IN *Proc. Int. Conf. on Human-Computer Interaction*, 2009

Partially Observable Markov Decision Process (POMDP) Technologies for Sign Language based Human-Computer Interaction

Sylvie C.W. Ong, David Hsu, Wee Sun Lee, Hanna Kurniawati

School of Computing, National University of Singapore, Computing 1,
13 Computing Drive, Singapore 117417
{song, dyhsu,leews, hannakur}@comp.nus.edu.sg

Abstract. Sign language (SL) recognition modules in human-computer interaction systems need to be both fast and reliable. In cases where multiple sets of features are extracted from the SL data, the recognition system can speed up processing by taking only a subset of extracted features as its input. However, this should not be realised at the expense of a drop in recognition accuracy. By training different recognizers for different subsets of features, we can formulate the problem as the task of planning the sequence of recognizer actions to apply to SL data, while accounting for the trade-off between recognition speed and accuracy. Partially observable Markov decision processes (POMDPs) provide a principled mathematical framework for such planning problems. A POMDP explicitly models the probabilities of observing various outputs from the individual recognizers and thus maintains a probability distribution (or belief) over the set of possible SL input sentences. It then computes a policy that maps every belief to an action. This allows the system to select actions in real-time during online policy execution, adapting its behaviour according to the observations encountered. We illustrate the POMDP approach with a simple sentence recognition problem and show in experiments the advantages of this approach over "fixed action" systems that do not adapt their behaviour in real-time.

Keywords: Sign language recognition, human-computer interaction, planning under uncertainty.

1 Introduction

Ensuring that information technology products and services are accessible to the deaf requires human-computer interaction systems that perform tasks in response to user commands and requests presented in the form of sign language (SL) input. An important requirement for such a system to act correctly is reliable recognition of the SL input. At the same time, due to the real-time nature of the interactions, the recognition process must also be fast.

In general, SL expression involves the hands, facial expression, head movement and body posture. Thus SL recognition systems which extract multiple features from the hands, face and body, and use these features as input to recognizers have been built in recent years (for e.g., [1]). While possibly providing better recognition accuracy, executing recognizers with multiple input features may be more time-consuming than executing recognizers with fewer input features. Conversely, not all of the features extracted from a SL expression are necessary in all circumstances for disambiguating signs. For example, some signs may be easily disambiguated with just the features from the dominant hand, without the need for features of the non-dominant hand and facial expression. Ideally, a SL recognition system operating under time constraints would process only as many of the extracted features as are necessary to reliably recognize the SL input. We can think of the system as having access to a set of recognizers, each recognizer taking a different set of extracted features as its input. The system's task is to sequentially (and selectively) apply the available recognizers to the SL input, until it is sufficiently confident of its interpretation. At this point it submits its interpretation of the SL input as the final recognition result.

The task of deciding which recognizers to apply in sequence and when to submit the final recognition result can be formulated as a planning problem for an intelligent agent where the agent's observations are the output of the available recognizers. Since these outputs are generally not a hundred percent accurate, the agent will always have a degree of uncertainty about its interpretation of the SL input. At the same time, the agent must take into account the trade-off between the computational cost (time) of applying the available recognizers and the confidence level of the recognition result it finally submits. We propose to formulate this planning problem as a partially observable Markov decision process (POMDP) which explicitly models both the agent's uncertainty and the desired trade-off.

POMDP is a powerful framework for planning under uncertainty [2]. It has a solid mathematical foundation, and has been applied to human-computer interaction applications such as spoken dialogue systems [3]. The most closely related work to ours is [4] where visual operators are (repeatedly) applied to find the color or shape of objects in a scene. Our work differs in that SL input data is sequential and dynamic in nature – we cannot repeatedly acquire data and apply the same recognizer to the same sentence utterance, unlike in [4], where a visual operator can repeatedly acquire data from the same static scene. In our input data, signs appear sequentially in a sentence, hence any information we have about the sentence model can be advantageously used when determining the final recognition result.

In the next section we give a short background on POMDPs, followed by a presentation of our proposed model in Section 3.

2 POMDPs

A POMDP models an agent taking a sequence of actions under uncertainty to achieve a goal. Formally a POMDP is specified as a tuple (S, A, O, T, Z, R), where S is a set of states, A is a set of actions, and O is a set of observations. In each time step, the agent lies in some state s in S; it takes some action a in A and moves from s to a new state s'. Due to the uncertainty in action, the end state s' is modeled as a conditional probability function $T(s, a, s') = p(s' \mid s, a)$, which gives the probability that the agent transits to s', after taking action a from state s. The agent then makes an observation to gather information on its state. Due to the uncertainty in observation, the observation result o in O is also modeled as a conditional probability function $Z(s, a, o) = p(o \mid s, a)$. In each step, the agent receives a real-value reward $R(s,a)$, if it takes action a from state s, and the agent's goal is to maximize its expected total reward by choosing a suitable sequence of actions. We control the agent's behavior by defining a suitable reward function.

For a POMDP, planning means computing an optimal policy that maximizes the expected total reward. Since the agent's state is partially observable and not known exactly, we rely on the concept of a belief b, which is a probability distribution over S. A POMDP policy π maps a belief b to a prescribed action a in A. Policy computation is usually performed offline.

Given a policy, the control of the agent's actions (i.e., policy execution) is performed online in real time. It consists of two steps executed repeatedly. The first step is action selection based on the policy π. The second step is belief estimation. After the agent takes an action a and receives an observation o, its new belief b' is given by

$$b'(s') = k\, Z(s', a, o)\, \Sigma_s T(s, a, s')\, b(s),$$

where k is a normalizing constant. The process then repeats.

3 Problem Formulation

As an illustration of how SL recognition in human-computer interaction systems can be formulated as a POMDP, we present a problem where the SL input consists of 2-word sentences made up of signs from a vocabulary of 3 signs: {sign-*0*, sign-*1*, sign-*2*}. We denote the word[1] positions in each sentence as position-*a* (first word) and position-*b* (second word). The sentences have been explicitly segmented into individual words (for example, by using the approach in [5]), and the features of the dominant and non-dominant hands have been extracted[2]. The system has available to it a single-hand recognizer which takes features from the dominant hand as its input; and a double-hand recognizer which takes features from both the dominant and non-dominant hands as its input. The recognizers are assumed to have been previously trained on representative data such that when presented with new (previously unseen) data, each recognizer will output one of the signs in the vocabulary as its recognition result.

The recognizer actions available to the system are:

1) *apply_ag*: apply the sin**g**le-hand recognizer to the the word at position-***a***,
2) *apply_ad*: apply the **d**ouble-hand recognizer to the the word at position-***a***,

[1] We use the terms word and sign interchangeably in this paper.

[2] Our focus is on planning the steps for applying existing recognizers to input data, and not on designing/training the individual recognizers. Hence, we abstract over details about the method of data acquisition, and feature extraction/selection.

3) *apply_bg*: apply the sin**g**le-hand recognizer to the the word at position-***b***,
4) *apply_bd*: apply the **d**ouble-hand recognizer to the the word at position-***b***.
The system submits its final recognition result by executing actions:
1) *submit_a0*: submit sign-***0*** as the recognized sign at position-***a***;
2) *submit_a1*: submit sign-***1*** as the recognized sign at position-***a***;
3) *submit_a2*: submit sign-***2*** as the recognized sign at position-***a***;
4) *submit_b0*: submit sign-***0*** as the recognized sign at position-***b***;
5) *submit_b1*: submit sign-***1*** as the recognized sign at position-***b***;
6) *submit_b2*: submit sign-***2*** as the recognized sign at position-***b.***

Once the sign id at both word positions have been submitted, the system is considered to have submitted its final recognition result for the SL input.

Below we describe the parameters in the POMDP tuple, (S, A, O, T, Z, R), for modeling the problem above.

- $S : S_s \times S_{ag} \times S_{ad} \times S_{bg} \times S_{bd} \times S_{at} \times S_{bt}$, the set of states, is a cross product of seven subspaces. The overall state s can be denoted as $s_s\, s_{ag}\, s_{ad}\, s_{bg}\, s_{bd}\, s_{at}\, s_{bt}$.

  S_s : {sxy; x in {0,1,2}, y in {0,1,2}} represents the sign ids in the sentence, where sxy denotes sign-x in position-a and sign-y in position-b. In general, given a vocabulary of size N_v and sentences of length N_s, not all $(N_v)^{Ns}$ combinations of sign sequences are valid sentences in the language. The POMDP model should reflect this by having $|S_s| < 3^2$. The set S_s of valid sentences constitutes the sentence model for the SL recognition task.

  S_{ag} : {f_{ag}, t_{ag}}, S_{ad} : {f_{ad}, t_{ad}}, S_{bg} : {f_{bg}, t_{bg}}, S_{bd} : {f_{bd}, t_{bd}} are indicators which represent whether the agent has applied the available recognizers to the words in the sentence. $s_{ag} = f_{ag}$ (i.e. *false*) indicates that the single-hand recognizer has not been applied to the word at position-a. $s_{ag} = t_{ag}$ (i.e. *true*) indicates that it has. Similarly, $s_{ad} = f_{ad}/t_{ad}$ indicates whether the double-hand recognizer has been applied at position-a. $s_{bg} = f_{bg}/t_{bg}$ and $s_{bd} = f_{bd}/t_{bd}$, respectively, indicate whether the single and double-hand recognizers have been applied to the word at position-b.

  S_{at} : {f_{at}, t_{at}}, S_{bt} : {f_{bt}, t_{bt}} are indicators which represent whether the agent has submitted the sign ids in the sentence. $s_{at} = f_{at}/t_{at}$ indicates whether the agent has submitted the sign id at position-a. $s_{bt} = f_{bt}/t_{bt}$ indicates whether the agent has submitted the sign id at position-b.

- A : {*submit_a0*, *submit_a1*, *submit_a2*, *submit_b0*, *submit_b1*, *submit_b2*, *apply_ag*, *apply_ad*, *apply_bg*, *apply_bd*} is the set of actions. Executing any of the three *submit_a* actions leads to s_{at} being set to t_{at}. Executing any of the three *submit_b* actions leads to s_{bt} being set to t_{bt}. Action *apply_ag* leads to s_{ag} being set to t_{ag} . Similarly for *apply_ad* and s_{ad}. Actions *apply_bg* and *apply_bd* have analogous effects on s_{bg} and s_{bd}, respectively.

- O : {*o0*, *o1*, *o2*} is the set of observations, i.e. the outputs of the recognizers.

- $T : S \times A \times S \rightarrow [0, 1]$ is the state transition function. This function reflects the fact that all of the actions have no effect on the state s_s of the subspace S_s since the sign ids in a sentence do not change as a result of the agent's actions. The indicator states, s_{ag} , s_{ad} , s_{bg} , s_{bd} , s_{at} and s_{bt}, are set to *true* when the corresponding actions are taken, as described above. Once set to *true*, an indicator remains as *true*.

- $Z : S \times A \times O \rightarrow [0, 1]$ is the observation function. For actions *apply_ag*, *apply_ad*, *apply_bg* and *apply_bd*, the function reflects the accuracy of the single-hand and double-hand recognizers (refer to Section 4.1 for details). For *submit* actions, the observation function is set to a uniform distribution.

- $R : S \times A \rightarrow$ real-valued rewards.
  The goal of the overall recognition system is to correctly recognize the words in the sentence. Hence, the reward function assigns a positive (negative) reward to the agent when it submits the right (wrong) sign ids. However, the agent is not allowed to submit the sign id at the same word position more than once. To enforce this, a very large penalty is set in the reward function for repeat submissions. The exact value of the penalty is not critical – setting it a few magnitudes bigger than the reward given for correct/incorrect sign id submission is quite sufficient. The reward function for *submit_a0* is shown below, the functions for *submit_a1*, *submit_a2*, *submit_b0*, *submit_b1* and *submit_b2* are analogous.

$$R(s, submit\_a0) = +/-10\alpha \text{ ; for all } s \text{ where } s_{at} \text{ is } f_{at}$$
$$R(s, submit\_a0) = -10000 \text{ ; for all } s \text{ where } s_{at} \text{ is } t_{at}$$

For actions that apply recognizers to words in the sentence, the reward values reflect the computational cost (time) of the recognizers. Here, we make a simplifying assumption by setting the cost of applying the double-hand recognizer to twice that of the single-hand recognizer. This is a reasonable assumption when there are twice as many features to process in the double-hand recognizer.

$$R(s, apply\_ag) = -1 \text{ ; for all } s \text{ where } s_{ag} \text{ is } f_{ag}$$
$$R(s, apply\_bg) = -1 \text{ ; for all } s \text{ where } s_{bg} \text{ is } f_{bg}$$
$$R(s, apply\_ad) = -2 \text{ ; for all } s \text{ where } s_{ad} \text{ is } f_{ad}$$
$$R(s, apply\_bd) = -2 \text{ ; for all } s \text{ where } s_{bd} \text{ is } f_{bd}$$

Applying a recognizer multiple times to the same word position violates the POMDP assumption that observations are independent. (The features from the word remain the same in each application, hence the recognizer outputs would be identical). To prevent a recognizer from being applied more than once to each word position, a very large penalty is set in the reward function.

$$R(s, apply\_ag) = -10000 \text{ ; for all } s \text{ where } s_{ag} \text{ is } t_{ag}$$
$$R(s, apply\_bg) = -10000 \text{ ; for all } s \text{ where } s_{bg} \text{ is } t_{bg}$$
$$R(s, apply\_ad) = -10000 \text{; for all } s \text{ where } s_{ad} \text{ is } t_{ad}$$
$$R(s, apply\_bd) = -10000 \text{ ; for all } s \text{ where } s_{bd} \text{ is } t_{bd}$$

The parameter α in the reward function for *submit* actions allows a trade-off between the computational cost of applying recognizers and the confidence level when submitting the sign ids.

4 Experimental Setup and Results

In this section, we describe the experiment setup and results on the POMDP model. The sentence model and observation function for the POMDP are defined in Section 4.1. Sentence recognition results and some example sequences of agent actions during policy execution are presented in the following sections.

4.1 POMDP Parameters

The experiments below are a preliminary investigation into the feasibility of using POMDPs for SL input recognition. As such, the experiments are simulations of how such a POMDP would perform with real-world data. Below we describe how the sentence model and observation function were defined for the simulation experiments, as well as how they could be learned from data when applied to real-world tasks.

Sentence Model. For a particular recognition task, the set of states of the subspace S_s should be defined to be the set of valid sentences that can occur with the given vocabulary. For the experiments below, we randomly chose 5 sentences from a total of 9 possible combinations of 2-word sentences : {*s00*, *s10*, *s21*, *s02*, *s12*}. (Admittedly, 2-word sentences consisting of the same sign repeated twice (as in *s00*) are rare, however this does occur in longer sentences, hence we didn't constrain the sentences to exclude repeated signs.)

Observation Function. For a particular recognition task, the observation function should reflect the accuracy of the available recognizers. This can be obtained from supervised training by presenting multiple instances of each sign in the vocabulary to each recognizer and estimating the probability of its outputs. For the experiments below, we set some reasonable values for the accuracy of the single-hand and double-hand recognizers, as shown in Table 1. We assumed that sign-*0* and sign-*1* are signs made with the dominant hand only while sign-*2* is made with both hands. Hence when the sign is sign-*0* or sign-*1*, the single-hand recognizer outputs the correct result with a higher probability as compared to the double-hand recognizer. The opposite is true for sign-*2*.

Table 1. Sign accuracy of single-hand and double-hand recognizers.

Actual sign	Probability of single-hand recognizer output		
	sign-*0*	sign-*1*	sign-*2*
sign-*0*	0.818	0.091	0.091
sign-*1*	0.091	0.818	0.091
sign-*2*	0.1	0.1	0.8

Actual sign	Probability of double-hand recognizer output		
	sign-*0*	sign-*1*	sign-*2*
sign-*0*	0.727	0.182	0.091
sign-*1*	0.182	0.727	0.091
sign-*2*	0.091	0.091	0.818

The observation functions for the POMDP were set according to the values in Table 1. For example,

$Z(s_s = s02, a = apply\_ag, o = (\text{sign-}0, \text{sign-}0, \text{sign-}2)) =$
$p(o = (\text{sign-}0, \text{sign-}1, \text{sign-}2) \mid s_s = s02, a = apply\_ag) = (0.818, 0.091, 0.091)$.

Note that the recognizer accuracy is not affected by which word position it is applied at, hence,

$Z(s_s = s10, a = apply\_bg, o = (\text{sign-}0, \text{sign-}0, \text{sign-}2)) =$
$p(o = (\text{sign-}0, \text{sign-}1, \text{sign-}2) \mid s_s = s10, a = apply\_bg) = (0.818, 0.091, 0.091)$.

4.2 Results

Experiments were performed on a PC with a 2.66GHz Intel processor and 2GB memory. A POMDP model was specified as described in Section 3 and Section 4.1, and with the parameter α in the reward function set to 10. We first ran the APPL solver [6] which implements SARSOP [7], a leading POMDP algorithm, on the model. We then ran a total of 5×10^6 simulation trials, to measure the performance of the solver's output policy. In each trial, the actual sentence is simulated with equal probability for each of the sentences in the set S_s. The outputs from the single-hand and double-hand recognizers are simulated with probabilities in accordance with Table 1. The actual sentence is not known to the agent executing the policy, only the recognizer outputs are observed.

The average sign accuracy obtained from the trials is 88% and average sentence accuracy 80% (Table 2). The average cost of applying the recognizers in each trial is 4.53.

Table 2. Overall sign and sentence recognition accuracy.

	POMDP model	Single-hand-recognizer-only	Double-hand-recognizer-only
Av. sign accuracy	88%	87%	83%
Av.sent. accuracy	80%	75%	68%
Cost of applying recognizers	4.53	2	4

As a baseline comparison, we estimated (based on the probability values from Table 1) the overall sign and sentence accuracy for a system which uses only one of the two available recognizers. For a system which uses the single-hand recognizer only and assumes that its output is always correct, it would, for example, correctly recognize the sentence *s02*, 65.4% of the time ($0.654 = 0.818 \times 0.80 = p(o = \text{sign-}0 \mid s_s = s02, a = apply\_ag) \times p(o = \text{sign-}2 \mid s_s = s02, a = apply\_bg)$).

We made adjustments to the probability values from Table 1 by taking into account the sentence model (this increases the calculated recognition rate since the model restricts the possible sequences of signs that could appear) and obtained the (estimated) overall accuracy rates for a single-hand-recognizer-only system. Similar calculations were made to estimate overall accuracy rates for a double-hand-recognizer-only system, as shown in Table 2.

The average cost (4.53) of applying recognizers in the POMDP model indicates that the policy *selectively* applies the available recognizers in the recognition task. It does not blindly apply both recognizers to each of the word positions in the sentence (which would entail a cost of 6), nor does it apply either only the single-hand or the double-hand recognizers. It achieves a higher recognition rate than both the single-hand-recognizer-only and the double-hand-recognizer-only systems with a slightly higher cost than both systems.

Note that we did not make a comparison with the recognition accuracy of a system that always uses both the available recognizers. Evidently, such a system would give a higher recognition accuracy. However, our goal is to ellicit a system that acts intelligently in applying recognizers when there is insufficient time to apply all the available recognizers at every word position, for every SL input. In the next section, we examine some examples of policy execution in the simulation trials to see how the POMDP selectively applies recognizers.

4.3 Examples of Policy Execution

Tables 3 and 4 show two examples of policy execution during the simulation trials. In terms of time steps, the tables should be read from left to right and from top to bottom.

Table 3. Policy execution example 1: actual sentence is *s02*, sentence recognized successfully.

Time	Belief over S_s	Action	Obs.
0	(0.2, 0.2, 0.2, 0.2, 0.2)	*apply_bg*	*o2*
1	(0.049, 0.049, 0.049, 0.427, 0.427)	*apply_bd*	*o2*
2	(0.006, 0.006, 0.006, 0.491, 0.491)	*submit_b2*	-
3	(0.006, 0.006, 0.006, 0.491, 0.491)	*apply_ag*	*o0*
4	(0.011, 0.001, 0.001, 0.888, 0.099)	*submit_a0*	-

In example 1 (Table 3), the system starts with equiprobable belief over the set of possible sentences, S_s : {*s00*, *s10*, *s21*, *s02*, *s12*}. The agent executes the first action *apply_bg*, applying the single-hand recognizer to the word at position-*b*. It receives observation *o2* and updates its belief over S_s. It subsequently executes action *apply_bd*, which applies the double-hand recognizer to the same word, receives another observation and updates its belief again. At this point, its belief indicates that the sentence is most likely to be *s02* or *s12*, both of which have sign-*2* in position-*b*. It thus submits sign-*2* as the recognized sign at position-*b*. Its next action is to execute *apply_ag* which results in observing *o0*. This resolves the sentence as *s02* and the system submits sign-*0* as the recognized sign at position-*a*.

Table 4. Policy execution example 2: actual sentence is *s21*, sentence recognized successfully.

Time	Belief over S_s	Action	Obs.
0	(0.2, 0.2, 0.2, 0.2, 0.2)	*apply_bg*	*o1*
1	(0.076, 0.076, 0.682, 0.083, 0.083)	*apply_ad*	*o2*
2	(0.012, 0.012, 0.951, 0.013, 0.013)	*submit_a2*	-
3	(0.012, 0.012, 0.951, 0.013, 0.013)	*submit_b1*	-

In example 2 (Table 4), the agent's first action is again to execute *apply_bg* which applies the single-hand recognizer at position-*b*. It receives observation *o1* and updates its belief over S_s. It subsequently executes action *apply_ad* which applies the double-hand recognizer to the word at position-*a*, receives another observation and updates its belief again. At this point, its belief indicates a very high probability for the sentence *s21* – a sufficient confidence level for the agent to submit the sign ids at both word positions without further recognizer actions.

The two examples above show that the agent adapts its actions according to the observations it receives (and its belief over S_s) during policy execution. Both the sequence and number of times it applies the available recognizers vary between trials as it adapts its behaviour to the observations it receives.

5 Conclusions and Future Work

We have shown how the SL recognition task in human-computer interaction systems can be formulated as a POMDP problem and how the solution to the problem allows for real-time adaptive behaviour. We illustrated the POMDP approach on a simple 2-word sentence recognition problem and experimentally showed that the computed POMDP policy performs recognizer actions only as many times as is necessary for sufficent sign disambiguation, and that it adaptively selects which recognizer to apply at each sequential step.

Admittedly however, the illustrated problem as we have presented it here is relatively simple and our experiments so far are just a preliminary investigation into the feasibility of using POMDPs in the SL recognition domain. As such there are many ways in which this work can be extended. We discuss some of these below and give some indications of how to tackle the resultant issues.

With the current 3-sign vocabulary and 2-word sentence recognition problem, we can experiment with different settings of the parameter α in the reward function (currently set to 10 in our experiments) and different sentence models, and measure the accompanying changes (if any) in the sign and sentence accuracies. So far, all our experiments have been done on simulation data. It would be highly informative to experiment with learning the sentence model and POMDP observation function from training examples of real-world data and executing the computed POMDP policy on real-world test data. Such experiments would be a big step in further validating the feasibility of the POMDP approach.

A major assumption that we have made in our problem formulation is the availability of individual words which have been explicitly segmented from the SL input sentence. Although there have been many previous works on explicit segmentation of SL sentences (see [5] for a short review), the approaches generally either rely on tuning threshold values or do not generalize well across different sets of sentences and/or different signers. One possible approach to doing away with explicit segmentation is to use one of the available recognizers to perform implicit segmentation on sentences. The segmented results are then used for the other recognizers. For example, a hidden Markov model network could be learned from training data consisting of features from the dominant hand, and then used to segment test sentences by applying the Viterbi algorithm. However, this forces the recognition system to always apply the recognizer that performs the segmentation. An ideal system would integrate implicit sentence segmentation with selective application of recognizers in a unified POMDP model. A promising direction to consider for the structure of such a model is hierarchical action and state spaces [8].

As we extend the current model to larger vocabularies and longer sentences, there are two main issues that need to be considered. Firstly, it will become infeasible to enumerate all possible sentences in the sentence model. Instead, a bigram model might be used instead. This would require restructuring the subspace $|S_s|$. Secondly, the difficulty of solving the corresponding POMDP model will increase very quickly with vocabulary size and sentence length. To get an idea of how quickly, we note that the number of subspaces which form part of the cross product that make up the overall state space S, increases linearly with vocabulary size and sentence length. The number of states, $|S|$, in turn increases exponentially with the number of subspaces in the model. And finally the difficulty of solving a POMDP model increases exponentially with $|S|$. To solve the scalability problem, we might consider approaches that compress the state space, such as in [3].

References

1. U. von Agris, J, Zieren, U. Canzler, B. Bauer, K.-F. Kraiss: Recent Developments in Visual Sign Language Recognition. In: Univ. Access Inf. Soc., vol. 6, pp. 323 --362 (2008)
2. L. Kaelbling, M. Littman, A. Cassandra: Planning and Acting in Partially Observable Stochastic Domains. In: Artificial Intelligence, vol. 101, no. 1–2, pp. 99–134 (1998)
3. J. Williams, S. Young: Scaling POMDPs for Spoken Dialogue Management. In: IEEE Trans. on Audio, Speech & Language Processing, vol. 17, no. 7 (2007)
4. M. Sridharan, J. Wyatt, R. Dearden: HiPPo: Hierarchical POMDPs for Planning Information Processing and Sensing Actions on a Robot. In: Intl. Conf. on Automated Planning & Scheduling (2008)
5. W.W. Kong, S. Ranganath: Automatic Hand Trajectory Segmentation and Phoneme Transcription for Sign Language. In: IEEE Conf. Automatic Face & Gesture Recog. (2008)
6. Available at http://motion.comp.nus.edu.sg/software/appl/appl.html.
7. H. Kurniawati, D. Hsu, W. Lee: SARSOP: Efficient Point-based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In: Proc. RSS (2008)
8. J. Pineau, S. Thrun: High-level Robot Behaviour Control Using POMDPs. In: AAAI (2002)