta and L. Davis, "Objects in action: An approach for combining action understanding and object perception," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8. 7, 16, 81

[62] V. Prasad, V. Kellokompu, and L. Davis, "Ballistic hand movements," in *AMDO*, 2006. 7

[63] Object Action Data Set, "http://www.cvmi.aau.dk/~san/objectActionDataset." 8, 69

[64] D. Herzog, A. Ude, and V. Krueger, "Motion imitation and recognition using parametric hidden markov models," in *Humanoids, IEEE-RAS International Conference on Humanoid Robots*, Daejeon, Korea, South, December 1-3, 2008. 12

[65] S. Tso and K. Liu, "Hidden markov model for intelligent extraction of robot trajectory command from demonstrated trajectories," in *Proceedings of The IEEE International Conference on Industrial Technology, 1996. (ICIT '96),*, Dec 1996, pp. 294–298. 12

[66] G. Hovland, P. Sikka, and B. McCarragher, "Skill acquisition from human demonstration using a hidden markov model," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, Apr 1996, pp. 2706–2711 vol.3. 12

[67] J. Yang, Y. Xu, and C. Chen, "Hidden markov model approach to skill learning and its application to telerobotics," *Robotics and Automation, IEEE Transactions on*, vol. 10, no. 5, pp. 621–631, Oct 1994. 12

[68] H. Jeung, H. T. Shen, and X. Zhou, "Mining trajectory patterns using hidden markov models," *Data Warehousing and Knowledge Discovery*, pp. 470–480, 2007, 9th Iinternational Conference, Data Warehousing and Knowledge Discovery. 12

[69] L. R. Rabiner, "A tutorial on hidden markov models and selected applications inspeech recognition," *Procceeding of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989. 12, 20

[70] S. Eickeler, A. Kosmala, and G. Rigoll, "Hidden markov model based continuous online gesture recognition," in *In Int. Conference on Pattern Recognition*, 1998, pp. 1206–1208. 12

[71] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern Recognition*, 1992, pp. 379–385. 12

[72] A. Chambaz, A. Garivier, and E. Gassiat, "A minimum description length approach to hidden markov models with poisson and gaussian emissions. application to order identification," *Journal of Statistical Planning and Inference*, vol. 139, no. 3, pp. 962–977, 2009. 12

[73] J. L. Sanz-Gonzalez, A. El-Jaroudi, and J. R. Boston, "An algorithm to determine hidden markov model topology," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1996, pp. 3577–3580. 12

[74] 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. 12, 53

[75] C. Stauffer and W. Grimson, "Learning Patterns of Activity Using Real-Time Tracking," *PAMI*, vol. 22, no. 8, pp. 747–757, 2000. 12, 42, 43, 80

[76] N. Robertson and I. Reid, "Behaviour Understanding in Video: A Combined Method," in *Internatinal Conference on Computer Vision*, Beijing, China, Oct 15-21, 2005. 12, 80

[77] K. Yamane, Y. Yamaguchi, and Y. Nakamura, "Human motion database with a binary tree and node transition graphs," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009. 13, 14, 53

[78] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern Recognition*, vol. 19, no. 1, pp. 41–47, 1986. 14

[79] Y. A. Ivanov and A. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 852–872, 2000. 14, 77

[80] G. Hayes and J. Demiris, "A robot controller using learning by imitation," in *Proc. 2nd International Symposium on Intelligent Robotic Systems*, 1994, pp. 198–204. 15

[81] N. Delson and H. West, "Robot programming by human demonstration: Adaptation and inconsistency in constrained motion," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 1996. 15

[82] T. Fukuda, Y. Nakauchi, K. Noguchi, and T. Matsubara, "Time series action support by mobile robot in intelligent environment," in *Proc. IEEE International Conference onRobotics and Automation*, 18-22 2005, pp. 2897–2902. 16

[83] D. Moore, I. Essa, and M. I. Hayes, "Exploiting human actions and object context for recognition tasks," in *CVPR 1999*, vol. 1, 1999, pp. 80–86 vol.1. 16, 81

[84] S. M. Omohundro, "Best-first model merging for dynamic learning and recognition," in *Advances in Neural Information Processing Systems*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds., vol. 4. Morgan Kaufmann Publishers, Inc., 1992, pp. 958–965. 20

[85] A. Stolcke and S. M. Omohundro, "Best-first model merging for hidden Markov model induction," ICSI, Berkeley, CA, 1947 Center Street, Berkeley, CA, Tech. Rep. TR-94-003, 1994. 20

[86] V. Kruger, D. Herzog, Sanmohan, A. Ude, and D. Kragic, "Learning actions from observations," *Robotics Automation Magazine, IEEE*, vol. 17, no. 2, pp. 30–43, june 2010. 29

[87] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, January 1997. 38

[88] C. H. Papadimitriou, *Elements of the theory of computation.* Prentice-Hall, 1982. 42

[89] A. V. Aho and J. D. Ullman, *The theory of parsing, translation, and compiling.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1972. 42

[90] KTH Gesture Data Set, "http://www.nada.kth.se/~danik/gesture_database/." 47, 69

[91] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting Motion Capture Data Into Distinct Behaviors," in *Proceedings of Graphics Interface.* London, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, pp. 185–194. 52

[92] 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. 53

[93] S. J. Julier and J. K. Uhlmann, *General Decentralized Data Fusion With Covariance Intersection (CI)*, D. Hall and J. Llinas, Eds. CRC Press, 2001, vol. Handbook of Data Fusion. 60

[94] Vicon Nexus Motion Capture System, "http://www.vicon.com/products/nexus.html." 68

[95] M. Pomplun and M. Matarić, "Evaluation metrics and results of human arm movement imitation," in *in: Proceedings of the 1st IEEE-RAS International Conference on Humanoid Robotics*, 2000. 68

[96] A. Stolcke, "An efficient probabilistic context-free parsing algorithm that computes prefix probabilities," *Comput. Linguist.*, vol. 21, no. 2, pp. 165–201, 1995. 77

[97] D. Bub N and Michael E. J. Masson, “Gestural knowledge evoked by objects as part of conceptual representations,” *Aphasiology*, vol. 20, pp. 1112–1124, 2006. 81

[98] S. Ullman, *High-Level Vision: Object Recognition and Visual Cognition.* The MIT Press, 1996. 81

[99] D. Kulic and Y. Nakamura, “Scaffolding on-line segmentation of full body human motion parameters,” in *IROS*, 2008, pp. 2860–2866. 82

Appendix A

Grammar for Object-Action-Dataset

Here we give the complete grammar derived for the object action data set we have used in Ch. 5.6. Only a part of the grammar was shown in Fig. 5.19.

```
S->p1 B2C1 [0.125]| p1 B6C3 [0.125]| p1 B11 [0.125]| p1 B13C11 [0.125]|

p1 B17C14 [0.125]| p1 B27 [0.125]| p1 B34 [0.125]| p1 B39C45 [0.125]

B2C45->p2 [1]

B2C1->p2 B3C1 [0.5]

B3C1->p3 [0.5]

B2C1->p2 [0.5]

B3C1->p3 B4C2 [0.5]

B4C2->p4 B5C2 [0.5]

B5C2->p5 [1]

B4C2->p4 [0.5]

B2C2->p2 [1]

B6C1->p6 [1]

B6C3->p6 B8C3 [0.5]

B8C3->p8 B7C4 [0.5]

B7C4->p7 B9C4 [1]

B9C4->p9 [1]

B11->p11 B10 [0.25]
```

```
B10->p10 B7C9 [0.5]

B7C9->p7 B9 [0.5]

B9->p9 [1]

B6->p6 [1]

B8C3->p8 [0.5]

B6C3->p6 [0.5]

B10->p10 B12C9 [0.5]

B12C9->p12 [1]

B6C9->p6 [1]

B13C11->p13 B14C11 [0.5]

B14C11->p14 B15C11 [0.5]

B15C11->p15 [0.5]

B13C11->p13 [0.5]

B14C11->p14 [0.5]

B15C11->p15 B16C12 [0.5]

B16C12->p16 [1]

B13C12->p13 [1]

B17C14->p17 B18C14 [0.5]

B18C14->p18 B19C14 [0.33333]

B19C14->p19 B20C14 [0.5]

B20C14->p20 B22C14 [0.33333]

B22C14->p22 B21C16 [0.5]

B21C16->p21 [0.5]

B20C14->p20 B21C14 [0.33333]

B21C14->p21 [1]

B17C16->p17 [1]

B22C14->p22 [0.5]
```

```
B17C14->p17 [0.5]

B19C14->p19 [0.5]

B20C14->p20 [0.33333]

B21C16->p21 B23C16 [0.5]

B23C16->p23 [1]

B18C14->p18 B24C14 [0.33333]

B24C14->p24 [1]

B18C14->p18 B25C14 [0.33333]

B25C14->p25 B26C70 [0.5]

B26C70->p26 [1]

B17C70->p17 [1]

B27->p27 B28 [0.5]

B28->p28 B29C22 [0.33333]

B29C22->p29 B30C23 [0.33333]

B30C23->p30 B5C24 [0.5]

B5C24->p5 [1]

B28->p28 B32C22 [0.33333]

B32C22->p32 B31C27 [1]

B31C27->p31 B30C27 [0.5]

B30C27->p30 B5 [1]

B5->p5 [1]

B27->p27 B2 [0.5]

B2->p2 [1]

B28->p28 B3C22 [0.33333]

B3C22->p3 [1]

B31C27->p31 B33C27 [0.5]

B33C27->p33 B5C30 [1]
```

```
B5C30->p5 [1]

B29C22->p29 B4C23 [0.33333]

B4C23->p4 [1]

B30C23->p30 B4C24 [0.5]

B4C24->p4 [1]

B29C22->p29 B31C23 [0.33333]

B31C23->p31 B30 [1]

B30->p30 B5 [1]

B34->p34 B35 [0.33333]

B35->p35 B10C34 [0.33333]

B10C34->p10 B8 [0.5]

B8->p8 [1]

B34->p34 B11 [0.33333]

B11->p11 B8C25 [0.25]

B8C25->p8 [1]

B11->p11 B8 [0.25]

B34->p34 B36 [0.33333]

B36->p36 B10C35 [1]

B10C35->p10 B8C35 [0.5]

B8C35->p8 [1]

B35->p35 B6C34 [0.33333]

B6C34->p6 [1]

B10C35->p10 B12C35 [0.5]

B12C35->p12 [1]

B7C9->p7 B12 [0.5]

B12->p12 [1]

B10C34->p10 B7 [0.5]
```

```
B7->p7 B12 [1]

B35->p35 B37C34 [0.33333]

B37C34->p37 B61C36 [0.5]

B61C36->p61 B12C37 [1]

B12C37->p12 [1]

B11->p11 B10C25 [0.25]

B10C25->p10 B7 [1]

B37C34->p37 B7C36 [0.5]

B7C36->p7 B12 [1]

B39C45->p39 B43C45 [0.33333]

B43C45->p43 B42C28 [1]

B42C28->p42 B38C28 [1]

B38C28->p38 B40C28 [0.5]

B40C28->p40 B41 [1]

B41->p41 B16 [1]

B16->p16 [1]

B39C45->p39 B42C45 [0.33333]

B42C45->p42 B38C39 [1]

B38C39->p38 B44C39 [0.33333]

B44C39->p44 B45C1 [0.5]

B45C1->p45 B15 [1]

B15->p15 [1]

B39C45->p39 B46C45 [0.33333]

B46C45->p46 B47 [0.25]

B47->p47 B40C46 [0.5]

B40C46->p40 B62 [0.5]

B62->p62 [1]
```

```
B46C45->p46 B44 [0.25]

B44->p44 B41C50 [0.5]

B41C50->p41 B16 [0.33333]

B47->p47 B45C46 [0.5]

B45C46->p45 B15C46 [1]

B15C46->p15 B16C46 [1]

B16C46->p16 [1]

B40C46->p40 B41 [0.5]

B38C39->p38 B63C39 [0.33333]

B63C39->p63 B49C56 [1]

B49C56->p49 B48C56 [1]

B48C56->p48 [1]

B44->p44 B45C50 [0.5]

B45C50->p45 B41C67 [0.5]

B41C67->p41 B16 [1]

B41C50->p41 B49 [0.33333]

B49->p49 B48 [1]

B48->p48 [1]

B44C39->p44 B41C1 [0.5]

B41C1->p41 B49C41 [1]

B49C41->p49 B48 [1]

B41C50->p41 B48 [0.33333]

B38C28->p38 B44C28 [0.5]

B44C28->p44 B45C28 [1]

B45C28->p45 B15 [1]

B46C45->p46 B38 [0.25]

B38->p38 B40 [1]
```

```
B40->p40 B41 [1]

B38C39->p38 B40C39 [0.33333]

B40C39->p40 B41 [1]

B45C50->p45 B15C67 [0.5]

B15C67->p15 [1]

B46C45->p46 B45 [0.25]

B45->p45 B15C52 [1]

B15C52->p15 [1]

B50C45->p50 B51C45 [0.5]

B51C45->p51 B52C14 [0.5]

B52C14->p52 B64C61 [0.5]

B64C61->p64 B60C64 [1]

B60C64->p60 B53C64 [1]

B53C64->p53 B23C64 [1]

B23C64->p23 [1]

B51C45->p51 B54C14 [0.5]

B54C14->p54 B55C14 [0.33333]

B55C14->p55 B56C66 [0.5]

B56C66->p56 B60C67 [0.33333]

B60C67->p60 B53C68 [1]

B53C68->p53 B23C68 [1]

B23C68->p23 [1]

B17C68->p17 [1]

B50C45->p50 B57C45 [0.5]

B57C45->p57 B18C69 [0.5]

B18C69->p18 B19C69 [0.33333]

B19C69->p19 [1]
```

```
B54C14->p54 B25C14 [0.33333]

B25C14->p25 B20C70 [0.5]

B20C70->p20 [0.5]

B18C69->p18 B25C69 [0.33333]

B25C69->p25 B26C69 [1]

B26C69->p26 [1]

B56C66->p56 B22C67 [0.33333]

B22C67->p22 B21 [1]

B21->p21 [1]

B54C14->p54 B19C14 [0.33333]

B55C14->p55 B58C66 [0.5]

B58C66->p58 B20C71 [1]

B20C71->p20 B22C71 [0.5]

B22C71->p22 B21C71 [1]

B21C71->p21 [1]

B52C14->p52 B55C61 [0.5]

B55C61->p55 B58C72 [1]

B58C72->p58 B20 [1]

B20->p20 B22 [0.5]

B22->p22 B21 [1]

B57C45->p57 B54C69 [0.5]

B54C69->p54 B25C73 [1]

B25C73->p25 B20C73 [1]

B20C73->p20 [1]

B56C66->p56 B59C67 [0.33333]

B59C67->p59 B53C74 [1]

B53C74->p53 B23C74 [1]
```

```
B23C74->p23 [1]

B20C70->p20 B22C70 [0.5]

B22C70->p22 B21C75 [1]

B21C75->p21 [1]

B20->p20 [0.5]

B20C71->p20 [0.5]

B18C69->p18 B24C69 [0.33333]

B24C69->p24 [1]
```