Learning Dynamic Audio/Visual Mapping with Input-Output Hidden Markov Models

Yan Li, Heung-Yueng Shum
Microsoft Research, China
{yli, hshum}@microsoft.com

Abstract

In this paper we formulate the problem of synthesizing facial animation from an input audio sequence (a.k.a. video rewrite, voice puppetry) as dynamic audio/visual mapping. We propose that audio/visual mapping should be modeled with an input-output hidden Markov model, or IOHMM. *An IOHMM is an HMM for which the emission and transition probabilities are conditional on the input sequence. We train IOHMMs using the expectation-maximization (EM) algorithm with a novel architecture to explicitly model the relationship between each transition probability and the input using a neural network. Given an input sequence, an output sequence is synthesized by a maximum likelihood estimation. Experimental results demonstrate that IOHMMs can generate good-quality and natural facial animation sequences from input audio.*

1. Introduction

Dynamic audio/visual mapping (or vocal/facial mapping) has recently received much attention as a powerful alternative to traditional facial animation techniques [6, 5, 7, 4, 9]. Instead of directly animating facial expression, a sequence of audio is used to drive the facial motion. While voice is generated from the vocal cords, and facial expressions are formed from facial skin and muscles, there exists a great deal of mutual information between audio and visual signals. Representative projects on learning dynamic audio/visual mappings, recently from the graphics community, include video rewrite [5] and voice puppetry [4]. A good survey on the importance and difficulties of audio/visual mapping can be found in Section 2 of [4].

As a powerful approach to model time-series data in the state-space, Hidden Markov Models [10, 12] have been adopted in many synthesis applications [5, 4]. Previous approaches assume that either the input or the output can be modeled by a Hidden Markov Model (HMM). For example, the "video rewrite" technique recognizes different phonemes from the input audio signal. Animation is generated by re-ordering the captured video frames which share similar phonemes as in the training video. On the other hand, the "voice puppetry" technique trains an HMM model for the visual signal. A remapping process is employed to give each state a dual mapping into both audio and visual signals.

There is a reason why the cumbersome remapping and analysis steps are needed in voice puppetry. Although HMM has been shown to be a powerful tool to model the dynamic process, it is quite sub-optimal for synthesis. Traditionally, for recognition, an HMM aims to model the dynamics of one kind of signal. For synthesis, we need to explore the mapping relationship between different signals, each of which might have a different probabilistic model. Moreover, the model parameters in conventional HMMs are fixed after training, which result in a homogeneous Markov chain. On the other hand, when our observations are two related input and output sequences, and the output sequence conditionally depends on the input sequence, the expected model should be inhomogeneous, or have the ability of adapting to the input.

In this paper, we propose that dynamic audio/visual mapping should be learnt by an input-output hidden Markov model, or *IOHMM*. IOHMM, a.k.a. conditional HMM originally introduced by Bengio [3, 1] for sequence processing, can be stated as follows: *An IOHMM is an HMM for which the emission and transition distributions are conditional on the input sequence.* Specifically in this paper, we present novel algorithms to tackle the following two problems:

- learning IOHMMs for dynamic vocal/facial mapping from synchronized audio and visual signals;
- synthesizing facial expressions from input audio and the learnt IOHMMs.

The remainder of this paper is organized as follows. The IOHMM model is introduced in Section 2. We explain why HMM needs to be augmented to IOHMM for the synthesis task. The audio/visual mapping is studied in Section 3. Experimental results are presented in Section 4. Finally we conclude our paper in Section 5.

2. IOHMM for synthesis

2.1. HMM

HMMs are statistical models of sequential data that have been used successfully in many applications, e.g., speech recognition. A Bayesian network [11] representing graphically the independence assumptions of an HMM is shown in Figure 1(a). The relationship between the observed (output) sequence $y_1^T = (y_1, y_2, ..., y_T)$ and the hidden state sequence $q_1^T = (q_1, q_2, ..., q_T)$ satisfies the conditional first-order independence assumptions [12].

The conventional HMM can be extended for the purpose of dynamic input/output mapping. The Bayesian network

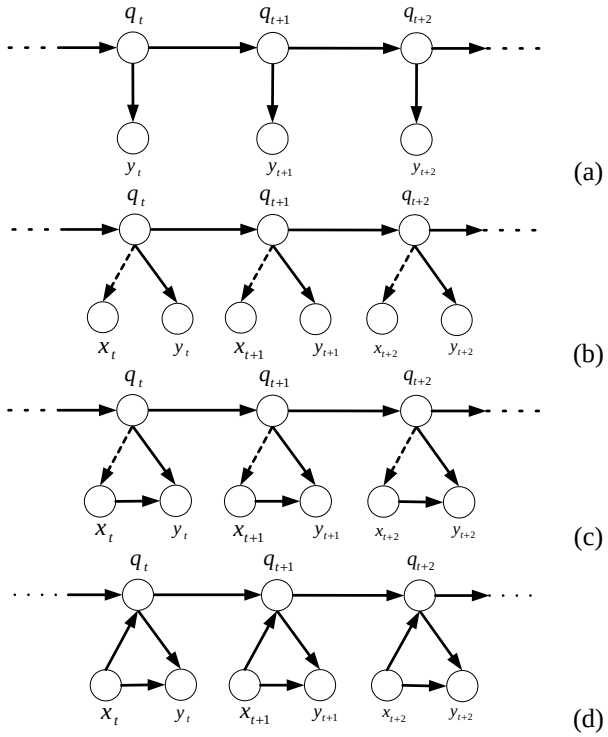


Figure 1. Bayesian networks for several hidden Markov models. (a) Conventional HMM where no input is in the network. (b) HMM remapped (with dotted lines) to the input sequence as well. (c) Remapped HMM (b) plus direct connection between input and output. (d) Input-output HMM. Dotted lines from q_t to x_t in (b)(c) indicate a remapping process, not a causal effect. Solid line from x_t to q_t in (d) shows that q_t and transition from q_t to q_{t+1} are conditional to x_t.

shown in Figure 1(b) illustrates that the learnt HMM from the observed output sequence can be remapped to the input sequence (in dotted lines). This is exactly the approach adopted in [4] for synthesizing facial gesture from voice. Although compelling results have been shown, this technique has two problems.

First, at the synthesis step, vocal signals are only used to generate the most likely state sequence. Animation is generated by solving a global trajectory in the visual state space, which obliterates the relationship between vocal and visual signals. This problem can be partially addressed by enforcing the local input/output relationship, i.e., adding a direct arc from input to output, as shown in Figure 1(c). At the synthesis stage, from the state sequence and input signal, we can generate the output sequence with the help of the local input/output mapping. Introducing a local model into HMM is necessary for the synthesis problem because we expect to obtain a continuous output, not to classify the input into a specific state (as expected in a recognition problem). If we have no prior knowledge of the relationship between input and output, the local mapping model can be obtained by some regression method such as a neural network. Generally speaking, a more explicit and compact distribution of the output can be learnt by introducing some prior knowledge or assumptions about the input and output signals.

Second, and more significantly, a remapping process is required to map the occupancy matrix (obtained from the HMM model for the output sequence) to the synchronized input so that each state has a dual mapping with both input and output. The underlying assumption made in the remapping process is that the input sequence shares the dynamic behavior exhibited in the HMM trained from the output. As a result, the learnt model is homogeneous for all input sequences.

These problems are addressed in the Bayesian network shown in Figure 1(d) where input and output are put together for training. The model proposed in Figure 1(d) is called IOHMM or conditional HMM because the model configuration is conditionally dependent on the input sequence. This is illustrated by the arc from the input to the state ($x_t q_t$) in Figure 1(d) having a direction reverse from that in Figure 1(b). It indicates the causal effect from the input to the output.

2.2. IOHMM

The main difference between standard HMMs and IOHMMs, is that the former represents the distribution $P(y_1^T)$ of output sequences, whereas the latter represents the conditional distribution $P(y_1^T|x_1^T)$ of the output sequence given the input sequence $x_1^T = (x_1, x_2, ..., x_T)$. IOHMMs are trained by maximizing the conditional likelihood $P(y_1^T|x_1^T)$. This is a supervised learning problem since the output y_1^T plays the role of a desired output in response to the input x_1^T. The Bayesian network for HMMs (Figure 1(a)) can be obtained by simply removing the input nodes and arcs from the IOHMM in Figure 1(d).

The arc from x_t to y_t in Figure 1(d) indicates that IOHMMs represent a conditional distribution of an (desired) output sequence when an (observed) input sequence is given. And the arc from x_t to q_t implies that in IOHMM, transition probabilities are conditional on the input and thus depend on time, resulting in inhomogeneous Markov chains. In comparison, standard HMMs are based on homogeneous Markov chains. Therefore, IOHMMs are better suited for learning to represent long-range context than HMMs. These properties of IOHMMs make them more suitable than traditional HMM for synthesis.

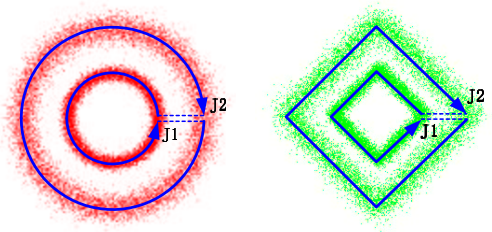
2.3. An example

We illustrate the difference between HMMs and IOHMMs in training and synthesis from a toy problem below.

2.3.1 Problem description

The input and output sequences shown in Figures 2(a) and (b) have the following properties:

- At any time instant, the input signal is assumed to move along one of the two concentric circles, clockwise along the outer circle, but counterclockwise along the inner one, indicated by circles with arrows. Gaussian noise proportional to the circle radius is further added to the point positions.



(a) (b)

Figure 2. A toy problem to map circles to squares. (a) Distribution of the input signal; (b) distribution of the output signal. Solid lines and curves are the paths in which the data move; dots are the actual samples perturbed by noise.

- The output signal is synchronous to the input signal. The output signal moves along one of the two corresponding diamonds, clockwise along the outer diamond, counterclockwise along the inner one.
- The input can only jump to adjacent circles from point J_1 to J_2, or vice versa, as shown in Figure 2(a). The output jumps accordingly.

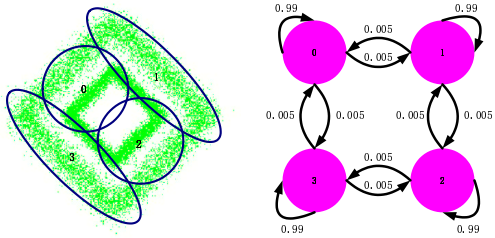
Our objective is to learn the dynamic mapping between the input and output. Furthermore, given a new input sequence, we would like to synthesize the most likely output sequence that best fits the learnt model.

2.3.2 HMM

To simplify the training problem, we assume four states in our HMM. It has been shown in [4] that the minimum entropy principle can be used to learn the number of states and the structure of HMMs. Applying the standard HMM to the output sequence, we obtain four states shown in Figure 3(a), each of which represents the data distribution along a specific side of the diamond. HMMs (with remapping from the output to the input as shown in Figure 1(b)) are inappropriate for synthesis because of the following two reasons.

First, HMMs do not represent any dynamics at a finer scale than a state. This causes blurring and muting of the output, and eliminates the fine-scale noise and texture that are expected for synthesis. For example, we might be able to recognize that the output is in state 0 in Figure 3(a), but we cannot determine if it is on the inner or outer diamond. Although the expressive power of the model can be ameliorated by adding more states, e.g., using 8 or 16 states for this toy problem, the complexity of the state machine increases (imagine that we have 100 concentric diamonds with different sizes for the output).

Second, as shown in Figure 3(b), the transition probabilities of an HMM are fixed after training. A transition probability represents an average transiting behavior between two states. The amount of uncertainty of the transition is, however, not modeled. Therefore, an HMM cannot distinguish whether a transition probability is highly volatile or fixed. With the fixed transition probability matrix, the HMM in Figure 3(a) cannot synthesize the correct change from one diamond boundary to another. To apply HMMs for synthesis, the emission and transition probabilities must depend on the input.



(a) (b)

Figure 3. The learnt HMM from the output sequence of Figure 2(b). (a) Four states of HMM; (b) fixed transition between HMM states. Solid lines in (a) indicate that model parameters are fully specified. Different shapes at the four states represent different distributions.

2.3.3 IOHMM

For IOHMM, we again use four states to model the output data distribution, as shown in Figure 4(a). Dotted lines for the states in Figure 4(a) indicate that the specific formulations of the emission probability and the transition probability are not fully determined unless an input is given.

Training an IOHMM is much more complex than training an HMM because the emission and transition probabilities are conditional on the input. In particular, for each entry in the transition matrix, its conditional distribution on the input may not have an analytical form. Therefore, the mapping from the input to the transition matrix should be trained by neural networks, as suggested by Bengio [2].

In general, the emission probabilities can be learnt using neural networks as well. But often they can be modeled by some radial basis functions (RBF) such as Gaussian distributions given some prior knowledge on the input/output relationship. Obviously, if we add more prior knowledge, we can obtain more compact and explicit output distributions. At the extreme, the training process degenerates to a regression problem between the input and the output.

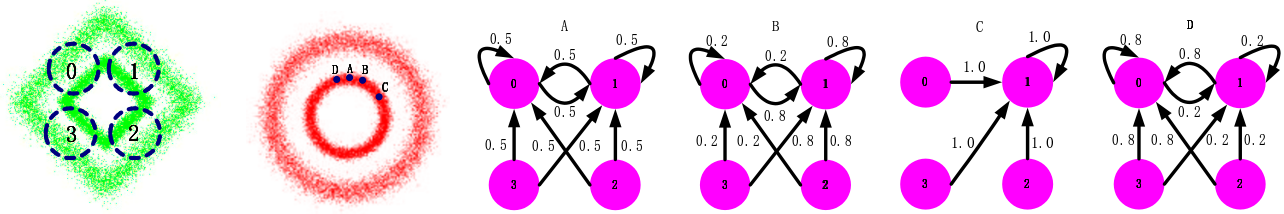
In this experiment, we simplify the emission distribution at each state as a Gaussian output whose mean and variance are determined by the input data. At each state S_i, the emission probability is given by

$$b_i = G(\mu_i \varphi(x_t), \Sigma_i \varphi^2(x_t)) \tag{1}$$

where μ_i is a vector and Σ_i is a matrix, and φ is the distance of the input signal from the origin. Learning the emission probability is then simplified to one of determining the values of μ_i and Σ_i.

We have developed a training algorithm for IOHMM. We follow Bengio's approach [3] to train IOHMMs under the EM framework. What is novel in our algorithm is the process of training the transition matrix with neural networks. Each entry in the transition matrix is trained with an independent neural network, after the M-step at each iteration. For this toy problem, our network has a single hidden layer with six nodes, two input nodes (2D coordinates of the input data) and one output node (the transition probability from state S_i to state S_j). A bias node is further added to the input and hidden layers.

The learnt transition probabilities are clearly dependent on the input, as shown in Figure 4(c)-(f) for four different



(a) (b) (c) (d) (e) (f)

Figure 4. The learnt IOHMM to map circles to diamonds. (a) Four states of IOHMM; (b) four points from the input; (c)-(f) corresponding transition matrices for the four points A, B, C, D shown in (b). Dotted lines in (a) indicate that the model parameters are not fully specified unless the input is given.

points. For example, the point A which is located at the boundary of state S_0 and state S_1 has a transition matrix in Figure 4(c). It shows that the next output can stay at either state S_0 or S_1, but not at S_2 or S_3 because $a_{22} = a_{33} = 0$. In other words, there is a strong tendency to transit to these two states no matter what the current state is, as long as the input falls at location A. As we move from A to B, it becomes more likely to transit to state S_1 than to S_0, as shown in Figure 4(d). When the input point C is at the mean (for the local observation distribution) of state S_1, the transition matrix will be simplified to a zero matrix except for the column corresponding to state S_1 whose entries are all equal to 1, as shown in Figure 4(e). It implies that the next state must be S_1 after the transition. Figure 4(f) shows the transition matrix when the input point is at point D. Figure 4(f) has a similar structure to Figure 4(d) except that the next output is more likely to be on state S_0. Because of the significant constraint by the transition matrix, the synthesis will most likely yield correct state transitions even if the output is sampled from a "wrong" state at some time instant.

From the input data in Figure 5(a), we obtain the synthesis result shown in Figure 5(b). As expected, the output is distributed around one of the two diamonds, similar to the training data. Moreover, transitions between different states are correct as shown by the arrows in Figure 5(b). Depending on whether the input is on the inner or the outer circle, the output samples form two Gaussian distributions that belong to the same state S_2 (the dotted blue circle in Figure 5(b)). Because temporal information is not used in training and only four states are used, the synthesized output trajectory does not follow the two diamonds exactly.

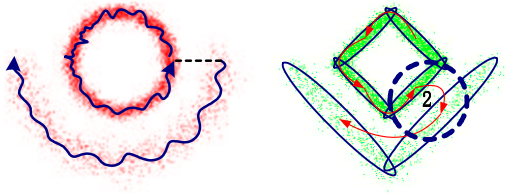
3. Synthesizing facial animation from audio

We apply IOHMMs to synthesize facial animations from audio.

3.1. Audio-visual signal representation

In our system, we use the trajectories of 3D points on the face of an actor and his voice as the training data. In total, 150 points are tracked. We use principal component analysis (PCA) to compress the 450 dimensional feature vector into a 15-dimensional feature vector that covers 97% of the variance.

We use an 18-dimensional feature vector to represent vocal signals. Instead of the traditional phoneme-viseme mapping, we use low-level acoustic features such as MFCC and energy as the input. The input audio sequence is blocked



(a) (b)

Figure 5. The synthesis results using IOHMM. (a) The input sequence. The dots are actual samples, curly lines with arrows show the moving trajectory and the dotted line indicates a jump. (b) The output. The dots are the sampled output signal. Solid ellipses are the local distributions fully specified given the input. The dotted circle indicates the state 2 which two local distributions belong to. Arrows show the transition between states and local distributions.

into frames with the same size as the captured video. In order to capture more dynamics in the vocal feature, we also calculate the delta parameters for MFCC and energy. Speech energy is an important vocal feature because it plays an important role in controlling facial expressions.

3.2. Training

In our application, both the input (vocal feature vector) and output (facial expression) are continuous high-dimensional random variables. Furthermore, we have no prior knowledge about the mapping relations between the input and output. Therefore, training such an IOHMM is much more complex than the toy problem proposed in the preceding section. In Bengio's work [3], the local mapping model and state transition probabilities are modeled as neural networks. Although this architecture can be trained using the generalized EM (GEM) algorithm [8], training so many neural networks is non-trivial. To simplify the training process, we first quantify the input into K classes, each of which has its own mean and variance. A new audio frame a can be classified by calculating the Mahalanobis distance:

$$a \in \text{class } m \tag{2}$$

if

$$m = \arg\min_i (a - \mu_{ai})^T \Sigma_{ai}^{-1} (a - \mu_{ai}), i = 1, ..., K \tag{3}$$

where μ_{ai} and Σ_{ai} are the mean and variance for class i. For each class, the conditional distribution is modeled by

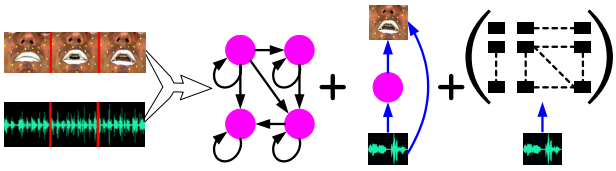


Figure 6. Training IOHMM from input audio signals and corresponding visual signals. The model consists of three parts: a state machine, emission probability conditional on the input, and transition probability matrix conditional on the input.

K Gaussians, each of which corresponds to a specific audio class. Then the emission probability for state i can be represented by

$$b_i(t) = G(\mu_{vik}, \Sigma_{vik}) \tag{4}$$

if a_t belongs to the class k. Note that μ_{vik} and Σ_{vik} are the mean and variance of the output distribution for class k at state i. These parameters need to be learnt.

In our system, the transition probabilities are modeled by $N \times N$ neural networks which are similar to those used in the toy problem in the last section. In the E-step, the emission probability should be computed by equation 4. In the M-step, the emission probability parameters are updated by:

$$\mu_{vik} = \frac{\sum_{t=1, a_t \in \text{class } m}^{T} \gamma_t(i) v_t}{\sum_{t=1, a_t \in \text{class } m}^{T} \gamma_t(i)} \tag{5}$$

$$\Sigma_{vik} = \frac{\sum_{t=1, a_t \in \text{class } m}^{T} \gamma_t(i)(v_t - \mu_{vik})(v_t - \mu_{vik})^T}{\sum_{t=1, a_t \in \text{class } m}^{T} \gamma_t(i)} \tag{6}$$

Figure 6 shows our training algorithm. A trained HMM consists of three parts: a state machine, an emission probability for each state conditional on the input, and a transition matrix conditional to the input.

3.3. Synthesis

Given a new audio sequence, we can apply the model to synthesize the most likely visual sequence that best fits the model. In IOHMM, the state output probabilities and transition probabilities are conditionally dependent on the input. Therefore, the synthesized sequence is the most likely one that satisfies

$$\bar{V} = \arg\max_{V} P(V|A, \lambda) \tag{7}$$

where V is the visual sequence and A the audio sequence. There are three steps in the synthesis process as shown in Figure 7:

- Initialization. At time 1 we choose q_1 according to the model prior state probabilities π. Then at time t we choose (randomly sample) q_t according to $P(q_t|x_t, q_{t-1})$ (where the R.H.S. is known), and we randomly sample y_t according to $P(y_t|x_t, q_t)$. We obtain an initial estimation of the output sequence by repeating this process, $V_1 = (v_{11}, v_{12}, ..., v_{1T})$.

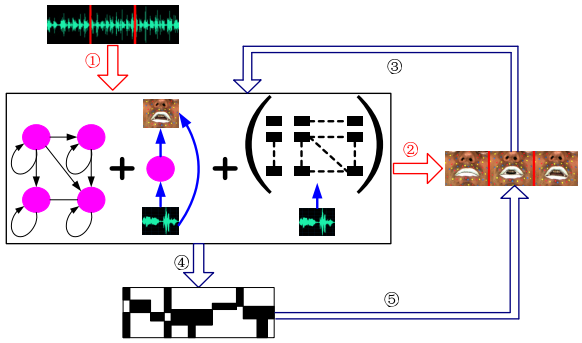


Figure 7. Synthesizing visual signals from an IOHMM and input audio signals. Steps 1 and 2 are the initialization. Steps 3, 4 and 5 are iterated until convergence.

Figure 8. A few frames from a synthesized video sequence of Dr. King's speech. The synthesis uses a single picture.

- Iteration. The observation is complete after we obtain the initial output sequence. For each iteration we run a forward-backward process, after which an occupancy matrix $\gamma_t(i)$ can be obtained. $\gamma_t(i)$ represents the probability of being in state i at time t, given the input/output sequence and model. Then the synthesized output can be updated by.

$$v'_t = \frac{\sum_{i=1}^{N} \gamma_t(i) \mu_i}{\sum_{i=1}^{N} \gamma_t(i)} \tag{8}$$

- Termination. Given the fixed model parameters and input sequence, the most likely output sequence can be obtained when the change of the likelihood is below a threshold.

It can be proven that the above iterative algorithm will converge to an optimal solution under the EM framework. In our experiments, we found that the synthesis sequence tends to converge to the means of the states. This can be explained by the blurring and muting effects in HMM. However, since we have K distributions for a given state which correspond to K audio classes, the expressive power is sufficient. In fact, those fine details which are expected for the synthesis are supplied mainly by the local mapping (one output distribution for each class at each state).

4. Experimental results

In our experiment, we have used 20000 frames or 667.33 seconds of video. The video consists of 189 short sentences. The input audio is clustered into 15 classes. The training process takes about 5 minutes to converge on a mid-level PC. We first apply the learnt IOHMM to synthesize facial

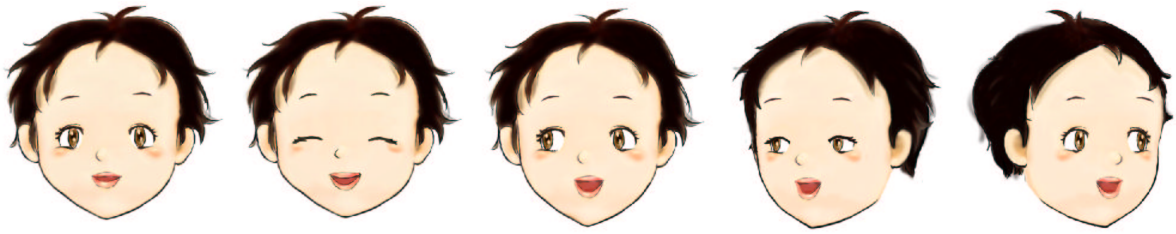


Figure 9. Several frames of synthesized cartoon video sequence. Several cartoon template images are used in the animation.

Comparison	Error
S1-B	0.0052
S2-B	0.0059
A-B	0.0015

Table 1. A comparison between the synthesized result and ground truth. A is the ground truth, B is the PCA reconstructed result (also ground truth), S1 is the synthesis result initialized by B, S2 is the synthesis result initialized by random sampling.

expressions from the audio in the training set. The audio sequence used for comparison is not used for training the IOHMM. Table 1 shows the comparison with ground truth. To compute the reconstruction error, we calculate the minimum distance of each feature point to the face model reconstructed after PCA (with 97% variance covered). The error shown in the table is the summed error normalized by all feature points. We can conclude from the table that reconstruction quality is good because the reconstruction errors (with two different initialization schemes) are on the same order of the error between the original and the PCA-reconstructed model.

Figure 8 shows the result of animating a single picture using our model. Several frames from the synthesized sequence of the famous speech of Dr. Martin Luther King, "I have a dream", are shown in the figure. The sequence shows significant facial movement. Using several cartoon templates with different poses and expressions, we can also animate a long sequence of cartoon. Several animated cartoon frames are shown in Figure 9. Because we use a 3D model with 150 feature points, we clearly observe facial expressions over the whole face in the animation sequences.

We have encountered some difficulties when using the IOHMM for synthesis. The most difficult problem is to synthesize facial expression when the character is silent. There is no clear mapping from silence to facial expression. Some high-level knowledge must be applied to tackle this problem. Similarly, we have difficulties synthesizing some facial expressions that do not correspond to vocal signals. An example is the frowning expression.

5. Conclusion and future work

In this paper we have studied the problem of dynamic audio/visual mapping, specifically, by formulating audio/visual mapping as an IOHMM problem. A key observation is that IOHMM is better suited than conventional HMM for synthesis because it can synthesize structures that are finer than states. Moreover, because transition probabilities in IOHMM are conditional to the input, it is more likely that the synthesized state sequence will be correct. An IOHMM model is trained under the EM framework, where each transition probability is modeled by a single neural network and updated at each iteration. Given the input audio signal, a facial animation sequence is generated by the maximum likelihood principle. Our experimental results from a single image and from a sequence of cartoon template images demonstrate that our synthesis results are of good quality.

While we have studied synthesizing facial expressions from audio in this paper, the very idea of IOHMM is also applicable to other dynamic input/output mappings. We plan to build a complete cartoon video rewrite system by combining cartoon animations from different poses/emotions.

References

[1] Y. Bengio. Markovian models for sequential data. *Neural Computing Survey 2*, pages 129–162, 1999.

[2] Y. Bengio. Personal email conmmunication, 2000.

[3] Y. Bengio and P. Frasconi. Input/output HMMs for sequence processing. *IEEE Trans on Neural Network*, pages 1231–1249, 1996.

[4] M. Brand. Voice puppetry. In *Proc. SIGGRAPH99*, pages 21–28, 1999.

[5] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proc. SIGGRAPH97*, pages 353–360, 1997.

[6] T. Chen and R. Rao. Audio-visual integration in multimodal communication. *Proceedings of IEEE*, pages 837–852, May 1998.

[7] K. H. Choi and J. N. Hwang. Baum-Welch hidden Markov model inversion for reliable audio-to-visual conversion. In *IEEE 3rd Workshop on Multimedia Signal Processing*, pages 175–180, 1999.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

[9] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Proc. SIGGRAPH98*, pages 55–66, 1998.

[10] S. Levinson, L. Rabiner, and M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell System Technical Journal*, 64(4):1035–1074, 1983.

[11] J. Pearl. *Probabilistic reasoning in intelligent system: Networks of plausible inference*. Morgan Kaufmann, 1988.

[12] L. Rabiner. A tutorial on hidden Markov models and selected applicaitons in speech recognition. *Proceedings of IEEE*, pages 257–286, Feb 1989.