Tracking the Progression of Reading Using Eye-Gaze Point Measurements and Hidden Markov Models

Stephen Bottos and Balakumar Balasingam, *Senior Member, IEEE*

***Abstract*— In this article, we consider the problem of tracking the eye gaze of individuals while they engage in reading. In particular, we develop the ways to accurately track the line being read by an individual using commercially available eye-tracking devices. Such an approach will enable futuristic functionalities, such as comprehension evaluation, interest level detection, and user-assisting applications such as hand-free navigation and automatic scrolling. Furthermore, the proposed approach will pave the way to develop technology that may generate valuable feedback to content makers, such as web designers, authors, educators, and social media users. The existing commercial eye trackers provide an estimated location of the eye-gaze points every few milliseconds. However, these estimated gaze points are not sufficient to quantify reading progression—a specific eye-gaze activity. In this article, we propose algorithms to bridge the commercial gaze tracker outputs and informative eye-gaze patterns while reading. The proposed system consists of Kalman filters and hidden Markov models to parameterize these statistical models and to accurately detect the line being read. The proposed approach is shown to yield an improvement of 27.1% in line detection accuracy over line tracking using estimated eye-gaze points alone by the eye tracker.**

***Index Terms*— Eye-gaze measurements, hidden Markov models, information fusion, Kalman filter, machine learning, eye tracking.**

NOMENCLATURE

L_x	Width of the surveillance area within with the passage of text to be read is contained.
L_y	Height of the surveillance area.
$\text{gp}(t)$	Eye-gaze point at time t.
$z_y(t)$	y-coordinate of the eye-gaze point at time t. Since this article deals only with line-by-line detection, and only the y-coordinate is necessary to consider.
$\hat{y}(t)$	Estimated y-coordinate corresponding to $z_y(t)$, the output from the Kalman filter.
N_l	Number of lines in a portion of text of interest.
$o(t)$	Discretized observation or observed state (i.e., observed line number).
$S(t)$	True state, representing the true line number.
$\hat{S}(t)$	Estimate of $\hat{S}(t)$, the estimated line number.
$\boldsymbol{\pi}$	Prior probabilities.
$\mathbf{A}$	State transition probability matrix.
$\mathbf{B}$	Observation probability matrix.

Manuscript received June 14, 2019; revised March 3, 2020; accepted March 9, 2020. Date of publication April 2, 2020; date of current version September 15, 2020. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grants (DG) Program under Grant RGPIN-2018-04557. The Associate Editor coordinating the review process was Gaigai Cai. *(Corresponding author: Stephen Bottos.)*

The authors are with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9G 3P4, Canada (e-mail: bottos@uwindsor.ca; singam@uwindsor.ca).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2020.2983525

I. INTRODUCTION

EYE-GAZE tracking has the potential to reveal valuable information regarding an individual's cognitive state. In the case of emerging applications such as advanced driver assistance systems, eye-tracking hardware is becoming ubiquitous due to their noninvasive ability to collect data. Advances in hardware have provided the means to perform complex experiments with unprecedented accuracy, similar to the manner by which the processing power of modern-day computers has ushered in the age of big data [1]. As eye-tracking technology becomes more and more inexpensive and accurate, possible new applications based on this technology are likely to emerge.

The origins of eye-tracking research could be traced back to over a century ago when psychologist Edmund Huey first began to draw connections between cognition and eye-gaze fixation patterns during reading [2]—marking the first era of eye movement research in the context of cognitive process and behavioral inference. It was later postulated in 1998 in a lengthy collection of works and studies in the field of eye movements during reading and informational processing [3] that, due to advances in technology and a rapidly growing interest in the field, the advent of the third era had occurred some 20 years prior following the second era, which was characterized by its focus on applied, experimental psychology. At present, 20 years later, it may be that we have advanced from the third era to the fourth. Advanced eye-tracking technology, such as the Gazepoint [4], provides affordable, constraint-free, and noninvasive methods of obtaining reasonably accurate eye-gaze point data.

Applications of eye-tracking technology, in general, exist in the field of human–machine systems [5], assistive technology, i.e., robotic prosthetics [6], and interactive applications with multimedia devices such as computers and televisions [7], [8]. Other notable emerging technologies include adaptive interactive training [9], in which content, presentation format, and

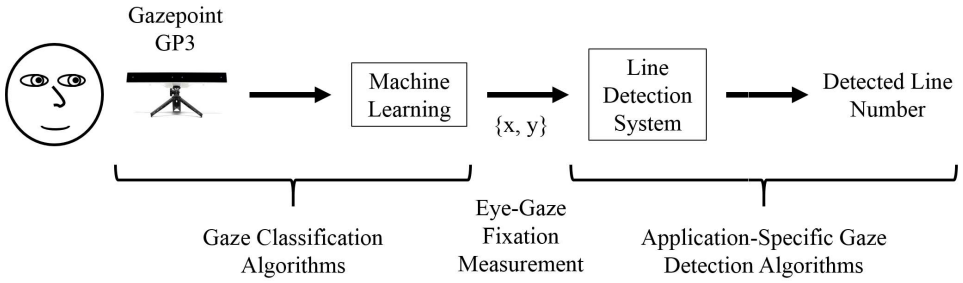

Fig. 1. LDS: proposed LDS relies on estimated eye-gaze points provided by commercial eye-tracking devices for eye-gaze data. An eye tracker uses machine-learning models to convert the instantaneous images captured by multiple infrared and video cameras into eye-gaze points. The Gazepoint GP3 eye tracker has four cameras, aligned along a horizontal line.

pacing of training modules are modified according to mental fatigue—measured by observing visual scanning strategies. Another explorative application aims to decrease operator error during unmanned aircraft missions by detecting operator fatigue using eye-gaze metrics [10].

Reading activity detection and classification [11], and general browsing activity classification, have potential applications in content placement and recommendation for maximal user experience. Such algorithms can be optimized with the use of reliable focal point information collected from users [12]–[14]. Indeed, human consumption of information is occurring more and more through interactions with a screen. Content makers, such as web designers, authors, professors, and social media users, are interested in actionable feedback to their content. Reliable, inexpensive and noninvasive technology that is able to answer questions such as did my audience spend time reading the descriptions that we provided? which parts/styles of writing were the most interesting to my readers? did my students read my lecture notes and descriptions? and what did my followers specifically like about the statement that I posted on my social media account? The proposed work in this article helps to provide a basis for technology that aims to answer some of these questions.

Recent work [15] has explored the possibilities of improving the efficacy of content recommender algorithms by answering similar questions using eye-gaze data. It was shown that eye-gaze data collected from a sample can be used to predict the behavior of a population at a very small scale—in an experiment that involved 17 participants in total, four of which were used for training their proposed model. These data were collected using expensive equipment (the Tobii T60 [16]) in a controlled laboratory environment. With such cost limitations on hardware, the applications of eye-tracking technology will not be able to scale up in order to incorporate learning from hundreds of thousands of individuals. Eye-tracking technology needs further improvement to work with inexpensive hardware or preferably existing hardware, such as the built-in camera of a tablet computer so that scalable applications can be developed. For instance, ubiquitous eye-tracking technology facilitates the deployment of sophisticated machine-learning applications requiring large amounts of training data, using crowd-sourcing platforms such as Amazon's Mechanical Turk [17] that enables the collection of large amounts of data from consenting individuals.

In this article, we present an approach to estimate the reading patterns of an individual using eye-gaze points obtained from an inexpensive eye tracker. Similar attempts to track reading progression [18]–[20], while effective, required the use of expensive hardware such as the Tobii T60 [16] or the SensoMotoric Instrument iViewX [21]—technology that is unlikely to be found outside of a laboratory—as well as highly regulated conditions for data collection. In order for such applications to reach wider recipients, it must not only be able to be performed with low-cost eye-trackers but also remain reasonably constraint-free as to not interfere with natural reading behavior.

Specifically, we present the algorithmic aspects of a line detection system (LDS) that will be useful in a futuristic reading pattern classifier that can generate informative feedback to content makers, such as advertisers, enterprises, researchers, educators, and parents. The proposed LDS algorithm takes noisy eye-gaze point measurements from a low-cost eye tracker, such as the Gazepoint GP3 [22], [23] as an input. Then, it outputs the line of text that was being read corresponding to each noisy measurement. The particular contribution of the LDS is that for the first time, statistical models, suited for eye trackers in general and low-cost eye trackers in particular, are presented in order to represent reading activity.

The remainder of this article is organized as follows. In Section II, the problem to be solved is discussed. Section III describes our proposed approach to line detection while reading. In Section IV, we present the performance analysis of the proposed line detection algorithm, tested using real-world eye-gaze data collected using a commercial eye-gaze tracking device. This article is concluded in Section V.

II. Problem Definition

It can be seen that eye-tracking technology itself is no new arrival. "Fourth-era" hardware, such as the Gazepoint GP3 [4] used to collect data during the research described in this article, utilizes machine-learning techniques in order to first recognize the location of a user's face and then to extract the location of the eyes. Next, by analyzing the position of the eyes using machine-learning algorithms, eye-gaze points in the form of (x, y) coordinates are generated. Fig. 1 shows a visual representation of this step, referred to as gaze classification. It is up to the "users" to process these gaze point estimates and interpret them for different applications. For example,

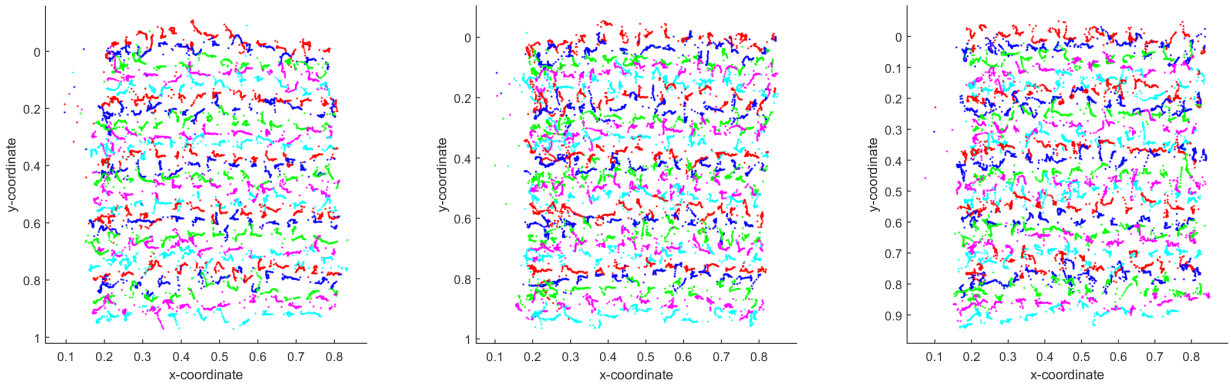

Fig. 2. Real eye-gaze points, color-coded per line according to ground truths. Real eye-gaze points collected during the reading of three separate sample pages, each containing a 25-line passage of text, are shown. Note that the y-axis has been reversed to account for the coordinate system used by the eye tracker. The top-left corner corresponds to the origin.

a previous work used the gaze point estimates in order to determine the points of interest in films and pictures using clustering algorithms [24], [25]. In this article, we develop an approach to track the progression of reading using these gaze point estimates.

The goal is to track a reader's progression through a block of text with no prior knowledge regarding the layout of the page displayed on the screen in terms of text position, text size, and distance between individual lines of text. If one is to consider a page in a novel, containing multiple lines of unbroken text of equal length and equal spacing, then the act of reading, speaking in terms of non-Semitic languages, will in the ideal case begin at the top left of the page and track left to right. When one line of text is completed, the reader begins the same left-to-right progression on a new line immediately below that which was previously read. A less ideal assumption, albeit a more reasonable one, would be to consider that the reader may also decide to occasionally return to previously read text rather than progressing in one direction only or skip forward through text by multiple lines.

The need for a special algorithm for the LDS arises due to the following three noise factors.

1) The measurement noise introduced as a result of classification errors made by the gaze tracker's computer algorithms [26], [27]. For eye trackers that have their sensors (cameras) located in a small area, it is also possible that the measurement error increases with the angle of alignment between the face and the eye-tracking device. The Gazepoint GP3 eye tracker shown in Fig. 1 has four cameras spanning a straight line, and as such, its vertical gaze measurement noise will be higher than its horizontal counterpart and both noise sources will increase with the angle of alignment.
2) The MIDAS[1] touch problem in eye-gaze tracking is an intrinsic characteristic of the human eye that it rarely stays focused in one point regardless of where the mind is concentrated [28], [29].
3) External circumstances, such as head movements, poor calibration, and distractions, all of which are introduced as a result of the "minimal constraints" environment which the authors strive to preserve.

[1]In reference to the Greek mythological King who got his wish, of turning everything he touched into gold, granted.

As a result of the above mentioned sources of noise, the data obtained from the Gazepoint GP3 [4] and similar state-of-the-art eye-tracking devices need further processing. In this article, we consider the above mentioned noise sources together and refer to them as "gaze measurement noise."

Fig. 2 shows the eye-gaze measurements collected during the reading of three separate pages, each containing a 25-line passage of text. The key point to note is that gaze points belonging to a single line tend to overlap with those belonging to other lines and do not necessarily follow predictable patterns. As the amount of overlap between individual lines increases with the increase of noise, the accurate detection of discrete lines of text becomes more challenging. The line detection algorithm must be robust enough to accurately predict a reader's progression even in the case of input data that are highly corrupted by noise.

In addition to the primary challenge of the LDS that is outlined so far, there are other practical limitations that it must adhere to in order to be viable in practical applications. It is necessary for the LDS to perform calculations in real time, considering that available computing resources could be limited. Furthermore, since the system must be tracking the reader's progress throughout the duration of their task, a minimal amount of stored memory required for the LDS to reach a decision is desirable. The approach we propose in this article, based mainly on Kalman filters and discrete hidden Markov models, is well suited for such real-time implementation.

III. Proposed Approach

A. Proposed Line Detection System

Fig. 3(a) shows the geometry of a block of text consisting of ten lines. During a reading activity, the eye-gaze measurements are produced by an eye-tracking device every few milliseconds; these measurements indicate the point on which

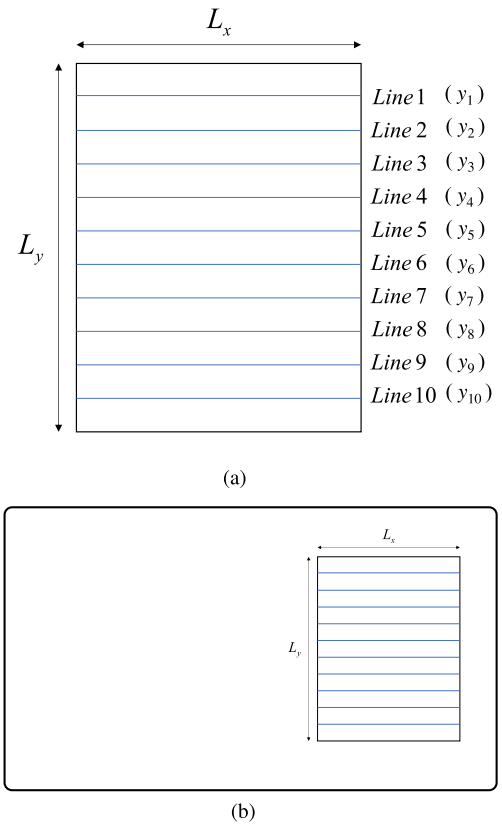

Fig. 3. Surveillance region. An imaginary sample paragraph containing ten lines of text, with dimensions $L_x \times L_y$, where the y-coordinate of each line is indicated as $y_1, \ldots, y_{10}$. (a) Surveillance region (area of the text in focus). (b) Position of the surveillance region on computer monitor.

the eyes were focused during that short time interval. Let us denote a particular measurement of gaze point, obtained at time t as

$$\text{gp}(t) = [z_x(t), z_y(t)]^T \tag{1}$$

where $z_x(t)$ is the x-coordinate of the gaze point measurement at time t and $z_y(t)$ is the y-coordinate of the gaze point measurement at time t. We use a coordinate system where the top-left corner of the surveillance region [see Fig. 3(a)] is considered as the origin $(0, 0)$.

Only gp(t) points within the $L_x \times L_y$ surveillance region [see Fig. 3(a)] containing the text are of interest. However, in reality, eye-gaze points from outside the surveillance region will be often recorded—this is due to the person looking away momentarily due to visual distractions. If the boundaries of the surveillance regions are known, then the measurements that fall outside those boundaries can be simply discarded; otherwise, the statistical model of the line detection system can be trained to ignore them. In this article, the surveillance region is always considered to be filled with a block of text and located in the center of the computer screen with L_y spanning the full height of the monitor's display.

The proposed LDS consists of the following three main components:

1) Kalman filter, which is employed to remove some noise from the raw input data
2) discretization, which translates the continuous y-coordinate component of gp(t) into the discrete observed state, $o(t)$
3) hidden Markov model, which decodes the discretized observation sequence $o(t)$ in order to produce the corresponding state sequence $S(t)$ that denotes the line being read at time t

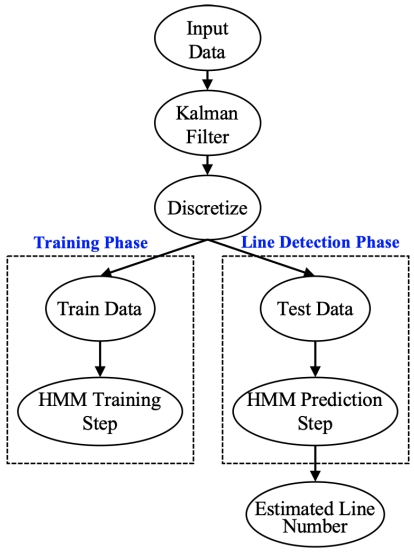

Fig. 4. LDS workflow. In the training phase, the parameters of the HMM, observation matrix **B** and the transition matrix **A**, are estimated. Once the HMM parameters are estimated, the LDS can continuously operate.

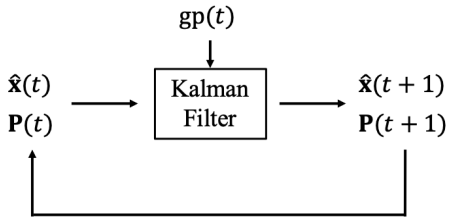

Fig. 5. Kalman filter recursion. Given the estimate $\hat{\mathbf{x}}(t)$ and its error covariance $\mathbf{P}(t)$ at time t, the Kalman provides the updated estimate $\hat{\mathbf{x}}(t+1)$ and its error covariance $\mathbf{P}(t+1)$ based on the new measurement $z(t+1) =$ gp(t + 1).

Fig. 4 shows the workflow of the LDS, and we describe each of these three processes in detail in the following sections.

B. Kalman Filtering Process

Given the gaze point measurements

$$\mathbf{z}(t) \triangleq \text{gp}(t) = [z_x(t), z_y(t)]^T \tag{2}$$

the Kalman filter [30] is employed to filter some of the noise arising due to measurement errors. The proposed Kalman filter functions according to the following state-space model:

$$\mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{v}(t) \tag{3}$$

$$\mathbf{z}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{w}(t) \tag{4}$$

where the 4×1 state vector

$$\mathbf{x}(t) = [x(t) \quad \dot{x}(t) \quad y(t) \quad \dot{y}(t)]^T \tag{5}$$

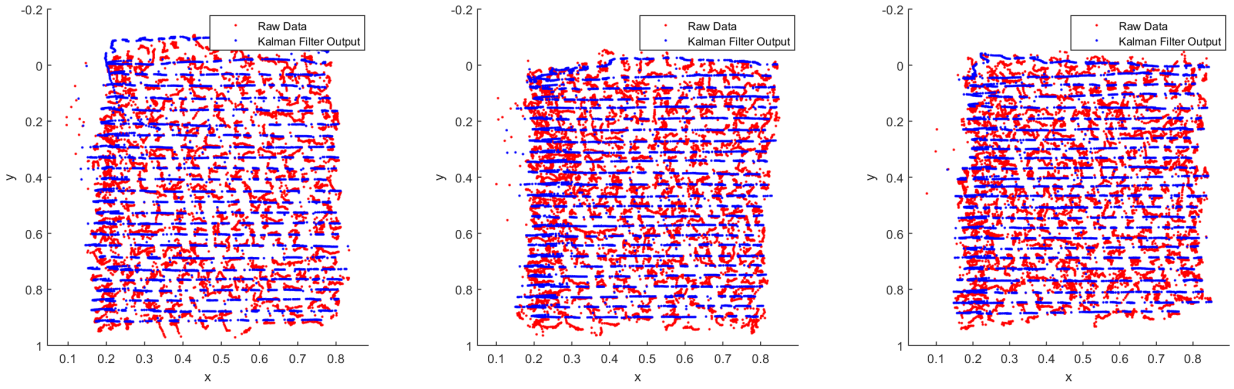

Fig. 6. Raw eye-gaze points overlaid with Kalman filter output. The raw eye-gaze points, in red, are far more difficult to interpret than their filtered counterparts, in blue.

contains the gaze displacement in the x-direction, rate of gaze-displacement (velocity) in the x-direction, gaze displacement in the y-direction, and rate of gaze displacement in the y-direction, respectively, the measurement matrix is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6}$$

and the process noise and the measurement noise of the state-space model (3) and (4) are indicated by $\mathbf{v}(t)$ and $\mathbf{w}(t)$, respectively.

Fig. 5 summarizes the Kalman filter [30] recursion based on the process model (3) and measurement model (4). The Kalman filter estimate is denoted as

$$\hat{\mathbf{x}}(t) = \left[\hat{x}(t) \;\; \hat{\dot{x}}(t) \;\; \hat{y}(t) \;\; \hat{\dot{y}}(t)\right]^T \tag{7}$$

where $(\hat{x}(t), \hat{y}(t))$ denote the estimated gaze location at time t and $(\hat{\dot{x}}(t), \hat{\dot{y}}(t))$ denote the estimated rate of gaze change (velocity) in the respective directions. First, we require initial estimates for the initial state at $t = 1$, denoted as $\hat{\mathbf{x}}(1)$, and the state prediction covariance matrix, $\mathbf{P}(1)$—a square matrix having rows and columns equal to the length $\hat{\mathbf{x}}(1)$. Given these (initial) estimates, the Kalman filter [30] can be used to recursively obtain the updated estimate of $\hat{\mathbf{x}}(t)$ and the state covariance matrix $\mathbf{P}(t)$ as each new measurement $\text{gp}(t)$ arrives, for $t = 2, 3, \ldots, T$. While it is more common to denote the initial estimate as $\hat{\mathbf{x}}(0)$, in this particular case, we consider the first eye-gaze point recorded by the eye tracker as the initial state and thus use $\hat{\mathbf{x}}(1)$ to stay true to the indexing. The Kalman filter is initialized as follows:

$$\hat{\mathbf{x}}(1) = \begin{bmatrix} z_x(1) \\ \dot{z}_x(1) = (z_x(2) - z_x(1))/2 \\ z_y(1) \\ \dot{z}_y(1) = (z_y(2) - z_y(1))/2 \end{bmatrix} \tag{8}$$

$$\hat{\mathbf{P}}(1) = \gamma\, \mathbf{I}_4 \tag{9}$$

where γ is selected to be an appropriately large value [30].

Fig. 6 shows a sample output of the Kalman filter. Here, the Kalman filter estimates $\hat{x}(t)$ and $\hat{y}(t)$ are overlaid atop their corresponding raw eye-gaze point measurements $[z_x(t), z_y(t)]$ (using the same three sample pages given in Fig. 2). It can be seen that the Kalman filter estimates have a relatively less noise in the y-direction and the estimates resample the lines of text much clearly than the raw measurements. This is because the Kalman filter is tuned mainly to filter only $z_y(t)$ measurements. However, it offers little to no filtering effect in terms of the x-coordinates—since our focus is only on detecting the line being read. It is also important to note that the measurement noise in the x-direction is much smaller than that in the y-direction due to the geometry of the Gaze point GP3, shown in Fig. 1, which had four sensors (cameras) arranged on a horizontal line. The Kalman filter estimates are passed to the next block of the LDS described in Fig. 4.

C. Discretization

The discretization process (see Fig. 4) starts with the Kalman filter output, denoted $\hat{\mathbf{x}}(t) = [\hat{x}(t), \; \hat{y}(t)]$. In particular, $\hat{y}(t)$ is used from here onward in the LDS. It must be noted that the Kalman filter estimated $\hat{y}(t)$ can take any value in the surveillance region, i.e., $\hat{y}(t) \in [0, L_y]$, and as such, we call $\hat{y}(t)$ a continuous value. In this section, we discuss the procedure to translate the continuous measurements $\hat{y}(t)$ to their corresponding discrete observations $o(t) \in \{1, \ldots, N_l\}$, where N_l is the number of lines. First, it is necessary to determine the height of the page, L_y. Then, the text region can be split evenly according to the number of lines within it as follows:

$$y_{\text{step}} = \frac{L_y}{N_l}. \tag{10}$$

Let us denote y_j to indicate the y-coordinate of the jth line as it was shown in Fig. 3(a). Using the height of each region and assuming that lines are evenly spaced and centered in the middle of each region, we compute y_j for each line according to its distance from the top of the page, y_{top}

$$y_j = y_{\text{top}} + \left((j-1) \times y_{\text{step}} - \frac{y_{\text{step}}}{2}\right). \tag{11}$$

Note that (11) is adapted for the Gazepoint GP3 coordinate system, as it was shown in Fig. 2.

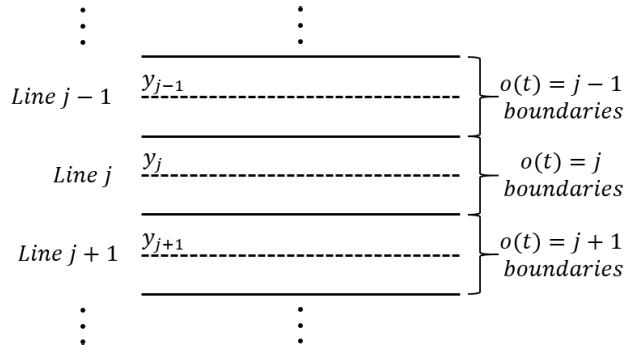

Fig. 7. Discretization of continuous eye-gaze point observations. The discretization boundaries are marked using solid lines.

The objective of the discretizer is to classify each gaze measurement $\hat{y}(t)$ in terms of line number; this now can be accomplished as follows (also, see Fig. 7):

$$o(t) = \left\{ j \middle| y_j + \frac{y_{\text{top}}}{2} < \hat{y}(t) < y_j - \frac{y_{\text{top}}}{2} \right\} \tag{12}$$

where $o(t)$ is the discretized observation corresponding to the eye-fixation measurement at time t.

From the discretization procedure described earlier, it must be clear that

$$o(t) \in \{1, \ldots, N_l\} \tag{13}$$

where N_l denotes the number of lines in a page of interest. For simplicity, we will assume that N_l is the same in each page considered for training and testing.

Now, the LDS problem can be formally stated as follows. Given a batch of T discretized observations $o(t)$, where $t = 1, \ldots, T$ and $o(t) \in \{1, \ldots, N_l\}$, estimate the corresponding lines on which a reader was focused, i.e., obtain the true state $S(t)$ for $t = 1, \ldots, T$, where $S(t) \in \{1, \ldots, N_l\}$ with $S(t)$ the line being read at time t.

D. Discrete State-Space Model

Similar to (3) and (4), a discrete state-space model can be written as follows:

$$S(t+1) = f(S(t)) \tag{14}$$

$$o(t) = g(S(t)) \tag{15}$$

where $S(t)$ denotes the state (line being read) at time t and $o(t)$ denotes the observation (observed line being read) at time t. It must be noted that $o(t)$ is obtained through the discretization step described in Section III-C. Also, it is worth reemphasizing that $S(t), o(t) \in \{1, \ldots, N_l\}$. In contrast, the state and observation in (3) and (4) take continuous values, i.e., $x(t) \in [0, L_x]$, $y(t) \in [0, L_y]$, $z_x(t) \in [0, L_x]$, and $z_y(t) \in [0, L_y]$.

The true line of focus at time $t+1$, $S(t+1) \in \{1, \ldots, N_l\}$, is dependent on the present state $S(t)$ as well as all previous states $S(t-1), S(t-2), \ldots, S(1)$, that is

$$S(t+1) = f(S(t), S(t-1), \ldots, S(1), S(0)). \tag{16}$$

Under the Markov assumption, this dependence can be relaxed to the previous state only—resulting in the following process model:

$$S(t+1) = f(S(t)). \tag{17}$$

The above mentioned process model can be stated as an $N_l \times N_l$ state transition probability matrix $\mathbf{A}$ where the (i, j)th element of $\mathbf{A}$ can be written as

$$a_{ij} = P(S(t+1) = j | S(t) = i). \tag{18}$$

Each observed data $o(t)$ relates to the true state through a nonlinear function $g(\cdot)$, that is

$$o(t) = g(S(t)). \tag{19}$$

Now, considering that both $o(t)$ and $S(t)$ take only discrete values, i.e., $o(t) \in \{1, \ldots, N_l\}$ and $S(t) \in \{1, \ldots, N_l\}$, the above mentioned observation model can be stated as the following $N_l \times N_l$ observation probability matrix $\mathbf{B}$ where the (i, j)th element of $\mathbf{B}$ can be written as

$$b_{ij} = P(S(t) = i | o(t) = j). \tag{20}$$

It is easy to see that what is described through (14) to (20) is a discrete hidden Markov model [31] with $\mathbf{B}$ as its observation probability matrix and $\mathbf{A}$ as its transition probability matrix. In addition, let us define the prior probability as follows:

$$\boldsymbol{\pi} = \left[p(S(0)=1),\ p(S(0)=2), \ldots, p(S(0)=L_y) \right]^T \tag{21}$$

where $S(0)$ denotes the initial sate (line read at time $t = 0$). Given a batch of observations $o(t)$, the parameters of the HMM, $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{B}$, can be estimated using the Baum–Welch algorithm (this is referred to as the "training phase" in Fig. 4. Given a batch of observations $o(t)$ and the model parameters $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{B}$, the corresponding state $S(t)$ (line being read at time t) can be estimated using the Viterbi algorithm—this step is referred to as the "line detection phase" in Fig. 4.

In Section III-E, we describe a general training procedure to estimate the parameters of an HMM and discuss training specific to the proposed LDS using real-world data in Section IV-C.

E. Discrete Hidden Markov Model Design

The proposed LDS consists of two important steps that are given in the following.

1) *Parameter Estimation:* The objective here is to estimate the parameters of the discrete state-space model defined in Section III-D to represent eye-gaze movement while reading. Specifically, the objective is to estimate, $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{B}$, the parameters of the discrete HMM. This step is also referred to as training phase in the workflow Fig. 4; we also refer this as HMM training in this article.
2) *State Estimation:* Once the (HMM) parameters are estimated, the proposed LDS becomes functional: given the discretized observations $o(t)$, estimate the discrete state $S(t)$ which refers to the line being read—based

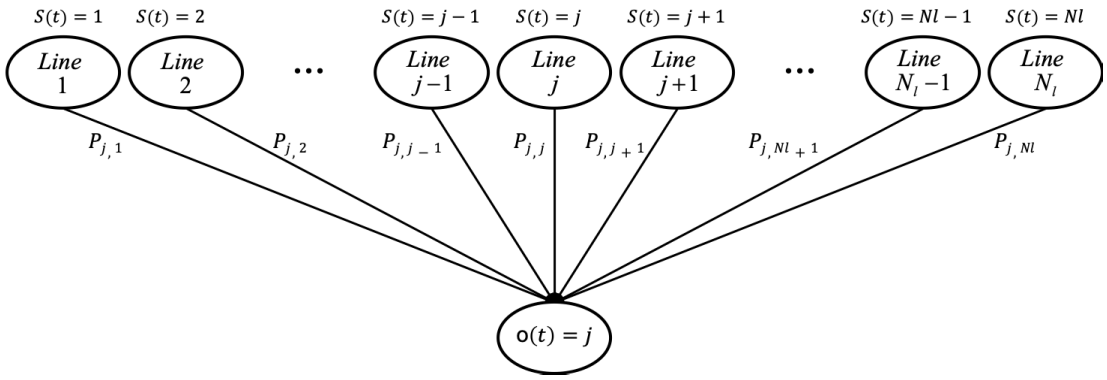

Fig. 8. Observation matrix. Given an observation $o(t)$, the observation probability matrix shows the probability of each possible state $S(t) \in \{1, \ldots, N_l\}$ (i.e., line number) to be the true state.

on the notations defined in Nomenclature. This step is also referred to as the line detection phase in the workflow Fig. 4. In this article, we also refer to the state estimation step as testing stage or simply as HMM testing.

The objective of the HMM training module is to estimate the model parameters $\boldsymbol{\pi}, \mathbf{A}$, and $\mathbf{B}$. This requires the discretized y-data, $o(t)$, that is obtained from the gaze measurement $\text{gp}(t)$. Estimating $o(t)$ requires the knowledge of the surveillance region as well as the exact locations of the y-coordinates of each line. If they are fixed and known (e.g., ebook in a tablet), then they can be used to estimate $o(t)$; if they are not known, then they can be estimated—see Section IV-B for a discussion on this.

Discrete HMM training is a well-studied subject, and the Baum–Welch algorithm [31] provides the best known approach to train an HMM with the use of the expectation–maximization (EM) technique. Several software platforms, including MATLAB and Python, provide optimized HMM training libraries. The training process requires an initial guess for the parameters, after which it updates them recursively. Assuming the initial parameters to be $\boldsymbol{\pi}_0, \mathbf{A}_0$, and $\mathbf{B}_0$, the trained HMM parameters $\hat{\boldsymbol{\pi}}, \hat{\mathbf{A}}$, and $\hat{\mathbf{B}}$ can be obtained.

A good initial guess is often conducive to fast and accurate convergence during HMM training. Next, we describe an approach to select suitable initial parameters $\boldsymbol{\pi}_0, \mathbf{A}_0$, and $\mathbf{B}_0$ for the proposed HMM training step. The selections described next are intended for $N_l = 10$ lines; however, the approach can be extended to any number of lines.

First, the prior probabilities can be selected by considering the fact that, most of the time, the reader will begin with the first line of text

$$\boldsymbol{\pi}_0^T = \begin{bmatrix} .91 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 \end{bmatrix} \quad (22)$$

where a significantly high probability assigned to the first line. Also, it was observed that the HMM training accuracy is not significantly affected by the selection for π_0.

The transition probabilities are selected based on the intuitive observation that the transition between two adjacent gaze points, $\text{gp}(t)$ and $\text{gp}(t+1)$, mostly occurs within the same line. The next most common transitions will intuitively occur between the immediate-next or immediate-previous lines; immediate-next line transitions occur when the person progresses from one line to another in natural progression, and the immediate-previous line transition occurs due to backtracked reading of lines. Assuming that the probability of immediate-next line transition and the immediate-previous line transition is the same, and the transition probability is zero for gaze transitions separated by more than one line, the initial guess of the transition probability is given as follows:

$$\mathbf{A}_0 = \begin{bmatrix} .9 & .1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .05 & .9 & .05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .05 & .9 & .05 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .05 & .9 & .05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 & .9 & .05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .05 & .9 & .05 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .05 & .9 & .05 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .05 & .9 & .05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .05 & .9 & .05 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .1 & .9 \end{bmatrix}.$$

Ideally, when there is no noise, the observation probability matrix $\mathbf{B}$ approaches an identity matrix, i.e., it means that the observations $o(t)$ represent the true line being read $S(t) = o(t)$. When there is observation noise, which is the realistic case, the observation probability matrix $\mathbf{B}$ takes a Toeplitz form. Fig. 8 shows this further; with each line j having the ability to generate $o(t)$ with some probability P_j, intuitively, one can say that the greatest probability occurs at P_j when $o(t) = j$, that

$$P_{j,j} > P_{j,j+1} > P_{j,j+2} > \ldots > P_{j,N_l} \ldots \text{ when } o(t) = j \quad (23)$$

and

$$P_{j,j} > P_{j,j-1} > P_{j,j-2} > \ldots > P_{j,1} \ldots \text{ when } o(t) = j. \quad (24)$$

In the following, we illustrate an initial guess for $\mathbf{B}_0$ that is a simplified version derived from the earlier discussion

Fig. 9. Eye-tracking data collection. A screenshot of the test program, displaying the full 25 lines of text. A human subject will read the lines from top-to-bottom, while the the eye tracker collects the gaze points of the subject at approximately 60 samples/s. A single line of text at a time is displayed during data collection in order to accurately record the ground truth.

where we assume $P_{j,1} = P_{j,2} = \ldots = P_{j,j-1} = P_{j,j} = P_{j,j+1} = P_{j,N_l}$

$$\mathbf{B}_0 = \begin{bmatrix} .91 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 \\ .01 & .91 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 \\ .01 & .01 & .91 & .01 & .01 & .01 & .01 & .01 & .01 & .01 \\ .01 & .01 & .01 & .91 & .01 & .01 & .01 & .01 & .01 & .01 \\ .01 & .01 & .01 & .01 & .91 & .01 & .01 & .01 & .01 & .01 \\ .01 & .01 & .01 & .01 & .01 & .91 & .01 & .01 & .01 & .01 \\ .01 & .01 & .01 & .01 & .01 & .01 & .91 & .01 & .01 & .01 \\ .01 & .01 & .01 & .01 & .01 & .01 & .01 & .91 & .01 & .01 \\ .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .91 & .01 \\ .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .01 & .91 \end{bmatrix}$$

where it is assumed that each of the off-diagonal elements is the same. It must be noted that this is a special case of the Toeplitz matrix.

Once training is complete and the parameters $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{B}$ are obtained, a sequence of observations from one page of reading

$$\mathbf{o}^T = [o(1), o(2), \ldots, o(T)] \tag{25}$$

can be used to estimate the lines on which the reader's eye gaze was fixated corresponding to each observation $o(t)$. Since a batch of T observations is considered, this proposed approach falls under the batch estimation category. In the batch estimate mode, the Viterbi algorithm [31] can be used to decode the true line sequence while reading.

The complexity of the Viterbi algorithm is $\mathcal{O}(N^2\ T)$, where N is the number of states and T is the length of the sequence. Assuming that the approximate reading speed is 2 s/line and that the sampling rate of the eye-tracking device is 60 Hz, waiting for 1200 samples, i.e., $T = 1200$, will have gaze-data spanning approximately 10 lines. The complexity of decoding will be in the order of 12k flops.

IV. Experimental Evaluation of the Proposed Approach

A. Data Collection Procedure

Eye-gaze data were collected from a single test subject (a male in his twenties), using a Gazepoint GP3 [4] desktop device. A simple data collection program was written in Python such that communication to and from the Gazepoint device was enabled. The program first initiated a calibration step, which is required by the Gazepoint device before tracking is possible. Upon completion of the calibration step, the test subject was required to press the space key which would simultaneously cue the device to begin logging the x and y eye-gaze fixation coordinates, at 60 Hz, and reveal a single line of text against a solid background, near the top of the display (in this case, a 1920×1080 computer monitor) for which the device was calibrated. Fig. 9 contains a visualization of the display, and with each line of text shown, however, only a single line of text at a time was displayed during data collection in order to accurately record the ground truth. While the topmost line of text—Line 1—was displayed, each gaze point corresponding to this line was labeled with a "1" to allow for comparison between ground truths and predictions.

Upon completion of Line 1, the space key was pressed, which would cause Line 1 to disappear from view and prompt Line 2 to be displayed on screen at the corresponding location of Line 2, as well as increment the ground-truth label to match the current line displayed on the screen. In such a manner, eye-gaze points were collected for 25 lines of text, which would represent one page worth of data. Text was fed to the program by a file containing a passage taken from a publicly

available copy of Moby Dick by Herman Melville [32], from which 25 full pages worth of data (having 25 full lines of text each) were collected.

B. Surveillance Region Detection

In this section, we discuss an approach of estimating the surveillance region based on collected eye-gaze points only, without knowing explicitly where, on the screen, the surveillance area is located. It was previously discussed in Section III-C that, given a known dimension L_y, the location of each line on the page y_j could be determined. Fig. 3(b) shows the need for surveillance area detection; here, the gaze point data may span not only the parameters of the surveillance area but may also be scattered across the region of the computer monitor. However, if the text is being read in full screen, then there is still a need to detect the surveillance area. Indeed, it was observed, while reading a full-screen text, that eye-gaze points were collected not only outside of the surveillance region but also outside of the region for which the eye tracker was calibrated—if an individual's gaze falls anywhere within the detectable field of the tracker, it will register as a data point. As such, it is necessary to discard outliers and establish the true surveillance region of interest. Both L_x and L_y were evaluated using the same method described in the next paragraph—avoid redundancy, we discuss the procedure of estimating L_y only.

It is assumed that the primary task of the participant during the data collection is reading, and as such, the vast majority of eye-gaze points will fall in the area of the text being read. With this intuition, first, an outlier removal procedure is implemented based on the z-score defined as

$$z_{\text{score}}(t) = \frac{z_y(t) - \mu_y}{\sigma_y} \tag{26}$$

where μ_y and σ_y are the average and standard deviation, respectively, of $z_y(t)$, $t = 1, \ldots, T$. Then, all measured gaze points $z_y(t)$ that fell outside a z-score of $[-1.9\ 1.9]$ were discarded (here, the cutoff number of 1.9 was reached with the help of visual empirical analysis). Fig. 10 shows the results of outlier removal for one page of data. Similarly, separate outlier detection can be implemented on the horizontal axis. Once the outliers are removed, the next task is to find the surveillance region, i.e., to identify the coordinates of the most top and the most bottom lines, respectively. For this, we select the average of M highest $z_y(t)$ data as the top y-coordinate of the top line and the average of M lowest $z_y(t)$ data as the bottom y-coordinate. Now, L_y (see Fig. 3) is the difference between the y-coordinate of the top line and the y-coordinate of the bottom line. Once an estimate for L_y was obtained, discretization could be performed as usual to estimate the location of each line of text, y_j. A demonstration of the effect of cleaning is shown in Fig. 10 Once an estimate for L_y was obtained, discretization could be performed as usual to estimate the location of each line of text, y_j. A demonstration of the effect of cleaning is shown in Fig. 10.

It is relevant to point out that the average of a batch of M points, rather than single extremum point, was taken by estimating the upper or lower boundaries of the page—this added robustness against selecting the single extreme point that could be an outlier.

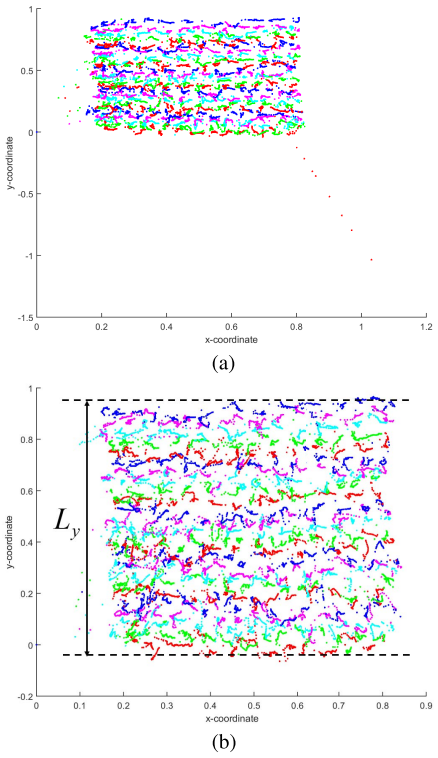

Fig. 10. Surveillance region detection. After the discarding of outliers, a batch of both maximum and minimum points may be collected in order to establish the range of L_y, after which discretization can be performed. (a) Area of interest, shown with stray gaze points which would skew the results of the discretizer. (b) Surveillance area after the removal of outliers. Relatively accurate discretization is now possible.

C. Hidden Markov Model Training

Given a batch of observations $o(t)$, $t = 1, \ldots, T$ and true states $S(t)$, $t = 1, \ldots, T$, the goal in HMM training is to estimate the parameters π, $\mathbf{A}$, and $\mathbf{B}$. However, in practice, it may be difficult to obtain the true state data $S(t)$. Considering this, an alternate approach is used to train the HMM.

First, parameters were initialized per Section III-E. Next, a synthetic data were created as follows.

1) A surveillance region of size $L_x \times L_y$ was created.
2) With the assumed number of lines N_l, the text "text lines" were drawn.
3) Assuming a constant reading speed of 10 s/line, true gaze points were simulated along the text lines in natural reading progression.
4) In order to emulate the measurement noise, unidirectional Gaussian noise of standard deviation σ_n was added to the simulated gaze points and the synthetic measurements were obtained.
5) The synthetic measurement was used in the training phase of Fig. 4 to obtain the HMM parameters.

It should be noted that σ_n is selected to match the measurement noise of the eye-tracking device and that the 10 s/line reading

speed was selected to be realistic. Models trained using simulated data each yielded state transition probability matrices containing zero-transition probabilities for some states. In practice, this meant that a single stray data point could break the LDS. To account for such stray data points, a small portion of the learned probabilities in each row of the state transition matrix was subtracted and distributed among each zero-transition probability in that row while ensuring that a stochastic matrix was maintained (i.e., the rows should sum to 1). With HMMs trained and adapted to handle real-world data, each was then tested for performance with the LDS and the most accurate model was chosen as the LDS's permanent, built-in HMM.

An important point to note is that the raw data at a sampling rate of 60 Hz were found to introduce a significant amount of noise attributed to the MIDAS touch problem discussed in Section II. As a remedy to this, several downsamples were experimented at the HMM training and HMM testing level and finally downsampled at a rate of 6 Hz was used for analysis in the remainder of this article.

D. Line Detection Using the Proposed Approach

Using the setup and procedure described in Section IV-A, 25 pages worth of real-world eye-gaze point data were collected, while the subject read text each containing 25 lines per page. The data collection procedure is designed in such a way that the true lines being read, $S(t)$, can be recorded. With $S(t)$, it was possible to quantify the error associated with a predicted state sequence for a given page through the comparison of estimated parameters to ground-truth parameters. The line detection error for a certain page p is defined as follows, with the subscript p included for clarity:

$$e_p = \left[\frac{\sum_{t=1}^{t=N_p} \hat{S}_p(t) \neq S_p(t)}{N_p} \right] \times 100 \tag{27}$$

where $\hat{S}_p(t)$ denotes the estimated line number, $\hat{S}_p(t)$ denotes the true line number (ground truth), and N_p denotes the number of data points from page p. The average error across all pages, e_{avg}, was then computed as

$$e_{avg} = \frac{\sum_{p=1}^{25} e_p}{25}. \tag{28}$$

In order to evaluate the performance, the average error e_{avg} was computed for the following three types of approaches.

1) *Discretized Output Only:* Here, the gaze point measurements were directly sent to the discretizer (Section III-C) and the discretized observations are treated as the estimated states, i.e., $\hat{S}(t) \triangleq o(t)$. In other words, the Kalman filter and the HMM modules were turned off in the overall procedure described in Fig. 4. This approach is summarized as [gp(t) $\rightarrow$ discretize $\rightarrow$ $o(t) = \hat{S}(t)$].
2) *Discretized Output + HMM:* Here, the gaze point measurements gp(t) were first discretized and the HMM was used to decode the true lines. In other words, the Kalman module was turned off in the overall procedure described in Fig. 4. This approach is summarized as [gp(t) $\rightarrow$ discretize $\rightarrow$ $o(t)$ $\rightarrow$ HMM $\rightarrow$ $\hat{S}(t)$].
3) *Kalman Filter + Discretize Function + HMM:* In this trial, the full LDS consisting of all the modules shown in Fig. 4 was implemented. This approach is summarized as [gp(t) $\rightarrow$ Kalman filter $\rightarrow$ $\hat{\mathbf{x}}(t)$ $\rightarrow$ discretize $\rightarrow$ $o(t)$ $\rightarrow$ HMM $\rightarrow$ $\hat{S}(t)$].

TABLE I
AVERAGE ERRORS

Prediction Method	Average Error
Discretize Only	39%
Discretize + HMM	16%
Full LDS	11.9%

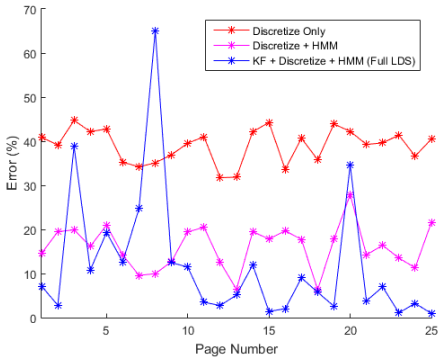

Fig. 11. Performance improvement achieved through the proposed method. Note that in each case, a significant performance improvement is witnessed with the inclusion of the LDS, and detection errors of less than 10% are achievable.

Table I shows the average line detection error from each approach described in the previous paragraph. It shows the benefits of different modules that consist of the proposed LDS described in Fig. 4. Similarly, Fig. 11 shows the line detection errors associated with each trial on a page-by-page basis.

V. Conclusion and Discussion

In this article, we proposed an approach to track the line being read in a reading activity based on an eye-tracking device. The proposed approach utilizes a low-cost eye tracker, the Gazepoint GP3, to measure eye-gaze points that are then fed through the proposed LDS, which applies a Kalman filter to smooth the data and translates the continuous measurement points to discrete observations. The resultant discrete observation sequence is then decoded using a hidden Markov model algorithm.

The LDS approach was designed with the intent to provide a basis for continued development, with the end goal being accurate reading detection using readily available hardware, such as the built-in camera of a laptop or tablet, without the need for physical constraints that restrict the natural behavior of the reader. The results presented in this article show promise as well as openings for continued improvements. We will discuss these improvements after discussing the results.

From Table I, it can be seen that the raw data, after discretization, produced the most average line detection error

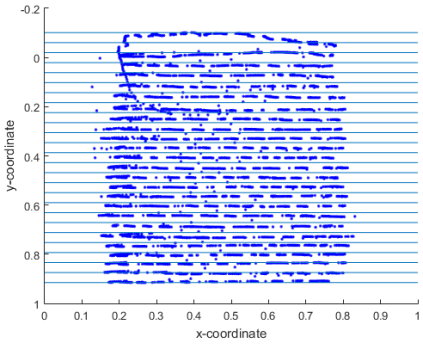

Fig. 12. Performance improvement achieved through the proposed method. Note that in each case, a significant performance improvement is witnessed with the inclusion of the LDS, and detection errors of less than 10% are achievable.

of 39%. The LDS, with the Kalman filter turned off, utilizing only the discretization and HMM capabilities of the algorithm, yielded significant improvement and reduced the average error to 16%. Referring to Fig. 11, in each test case, the HMM offered improved accuracy compared with raw data.

Considering the fully functional LDS, with smoothing, discretization, and HMM decoding, results were somewhat mixed. For the test pages 3, 7, 8, and 20, the Kalman filter worsened results. The reason for this is explainable. Referring to Fig. 12, which showcases the observation boundaries drawn during discretization for Page 8—the worst case—we can see that the region in which Line 2 should lie is empty, and Line 2 is present in Line 3's region instead. This has caused a shift in the observation sequence which not even the HMM can fix. A similar effect was witnessed for pages 3, 7, and 20. Nonetheless, the average error of the full LDS is 11.9%—the fully functional LDS yields the most accurate results in most cases. Indeed, if page 8 is considered an extreme anomaly, the average error in line detection drops to 9.7%, meaning that the LDS is above 90% accurate in its current state.

Some considerations for future work and potential improvements that can be made are listed as follows.

1) *Kalman Filter:* Pages 3, 7, 8, and 20 all experience a drop in accuracy due to the fact that the line boundaries that facilitate discretization are skewed. This skewness is due to the position of the filtered points (by the Kalman filter) belonging to Line 1. It seems as though convergence for Line 1 is slow, while all other lines follow a predictable pattern with a relatively even spacing between each line. It is also possible that the eye tracker exhibits a bias at the top of the screen—the furthest angle of its operation. An improved Kalman filter by incorporating several extra sources of information, such as the angular characteristics of the eye tracker and the true/predicted locations of line boundaries.
2) *Discretization:* In light of the results yielded by Pages 3, 7, 8, and 20, an improved discretization procedure that does not distribute line boundaries and regions evenly throughout the height of the surveillance region, but rather splits according to the characteristics of the eye tracker as well as the distributions created by the smoothed data points may be beneficial.
3) *Discrete HMM Model:* The hidden Markov model designed for the proposed LDS in this article is a discrete one. Advanced versions of the HMM, such as the continuous HMM and temporal HMM [31], might prove to be beneficial. Continuous HMM models the observed data directly through an output distribution, and a temporal HMM takes the time information in the model.
4) *Fixed Number of Lines:* The LDS as described in this article is designed to work with blocks of text having a fixed number of lines—to which reading an ebook in a tablet is a good example—in which each page will be of similar structure. Adapting the LDS to any number of lines is a straightforward extension. The proposed approach needs to be extended to pages with a different number of lines.

The proposed LDS is tested using 25 pages of data, each containing 25 lines of text, while a participant read through them. The ground truth—the actual lines being read–was recorded using a specially designed software. The HMM training was carried out using a synthetic data set that was created based on the noise standard deviation of the eye tracker. As such, testing the proposed LDS on multiple participants will not require additional data for training as long as we use the same eye tracker to collect data from each participant. The already trained LDS can be used on data collected from new participants. Testing the proposed LDS on data collected from a statistically significant number of participants is left as a future work due to the amount of time and resources required to recruit participants for such data collection.

As a final point of consideration, different eye-tracking devices offer varying degrees of accuracy and sampling rates. The proposed LDS is tested using data collected at 60 Hz from the Gazepoint GP3 eye tracker. It must be noted that accuracy of eye trackers varies based on the manufacturer and their models. The proposed approach depends on an eye tracker; as such, the LDS algorithm cannot be directly tested in laptops and tablets. However, open-sourced eye-tracking apps can be utilized to test the proposed algorithm using the embedded camera in a tablet, laptop, or desktop computer. Future work shall consider designing the LDS to work with devices that are prone to a higher degree of spatial error in their gaze point estimation.

ACKNOWLEDGMENT

The authors would like to thank Rajankumar Patel (third-year electrical engineering student at the University of Windsor) for his assistance in collecting the data from the eye tracker. They would also like to thank the anonymous reviewers for patiently reading this manuscript very thoroughly and providing numerous critical and useful comments that significantly improved the quality of this manuscript.

REFERENCES

[1] C. Robert, "Machine learning, a probabilistic perspective," Tech. Rep., 2014.
[2] E. B. Huey, *The Psychology and Pedagogy of Reading*. New York, NY, USA: The Macmillan, 1908.

[3] K. Rayner, "Eye movements in reading and information processing: 20 years of research.," *Psychol. Bull.*, vol. 124, no. 3, pp. 372–422, 1998.
[4] *Gazept Gazepoint Eye Tracker Web Site*. Accessed: Nov. 11, 2018. [Online]. Available: https://www.gazept.com/
[5] A. Poole and L. J. Ball, "Eye tracking in HCI and usability research," in *Encyclopedia Human Computer Interaction*. Hershey, PA, USA: IGI Global, 2006, pp. 211–219.
[6] D. P. McMullen *et al.*, "Demonstration a semi-autonomous hybrid brain–machine interface using human intracranial eeg, eye tracking, Comput. Vis. To control a robotic upper limb prosthetic," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 4, pp. 784–796, May 2014.
[7] X. Zhang, X. Liu, S.-M. Yuan, and S.-F. Lin, "Eye tracking based control system for natural human-computer interaction," *Comput. Intell. Neurosci.*, vol. 2017, Dec. 2017, Art. no. 5739301.
[8] J. P. Hansen, A. W. Andersen, and P. Roed, "Eye-gaze control of multimedia systems," in *Advances Human Factors/Ergonomics*, vol. 20. Amsterdam, The Netherlands: Elsevier, 1995, pp. 37–42.
[9] J. T. Coyne, C. Baldwin, A. Cole, C. Sibley, and D. M. Roberts, "Applying real time physiological measures of cognitive load to improve training," in *Proc. Int. Conf. Found. Augmented Cognition*. Springer, 2009, pp. 469–478.
[10] P. Mannaru, B. Balasingam, K. Pattipati, C. Sibley, and J. Coyne, "Cognitive context detection in UAS operators using eye-gaze patterns on computer screens," *Proc. SPIE Int. Soc. Opt. Photon.*, vol. 9851, May 2016, Art. no. 98510F.
[11] A. Hyrskykari, "Utilizing eye movements: Overcoming inaccuracy while tracking the focus of attention during reading," *Comput. Hum. Behav.*, vol. 22, no. 4, pp. 657–671, Jul. 2006.
[12] L. A. Granka, T. Joachims, and G. Gay, "Eye-tracking analysis of user behavior in WWW search," in *Proc. 27th Annu. Int. Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2004, pp. 478–479.
[13] S. Xu, H. Jiang, and F. C. M. Lau, "Personalized online document, image and video recommendation via commodity eye-tracking," in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2008, pp. 83–90.
[14] K. Puolamäki, J. Salojärvi, E. Savia, J. Simola, and S. Kaski, "Combining eye movements and collaborative filtering for proactive information retrieval," in *Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2005, pp. 146–153.
[15] Q. Zhao, S. Chang, F. M. Harper, and J. A. Konstan, "Gaze prediction for recommender systems," in *Proc. 10th ACM Conf. Recommender Syst. (RecSys)*, 2016, pp. 131–138.
[16] *Tobii t60 Web Site*. Accessed: Jul. 31, 2019. [Online]. Available: https://imotions.com/hardware/tobii-t60-t120/.
[17] *Amazon's Mechanical Turk Web Site*. Accessed: Nov. 29, 2018. [Online]. Available: https://www.mturk.com/
[18] R. Paeglis, K. Bagucka, N. Sjakste, and I. Lacis, "Maximizing reading: Pattern analysis to describe points of gaze," *Proc. SPIE Int. Soc. Opt. Photon.*, vol. 6315, Jun. 2006, Art. no. 63150R.
[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[20] S. S. Mozaffari, F. Raue, S. D. Hassanzadeh, S. Agne, S. S. Bukhari, and A. Dengel, "Reading type classification based on generative models and bidirectional long short-term memory," in *Proc. IUI Workshops*, 2018, pp. 1–5.
[21] *Sensomotoric Iviewx Product Page*. Accessed: Jul. 31, 2019. [Online]. Available: https://www.inition.co.uk/product/sensomotoric-instruments-iview-x-hi-speed/
[22] S. Zugal and J. Pinggera, "Low–cost eye–trackers: Useful for information systems research?" in *Proc. Int. Conf. Adv. Inf. Syst. Eng.* Springer, 2014, pp. 159–170.
[23] E. Dalmaijer, "Is the low-cost EyeTribe eye tracker any good for research?" PeerJ Pre Prints, Tech. Rep., 2014.
[24] A. Martinet, J. Martinet, N. Ihaddadene, S. Lew, and C. Djeraba, "Analyzing eye fixations and gaze orientations on films and pictures," in *Proc. 16th ACM Int. Conf. Multimedia*, 2008, pp. 1111–1112.
[25] K. Naqshbandi, T. Gedeon, and U. A. Abdulla, "Automatic clustering of eye gaze data for machine learning," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 1239–1244.
[26] P. Mannaru, B. Balasingam, K. Pattipati, C. Sibley, and J. T. Coyne, "Performance evaluation of the gazepoint GP3 eye tracking device based on pupil dilation," in *Proc. Int. Conf. Augmented Cognition*. Springer, 2017, pp. 166–175.
[27] Z. Ramdane-Cherif and A. Nait-Ali, "An adaptive algorithm for eye-gaze-tracking-device calibration," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 4, pp. 716–723, Apr. 2008.
[28] H. Istance, R. Bates, A. Hyrskykari, and S. Vickers, "Snap clutch, a moded approach to solving the midas touch problem," in *Proc. Symp. Eye Tracking Res. Appl. (ETRA)*, 2008, pp. 221–228.
[29] B. Velichkovsky, A. Sprenger, and P. Unema, "Towards gaze-mediated interaction: Collecting solutions of the Midas touch problem?" in *Human-Computer Interaction*. Springer, 1997, pp. 509–516.
[30] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Application to Tracking Navigation: Theory Algorithms Software*. Hoboken, NJ, USA: Wiley, 2004.
[31] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Readings Speech Recognition*. Amsterdam, The Netherlands: Elsevier, 1990, pp. 267–296.
[32] *Moby Public Copy of Moby Dick by Herman Melville*. Accessed: Nov. 20, 2018. [Online]. Available: https://www.gutenberg.org/files/2701/2701-h/2701-h.htm

Stephen Bottos was born in Canada, in 1992. He received the Bachelor of Science in mechanical engineering with a focus on materials science and the Master of Applied Science in electrical engineering from the University of Windsor, Windsor, ON, Canada, in 2016 and 2019, respectively.

Since the completion of his master's degree in September 2019, he has been working as a Machine Learning Engineer at Qimia Inc., San Diego, CA, USA. His research interests include signal processing, machine learning, and their applications in various autonomous systems.

Balakumar Balasingam (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from McMaster University, Hamilton, ON, Canada, in 2008.

After his Ph.D. studies, he held a post-doctoral position at the University of Ottawa, Ottawa, ON, Canada, from 2008 to 2010, and then a University post-doctoral position at the University of Connecticut from 2010 to 2012. From 2012 to 2017, he was an Assistant Research Professor with the Department of Electrical and Computer Engineering, University of Connecticut, Mansfield, CT, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON. His research interests are in signal processing, machine learning, and distributed information fusion and their applications in autonomous cyber–physical–human systems. In these areas, he has published over 70 research articles.

Dr. Balasingam received the ISIF Jean-Pierre Le Cadre Best Paper Award (second runner-up) for his paper titled "Maximum likelihood detection in images" at the IEEE International Conference on Information Fusion, Xi'an, China, in 2017.