Hidden Markov Models for Endpoint Detection in Plasma Etch Processes

Technical Report UCI-ICS 01-54
Department of Information and Computer Science
University of California, Irvine

Xianping Ge, Padhraic Smyth
Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
{xge,smyth}@ics.uci.edu

September, 2001

Abstract

We investigate two statistical detection problems in plasma etch endpoint detection: change-point detection and pattern matching. Our approach is based on a segmental semi-Markov model framework. In the change-point detection problem, the change-point corresponds to state switching in the model. For pattern matching, the pattern is approximated as a sequence of linear segments that are modeled as segments (states) in the model. The segmental semi-Markov model is an extension of the standard hidden Markov model (HMM), from which learning and inference algorithms are presented to solve the problems of change-point detection and pattern matching in a Bayesian framework. Results on both simulated data and real data from semiconductor manufacturing illustrate the flexibility and accuracy of the proposed framework.

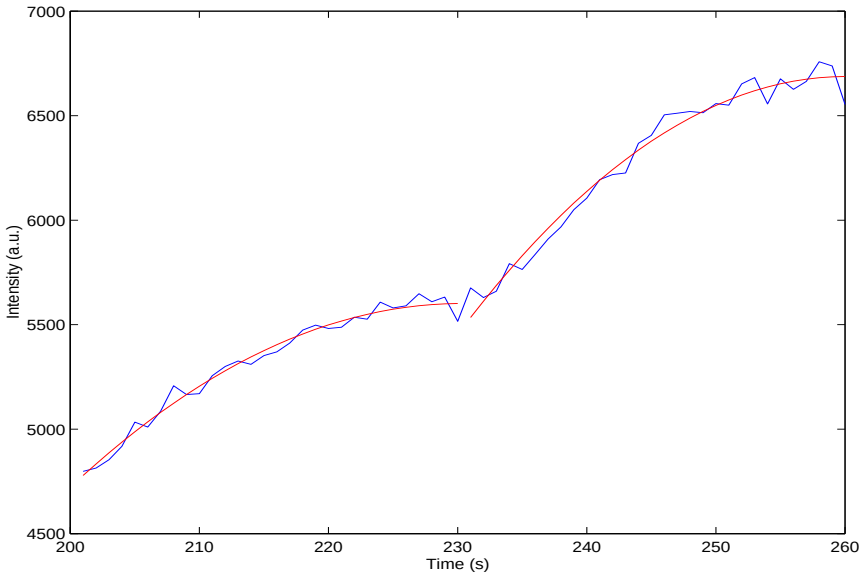

Figure 1: An illustrative example of a change-point detection problem from plasma etching. Data are from a commercial LAM 9400 plasma etch machine.

1 Introduction

Plasma etch is a critical process in semiconductor manufacturing. Accurate automatic detection of the end of the etch process is essential for reliable wafer processing. Figures 1 and 2 show two techniques for endpoint detection [19, 20, 2, 25] using interferometry sensor data from plasma etching:

1. *Change-point detection*: In Figure 1, a "change-point" occurs at the boundary between the two fitted quadratic curves. Process engineers are interested in detecting this change-point in real time for the purpose of endpoint detection.

2. *Pattern matching*: In Figure 2, a signature pattern (enlarged in the box in Figure 2(a)) is visually determined by process engineers to be a good detector of the endpoint. Given one example pattern (e.g., from a test run of the process), can we find similar patterns in subsequent runs? (E.g., in Figure 2(b).)

Solving either of these problems in an automated manner is non-trivial due to the inherent variability in end point signature from run to run. We address both problems in this paper using the general framework of a segmental semi-Markov model. This framework extends the standard hidden Markov model (HMM) to allow explicit state duration modeling (the semi-Markov model, e.g., [10]), and non-constant observations (the segmental model, e.g., [21]).

There has been extensive research into the plasma etch endpoint detection using optical emission spectroscopy and interferometry signals in recent years [20] [25] [24] [1] [5] [27] [23]. As mentioned

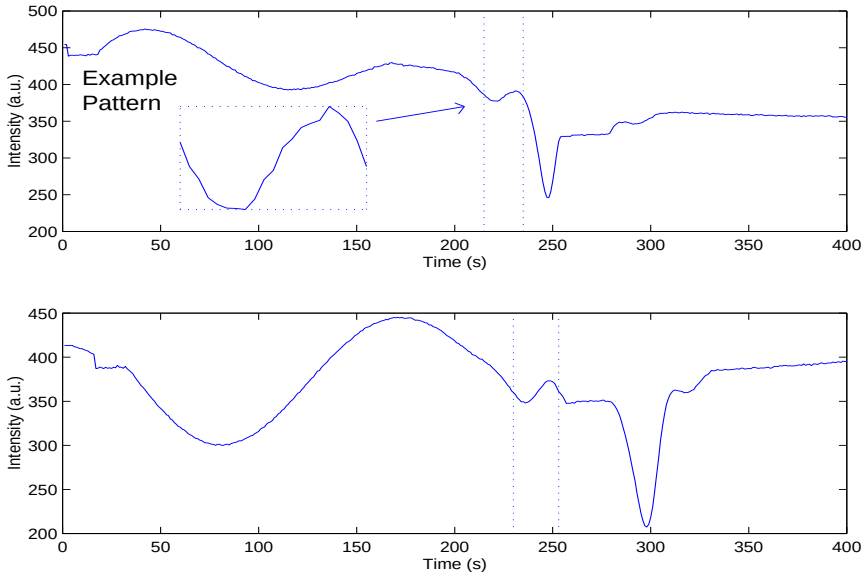

Figure 2: An example of an interferometry sensor data from plasma etching: (a) (top) a waveform pattern indicating the end of the plasma etch process is indicated with dotted line, (b) (bottom) another run of the same process where we wish to detect a similar pattern (indicated by dotted lines).

above, experienced process engineers can manually find the endpoint by inspecting the shape of the time waveform of selected spectral lines, and pattern recognition methods such as neural networks are then trained on example data for endpoint detection. One limitation with the neural network approach is that a substantial number of training examples may be needed to build the model. To remedy this problem Mundt [20] proposed synthesizing training data. To do this, a mathematical model has to be constructed for the endpoint. This is a nontrivial task to perform since it requires detailed prior knowledge and may need to be repeated for each new pattern. Another limitation of the neural network approach is that the learned model is a "black box." It is hard to interpret and it is difficult to incorporate expert knowledge of process engineers into the network model. Other pattern matching techniques have also been explored. Allen et al [2] used the Haar wavelet representation to model an endpoint pattern over many resolutions. Again, it is difficult to directly incorporate prior knowledge using this approach.

In this paper, we propose a novel approach to the modeling of the endpoint shape signature, based on the segmental semi-Markov model. Our approach can directly capture a process engineer's knowledge about endpoint characteristics, for example, the nature of a change-point between two polynomial segments, the shape of an example pattern, the expected location in time of the endpoint, and so forth. The segmental semi-Markov model was originally proposed in the speech recognition literature [10, 21] but to our knowledge this paper describes the first application to process sensor data analysis. The remainder of the paper is organized as follows. In Section 2, we describe the segmental semi-Markov model and the inference algorithms to compute the hidden states from observed data. In Sections 3 and Sections 4 we apply the model to the problems of change-point detection and pattern matching. We show how to build the model, apply the inference algorithms, and describe experimental results on both simulated and actual plasma etch data. Finally, Section 5 concludes with a discussion of future work.

2 Background on Segmental Hidden Semi-Markov Models

2.1 Hidden Markov Models

We begin with a brief review of the standard discrete-time finite-state Markov model. There are M states in the model: $1, 2, \ldots, M$. The state at time t is denoted s_t, where $t = 1, 2, \ldots$ is the time index. At time $t = 1$, the probability distribution over the states is given by $\boldsymbol{\pi}$, the initial state distribution, i.e., $p(s_1 = i) = \pi_i$, for $i \leq i \leq M$. Given the state s_t at time t, the state s_{t+1} at time $t + 1$ is given by the conditional probabilities $p(s_{t+1}|s_t)$, and conditionally independent of $s_{t-1}, s_{t-2}, \ldots, s_1$. The state transition matrix $\mathbf{A}$ specifies the conditional probabilities $A_{ij} = p(s_{t+1} = j|s_t = i)$, for $1 \leq i, j \leq M$, and $\sum_j A_{ij} = 1$, for $1 \leq i \leq M$.

A hidden Markov model (HMM) is a Markov model where the states s_t are not directly observable [22]. Instead, we can observe another measurement y_t (real-valued for the purpose of this paper) that is related to s_t by the stationary probability distribution $p(y_t|s_t)$, i.e., y_t is a stochastic function of the unobserved state s_t. Given the state s_t, the observation y_t is independent of all previous states $s_1, \ldots, s_{t-1}$, and all previous observations $y_1, \ldots, y_{t-1}$. Let θ_i be the set of parameters that specify the observation-state model $p(y_t|s_t = i)$, for $1 \leq i \leq M$, and where $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_M\}$. The functional form of $p(y_t|s_t = i)$ is often chosen from some relatively simple parametric family,

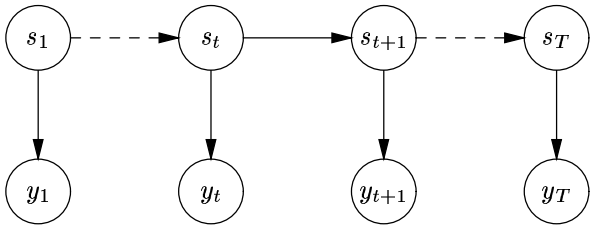

Figure 3: The graphical structure of a hidden Markov model.

e.g., Gaussian for real valued y_t, or a multinomial for categorical y_t. Note that θ_i does not depend on t, i.e., the model parameters do not change with time.

The graphical structure of the hidden Markov model is illustrated in Figure 3. The joint distribution of the model, based on the assumptions stated above, can be written in factored form as:

$$p(s_1, \ldots, s_T, y_1, \ldots, y_T) = p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t) \prod_{t=1}^{T} p(y_t|s_t). \tag{1}$$

There exists an efficient algorithm, called the Viterbi algorithm, that computes the most likely state sequence $\mathbf{s} = s_1 \ldots s_T$ from a given observation sequence $\mathbf{y} = y_1 \ldots y_T$, given the parameters $\lambda = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\theta}\}$ of the model. Furthermore, given an observed sequence $\mathbf{y}$ and fixing M, one can perform maximum likelihood estimation of the parameters λ of the model using the Expectation-Maximization (EM) algorithm [22].

We can model the change-point detection problem from Figure 1 with two states: "before the change-point" (first segment), and "after the change-point" (second segment). The data $\mathbf{y} = y_1 \ldots y_T$ are observed, but the corresponding states $\mathbf{s} = s_1 \ldots s_T$ (i.e., segment labels) are hidden. If we can estimate the state sequence $\mathbf{s} = s_1 \ldots s_T$, the change-point can be estimated by noting that the change-point is simply the first t such that $s_t = 2$.

The pattern matching problem of Figure 2 can also be formulated within the framework of hidden Markov models. The pattern can be approximated by a piecewise linear representation (see Figure 4), consisting of a number of linear segments, where the hidden states correspond to the segment labels. We will elaborate on the details of this model later in the paper.

2.2 Semi-Markov Models

For a finite state Markov model, the state duration d_i is the number of consecutive time-steps that the system stays in state i, after entering the state. In the standard Markov model, the distribution of d_i is given by

$$p(d_i = n) = A_{ii}^{n-1}(1 - A_{ii}), \quad n = 1, 2, \ldots \tag{2}$$

where A_{ii} is the self-loop transition probability of state i and n is the number of time-steps spent in state i. In other words, the state-duration distribution is constrained to be geometric in form. In reality other kinds of distributions, such as log-normal, may provide a more realistic model for certain applications. For example, in Figure 1 we have prior knowledge from the physics of the

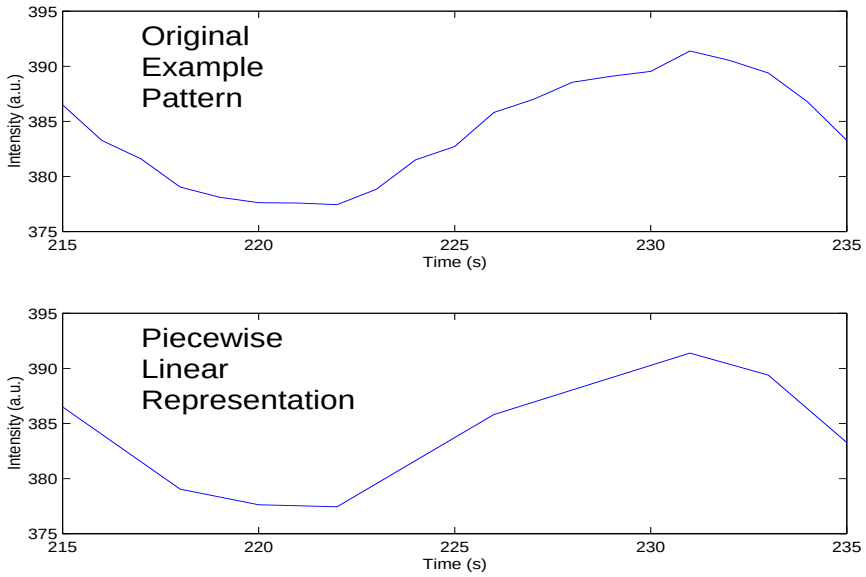

Figure 4: The example waveform pattern of Figure 2(a) and a piecewise linear representation.

plasma etch process that a change is more likely to occur about half-way through the process, rather than at the beginning or at the end. Thus a geometric duration, monotonically decreasing from its mode at time 1, is physically implausible.

The problem of modifying the standard Markov model to allow for arbitrary state-durations can be addressed by the use of *semi-Markov* models (e.g., [10]). A semi-Markov model has the following generative description:

- On entering state i a duration time d_i is drawn from a state-duration distribution $p(d_i)$.
- The process remains in state i for time d_i.
- At time d_i the process transitions to another state according to a transition matrix $\mathbf{A}$, and the process repeats.

The state-duration distributions, $p(d_i), 1 \leq i \leq M$, can be modeled using parametric distributions (such as log-normal, Gamma, etc) or non-parametrically by mixtures, histograms, kernel densities, and so forth. If d_i is constrained to take only integer values we get a discrete-time semi-Markov model. The discrete-time model is used throughout this paper since the observed measurements y_t are discrete-time sampled sensor signals. In a change-detection context, by including the state-duration distributions in the model, we can encode a prior on how long we expect the process to remain in each state. In applications where we have multiple runs of the same process, the prior can be adapted to the data over multiple runs, i.e., the parameters for the prior $p(d_i)$ can be recursively updated using a Bayesian estimation framework.

2.3 Segmental Observation Models

We have not yet described the functional form of the conditional densities $p(y_t|s_t)$ that relate the observed data to the hidden states. In the standard HMM framework, the observed y_t's depend only on the state s_t, not on the time t. This effectively models the data $\mathbf{y}$ as being piecewise constant with additive noise, as y_t will be governed by a constant mean over time within each state. There are many examples of real-world time-series where this piecewise-constant model is inappropriate, e.g., the data in Figure 1. A natural generalization of the piecewise-constant model is to allow each state to generate data in the form of a regression curve [21], i.e.,

$$y_t = f_i(t|\theta_i) + e_t \tag{3}$$

where $f_i(t|\theta_i)$ is a deterministic state-dependent regression function with parameters θ_i and e_t is additive independent noise (often assumed Gaussian, but not necessarily so). In the Gaussian noise case we get that $p(y_t|s_t = i)$ is Gaussian with a time-dependent mean $f_i(t|\theta_i)$ and with variance σ^2.

This segmental model allows us to directly model the shapes of the segments in the data, for example, the quadratic curves in the change-point detection problem (Figure 1), and the slopes of the linear segments in the pattern matching problem (Figures 2 and 4).

2.4 Segmental Semi-Markov Model

The discussion so far has generalized the standard hidden Markov model to the segmental hidden semi-Markov model, specified by

- $\boldsymbol{\pi}$, the initial state distribution,
- $\mathbf{A}$, the state transition matrix,
- $p(d_i)$, for $1 \leq i \leq M$, the state duration distributions,
- $y_t = f_i(t|\theta_i) + e_t$, for $1 \leq i \leq M$, the stochastic regression functions of the M states.

Let $n_0 = 0$ and $n_r = T$, and let there be r "same-state runs" in the data, where n_j, $1 \leq j \leq r$, denotes the final time index for each of the r runs, such that $s_1 = \cdots = s_{n_1}$, $s_{n_1+1} = \cdots = s_{n_2}$, ..., $s_{n_{r-1}+1} = \cdots = s_{n_r}$. The joint distribution of the model is

$$p(s_1, \ldots, s_T, y_1, \ldots, y_T) = p(s_1) \prod_{j=1}^{r-1} p(s_{j+1}|s_j) \prod_{j=1}^{r} \left[p(d_{s_{n_j}} = n_j - n_{j-1}) p(y_{n_{j-1}+1}, \ldots, y_{n_j}|s_{n_j}) \right]. \tag{4}$$

2.5 A Viterbi-like Algorithm to Compute the Most Likely State Sequence (MLSS) for a Segmental Semi-Markov Model

To find the most likely state (i.e., segment label) sequence $\hat{\mathbf{s}} = s_1 \ldots s_T$ for a data sequence $\mathbf{y} = y_1 \ldots y_T$, we have the following recursive Viterbi-like algorithm based on dynamic programming

(this is essentially the same algorithm as in [21]). At each time $t < T$, this algorithm calculates the quantity $\hat{p}_i^{(t)}$ for each state i, $1 \le i \le M$, where $\hat{p}_i^{(t)}$ is defined as

$$\hat{p}_i^{(t)} = \max_{\mathbf{s}} \{p(\mathbf{s}|y_1 \dots y_t) | \mathbf{s} = s_1 \dots s_t, s_t = i, s_{t+1} \neq i\}. \tag{5}$$

In other words, $\hat{p}_i^{(t)}$ is the likelihood of the most likely state sequence that ends with state i, and y_t is the last point of segment i. At time $t = T$, we define $\hat{p}_i^{(T)}$ as

$$\hat{p}_i^{(T)} = \max_{\mathbf{s}} \{p(\mathbf{s}|y_1 \dots y_T) | \mathbf{s} = s_1 \dots s_T, s_T = i\}. \tag{6}$$

By definition, the most likely state sequence for the data sequence $y_1 \dots y_T$ will be the state sequence $s_1 \dots s_T$ with likelihood $\max_i \hat{p}_i^{(T)}$.

The recursive function for calculating $\hat{p}_i^{(t)}$ is

$$\hat{p}_i^{(t)} = \max_{t'} \Big(\max_j \Big[\hat{p}_j^{(t')} A_{ji} \Big] p(d_i = t - t') p(y_{t'+1} \dots y_t | \theta_i) \Big), \tag{7}$$

for $1 \le i \le M$. In the above equation, t' is either the last point of the previous segment, or 0. If $t' = 0$, $\max_j \left[\hat{p}_j^{(t')} A_{ji} \right]$ is replaced with π_i. If $t = T$, $p(d_i = t - t')$ is replaced with $p(d_i \ge T - t')$.

The time t' and the state j that maximize Equation 7 are recorded in a table: $PREV(i,t) \leftarrow (j,t')$. Let $j = \arg\max_i p_i^{(T)}$. We can trace back from $PREV(j,T)$ through the table $PREV$ to get the most likely state sequence. Figure 5 summarizes the procedure in pseudo-code.

The space complexity of the MLSS algorithm is $O(MT)$ for storing $\hat{p}_i^{(t)}$, $PREV(i,t)$, for $1 \le i \le M$, $1 \le t \le T$. Let $D_i = \max d_i - \min d_i + 1$ and $D = \sum_{i=1}^{M} D_i$. At time t, when y_t is available, the computation time complexity for calculating $\hat{p}_i^{(t)}$ (Equation 7) is $O(D_i M)$, so the total time complexity (for calculating $\hat{p}_i^{(t)}$ for all $1 \le i \le M$) is $O(DM)$.

2.6 Forward-backward Algorithm

Another inference task is to calculate $p(s_t = i | y_1, \dots, y_T)$, the posterior probability that a data point belongs to state i, given all the observations. In the standard hidden Markov model, this can be done by using the forward-backward algorithm [22]. Here we extend this algorithm for the segmental semi-Markov model, again in a manner similar to that described in [21].

Let $s_{t_1,t_2} = i$ be an abbreviation for

$$s_{t_1-1} \neq i, s_{t_1} = \dots = s_{t_2} = i, s_{t_2+1} \neq i, \tag{8}$$

and define

$$\begin{aligned} \alpha(i,t_1,t_2) \quad = \quad & p(s_{t_1,t_2} = i, y_1, \dots, y_{t_2}), \\ & \text{for } 1 \le i \le M, \\ & \quad 1 \le t_1, t_2 \le T, \\ & \quad p(d_i = t_2 - t_1 + 1) > 0, \end{aligned} \tag{9}$$

```
function s_1 ... s_T = MLSS(y_1 ... y_T)
 1. for t=1 to T
 2.     for i=1 to M
 3.         Compute p̂_i^(t), PREV(i,t);
 4.     end for
 5. end for
 6. j = arg max_i p̂_i^(T);
 7. t = T;
 8. [j', t'] = PREV(j,t);
 9. for k = t' + 1 to t
10.     s_k = j;
11. end for
12. if (t' > 0)
13.    [j,t] = [j',t'];
14.    goto 8;
15. else
16.    return;
17. end if
```

Figure 5: Pseudo-code for MLSS (finding the most likely state sequence $s_1 \dots s_t$ for data sequence $y_1 \dots y_t$).

and

$$\begin{aligned} \beta(i,t) &= p(y_{t+1}, \dots, y_T | s_t = i), \\ &\quad \text{for } 1 \le i \le M, \quad 1 \le t \le T. \end{aligned} \tag{10}$$

The forward pass of the algorithm calculates α's:

1. Initialization ($t_1 = 1$).

$$\alpha(i, 1, t_2) = \pi_i p(d_i = t_2) p(y_1, \dots, y_{t_2} | \theta_i), \tag{11}$$

   where $p(d_i = t_2)$ is replaced with $p(d_i \ge T)$ when $t_2 = T$.

2. Forward recursion ($t_1 > 1$). For $t_2 = 2, 3, \dots, T$,

$$\begin{aligned} \alpha(i, 1, t_2) &= \sum_j \Big(A_{ji} \sum_{t_1'} \alpha(j, t_1', t_1 - 1) \Big) \\ &\quad p(d_i = t_2 - t_1 + 1) p(y_{t_1}, \dots, y_{t_2} | \theta_i), \end{aligned} \tag{12}$$

   where $p(d_i = t_2 - t_1 + 1)$ is replaced with $p(d_i \ge T - t_1 + 1)$ when $t_2 = T$.

The backward pass calculates β's:

1. Initialization ($t = T$).

$$\beta(i, T) = 1, \qquad \text{for } 1 \le i \le M. \tag{13}$$

2. Backward recursion ($t < T$). For $t = T-1, T-2, \ldots, 1$,

$$\begin{aligned} \beta(i,t) &= \sum_j \Big(A_{ij} \sum_{t'} \big[\beta(j,t') \\ &\qquad p(d_j = t' - t) p(y_{t+1}, \ldots, y_{t'} | \theta_j) \big] \Big), \end{aligned} \tag{14}$$

where $p(d_j = t' - t)$ is replaced with $p(d_j \ge T - t)$ when $t' = T$.

Let $\mathbf{y} = y_1, \ldots, y_T$. Now we have

$$\begin{aligned} p(s_{t_1,t_2} = i, \mathbf{y}) &= p(s_{t_1,t_2} = i, y_1, \ldots, y_{t_2}, y_{t_2+1}, \ldots, y_T) \\ &= p(s_{t_1,t_2} = i, y_1, \ldots, y_{t_2}) \\ &\qquad p(y_{t_2+1}, \ldots, y_T | s_{t_1,t_2} = i, y_1, \ldots, y_{t_2}) \\ &= p(s_{t_1,t_2} = i, y_1, \ldots, y_{t_2}) \\ &\qquad p(y_{t_2+1}, \ldots, y_T | s_{t_2} = i) \\ &= \alpha(i, t_1, t_2) \beta(i, t_2). \end{aligned} \tag{15}$$

and

$$p(s_t = i, \mathbf{y}) = \sum_{t_1 \le t \le t_2} p(s_{t_1,t_2} = i, \mathbf{y}), \tag{16}$$

$$\begin{aligned} p(s_t = i | \mathbf{y}) &= \frac{p(s_t = i, \mathbf{y})}{p(\mathbf{y})} \\ &\propto p(s_t = i, \mathbf{y}) \\ &= \sum_{t_1 \le t \le t_2} \alpha(i, t_1, t_2) \beta(i, t_2). \end{aligned} \tag{17}$$

Let $D_i = \max d_i - \min d_i + 1$ and $D = \sum_{i=1}^{M} D_i$. The space complexity of the forward-backward algorithm is $O(DT)$ for storing all the α's and β's. At time t, when y_t is available, the computation time complexity is $O(DM) + O(DMt) = O(DMt)$, where $O(DM)$ is the time complexity of one forward step of calculating α's, and $O(DMt)$ is the time complexity of t backward steps of calculating β's.

3 Change-based Endpoint Detection

In this section, we apply the segmental semi-Markov model to solve the problem of change-point detection for plasma etch. There is a long history of work on change detection in statistics and engineering [3, 17]. Of direct relevance to the type of data in Figure 1 is the prior work on piecewise regression [13, 12], also called segmented regression [18, 9], or multi-phase regression [14]. When

the number of segments is known a priori, these techniques can be viewed simply as trying to minimize the sum of squared errors (SSE) when fitting regression functions to the segments. This SSE approach is quite similar to the semi-Markov model approach with the important exception that it does not use any prior information on where the change is expected to occur. The SSE based method will be compared with our new method which is based on the segmental semi-Markov model. We will denote the former by "SSE," the latter by "SEGHMM." Details of the SSE method are provided in Appendix A.

3.1 Change-point Detection Algorithms

For the problem of change-point detection, we propose a 2-state segmental semi-Markov model:

- State 1: before the change point (first segment),
- State 2: after the change point (second segment).

As the process will start with state 1 and transition to state 2, the initial state distribution is

$$\boldsymbol{\pi} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{18}$$

and the state transition matrix is

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{19}$$

The state duration distribution of state 1 is set to reflect prior knowledge about when the change-point will occur. For example, if we expect that the change will occur approximately at time $\mu_c \pm 20\%$, we can use a truncated normal distribution

$$p(d_1) \propto \begin{cases} \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(d_1-\mu_c)^2}{2\sigma_c^2}}, & \mu_c - 3\sigma_c \le d_1 \le \mu_c + 3\sigma_c \\ 0, & \text{otherwise,} \end{cases} \tag{20}$$

where $3\sigma_c = \mu_c \times 20\%$. As we are not interested in the duration of state 2, we set its distribution to be

$$p(d_2) \propto 1, \qquad \text{for } d_2 \ge 0. \tag{21}$$

3.2 Estimating the Regression Parameters by the EM Algorithm

If the regression parameters of the segments are not known *a priori*, they can be learned from data by the Expectation-Maximization (EM) algorithm [6], which is a powerful framework for parameter estimation from incomplete data. For our change-point detection problem, if we knew which data points belong to state 1, and which data points belong to state 2, we could directly estimate the regression parameters. Conversely, if we knew the regression parameters, we could calculate for each data point the probabilities that it belongs to state 1 and state 2. The EM algorithm solves this "chicken-and-egg" problem as follows:

- Start with $\hat{\boldsymbol{\theta}}$, an initial "guess" of the regression parameters.
- Iterate over the two steps below until convergence:
  1. Calculate $p(s_t = i|\mathbf{y}, \hat{\boldsymbol{\theta}})$, for $i = 1, 2$, $t = 1, 2, \ldots$, using the forward-backward algorithm in Section 2.6.
  2. Re-estimate $\hat{\boldsymbol{\theta}}$. Weighted linear regression [7] is used to estimate the regression parameters for each segment. For example, the weights used for segment 1 are $p(s_t = 1|\mathbf{y}, \hat{\boldsymbol{\theta}})$, for $t = 1, 2, \ldots$.

It can be shown that EM converges to at least a local maximum of the likelihood function in the parameter space [6].

3.3 Estimating the Change-Point

After the regression parameters for the two segments are estimated via EM, our model is fully specified, and we can apply the algorithm in Section 2.5 to find the most likely state sequence (MLSS) from the observed data $\mathbf{y} = y_1 \ldots y_t \ldots$. The change-point will be the smallest t such that $s_t = 2$. We denote this variant of the SEGHMM method as "MLSS."

For a given sequence of observed data $\mathbf{y} = y_1 \ldots y_T$, there are, in theory, $T - 1$ possible state sequences:

	s_1	s_2	s_3	$\cdots$	s_{t-1}	s_t	s_{t+1}	$\cdots$	s_{T-1}	s_T
$\mathbf{s}^{(2)}$:	1	2	2	$\cdots$	2	2	2	$\cdots$	2	2
$\mathbf{s}^{(3)}$:	1	1	2	$\cdots$	2	2	2	$\cdots$	2	2
						$\cdots$				
$\mathbf{s}^{(t)}$:	1	1	1	$\cdots$	1	2	2	$\cdots$	2	2
						$\cdots$				
$\mathbf{s}^{(T)}$:	1	1	1	$\cdots$	1	1	1	$\cdots$	1	2

Each state sequence provides an estimate of the location of the change-point. For example, in $\mathbf{s}^{(t)}$ the location of change-point is t. Instead of relying solely on the decision of the single most likely state sequence, we can pool together the decisions of all the possible state sequences $\mathbf{s}$, weighted by their posterior probabilities $p(\mathbf{s}|\mathbf{y})$. The estimated change time is the weighted average

$$\hat{t}_c = \sum_{t=2}^{T} t \times p(\mathbf{s}^{(t)}|\mathbf{y}). \tag{22}$$

Note that

$$p(s_t = 1|\mathbf{y}) = \sum_{t'=t+1}^{T} p(\mathbf{s}^{(t')}|\mathbf{y}), \tag{23}$$

and similarly,

$$p(s_{t-1} = 1|\mathbf{y}) \quad = \quad \sum_{t'=t}^{T} p(\mathbf{s}^{(t')}|\mathbf{y}) \tag{24}$$

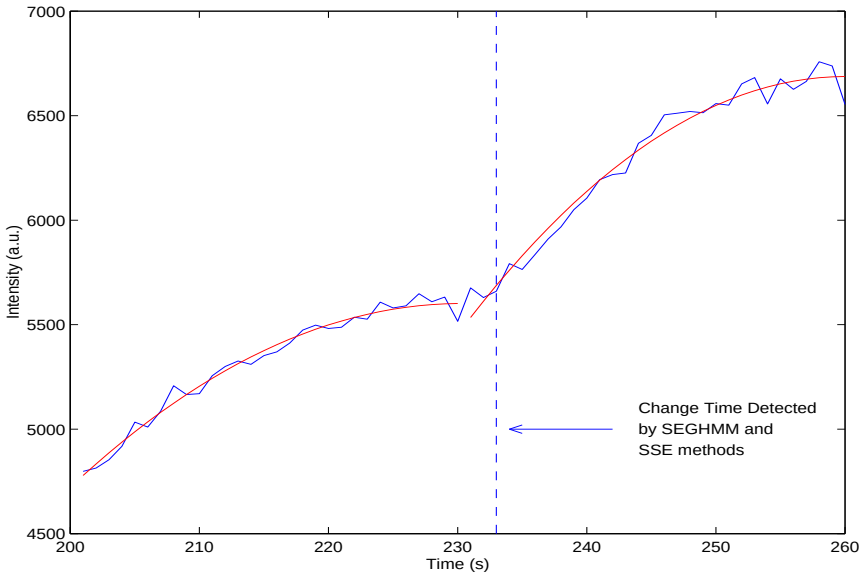

Figure 6: Detected change-points by SEGHMM and SSE methods for plasma etch data.

$$= p(\mathbf{s}^{(t)}|\mathbf{y}) + \sum_{t'=t+1}^{T} p(\mathbf{s}^{(t')}|\mathbf{y}) \tag{25}$$

$$= p(\mathbf{s}^{(t)}|\mathbf{y}) + p(s_t = 1|\mathbf{y}), \tag{26}$$

so we have

$$p(\mathbf{s}^{(t)}|\mathbf{y}) = p(s_{t-1} = 1|\mathbf{y}) - p(s_t = 1|\mathbf{y}), \tag{27}$$

where $p(s_{t-1} = 1|\mathbf{y})$ and $p(s_t = 1|\mathbf{y})$ can be computed efficiently by the forward-backward algorithm in Section 2.6. Substituting the above equation into Equation 22, we have

$$\hat{t}_c = \sum_{t=1}^{T} \Big(t \times \Big[p(s_{t-1} = 1|\mathbf{y}) - p(s_t = 1|\mathbf{y}) \Big] \Big). \tag{28}$$

We denote this variant of the SEGHMM method as "weighted."

3.4 Experimental Results on Change-based Endpoint Detection

We applied the SSE method, and both the MLSS and weighted variants of the SEGHMM methods to interferometry sensor data from an etch run on a LAM 9400 plasma etch machine (Figure 1). For the SEGHMM methods, the prior on the location of the change-point (i.e., the duration of the first state) is set to be flat ($p(d_1) \propto 1$). All three methods estimated the change-point to be at 233, close to the manually marked time of 231.

To systematically compare the SEGHMM and SSE methods, they were also tested on simulated data in the following manner:

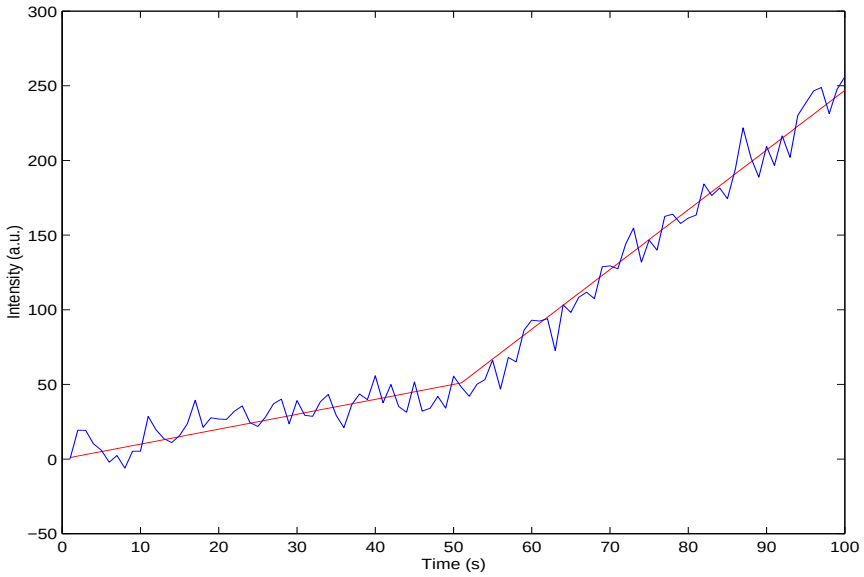

Figure 7: Synthetic data with 2 linear segments. The slopes are $k_1 = 1$, $k_2 = 2$. The noise $\sigma_y = 10$.

- Data were simulated from a waveform of 100 points consisting of the two linear segments with additive Gaussian noise (zero mean and variance σ^2). The slopes of the two segments are $k_1 = 1$, $k_2 = 4$, respectively. See Figure 7 for an example of the simulated data.
- The true change-point from one segment to the other is sampled from a truncated normal distribution with mean 50 and standard deviation 5. This distribution was used as the prior in the segmental semi-Markov model. To test the sensitivity of the SEGHMM method to specification of the prior, we also looked at the case when no information on prior is available, in which case we used the flat prior, i.e., all points are equally likely to be a change point. Depending on whether the prior is used, and whether the "MLSS" or "weighted" approach is used to find the change point, we have four different variations of the SEGHMM method.
- 10000 random realizations of the process were generated and detections were made by the SSE method and the four variations of the SEGHMM method.
- The experiment was repeated for three different noise levels: $\sigma = 5$, $\sigma = 10$, $\sigma = 15$.

Figure 8 shows a histogram of the errors of the detected change time for each of the SSE method and the SEGHMM method (with prior, "weighted") for $\sigma = 5$. The SEGHMM method is clearly superior in that the errors tend to be much smaller. Figure 9 shows the mean absolute errors. As the noise level increases the relative improvement from using the SEGHMM method also increases. This is as we might expect, since as the ambiguity in the data increases a probabilistic model will be better able to deal with the ambiguities in the data compared to a non-probabilistic approach.

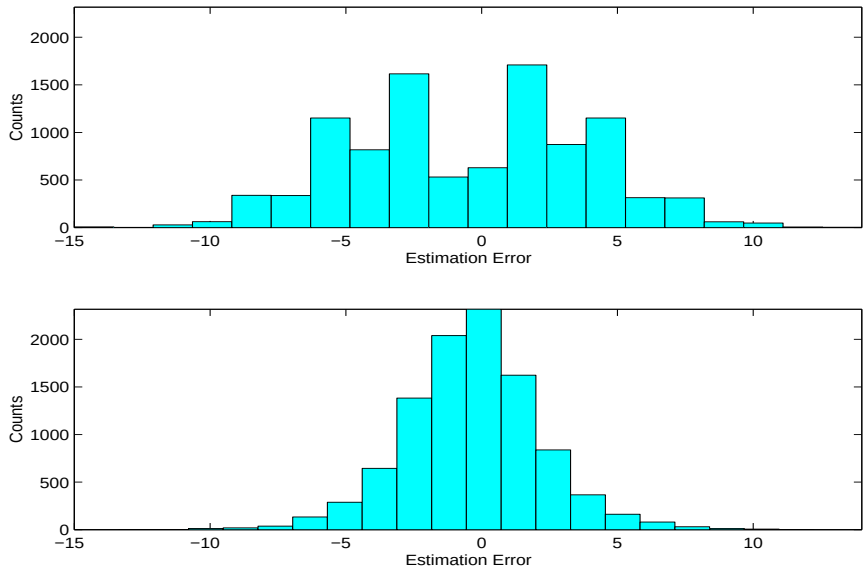

Figure 8: Histograms of the detection time errors for the SSE method (top) and the SEGHMM method (with prior, "weighted") (bottom), $\sigma = 10$.

Conversely, for low-noise situations, the detection problem will be relatively easy and we can expect relatively little difference between the two methods.

Additionally, we can see from the figure that the "weighted" variations of the SEGHMM method are much better than the "MLSS" variations, and that, not surprisingly, "with prior" is better than "without prior." So the knowledge about the prior improves the performance, but it is not essential: without it, the performance is still much better than the SSE method and relatively close to performance with the prior. The main advantage of the "weighted" SEGHMM method appears to come from the fact that it averages out the uncertainty over the location of the change-point rather than picking the single best segmentation (MLSS and SSE methods).

For the results above, we have applied the SEGHMM method in an off-line manner; that is, at time T after all y_t measurements are available. In this case, the SEGHMM method has been shown to be very effective in accurately pinpointing the location of the change point. We also tested the SEGHMM method in an on-line manner, but found it to not significantly superior than the SSE method, in part due to the trade-off between the false alarm rate and detection delays.

4 Pattern-Based End-point Detection

4.1 Building A Segmental Semi-Markov Model From An Example Pattern

For certain sensor and endpoint problems, the endpoint is indicated by a distinctive pattern or waveform, rather than a simple change-point (e.g., see Figure 2). From the interferometry data in Figure 2(a), an engineer can manually detect the endpoint by marking the pattern (enclosed in the

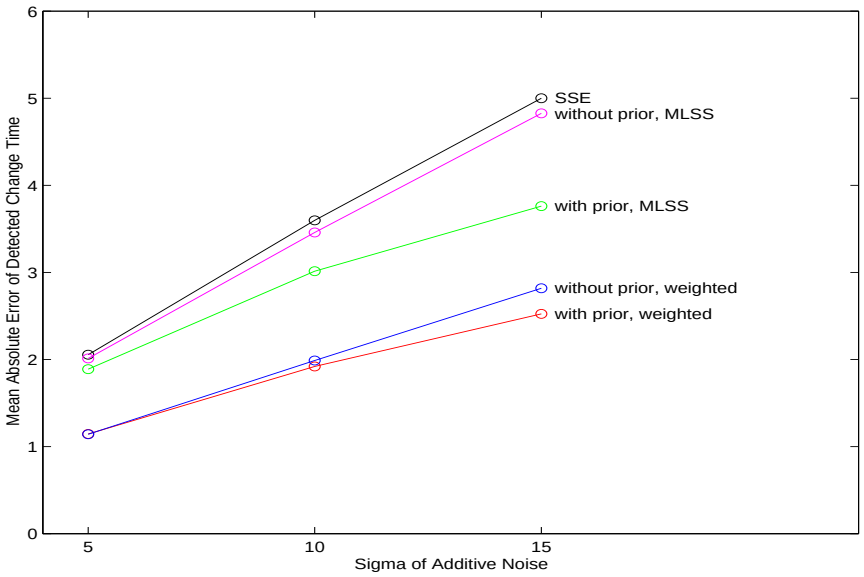

Figure 9: Comparison of the SSE method and the four variations of the SEGHMM method (with or without prior, "weighted" or MLSS), in terms of the mean absolute errors of detected change times.

dotted rectangle). The problem is to automatically detect similar patterns in the interferometry data of future runs, e.g., Figure 2(b).

To build a segmental semi-Markov model from the example pattern, we first approximate the example pattern as a sequence of linear segments. This process is called piecewise linear segmentation, for which there exist many algorithms, e.g., [15], [28]. For simplicity, we are using a simple recursive algorithm (similar to the algorithm of [8]; see also [16], p. 364), which takes as input a sequence of points $(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)$ and the error tolerance ε, and output a sequence of linear segments:

1. Connect the end points A, B. Let the equation for the line AB be $y = kx + b$.
2. Find the point C between A, B with the maximum error $|y_C - (kx_C + b)|$.
3. If $|y_C - (kx_C + b)| > \varepsilon$, break the line AB into two segments AC, CB, and repeat this process on these two segments.

The error tolerance ε can be set manually, or estimated in a robust manner from the data by median filtering as follows:

1. Run the waveform $y_1 \ldots y_n$ through a median filter, and let the resulting smooth signal be $z_1 \ldots z_n$.
2. Let $d_i = |z_i - y_i|$, for $i = 1, \ldots, n$.

3. ε is set to be the third quartile of $\{d_1, \ldots, d_n\}$.

Let M be the number of segments in the piecewise linear representation of the example pattern. We build an M-state segmental semi-Markov model. The initial state distribution is $\boldsymbol{\pi} = [1, 0, \ldots, 0]$. The transition matrix $\mathbf{A}$ is defined as

$$A_{ij} = \begin{cases} 1, & \text{if } i = j + 1 \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

Let l_i be the length of the ith linear segment in the piecewise linear representation. The duration distribution of the ith segment is modeled as a truncated normal distribution with mean $\mu_{d_i} = l_i$, and standard deviation $\sigma_{d_i} = l_i \times 20\%/3$, similar to Equation 20.

The regression function of the ith segment is a linear function

$$y_t = k_i t + b_i + e_t, \tag{30}$$

where k_i, b_i are the slope and the intercept, repsectively, and e_t is the additive Gaussian noise. The slope k_i is set to that of the ith linear segment in the piecewise linear representation. We let the intercept b_i to float freely in our model as it depends on the starting point of the segment. Furthermore, we assume that the noise variances of all the segments are the same: σ^2, which is estimated as the noise in the training example, given its piecewise linear representation.

4.2 Pattern-matching Algorithm

To detect such a waveform inside a (much longer) time series $y_1 \ldots y_t \ldots$, an obvious approach would be to match the model against every subwindow $y_i y_{i+1} \ldots y_j$, find the most likely state sequence $s_i s_{i+1} \ldots s_j$, and declare "found" if the likelihood is above a certain threshold. The problems with this approach are (1) how to set the likelihood threshold and (2) the redundant computation from the fact that the computation for every subwindow will need to be completely redone, even if a subwindow overlaps with another subwindow.

To deal with these problems, we augment the model with two extra "background" states: a *pre-pattern* background state (state 0) to model the data before the pattern, and a *post-pattern* background state (state $K + 1$) for the data after the pattern. This augmented model may be seen as a "global" model that can be matched directly against the whole time series (instead of the subwindows). Similar ideas have been used in speech recognition to detect specific words within long speech sequences [26]. With this augmented model, we run the MLSS algorithm of Section 2.5 on-line as new data points $y_1 \ldots y_t \ldots$ are coming in. If, at time t, the most likely state sequence ends with $s_t = K$, where K is the last segment in the waveform, we declare that the waveform is detected with end time at y_t. See Figure 10 for pseudo-code.

4.3 Experimental Results on Pattern-Based End-point Detection

To test the above algorithm, a segmental semi-Markov model was fitted to the example pattern in Figure 2(a), using piecewise linear segmentation and duration parameters as described in Section 4.1, and the pattern matching algorithm was run on the data in Figure 2(b). The algorithm

```
procedure DETECT-PATTERN(y_1 ... y_t ...)
1. t = 1;
2. s_1 ... s_t = MLSS(y_1 ... y_t);
3. if (s_t == K)
4.    declare 'found';
5.    stop;
6. else
7.    t = t + 1;
8.    goto 2;
9. end if
```

Figure 10: Pseudo-code for DETECT-PATTERN (on-line detection of waveform).

correctly found the matched pattern between time 230 and 252. See Figure 11 for another example of applying the pattern matching algorithm, again using the same methodology for pattern modeling and detection.

4.4 Comparison with Squared-errors Based Methods

It is informative to compare our pattern matching method with a baseline squared-error based method that defines the distance or dissimilarity between two patterns as the root mean squared error, i.e., template matching or cross-correlation. To apply such method in an on-line manner, we need to set a threshold on the squared error distance measure. If the distance is lower than the threshold, we can declare that two patterns are similar to each other. But the setting of such a threshold must often be done in *ad hoc* manner. Also, in our problems, similar patterns may have different dynamic ranges, so the data require preprocessing to allow shifting and scaling in amplitude. It is difficult to appropriately incorporate domain knowledge into this kind of preprocessing. To allow more flexible matching of patterns, dynamic time warping (DTW) can be used to allow warping of the time axis (see Appendix B). In other words, time can be compressed or stretched, so that two patterns can align together [4]. However, it is also difficult to incorporate domain knowledge into the definition of the distance function for time warping. (See [11] for further discussion.)

The results of running the squared-error based method and dynamic time warping on the same data used in Section 4.3 are shown in Figure 12 and Figure 13. In both cases, the patterns are mean-shifted (i.e., to make the means 0) when they are compared. See Appendix C for details. The global minima in the two resulting error curves do not correspond to the best match, i.e., both minima correspond to false alarms around time $t = 90$ seconds. Similarly, for the data in Figure 11, the global minima of the squared-error based method and dynamic time warping are at times $t = 327$ seconds and $t = 381$ seconds, respectively, neither of which corresponds to the best match.

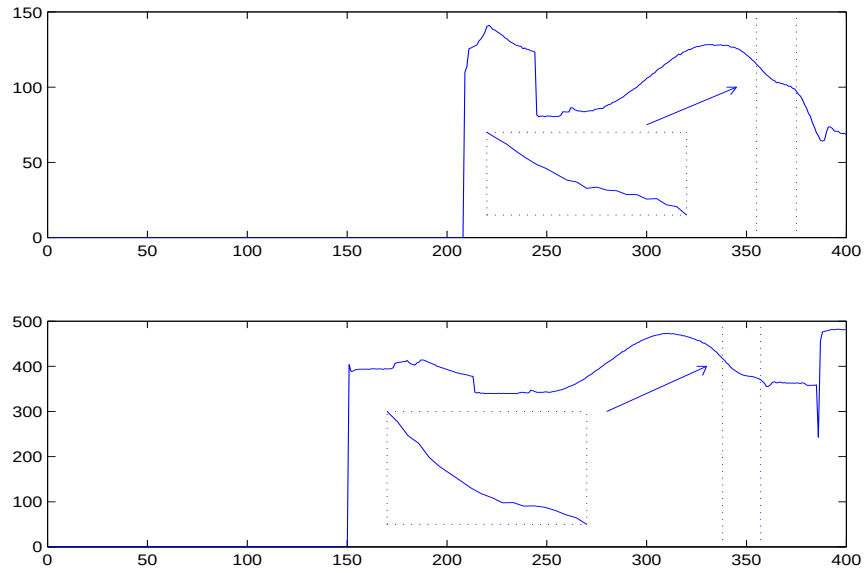

Figure 11: Another example of applying our pattern matching algorithm to plasma etching endpoint detection.

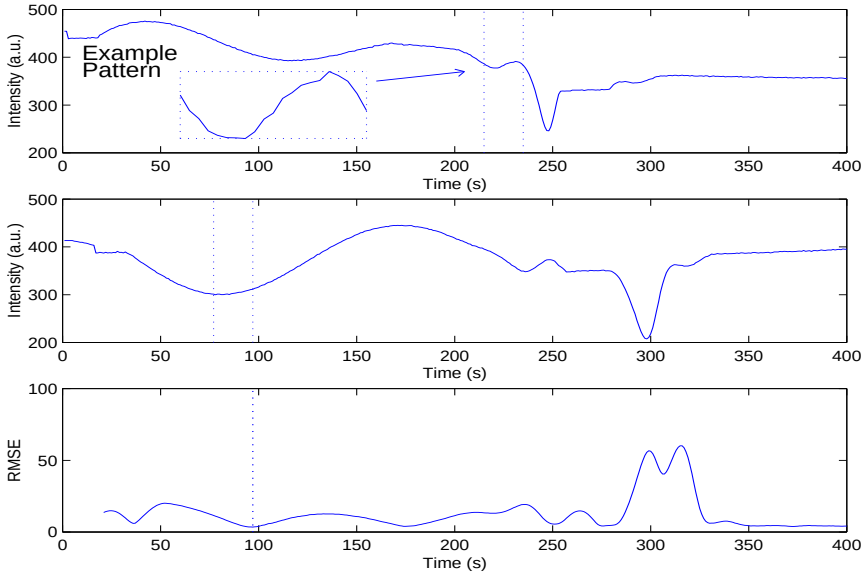

Figure 12: Results of applying a simple template matching technique to the data in Figure 2.

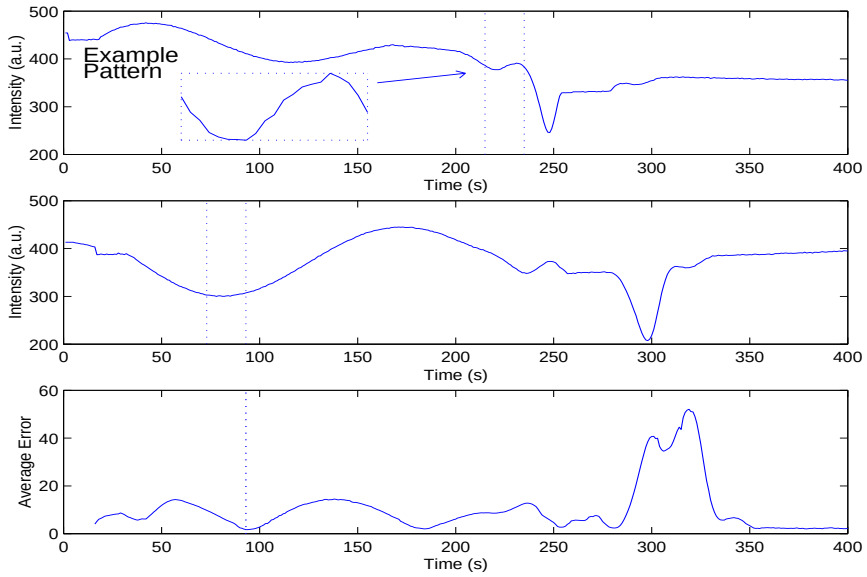

Figure 13: Results of applying dynamic time-warping to the data in Figure 2.

5 Conclusion and Future Work

In this paper, we applied segmental hidden semi-Markov models to the problems of plasma etch endpoint detection. The models provide a useful, flexible, and accurate framework for change-point detection and pattern matching. By modeling the problem within a generative model framework (including notions of state and time explicitly) one can incorporate prior knowledge in a principled manner and use the tools of probabilistic inference to infer change-points and pattern in an optimal manner. The proposed techniques were shown to be more accurate than non-probabilistic alternatives such as dynamic time-warping on both real and simulated plasma etch data.

Acknowledgments

We would like to thank Wenli Collison, Tom Ni, and David Hemker of LAM Research for providing the plasma etch data and for discussions on change-point detection in plasma etch processes.

References

[1] R. L. Allen, R. Moore, and M. Whelan. Multiresolution pattern detector networks for controlling plasma etch reactors. In *Proceedings of the SPIE - The International Society for Optical Engineering, vol.2637*, pages 19–30, October 1995.

[2] R. L. Allen, R. Moore, and M. Whelan. Application of neural networks to plasma etch end point detection. *Journal of Vacuum Science & Technology B (Microelectronics and Nanometer Structures)*, 14:498–503, 1996.

[3] Michèle Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Inc., Englewood Cliffs, N.J., April 1993.

[4] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, July 1994.

[5] P. Biolsi, D. Morvay, L. Drachnik, and S. Ellinger. An advanced endpoint detection solution for $< 1\%$ open areas. *Solid State Technology*, 39(12):59,61,62,64,67, December 1996.

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

[7] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, Inc, 1998.

[8] R. O. Duda and P. E. Hart. *Pattern Recognition and Scene Analysis*. John Wiley, 1973.

[9] S. R. Esterby and A. H. El-Shaarawi. Inference about the point of change in a regression model. *Applied Statistics*, 30(3):277–285, 1981.

[10] J. D. Ferguson. Variable duration models for speech. In *Proc. Symposium on the Application of Hidden Markov Models to Text and Speech*, pages 143–179, October 1980.

[11] X. Ge and P. Smyth. Deformable Markov model templates for time-series pattern matching. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 81–90, August 20-23, 2000.

[12] Fredrik Gustafsson. *Adaptive Filtering and Change Detection*. John Wiley & Sons, Ltd, 2000.

[13] D. M. Hawkins. Point estimation of the parameters of piecewise regression models. *Applied Statistics*, 25(1):51–57, 1976.

[14] D. V. Hinkley. Inference in two-phase regression. *Journal of the American Statistical Association*, 66:736–743, December 1971.

[15] H. Imai and M. Iri. An optimal algorithm for approximating a piecewise linear function. *Journal of Information Processing*, 9(3):159–162, 1986.

[16] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.

[17] T. L. Lai. Sequential changepoint detection in quality control and dynamical systems. *Journal of the Royal Statistical Society, Series B (Methodological)*, 57(4):613–658, 1995.

[18] P. M. Lerman. Fitting segmented regression models by grid search. *Applied Statistics*, 29(1):77–84, 1980.

[19] H. Maynard, E. Rietman, J. T. C. Lee, and D. Ibbotson. Plasma etching endpointing by monitoring RF power systems with an artificial neural network. In M. Meyyapan, D. J. Economou, and S. W. Butler, editors, *Proceedings of the Symposium on Process Control, Diagnostics, and Modeling in Semiconductor Manufacturing Proceedings of Process Control Diagnostics, and Modeling in Semiconductor Manufacturing*, pages 189–207, Pennington, NJ, USA, May 1995. Electrochem. Soc.

[20] R. Mundt. Model based training of a neural network endpoint detector for plasma etch applications. In M. Meyyapan, D. J. Economou, and S. W. Butler, editors, *Proceedings of the Symposium on Process Control, Diagnostics, and Modeling in Semiconductor Manufacturing Proceedings of Process Control Diagnostics, and Modeling in Semiconductor Manufacturing*, pages 178–188, Pennington, NJ, USA, May 1995. Electrochem. Soc.

[21] M. Ostendorf, V. V. Digalakis, and O. A. Kimball. From HMM's to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378, September 1996.

[22] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[23] S. Rangan, C. Spanos, and K. Poolla. Modeling and filtering of optical emission spectroscopy data for plasma etching systems. In *1997 IEEE International Symposium on Semiconductor Manufacturing Conference Proceedings*, pages B41–4, New York, NY, USA, October 1997. Semiconduct. Equipment and Mater. Int, IEEE.

[24] E. A. Rietman, R. C. Frye, E. R. Lory, and T. R. Harry. Active neural network control of wafer attributes in a plasma etch process. *Journal of Vacuum Science & Technology B (Microelectronics Processing and Phenomena)*, 11(4):1314–1316, 1993.

[25] D. A. White, B. E. Goodlin, A. E. Gower, D. S. Boning, H. Chen, H. H. Sawin, and T. J. Dalton. Low open-area endpoint detection using a PCA-based T^2 statistic and Q statistic on optical emission spectroscopy measurements. *IEEE Transactions on Semiconductor Manufacturing*, 13(2):193–207, May 2000.

[26] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. R. Goldman. Automatic recognition of keywords in unconstrained speech using hidden markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(11):1870–1878, November 1990.

[27] K. Wong, A. D. S. Boning, B. H. H. Sawin, S. W. Butler, and E. M. Sachs. Endpoint prediction for polysilicon plasma etch via optical emission interferometry. *Journal of Vacuum Science & Technology A (Vacuum, Surfaces, and Films)*, 15(3):1403–1408, 1997.

[28] Y. Zhu and L. D. Seneviratne. Optimal polygonal approximation of digitized curves. *IEE proceedings. Vision, image, and signal processing*, 144(1):8–14, February 1997.

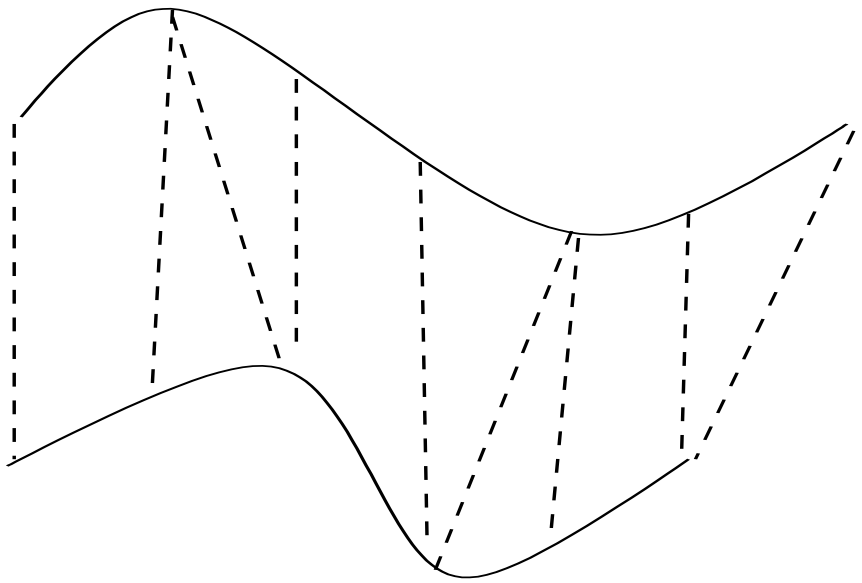

Figure 14: Matching two time series patterns of different lengths by dynamic time warping (DTW).

A SSE Method of Change-point Detection

When there is one change-point in a given data sequence $\mathbf{y} = y_1 \dots y_T$, the SSE method of change-point detection tries to minimize the sum of squared errors (SSE) when fitting regression functions to the two segments. In other words, the change-point is

$$\text{argmin}_t \left\{ \text{SSE}(y_1 \dots y_t) + \text{SSE}(y_{t+1} \dots y_T) \right\}, \tag{31}$$

where $\text{SSE}(y_i \dots y_j)$ is the minimum sum of squared errors when fitting a regression function on $y_i \dots y_j$.

B Dynamic Time Warping

Dynamic time warping (Figure 14) is a method of matching two time series patterns of different lengths. Let x_1, x_2, …, x_m be the first pattern, and y_1, y_2, …, y_n be the second pattern. Let a matching be such that the matched pairs are (x_{i_1}, y_{j_1}), …, (x_{i_k}, y_{j_k}), …, (x_{i_K}, y_{j_K}), where $i_1 = 1$, $j_1 = 1$, $i_K = m$, $j_K = n$. The cost of the matching is

$$C(m, n) = \sum_{k=1}^{K} (x_{i_k} - y_{j_k})^2 . \tag{32}$$

Dynamic time warping minimizes $C(m, n)$ by dynamic programming:

$$C(i,j) = (x_i - y_j)^2 + \min \begin{cases} C(i-1,j) \\ C(i,j-1) \\ C(i-1,j-1) \end{cases} \tag{33}$$

where we define $C(0,0) = 0$, $C(i,0) = \infty$, $C(0,j) = \infty$, for $i, j > 0$.

C Searching for Similar Patterns Using Template Matching and Dynamic Time Warping

To search a time series $y_1 y_2 \dots y_T$ for a pattern similar to a given example pattern $x_1 x_2 \dots x_P$ using template matching (i.e., with root mean squared error as the distance function), we calculate, at each $t >= P$, the distance between the example pattern $x_1 x_2 \dots x_P$ and the subsequence $y_{t-P+1} \dots y_t$ with the means of both patterns shifted to 0:

$$\text{RMSE}(t) = \sqrt{\frac{1}{P} \sum_{i=1}^{P} (\hat{x}_i - \hat{y}_{t-P+i})^2}, \tag{34}$$

where $\hat{x}_i = x_i - \text{mean}(x_1 \dots x_P)$, and $\hat{y}_{t-P+i} = y_{t-P+i} - \text{mean}(y_{t-P+1} \dots y_t)$.

We can calculate the distance function using dynamic time warping in a similar manner. Instead of comparing the example pattern with just one subsequence of length P ending at t, any subsequence of length Q, where $(1 - 20\%)P \leq Q \leq (1 + 20\%)P$, and ending at t, is a candidate pattern. We pick the subsequence with the lowest cost $C(P, Q)$, and define

$$\text{AverageError}(t) = \sqrt{\frac{1}{K} C(P,Q)}. \tag{35}$$

These distance functions were used to get the error curves in Figure 12 and Figure 13.