

Second Year PhD Report

Extracting Motion Primitives from Natural Handwriting Data

Ben H Williams
Institute of Adaptive and Neural Computation
School of Informatics
University of Edinburgh
5 Forrest Hill, Edinburgh UK
ben.williams@ed.ac.uk

September 15, 2006

Abstract

Biological movement control and planning is based upon motor primitives. Each motor primitive takes responsibility for controlling a small sub-block of motion, containing coherent muscle activation outputs. A central timing controller cues these subroutines of movement, creating complete movement strategies that are built up by overlaying primitives, thus creating synergies of muscle activation. This partitioning allows the movement to be defined by a sparse code representing the timing of primitive activations. We have shown that it is possible to use a factorial hidden Markov model to infer primitives in handwriting data. The variation in the handwriting data can to a large extent be explained by timing variation in the triggering of the primitives. Once an appropriate set of primitives has been inferred, the characters can be represented as a set of timings of primitive activations, along with variances, giving a very compact representation of the character. The model is naturally partitioned into a low level primitive output stage, and a top-down primitive timing stage. This partitioning gives us an insight into behaviours such as scribbling, and what is learnt in order to write a new character.

1 Introduction

Biological systems are strongly superior to current robotic movement control systems, despite having very noisy sensors, and unpredictable muscles. Therefore, the amount and nature of pre-planning in biological movement is extremely interesting. Strong evidence exists to suggest that biological motor control systems are modularised, with *motor primitives* first being conclusively found in

frogs (Bizzi et al., 1995; d’Avella and Bizzi, 2005; d’Avella et al., 2003), where stimulation of a single spinal motor afferent triggered a complete sweeping movement of the frog’s leg. For a review of modularisation of motor control in the spine, see (Bizzi et al., 2002).

Motor primitives are defined as time extended sub-blocks of movement that can be overlaid to create complete movements, with new movements being learnt and stored as new timings of primitives. Evidence suggests that once a particular subsection of movement has commenced, it cannot be unexpectedly switched off. Rather, to quickly modify a movement trajectory, the movement primitives are superimposed (Kargo and Giszter, 2000). Therefore, if the primitives are fixed in the short term at least, and once a primitive has been triggered, it must play out its whole length, it means that a sufficient code for all movements would be the onset timings of these primitives.

To reliably model natural handwriting, we introduce a fully generative model, allowing proper modelling of handwriting variation and adaptation irrespective of character drawn. If the handwriting output is made up of primitive type sub-blocks then the generative model must represent these primitives, to allow it to efficiently model the internal handwriting encoding.

2 Project Outline

A generative handwriting model must be capable of reproducing the class of characters upon which it has been trained. Assuming that all motor commands are made up of motor primitives, handwriting must therefore contain projections of these primitives. Assuming also that motor primitives are fixed, or adaptable over long time scales, any short term adaptability and learning must come from the timing and selection of different primitives.

Assuming the individual primitives are independent of each other, and are linearly superimposed, a controller to select primitive onset timing is necessary, similar in nature to a Piano Model, where key pressing controls the onset of time extended clips of sound that the listener hears as a superimposed waveform.

2.1 A Simple Generative Model: The Piano Model

Unlike many previous studies which analysed motor primitives directly from given data, we base our approach on a generative model of motion. With probabilistic inference methods, we infer the primitives inherent in given data. We assume that the activation of motor primitives can be overlapping, and that they do not have uniform fixed length. The primitives are therefore the output vocabulary, which may either be inherited, or learnt over long time-scales. We assume that the primitives have little or no dependence on each other. The independence of one primitive from another, and post-activation persistence of the motor primitives, give rise to a model that is similar in nature to that of a piano. (See Figure 1)

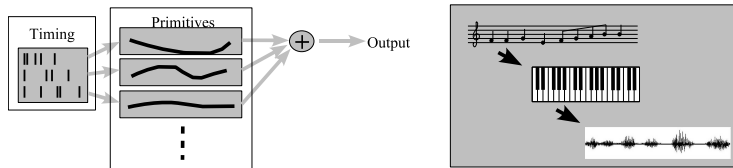


Figure 1: The Piano Model. The model is segmented into a timing part and a movement subroutine, or primitive part. The timing information is encoded in spike positions, possibly in a similar way to how biological neurons may encode the movement. The movements are encoded as multidimensional muscle activation synergies in biology. Here they have been simplified to one-dimensional signals. The analogy with written music being translated into auditory music is also shown.

To formalise the model in a generative way, the output of the system Y at time t is defined as

$$Y(t) = \sum_{m,n} \alpha_{mn} W_m(t - \tau_{mn}) , \quad (1)$$

where $W_m(t)$ are the primitives, and τ_{mn} represents the time of the n^{th} activation of primitive m , and α_{mn} defines the activation strengths of the primitives.

In this definition, m enumerates the primitives, whilst n enumerates the occurrence of the primitive within the sample window, at time τ_{mn} . We have called this model the Piano Model because of its similarities to the operation of a piano being played, where the timing controller (the pianist) presses each key at the appropriate time in the piece of music. The keys on the piano produce time extended clips of sound, which are superimposed to create the music that is heard by the listener. The crucial point is that the only dependence that the music has on the pianist is the timing and choice of keys he presses, with associated pressure¹. In the same way, in our motor primitive model, a central controller does not control the precise movements but rather the timings of particular temporal movement sequences. Figure 1 shows a diagram to illustrate the Piano Model.

The Piano Model neglects noise effects and learning the parameters of the model is better realised in a probabilistic framework. Assuming discrete time steps, an appropriate modelling framework is that of Factorial Hidden Markov Models (fHMMs). These are the same as standard HMMs, but with multiple, parallel and independent hidden state chains, as seen in Figure 2.

2.2 The Factorial Hidden Markov Model

A graphical model of the fHMM can be seen in Figure 2. At each time step, the observable output Y_t , a vector of dimension D , is dependent on M hidden

¹This is debatable, as pianos may respond to the speed of key press, the duration of the key press, and what other keys are being pressed at the time. Pianos also have pedals to create different effects. The analogy is not intended to be exact.

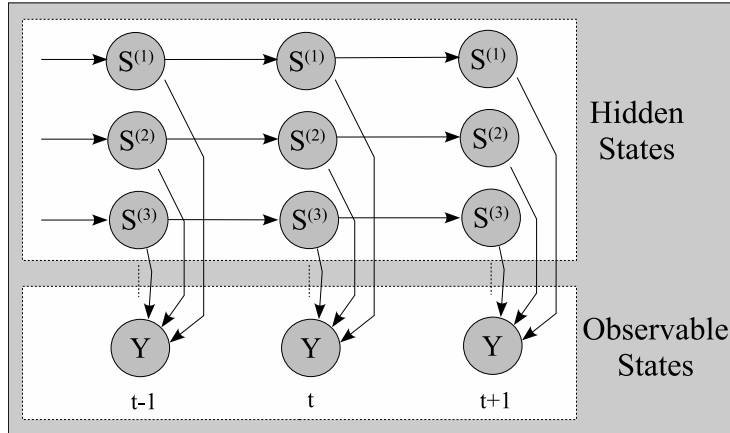


Figure 2: Graphical representation of a Factorial Hidden Markov Model, showing the independence of the separate Markov chains. Although the observable output is dependent upon the state of the entire system, the internal states evolve with no interdependencies. S_t^m denotes the hidden state vector at time t , in factor m .

variables $S_t^{(1)}, \dots, S_t^m$. The output is a multivariate Gaussian, such that

$$Y_t \sim \mathcal{N}(\mu_t, C), \quad (2)$$

where C is a $D \times D$ parameter matrix of output covariance, and

$$\mu_t = \sum_{m=1}^M W^m S_t^m \quad (3)$$

is the D -dimensional output mean at time t . W^m is a $D \times K$ parameter matrix giving the output means for each factor m , such that the output mean μ_t is a linear combination of its columns weighted with the hidden state activations.

Each of the M hidden variables can be in K different states. In equation (3) this is encoded in the K -dimensional state vector S_t^m using a 1-in- K code, i.e., $S_{t,i}^m = 1$ if the m -th factor is in state i and zero otherwise. This allows us to write expectations of the hidden states as $\langle S_t^m \rangle$, which is also the probability distribution over the individual states S_t^m . Each latent factor is a Markov chain defined by the state transition probabilities and the initial state distribution as

$$P(S_1^m = i) = \pi_i^m, \quad P(S_t^m = i | S_{t-1}^m = j) = P_{i,j}^m, \quad (4)$$

where π^m is a K -dimensional parameter vector giving the initial hidden state distribution, and P^m is a $K \times K$ parameter matrix denoting the state transition probabilities. As can be seen in Figure 2, each factor is independent. This means

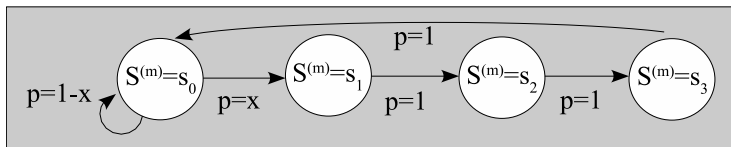


Figure 3: State change probabilities. This shows how the state change probabilities were constrained so that the individual Markov chains can correspond to a time extended primitive.

that the joint probability distribution can be factorised as

$$P(\{Y_t, S_t\}) = P(S_1)P(Y_1|S_1) \prod_{t=2}^T P(S_t|S_{t-1})P(Y_t|S_t) \quad (5)$$

$$= \prod_{m=1}^M \pi^m P(Y_1|S_1) \prod_{t=2}^T \prod_{m=1}^M P^m P(Y_t|S_t) . \quad (6)$$

The fHMM model was based upon that described in (Ghahramani and Jordan, 1997), which provides arguments for using a distributed state representation as is used here, and discusses the model in further detail.

To use the fHMM framework as a probabilistic implementation of the Piano Model, we must attribute each hidden Markov chain, or factor, to one primitive. The observables, being real-valued output vectors describing pen position derivatives are modelled by the multivariate output Gaussian distribution, dependent upon the hidden state values at time t . The model has M primitives, with each primitive having K states. This is similar to the Piano Model, given some extra constraints.

2.3 Constraints

The Piano Model differs from the fHMM model by allowing the primitives to be inactive, giving a zero output contribution. In fact, it is a tacit assumption that the primitive activation is fairly sparse, making the periods of inactivity important to the model. In the fHMM model, the output is always a linear combination of all the factors, thus a major constraint imposed was that state 0 for all Markov chains should contribute towards a zero output mean (i.e., to not contribute, as the data mean is zero).

In the Piano Model, each primitive should keep its shape. This means that the possible hidden state transitions in the fHMM model needed to be constrained. When a particular primitive was triggered, the hidden state vector for that factor changes from state 0 to state 1, then is constrained to progress monotonically through the states until the last state is reached, and returns to state 0. These state change restrictions can be seen graphically in Figure 3.

2.4 Learning the Model

Given the fully parameterised modelling framework, learning of the parameters can be done using an Expectation-Maximisation (EM) method. The structured variational approximation was chosen for the E-step inference. For more details on the various arguments for and against this choice, refer to (Ghahramani and Jordan, 1997). With the structured variational approximation, the inference in the fHMM is split up into M separate Hidden Markov Models, with single hidden state chains, with each HMM contributing a learnt proportion towards the output. With separate HMMs, the normal Baum-Welch Forward-Backward algorithm can be used to infer the hidden state expectations (Dempster et al., 1977). The only addition necessary is a responsibility factor h_t^m , which models the amount that the m^{th} HMM contributes towards the output. h_t^m takes the place of the observation likelihood in a standard HMM. See (13) for the details of calculating h_t . The M-step updates the parameters W^m , π^m , P^m , and C . The update equations are in Appendix A.

The fHMM can reconstruct the data by using a set of maximally statistically independent primitives, and the appropriate hidden state values. Due to the constraints imposed upon the hidden state transitions, these state values can be reduced to a set of primitive activation timings, or spikes. Without this spike timing information, the primitive model can still be run separately, as can be seen in Figure 6, which can be thought of as *primitive babbling*. To reconstruct a character, the primitives need to be coordinated, and activated at the appropriate times. This is achieved by introducing a separate part of the model, the centralised timing controller.

3 Timing Model

The centralised timing controller must be capable of reproducing spiking characteristics that in some areas of the character are variable, and others less so, both in time of spike, and existence of spike. In other words, some primitives are necessary, occurring in every character sample in roughly, but not exactly the same place. However, there are some primitives that occasionally occur in a reproduction, but not in every case. Crucially, on a short time scale, there is heavy dependency between time steps, whereas on the long term, they are simply dependent upon the character being drawn, ie. The parameterisation of the model.

We have chosen an Integrate and Fire (IF) model for the generation of spikes, as this model has a local temporal dependency characteristic, and also allows variance in the total number of spikes in a sample.

3.1 Integrate and Fire

The Integrate and Fire (IF) model originates from simplified models of biological neurons. It treats the neuron as a leaky capacitor, upon which a charge is built up by the inputs, over time. Once the voltage across the capacitor reaches a

threshold level, the neuron fires, producing a spike at its output, and discharging the capacitor. This means that, due to the leak term, over a long time scale, the inputs at different times are independent, however, on a short time scale, they are not, as it is the short-term running sum of the inputs that causes the neuron to fire. This is desirable for the primitive model, because the timing of a necessary primitive can be variable in the character samples, however, the IF neuron will still fire as long as it receives enough inputs during its temporal memory window. A probabilistic model of an IF neuron includes a noise term in the threshold.

The most straight forward model using IF neurons is to attribute one IF neuron to one primitive. The inputs to the neurons will then determine the timing of the primitives. For a particular primitive, m , the probability of a spike at time t , $P(\lambda_t^m)$ is given by:

$$P(\lambda_t^m | \lambda_{t-1}^m = 0) = P(\lambda_{t-1}^m) + I_t^m - L_t^m, \quad (7)$$

$$P(\lambda_t^m | \lambda_{t-1}^m = 1) = I_t^m - L_t^m, \quad (8)$$

$$L_t^m = \nu P(\lambda_{t-1}^m), \quad (9)$$

where I_t^m are the input excitations, and L_t^m is a leak term proportional to the accumulated probability. Therefore, given a common set of primitives, a character is defined by its *temporal excitation matrix*, I_t^m , which parameterises the IF model. This matrix is learnt from the spiking statistics of the character training set, as seen below.

During the E-step inference of the fHMM, the hidden state distribution $P(S_t^m)$ is inferred. As the transition matrix is constrained so that the primitives progress monotonically through their states, the information in $P(S_t^m)$ can be summarised as the onset probabilities of the different primitives, $P(S_{t,k=1}^{m,n})$ which are the rows of $P(S_t^m)$ for state 1, the first state in each primitive, in character sample n . For a set of primitives that fit the data well, these probabilities are close to zero or one, and form a very sparse matrix containing spikes representing primitive activation appropriate to reconstruct a single character sample from the data set. It is effectively an average over the samples of these spiking matrices that is needed to parameterise the IF model, as I_t^m . For ease of notation, let $\tau_{m,t}^n = P(S_{t,k=1}^{m,n})$.

To allow for differences in the start point time, and average speed of each character sample, two parameters are associated with each τ matrix, an offset, δt and a linear stretching factor, δl . These parameters are optimised so that the τ matrices best fit the average I_t^m matrix that is constructed by taking linear interpolations.

$$I_t^m = \frac{\sum_n \tau_{m,k^n}^n}{N}, \quad (10)$$

where

$$k^n = (t + \delta t^n) \delta l^n. \quad (11)$$

The τ matrices are then shifted to better fit I_t^m , thus optimising the quantity

$$\sum_{n,m,t} \tau_{m,k^n}^n I_t^m \quad (12)$$

w.r.t δt , and δl .

This finds an I_t^m matrix that best reflects an average primitive onset probability matrix, $P(S_{t,k=1}^{m,n})$, where t has a separate linear shifting and stretching factor associated with it for each character sample, n . This is used to parameterise the IF model, which generates the spike timing information needed to run the fHMM generatively.

3.2 Implementation

Handwriting data were gathered using an INTUOS 3 WACOM digitisation tablet

<http://www.wacom.com/productinfo/9x12.cfm>. This provided 5 dimensional data at 200Hz. The dimensions of the data were x-position, y-position, pen tip pressure, pen tilt angle, and pen orientation (0-360°). The normalised first differential of the data was used, so that the data mean was close to zero, providing the requirements for the zero state assumption in the model constraints (see section 2.1). The data collected were separated into samples, or characters, for processing purposes, and then the parameters were fitted to the data using our algorithm. Once the parameters were learnt, the hidden state expectations $\langle S \rangle$ were finalised, and the pen space reconstruction of the data could be calculated, along with the primitive timing statistics.

To clarify the operation of our algorithm, and the iterative nature of the EM inference, here are the pseudo-code and parameter settings.

Constants T =times of each sample, N =number of samples, K =max primitive length, M =number of primitives, D =dimension of data.

Initialisations The primitives W^m are initialized with a zero mean Gaussian distribution of the same variance as the data. The transition probabilities π^m and P^m are set as defined in Section 2.3 with primitive onset probability N/T , giving a prior expectation of each primitive to be used once in each character. The output covariance C is taken directly from the covariance of the data.

loop (EM loop)

E-step: initialize $\langle S_t^m \rangle$ to $P(S_t^m = 0) = 1$ for all t, m .

loop

compute h_t^m from equation (13).

$\forall m$: forward-backward algorithm gives $\langle S_t^m \rangle$ using h_t^m as obs. likelihood.

until expectations not changing significantly, or max 20 iterations

M-step: update the parameters as in (16)

until primitives don't changing significantly, or max 50 iterations.

4 Work accomplished / Underway

4.1 fHMM Modelling issues

After testing the algorithm on toy data sets that had been generated from a piano model, we noticed that two main constraints must be placed upon the fHMM to allow primitive extraction, as detailed in Section 2.3. These constraints allowed us to simplify the E-step inference, particularly in terms of space, as only a single $\langle S_{t,i}^{(m)} S_{t-1,j}^{(m)} \rangle$ value needed to be calculated per primitive. (See equation (16).)

Constraining the transition matrix to force the primitives to progress monotonically is effectively placing a strong prior on the model. If, at some point during the iterative learning procedure, a particular primitive is a bad representation of the data in its current form, the evidence will strongly suggest that the primitive should not be used at all. This is a clash of the evidence with the prior, and leads to the posterior distribution vanishing to zero. This causes the update equations to fail, as they try to invert the posterior. To stop this from happening a small flat prior ($1e^{-40}$) was added to the evidence, meaning that the priors have the advantage in a clash with the evidence.

Constraining the transition possibilities for the markov chains creates large local minima in the search space, which manifests itself most obviously by making the markov factors ‘get stuck’ on part of the primitive (which is represented at the start of the state chain progression, but the part of the primitive prior to this cannot be represented, because the markov chain gets ‘stuck’ on this bit (as one may expect, as this part of the data is then strongly represented). To get over this, a shifting algorithm was included. The shifting algorithm literally shifts the output values for each hidden state for a single primitive either to the right or the left, depending on the magnitude of the start of the primitive relative to the maximum value of the primitive. This works because prior to the activation of a particular primitive, the contribution of that primitive must on average be zero, therefore, forcing the first state (or section) of the primitive to be zero means that we can be sure that the start of the primitive is represented.

To stop over-fitting of the data, there is a routine that merges similar primitives. For merging to occur, the average difference between the primitive output contributions must be less than 10% of the maximum output contribution by default.

There is also a primitive checking routine that makes sure that the onset probability of any of the primitives does not approach zero, in which case all the parameters associated with the primitive are reset (including the shape of the primitive).

The primitives although not a fixed length on paper, are currently a fixed number of states in the model. This means that their duration is constant. The end of the primitives is not defined however, and is learnt by the model, due to the fact that the primitives are capable of giving a zero output contribution, so in this sense, they are variable length. However, there are a limited number of primitives, and small variations in the duration of a motif in the data re-

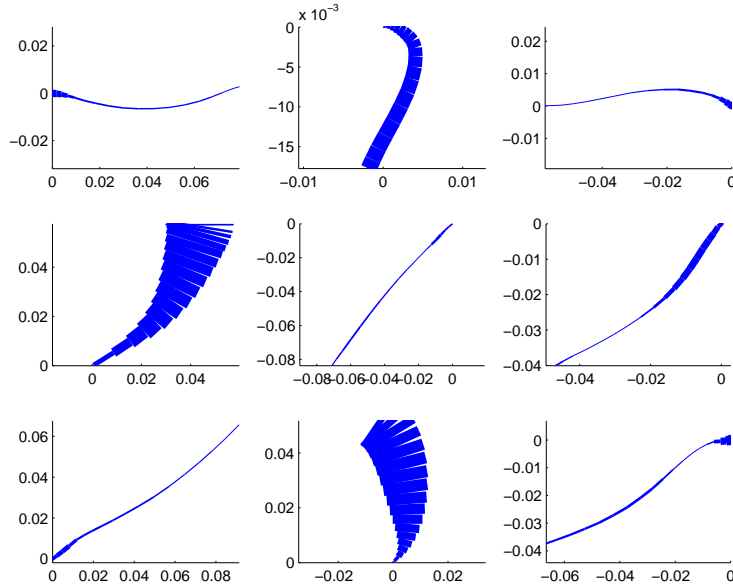


Figure 4: A sample of 9 primitives used for the reconstruction in Figure 5(a). The primitives are shown as a reconstruction in pen-space, rather than the raw data, which is in 3-dimensional velocity space. Refer to section 3.2 for an explanation.

quire different primitives to model it. This is unlikely to be the case in reality, with a certain amount of time duration variation in the primitives. This can be modelled by allowing self-transitions in the hidden state transition matrix. These can either be a constant prior, or a set of learnt parameters. So far trials have been unsuccessful, resulting in very poor modelling of the data, however ‘stretchy’ primitives may be further explored.

4.2 fHMM operation

Using data from the ‘*g*’ samples, a set of 27 primitives were inferred that could reconstruct the data well. A demonstration of what primitives look like if triggered alone is shown in Figure 4, which shows a sample of 9 primitives, showing them as reconstructions in pen-space, rather than their original form in velocity space. The first differential of the pen position data was used so that the average value of the data used would be zero, so that the primitives would not need to model any ‘drift’ in the writing. A drift would be a long term accumulated error in the position of the pen, and is not very interesting from a muscle control point of view, rather than some form of visual feedback to correct writing position.

The original data sets can be well reconstructed as is seen in Figure 5. Here we can see the reconstruction on the right, and the timing information is shown on the left.

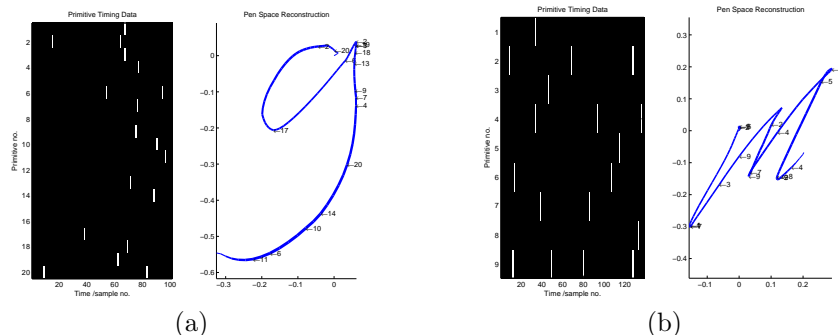


Figure 5: Two examples of primitive timing codes. In both (a) and (b), the timing information is shown on the left, and the reproduction of the sample on the right, with the onset of each primitive marked with an arrow. In (a), 20 primitives of length 40 time steps were used to model the ‘*g*’ dataset, of over 1000 characters. In (b), 9 primitives of length 30 were used to model the ‘*m*’ dataset, of over 100 characters.

These primitives can adequately reproduce the data set from which they were inferred. An interesting aspect of the primitives was found when examining how important the timing information is for the accurate reconstruction of a character. The fHMM model learns the parameters for the hidden states of the Markov model, and it is these parameters which are representations of the primitives. The parameters represent the shape of the primitives, the likelihood of their onset in the dataset, and the output covariance. The hidden state expectations, a probability distribution over the hidden states of the model effectively provides the precise timing information necessary for the reconstruction of the data set. However, the primitives can be ‘used’ generatively, by sampling the model states from the prior distribution (the transition matrices * previous states). This gives us pen output that can be seen in Figure 6. The scribbling nature of the output also seems to convey some aspect of the character set from which the primitives were learnt, for in Figure 6(a), certain characteristics of a ‘*g*’ can be seen, in contrast to Figure 6(b), which is learnt from an ‘*m*’ data set.

Without the precise timing information, the output of the model resembles a form of scribble. In Figure 7, we can see a data set of scribbles, rather than characters, from which primitives can be inferred. A reconstruction of a single sample can be seen on the right. This is shown to highlight the similarity between a scribble and the generative use of primitives, as it is impossible to tell whether or not any precise timing information was used in the reconstruction. (At least, without seeing the original data set.)

The original reason for creating the scribble data set was to demonstrate the generality of the primitives, in terms of modelling other data sets. In Figure 8 we can see reconstructions of two data sets, but using primitives from the scribble data set. The primitives are fit to the samples using a single E-step.

To be able to use the primitives as the basis for an efficient handwriting code, the distribution over the different character samples must be modelled. This

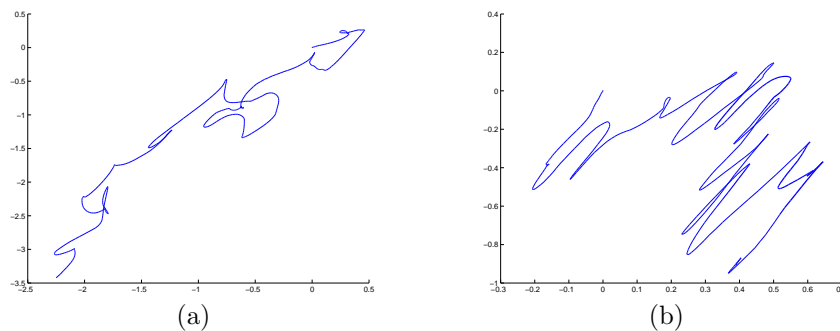


Figure 6: Two samples generated using primitives without specific timing information. (a) was generated using primitives inferred from the ‘g’ dataset, (b) was generated using primitives from the ‘m’ dataset. Starting point of both reconstructions is $(0,0)$.

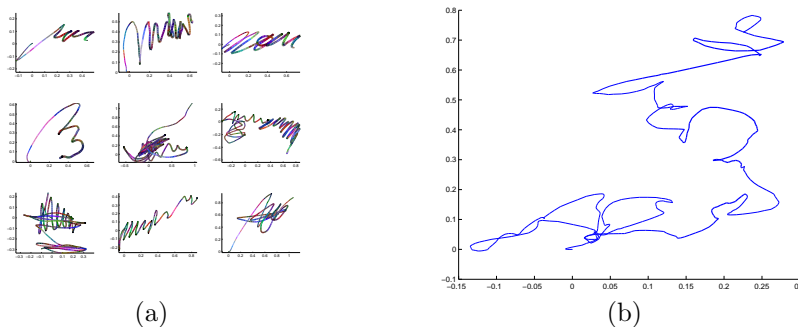


Figure 7: A reconstruction of some samples from the scribble dataset is shown in (a). This was a set of unconstrained scribbles. The primitives are coloured differently, to show different activation areas. (b) shows a sample generated using primitives from this dataset. Starting point of reconstruction is $(0,0)$.

can be accomplished by examining the timing characteristics of the primitives within the data.

The timing information, seen in Figure 5, is simply the onset probabilities of the primitives, which are very close to 1 or 0 for a given sample, therefore taking on the characteristics of neuronal spikes, with the important information being the timing of the spikes rather than their amplitude. The distribution over the spiking matrices is needed, as detailed in Section 3.1. The average spiking matrix, from Equation (3.1) is calculated by shifting and applying a linear stretch the individual samples, so that they best fit the average over them all. This is to allow for slightly different speeds and onset times for the different samples. The final average matrix, which is a distribution over the spike time matrices, taking each spike as independent, can be seen in Figure 9.

Using this matrix, samples of spike timings can be generated, and used to trigger primitives, which in turn reconstruct the written character. The end

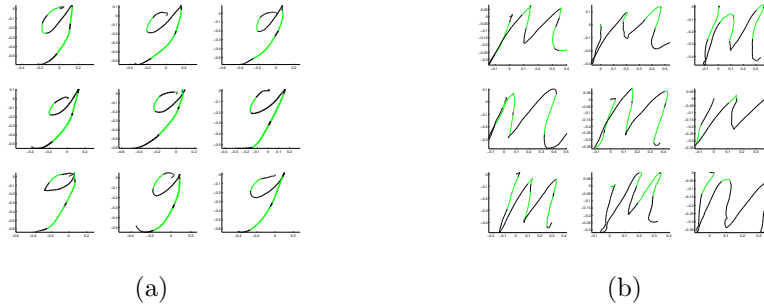


Figure 8: A reconstruction of some samples, using primitives from the scribble set. One of the 36 primitives is highlighted. (a) shows characters from the ‘*g*’ dataset, (b) shows characters from the ‘*m*’ dataset.

result of this process can be seen in Figure 10. These 9 samples were produced by sampling from the distribution in Figure 9, taking each spike as independent. Clearly, although these samples are more like the character ‘*g*’ than the scribble sample shown in Figure 6(a), they are not as good as the original data set, as some are too messy to be clearly identified as the character ‘*g*’. The problem with sampling the spikes independently from the probability distribution is that there is no time time dependence between samples. Some spikes reliably happen at the same time, in which case, their associated probability density is sufficiently concentrated for a spike to appear in the majority of samples. However, other spikes have larger variance in their position of occurrence, however, they do reliably occur. In the sampling procedure, however, their occurrence is not guaranteed, and their absence gives rise to missing parts of characters as can be seen in Figure 10.

To create a time dependence between the samples, a leaky integrate and fire (IF) model was tried, as detailed in Section 3.1. In the IF model, each primitive is associated with an IF neuron, with the inputs to the neurons taken from the spike timing distribution. The reasoning behind using this model is to provide a cumulative summing of the probability distribution, so that areas of lower, more distributed probability will still give rise to a spike when sampled from. The method produces good results although it should be noted that there are two more parameters associated with an IF neuron, that of its threshold, and its leak constant. Figure 11 shows some samples using the IF model to sample from the distribution in Figure 9. These are clearly ‘better’ samples than those generated with the independent spikes assumption.

The problem with using the spike timing distribution to parameterise the IF model is that the matrix of parameters is reasonably large for a single character. This is intuitively unnecessary, when all the parameters are doing is representing a mean time and a variance for each spike (the probability of the presence of a spike is really a function of the method of sampling from this distribution - it is related to the leak and threshold noise in the IF model). A more compact

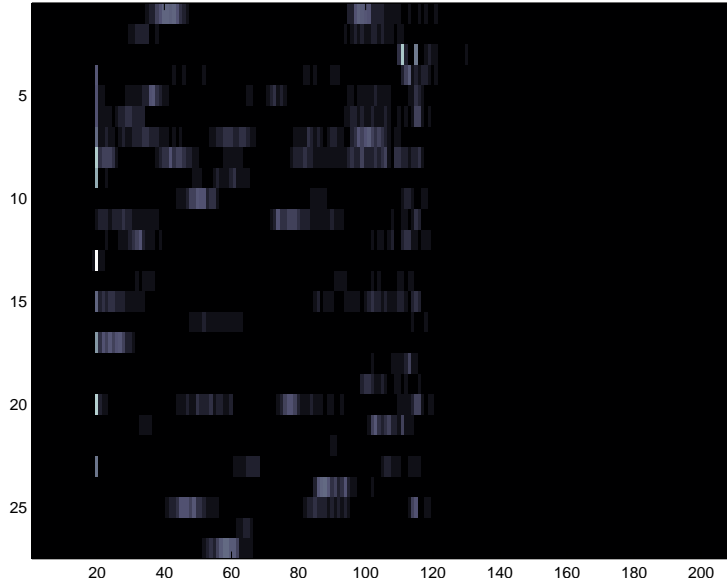


Figure 9: A graphical representation of the distribution of spikes obtained by averaging the individual character spike timing matrices from a 'g' character set.

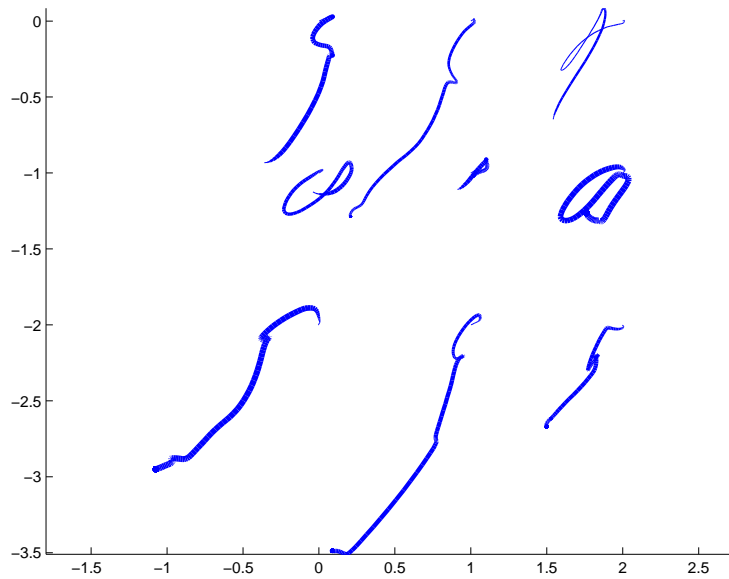


Figure 10: 9 generative samples from the spike distribution shown in Figure 9.

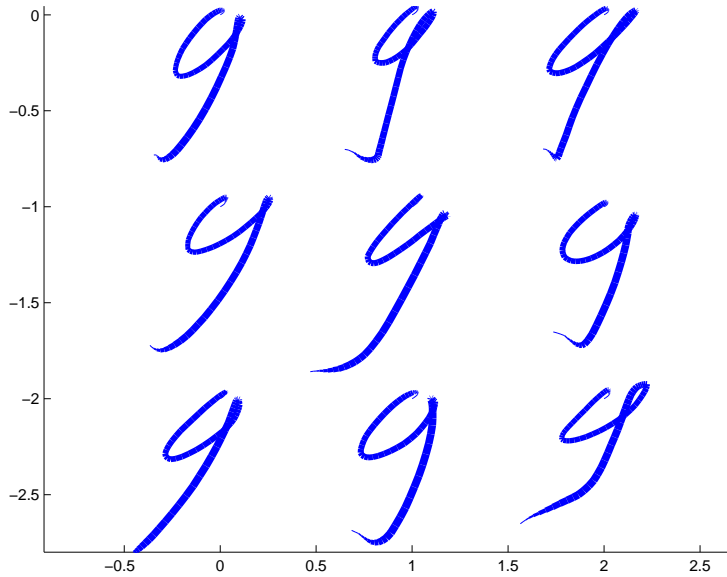


Figure 11: 9 generative samples from the spike distribution shown in Figure 9, using the IF model to sample from the distribution.

and intuitive model would be that of a mixture of Gaussians, with one Gaussian modelling the timing probability of a spike. Figure 12 shows samples from such a model. Again, the problem with a normal mixture of Gaussians is that each Gaussian does not guarantee a spike, and the more likely a single spike is, the more likely a second spike is also, which is not desirable. The benefit of the IF model is that the neuron has a ‘reset’ function when it fires a spike, making a second spike in the short term very unlikely.

4.3 Summary of Problem History and Discussion

Currently, the main problem is efficiently modelling the spike timing distribution for a particular character, and being able to sample from this distribution to reproduce samples within the variation of the data set. Different forms of Gaussian mixture models will be explored, and greedy approximations of these models may allow us to associate more directly the Gaussians with single spikes from particular samples.

Using the current shifting and stretching algorithm does seem to improve the sample output, however, examining the adjustment in the length of the samples shows that it does not act to normalise the lengths of the samples at all. A better method would be to employ some Gaussian fitting, allowing the variance of the Gaussians to model the variance in the sample times.

The next stage of model development will be focussed on the Gaussian mixture model. To solve the issue of multiple spikes originating from the same

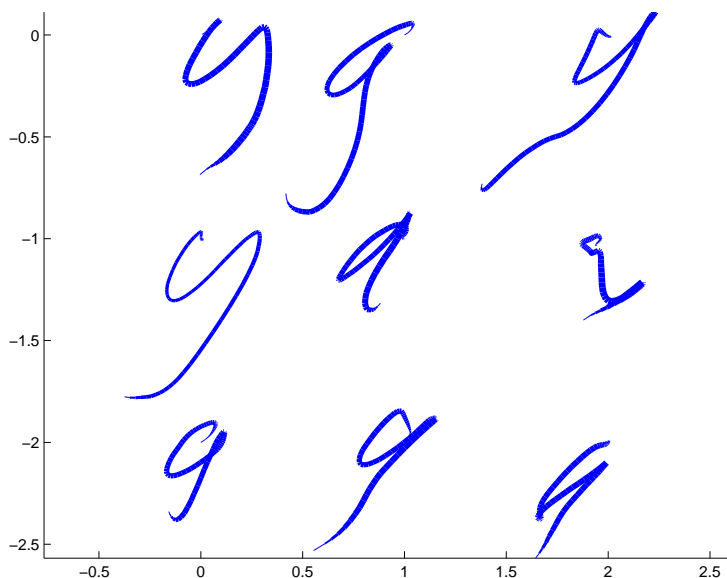


Figure 12: 9 generative samples using a mixture of Gaussians to model the spike timings.

Gaussian, we need to attribute sample spikes to particular Gaussians, and sample from a set of Gaussians, where the number of Gaussians is perhaps also a Gaussian distribution on the number of spikes in a sample.

Without the timing model, the primitives can still produce output, by running the fHMM model generatively, and sampling the hidden states at each time step, from the prior distribution conditioned on the previous time step. This produces an output sample that is similar to scribbling or doodling. Perhaps when we are absent-mindedly doodling, it is a disconnection of the timing control part of our motor system from the muscle output, or primitive section that produces these scribbles. Intuitively, we are putting a pen in our hand, and telling ourselves to write, but not dictating what exactly.

In this model, given a common set of primitives used to control the complete set of possible movements, it is clearly the timing model that is dictating what class of movement is required. The spikes are the internal encoding of the movement class. Using the spike timing representation of when the primitives are triggered, allows a very compact representation for a character. As a common set of primitives is capable of reconstructing several different characters, the spike encoding can therefore be divisive in differentiating one character from another, and may therefore be useful for efficient character recognition. The compact code would also be useful for data storage and transmission, in much the same way as the Ascii code allows efficient transmission of printed characters, a spike encoding could efficiently transmit handwritten ones.

This distributed model provides a framework both for learning new character

sets, and experimentation with primitive shapes. It is possible that primitive shape is a personal attribute, which may account for different handwriting styles. If this is so, then an efficient mapping from one style to another may be possible. This hypothesis requires further research, by examining the primitives inferred from different people’s handwriting samples. Another hypothesis may be that different people have a different number or average length of primitive, maybe this could account for more ‘messy’ styles of handwriting.

Given a common set of primitives, the spike timings encode the movement, but this does not address learning new movements. When we learn to write for instance, we must either learn new primitives, or adapt old ones, or perhaps some movements are simply not learnable or rather cannot be encoded by a sparse spike representation. The adaptation of primitives, and their association with learning new motor skills would be a very interesting area to explore, with the help of this model.

5 Future Targets

Immediately, the main target is to make a generative model of characters (or any handwritten symbol), which is capable of reproducing these characters, and their variance, with a combination of primitive modular output, and a simple timing model, such as a set of Gaussians.

With a working model, a compact code would be available in the form of spike timings, that encodes not only the drawn character, but the distribution over the variance of the character. This would be very useful for understanding how this information may be encoded in our brains, and what areas might be responsible for this encoding. A more thorough definition of what this code means, and how it could be represented using biological neurons would be interesting, and may form a hypothesis for neurological studies.

One of the founding motivations behind this project was to look at how children learn to write, and try to model this. So far we have looked at fully developed handwriting (despite its appearance), and we have assumed that the primitives are fixed functions. Presumably during childhood the motor primitives adapt to become optimal for the learnt tasks. Using the timing model as a prior for the ‘task at hand’ on top of the fHMM model would give us strong priors as to where the primitives occur in the samples, thus allowing further optimisation of the primitives. Currently the fHMM model, and the timing model are learnt separately. It might be interesting to run them as a joint model.

There is still some work to be done at the low level, in terms of validation of the primitives that the fHMM is extracting. The robustness of the extracted primitives to variables such as length and number is an area that needs exploration. With confidence in the extracted primitives, it would be interesting to explore how primitives differ from one individual to another, and whether the primitives themselves adapt when learning a new task in adults. Experiments could include learning to write a new character, trying to write characters at different speeds, timed distractions whilst writing to observe alternative prim-

itive overlaying, contrasting writing with visual feedback against writing with subject's eyes closed. These are just some examples of the possible experiments that could be performed using this model to extract primitives.

After work into categorisation of primitives, it will be possible to explore links between primitives and handwriting style, and whether a small change to certain primitives would effect a global handwriting style.

5.1 Thesis Plan

A preliminary plan of thesis chapters goes as follows:

Introduction

Chapter 1 Background - History of motor primitives and research into primitive modelling, and primitive based control. Literature review looking at evidence behind the existence of primitives, and how people have extracted them, and the implications of the model chosen on the characteristics of the primitives found so far. As the primitives rise from biological constraints and optimisations, this will be addressed here.

Chapter 2 Introduction of the Piano Model - Our base assumptions about primitive characteristics and how they motivate this particular model. Other examples of similar systems.

Chapter 3 Learning the Piano Model - Using a factorial hidden Markov model to learn model parameters. The main modelling details will be presented here, along with the constraints, and the algorithmic operation, in the form of pseudo-code. Details of how the algorithm was optimised, and the numerical issues, and local minima problems will be detailed. Possibly a modular description of the code will be included, to help with extensions.

Chapter 4 Timing Model - Description of different approaches, and detailed explanation of chosen method. As this is a separate part of the model, details of how it interacts with the primitive part, and how the model parameters are learnt will be crucial. An examination of the outputs from different types of generative timing models could be included here, or in the experimental results section.

Chapter 5 Experiments - Motivation behind experiments, hypotheses, experimental method. Experiments will include a comparison of the primitives extracted from different people, and different character sets. A paired-primitive experiment will compare style transformations using a primitive based mapping. A model suitability experiment will explore how the extracted primitives can be justified, by looking at the likelihood of the model, and the reconstruction error, and other model comparison objective functions. If time permits, then an experiment looking at using primitives in a recognition task will be included.

Chapter 6 Results

Chapter 7 Applications - Applications for a primitive based model are huge.

Examples of these, and details of why they would benefit from a generative primitive based model will be outlined here. An overview of similar research, especially in the sound encoding domain, and how applications have benefitted will also be included.

Chapter 8 Discussion - Possible extensions, personal views, other approaches, etc.

A major extension to the model would merit another modelling chapter between 4 and 5. Such an extension would need to be distinct from the fHMM and the timing part of the model. One possibility would be implementing a Kalman filter on the output to compensate for long term drift of output amplitude scaling, or perhaps some form of 'visual feedback', giving an absolute position feedback into the model, to minimise drift error.

Another chapter may well address the issue of handwriting recognition. Using a generative model of handwriting will help to make the recognition more robust to problems such as handwriting occlusion, as areas of observed handwriting would provide evidence for unobserved, or occluded parts.

A Appendix

In the E-step, the responsibility factor, h_t was calculated using a residual error,

$$h_t^{(m)new} = \exp\{W^{(m)T} C^{-1} \tilde{Y}_t^{(m)} - \frac{1}{2} \Delta^{(m)}\} \quad (13)$$

$$\Delta^{(m)} \equiv \text{diag}(W^{(m)T} C^{-1} W^{(m)}) \quad (14)$$

$$\tilde{Y}_t^{(m)} \equiv Y_t - \sum_{l \neq m}^M W^{(l)} \langle S_t^{(l)} \rangle \quad (15)$$

where \tilde{Y}_t is the residual error.

In the M-step, the parameter update equations used were

$$W \leftarrow \left(\sum_{t=1}^T Y_t \langle S_t^T \rangle \right) \left(\sum_{t=1}^T \langle S_t S_t^T \rangle \right)^\dagger \quad (16)$$

$$\pi^{(m)} \leftarrow \langle S_1^{(m)} \rangle \quad P_{i,j}^{(m)} \leftarrow \frac{\sum_{t=2}^T \langle S_{t,i}^{(m)} S_{t-1,j}^{(m)} \rangle}{\sum_{t=2}^T \langle S_{t-1,j}^{(m)} \rangle} \quad (17)$$

$$C \leftarrow \frac{1}{T} \sum_{t=1}^T Y_t Y_t^T - \frac{1}{T} \sum_{t=1}^T \sum_{m=1}^M W^{(m)} \langle S_t^{(m)} \rangle Y_t^T \quad (18)$$

$\langle S_t \rangle$ is the expected value of the hidden states at time t . \dagger denotes *pseudo-inverse*.

References

- E. Bizzi, S.F. Giszter, E. Loeb, F.A. Mussa-Ivaldi, and P. Saltiel. Modular organization of motor behavior in the frog's spinal cord. *Trends in Neurosciences*, 18(10):442–446, 1995.
- E. Bizzi, A. d'Avella, P. Saltiel, and M. Trench. Modular organization of spinal motor systems. *The Neuroscientist*, 8(5):437–442, 2002.
- A. d'Avella and E. Bizzi. Shared and specific muscle synergies in natural motor behaviors. *PNAS*, 102(8):3076–3081, 2005.
- A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature Neuroscience*, 6(3):300–308, 2003.
- A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm (with discussion). *J. R. Statist. Soc. B*, 39: 1–38, 1977.
- Z. Ghahramani and M.I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–275, 1997.
- W.J. Kargo and S.F. Giszter. Rapid corrections of aimed movements by combination of force-field primitives. *J. Neurosci.*, 20:409–426, 2000.