AUTOMATING THE SEGMENTATION OF EYE MOVEMENTS IN A ROBOTIC VIRTUAL REALITY ENVIRONMENT

by

MARGARET FRANCES SCHRAYER

(Under the Direction of Deborah Barany and Jaewoo Lee)

ABSTRACT

Modern eye-tracking technology can record eye movements at a high temporal and spatial resolution during behavioral tasks, allowing researchers to make inferences about sensorimotor integration and visual processing. Analysis of the gaze data often requires segmentation into fixations, saccades, and smooth pursuits; a process that is very labor intensive when annotated manually. Segmentation methods have been developed to automatically process user experience and reading data, but most use data from subjects viewing static stimuli and are not suited to gaze data containing smooth pursuits. A novel method using hidden Markov models is proposed to automate eye movement classification in virtual reality and robotic environments used in visuomotor neuroscience research. This method performed with 72 percent accuracy when vergence eye movements were considered and 82 percent accuracy when they were excluded.

INDEX WORDS: Hidden Markov Model, Markov Process, Continuous HMM, Eye Movement Classification, Time Series Segmentation, Eye Tracking

AUTOMATING THE SEGMENTATION OF EYE MOVEMENTS IN A ROBOTIC VIRTUAL REALITY ENVIRONMENT

by

MARGARET FRANCES SCHRAYER

B.S., University of Georgia, 2021

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree.

MASTER OF SCIENCE

ATHENS, GEORGIA

2022

AUTOMATING THE SEGMENTATION OF EYE MOVEMENTS IN A ROBOTIC VIRTUAL REALITY ENVIRONMENT

by

MARGARET FRANCES SCHRAYER

Major Professors: Deborah Barany
Jaewoo Lee

Committee: Frederick Maier
Khaled Rasheed

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
December 2022

ACKNOWLEDGMENTS

I would like to express my gratitude to my mentors: Dr. Tarkeshwar Singh, who began mentoring me as a first year undergraduate and proposed this research project; and Dr. Deborah Barany, who helped see the eye movement classification project through a pandemic and many setbacks. Thank you also to Professor Suchendra Bhandarkar and Dr. Jaewoo Lee from the Institute for Artificial Intelligence for supporting this research, and to Dr. Frederick Maier and Professor Khaled Rasheed for stepping in to join my committee for the defense. Finally, thank you to my mother, the other graduate student in the family, for your sage advice—I'm glad we both made it to the finish line this year.

TABLE OF CONTENTS

Acknowledgments **iv**

List of Figures **vi**

List of Tables **vii**

1 Introduction **1**
1.1 Problem Introduction 1
1.2 Overview of Present Contribution 3

2 Background **5**
2.1 Eye Tracking 5
2.2 Time Series Segmentation 8

3 Methods **14**
3.1 Data Collection 14
3.2 Data Labeling 19
3.3 Data Preprocessing 22
3.4 Piecewise Linear Segmentation 26
3.5 Hidden Markov Models 29

4 Results **33**
4.1 Unconsolidated NSLR Reconstruction Dataset 33

5 Discussion **41**
5.1 Limitations 41
5.2 Future Directions 42
5.3 Conclusion 43

Bibliography **44**

LIST OF FIGURES

3.1 A research participant seated at the KINARM End-Point Lab. Reprinted from Barany, D. A., Gómez-Granados, A., Schrayer, M., Cutts, S. A., Singh, T. (2020). Perceptual decisions about object shape bias visuomotor coordination during rapid interception movements. Journal of neurophysiology, 123(6), 2235-2248. 15
3.2 Sample trajectories of each type: Vertical, horizontal, diagonal, sinusoidal, and pseudo-random. 17
3.3 Process for determining the target movement during a trial. 19
3.4 The manual gaze data labeling process 21
3.5 A gaze point of regard trajectory before and after interpolating missing values during a blink. 24
3.6 The same sequence plotted with its manual labels and segmented linear reconstructions generated using three different estimated noise parameter values. 27
3.7 Transition matrices for each state space 31

4.1 Confusion matrices for the best performing models in each state space. 34

LIST OF TABLES

4.1 Supervised Model Classification Metrics for the Simplified State Space (D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity) . . 35
4.2 Supervised Model Classification Metrics for the Full State Space (D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity) 36
4.3 Semi-Supervised Model Classification Metrics for the Simplified State Space (D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity) . . 38
4.4 Semi-Supervised Model Classification Metrics for the Full State Space (D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity) . . 39
4.5 Summary of Most Accurate Classifiers by Category 40

CHAPTER 1

INTRODUCTION

§ 1.1 Problem Introduction

In psychology, vision, and neuroscience research, eye tracking data provides an important window into visuomotor processing. Assessing a research participant's ability to visually track a moving target, compensate for errors in visual tracking, and coordinate hand and eye movements allows us to quantify this ability and make inferences about the integration of visual information with motor control (Latash, 2008). Specially designed tasks performed in a controlled environment allow us to administer these tasks in a way that is not subject to interference from outside influences or unplanned variation in the task conditions. For example, the KINARM End Point Lab (KINARM, Kingston, Ontario, Canada) is designed to precisely control the presentation of visual targets, constrain hand movements to a horizontal plane, and measure the hand and gaze kinematics at a high sampling rate (up to 4 kHz for the hand and 500 Hz for the gaze) (Scott, 1999).

A key step in many eye tracking data analysis pipelines is the classification of gaze data into one of several eye movement types. Eye trajectories can be divided into disjoint periods of fixation, smooth pursuit, and saccade. In fixation, the eye focuses on a static area in the visual field; for example, when a person is looking at an object that is not moving. Though the gaze point of regard may waver during a fixation, it stays within a small radius of the fixated object. In smooth pursuit, the eye tracks an object that is moving along a continuous path. Finally, in saccade, the eye darts rapidly between two targets (Holmqvist et al., 2011).

In addition to the three primary eye movements, the eye makes post saccadic oscillations and vergence movements. Post saccadic oscillations are small corrective eye movements occurring immediately after saccades (Bahill et al., 1975). Vergence is the movement of the pupils closer together or further apart to

focus on an object that is closer to or further away from the eyes (Toates, 1974). Vergence can occur at the same time as either pursuit or saccade movements. Because each eye movement type is controlled by a different physiological mechanism, observing the eye movement sequence during a visuomotor task can reveal underlying neurological processes without more invasive electrophysiology or imaging methods (Young and Sheena, 1975).

Many attempts have been made to automate the process of classifying eye movements from a noisy eye tracking signal, with mixed results. However, most of these do not consider smooth pursuit and are thus not useful for applications involving continuously moving targets (Komogortsev and Karpov, 2013). Another challenge in eye movement classification is the use of augmented or virtual reality environments such as the KINARM End Point lab. Though these environments offer the ability to use more realistic simulations than computer based tasks, they also introduce vergence eye movements and more complicated ocular kinematics (Singh et al., 2016). Of the existing eye movement classification methods, only one that we are aware of is designed specifically for use with the KINARM End Point Lab or systems like it. This method, proposed in Singh et al., 2016, performs with accuracy in the 90s on some datasets but does not perform reliably on gaze data generated from a variety of target trajectories. Along with many other methods developed for other environments, it uses a combination of gaze signal denoising and transformation techniques and setting gaze velocity thresholds to distinguish the eye movements. However, as probabilistic sequential models and deep learning methods have taken hold as the state of the art approach to many time series segmentation problems, they have been applied for gaze data analysis as well (Pekkanen and Lappi, 2017, Zhu et al., 2020). The present study aims to develop a robust approach to eye movement classification in the KINARM End Point Lab environment using a hidden Markov model to predict the most likely sequence of eye movements given a raw gaze point of regard signal. The classification method should take as input a sequence of n two dimensional data points $(X_1, Y_1), (X_2, Y_2), ...(X_n, Y_n)$, representing the gaze point of regard X and Y coordinates for n consecutive samples recorded at $500Hz$, and output a sequence of n eye movements $C_1, C_2, ...C_n$ representing the eye movement class (fixation, saccade, or pursuit) corresponding to each time point.

§ 1.2 Overview of Present Contribution

In this work, a novel gaze data processing pipeline was developed based on existing methods with adaptations to improve performance using data collected during KINARM End Point Lab tasks. First, a body of raw gaze data was collected for use in developing a classification method. An algorithm was developed to generate a series of unpredictable two dimensional trajectories for a visual target. These trajectories were coded into a KINARM End Point Lab task. The moving target was projected to the robot screen while research participants were instructed to follow the target with their eyes. This task was designed to elicit an unpredictable sequence of eye movements at a variety of speeds and directions to create a varied representation of each eye movement type in the training dataset. Because existing KINARM tasks were each designed to study a particular behavioral neuroscience research question, the eye movement data collected during these studies is heavily biased according to the design of the study trials. For example, eye movements may be constrained to a small portion of the workspace where visual targets are presented, or a task may elicit only fixations and saccades with few smooth pursuits. By designing a task to elicit a mixture of eye movements at different speeds and directions across the whole workspace, we aimed to prevent overfitting of a classification method to gaze data from a single task and achieve reliability of classification across tasks.

After data collection was complete, a subset of the data was manually labeled with the onsets and offsets of each eye movement type, allowing a discrete class to be determined for the gaze data observation at each time point. These labels were used to train and test the supervised hidden Markov model classifiers. Data were denoised and preprocessed using methods similar to those described in Singh et al., 2016. The raw output of the EyeLink eye tracker (SR Research Ltd., Ottawa, Ontario, Canada) consisted of the two dimensional gaze point of regard (POR) and the gaze direction vector. Based on these signals, the foveal visual radius and gaze angular velocity were calculated for each time point.

Because of the high temporal resolution of the eye tracker, each eye movement consists of tens of samples for saccades and hundreds for fixations and pursuits. As a result, in a hidden Markov model, which assumes that the probability of transitioning from one state to the next depends only on the current state, the probability of self-transition between eye movements is well above 90% for each eye movements.

This results in predicted eye movement sequences consisting of a single eye movement repeated over and over again (e.g. "Pursuit, Pursuit, Pursuit... Pursuit"), since the extremely high transition probabilities outweigh the more ambiguous information provided by the feature output probability distributions for each eye movement. In this work, segmented linear regression was used to summarize each gaze data sequence as a series of approximately linear segments, reducing the length of input sequences by a factor of about 50. Expressing gaze data sequences in this manner helped tip the balance of transition probabilities between eye movements away from self-transition.

A segmentation method adapted from the one developed in Pekkanen and Lappi, 2017 was applied to all gaze data sequences, both labeled and unlabeled. Each sequence could now be expressed as a sequence of approximately linear segments rather than a sequence of individual 2 ms time points. In the labeled dataset, each segment was given a label based on the surrounding gaze event labels. Four features were developed and calculated for each segment, allowing each segment to be treated as a discrete, atomic unit within a sequence. One hundred four hidden Markov models were then trained and tested on the dataset using. To determine the most useful features and best way to model feature dependencies, model design was varied in four ways:

1. Different subsets of the features were used
2. Of the models using at least two features, some modeled the output probabilities for each feature as independent and some modeled them as dependent
3. Some models were trained and tested on the simplified dataset (not considering vergence eye movements) and some used the original dataset
4. Some models were trained on the labeled dataset alone (supervised learning) and some used semi-supervised learning

Accounting for each of these variations resulted in 104 different models. The top performing models in the simplified and original datasets performed with 83% and 72% accuracy, respectively. The probabilistic sequential model approach shows promise in labeling gaze datasets with smooth pursuits but requires refinements to the segmentation approach and a better training dataset to improve performance.

CHAPTER 2

BACKGROUND

§ 2.1 Eye Tracking

§ 2.1.1 Eye Movement

In order to perform essential functions of the visual system and interact with stimuli in the visual workspace, we must perform thousands of finely tuned precise eye movements every day. The four major classes of eye movements are fixation, saccade, smooth pursuit, and vergence (Holmqvist et al., 2011). During a fixation, the eyes foveate a static visual target by angling the eye to capture the target in the fovea, the central region of the retina that allows the eye to focus on objects in sharp detail. In a smooth pursuit, the eye moves the foveated region along a continuous path to track a moving object, almost like a mobile fixation. Saccades are rapid eye movements from one area of focus to another. For example, shifting the gaze from a coffee cup on a table to a clock on the wall is accomplished by a saccade. A saccade typically takes under 200 milliseconds, depending on the angular distance between targets (Boghen et al., 1974). Objects in between the cup and the clock are not foveated during the saccade. Finally, vergence occurs when the two eyes move in opposite directions to accommodate a change in depth of a foveated stimulus. Vergence occurs simultaneously with either saccades or pursuits; for example, when tracking a ball moving toward the head or glancing back and forth between a notebook on a desk and a clock on the wall (Toates, 1974).

Because eye movement primarily consists of rotation about a single point, eye movements are characterized in terms of the angular kinematics of the eye. The angular velocity of the eye peaks at around 100-250 ms during saccades (Boghen et al., 1974), stabilizes around zero during fixations, and varies from 0 to about 100 degrees per second during smooth pursuits, depending on the angular velocity of the pursuit target (Meyer et al., 1985). Many eye movement classification methods distinguish saccades from

fixations by establishing an angular velocity threshold and labeling sequences with angular velocity above the threshold as saccades (Singh et al., 2016).

Smooth Pursuit

In smooth pursuit eye movements, the eye continuously tracks moving stimuli to maintain objects of interest captured in the fovea, the region of the retina with the highest acuity. Smooth pursuits are driven by similar neural pathways to saccades, and the two movement types typically alternate during pursuit of a moving target as small saccades compensate for error in prediction of the target's motion (Krauzlis, 2004). The angular velocity of the eye during smooth pursuit movements typically stays below 30 deg/s but can be higher in the case of fast moving stimuli (Larsson et al., 2013).

Smooth pursuit can be characterized by statistics such as the smooth pursuit gain (the ratio of gaze point of regard velocity to target velocity), phase lag (the time between target movement onset and smooth pursuit onset), and number of catch up saccades occurring during a pursuit (De Brouwer et al., 2002). These metrics offer a snapshot of the quality of a subject's ability to track moving stimuli. The characteristics can be affected by the predictability of the target path, task demands (Barany et al., 2020), concurrent motor activity (Niehorster et al., 2015), age (Sharpe and Sylvester, 1978, Gómez-Granados et al., 2021), and neurological conditions. For example, manually tracking a moving target has been found to improve eye tracking of a moving target (Niehorster et al., 2015, Danion and Flanagan, 2018).

Smooth pursuits hold special significance in the study of mental and neurodegenerative disorders. Smooth pursuit dysfunction occurs in schizophrenia (O'Driscoll and Callahan, 2008), Alzheimer's Disease (Fletcher and Sharpe, 1988), and Parkinson's Disease (Helmchen et al., 2012). Because of the role of smooth pursuit in visuomotor function, analysis of smooth pursuits during a motor task can be used to quantify the motor deficits present in many of these disorders (Fletcher and Sharpe, 1988).

In contrast to other methods of measuring motor function such as task performance, electromyography, or force measurement; which are often either invasive or subject to interference from other factors; eye tracking provides a window into the visuomotor system without compromising the integrity of the

task (Young and Sheena, 1975). The high resolution EyeLink tracker integrated with the KINARM allows for complete control over the task stimuli and precise measurement of both hand and eye movement.

Though many algorithms exist to detect saccades in eye-tracking data, and a few have even been developed for data collected in three dimensional environments such as the KINARM workspace, few are able to accurately distinguish fixations from smooth pursuits. Santini et al. developed a probabilistic algorithm to segment all three eye movements based on both the raw eye position data and a "movement ratio" representing the proportion of recent time points at which movement was observed (Santini et al., 2016).

§ 2.1.2 Eye Tracking Technology

Eye trackers can be divided into head-mounted systems, which record the position of the gaze relative to the head, and table-mounted systems, also known as remote systems. Eye tracking methods include electrophysiological measurements of facial areas, contact lenses with embedded mechanical devices, and a host of videography based methods employing various image processing algorithms to derive the eye movements (Duchowski, 2017). Most of these methods measure eye movements relative to the head. To analyze interactions between the visual system and visual stimuli in the three dimensional visual workspace, we need to measure the gaze point of regard (POR), which is computed by estimating the orientation of the eye in three dimensional space and calculating the intersection point of the gaze vector and the plane of the target or stimulus. Video-based corneal reflection eye trackers, such as SR Research's EyeLink system, accomplish this goal by using image processing algorithms to locate two reference points on the eye: The center of the pupil and the reflection of an infrared light source (illuminator) off the cornea. Tracking both points allows the algorithm to distinguish eye movement from head movement (Duchowski, 2017).

§ 2.1.3 Eye Tracking Applications

Eye tracking technology is used to study attention, cognition, and visuomotor function in humans and non human primates. Notably, eye tracking has been used to improve our understanding of eye movement during reading and disorders such as dyslexia (Holsanova et al., 2006; Jarodzka and Brand-Gruwel, 2017;

Jones et al., 2008). Eye tracking is also used in the software industry for user experience research (Bergstrom and Schall, 2014).

In neuroscience, eye tracking has the potential to improve our understanding of visuomotor function by allowing us to analyze the relationship between eye movement and body movement during a variety of behavioral tasks; however, analysis methods developed for reading and user experience studies are not always suited to eye tracking data from other visuomotor tasks. Reading generally consists of alternating fixations and very small saccades between words and sentences (Jarodzka and Brand-Gruwel, 2017), and both reading and software use involve small amplitude saccades within a limited visual workspace. In contrast, both real-world physical activities and laboratory visuomotor tasks often involve eye movements within a larger visual workspace, necessitating larger saccades. In addition, tracking moving objects in the workspace elicits smooth pursuits, which are not usually present in reading and user experience data. These differences demand different eye movement data analysis techniques for reading research, user experience research, and visuomotor neuroscience research.

§ 2.2 Time Series Segmentation

The eye movement classification problem comprises a time series segmentation problem. Naturally occurring in language, physiology, behavior, and countless other domains, time series are the key to predicting future events and generating simulated sequences. In addition to sequence generation and prediction, time series analysis can be used to segment data streams into discrete categories. Time series segmentation problems arises often in the study of cognitive processing, where physiological or behavioral signals must be analyzed to detect incidences of a certain process or cognitive event. In order to classify portions of a stream of gaze data as fixations, saccades, or pursuits, it is necessary to divide the data into subsequences at the correct transition points and to classify each subsequence as one of the possible eye movements. Since multiple channels of eye tracking data are available, including angular velocity, Euclidean position, and pupil area, this problem is known as a multivariate time series segmentation problem. Proposed techniques for analyzing eye tracking data include both techniques developed specifically for gaze data segmentation and general time series segmentation techniques adapted for this purpose.

§ 2.2.1 Piecewise Linear Segmentation

Piecewise linear segmentation is a popular time series segmentation approach. Implementations vary but consist of the following steps:

1. Divide the entire time series into segments
2. Approximate each segment as a straight line using a linear regression method

In the eye tracking domain, this approach has been used as an alternative to low-pass or Savitzy-Golay filtering, in which filtered signals depend on the neighboring signal values. In a 2017 paper Pekkanen and Lappi propose approximating the maximum likelihood estimated piecewise linear segmentation of the signal, then classifying each linear segment using a hidden Markov model to identify the most likely sequence of gaze events (Pekkanen and Lappi, 2017). This approach outperformed most other models in agreement with human raters but faltered when it came to distinguishing fixations from smooth pursuits.

§ 2.2.2 Velocity Threshold Methods

One approach to segmenting gaze data is to establish hand picked or data driven angular velocity thresholds to distinguish fixations from smooth pursuits (lower threshold) and pursuits from saccades (higher threshold). Singh et al. developed a velocity-threshold-based eye-movement segmentation method tailored for the KINARM, but the method has had mixed success and adaptability to different subjects and tasks: Accuracy ranged from over 90% in some tasks to only 33% in others (Singh et al., 2016).

§ 2.2.3 NLP Approaches

Natural language processing (NLP) is the manipulation of text and speech to predict future language expression or classify samples of language. For example, NLP methods can be used to generate automatic responses to questions posted by humans or classify written stories into genres. Though natural-language processing is not considered a time series problem since it does not involve timestamped numerical data, a rich body of methods have been developed for NLP sequence-analysis problems, and some can be applied to time-series classification tasks. A few of these methods are reviewed here.

Language Models

A language model is a parametric model that learns to predict the probability of a sequence of words given the history of the language stream. After general training of a language model, fine-tuning on a task-specific corpus allows a model to perform well on a specialized language task. To eliminate the need for such extensive fine-tuning, researchers at OpenAI developed GTP-3 (Brown et al., 2020), an autoregressive language model with 175 billion parameters, ten times the status quo. The larger model could be trained for specific tasks using just a few examples instead of the large fine-tuning corpus. Results of this method were promising in some domains but not others. In the eye-tracking classification problem, this result is encouraging because the characteristics of eye-movements and eye-tracking data vary greatly between individuals, experimental tasks, and physical experimental setups. With sufficient data, it may be possible to train an eye-movement classification model on a sufficiently large corpus of general eye-tracking data, then fine-tune or use a "few-shot" approach on task-specific or individual-specific data to improve the accuracy of classification.

Attention-Based Sequence Transduction

Sequence transduction is the process of outputting a time step of predicted signal for each time step of input signal. Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a numerical representation of the sequence. Researchers at Google found that this attention technique alone can perform better than a combination of convolutional or recurrent deep neural networks and attention on NLP tasks, with lower network training time to boot (Vaswani et al., 2017). This result may have some promise in terms of applying the same technique to eye-tracking data analysis, but there is little discussion in the literature about applying attention techniques to sequence classification as well as transduction.

§ 2.2.4 Hidden Markov Models

Hidden Markov models (HMM) are a popular time series segmentation method that models time series data as a sequence of hidden states that are not readily observable. In a Markov process, successive states

depend only on the previous state, not on entire state histories (the "Markov assumption"). However, non-Markovian processes can be modeled as Markov processes by using a transformed state space in which state histories of a length greater than one from a non-Markovian process are encoded as single states in a Markov process. A hidden Markov model is a method to find hidden state given information about a visible state, which takes the form of one or more observations at each time point. A Markov process can thus be characterized by three sets of probability distributions: A "initial distribution" over the state space dictating the initial state in a sequence, a "transition matrix" dictating the probability of transitioning to each state from each predecessor state, and the uni- or multi-variate "emission distribution" describing the observation data given each possible hidden state. A classic example involves predicting whether the weather is sunny or rainy based on a man's activities: walking, shopping, or cleaning. If the emission distribution determining the man's activities based on the weather is known, it is possible to use Bayesian statistics to predict the weather even if it is not directly observable.

The ternary eye movement classification problem can be represented using a hidden Markov model as follows:

- The start probability distribution P_{x_1} consisting of the start state probabilities $P_{x_1}(Fixation)$, $P_{x_1}(Pursuit)$, and $P_{x_1}(Saccade)$
- The transition probability matrix $P(x_{k+1}|x_k) = T(x_{k+1}, x_k)$ enumerating the probabilities of transition to each state (fixation, pursuit, or saccade) given each possible current state (fixation, pursuit, or saccade)
- The observation probability distributions $P(y_k|x_k)$ describing the probability distributions of observed values for each feature y and given each state x.

Hidden Markov Models for Eye Movement Classification

In eye movement classification, the hidden state is the current eye movement: Fixation, saccade, or smooth pursuit. The observable state is composed of one or more features of the eye movements, such as absolute angular velocity, the x and y components of cartesian velocity, angular acceleration, pupil area, and calcu-

lated foveal visual radius. Zhu et al. compared a hierarchical method of classifying eye-movements using HMMs by first separating saccades from fixations and smooth pursuits, then separating fixations from smooth pursuits (Zhu et al., 2020). This approach worked better than a three-state HMM built to classify all three events at once. In another eye-movement study, HMMs were used to predict the region of interest (ROI) that would be foveated by the eye in subsequent time steps. Though this problem differs from ternary eye-movement classification, the results show some promise that HMMs could work similarly on this problem. In a binary ROI prediction problem, the algorithm performed with 81% accuracy (Coutrot et al., 2018).

Pekkanen and Lappi developed an eye movement classification method in which the gaze point of regard component signals are filtered by fitting a piecewise linear function to approximate the signal (Pekkanen and Lappi, 2017). Each linear segment can then be considered an atomic element of a gaze data sequence. The hidden Markov model-based classifier operates on sequences of linear segments rather than sequences of individual recorded data points, and each segment receives a single classification. Pekkanen and Lappi forgo other pre-classification filtering and denoising steps entirely, relying on the segmented linear regression algorithm to reduce noise while closely approximating the signal.

The HMM developed by Pekkanen and Lappi uses hand tuned transition probabilities rather than learning transition probabilities from a labeled dataset. Included in their transition model is a decreased probability of direct transitions from pursuit to fixation and fixation to pursuit, as a saccade usually occurs between these two eye movements. The model uses two output features: Gaze velocity and geometric angle between segments. The algorithm proved effective at distinguishing saccades and post saccadic oscillations from slower fixations and smooth pursuits; however, fixation and smooth pursuit overlapped extensively in the chosen feature set, and classification of these eye movements had low agreement with human coders.

Zhu, Yan, and Komogortsev describe a second hidden Markov model-based approach to the ternary eye movement classification problem (Zhu et al., 2020). They employ a hierarchical approach to eye movement classification: They fit an HMM using a single feature to perform a rough classification of the eye movements, then use a second HMM with a different feature to refine the classification. Their exact

method of weighing the results of the two classifiers is unclear, as is their training process. The second step in the hierarchical process uses position data, which is task- and environment-specific and therefore not robust to changes in the task demands and environment.

Coutrot et al. developed an HMM using gaze features, though they were classifying scan paths and not the eye movements themselves (Coutrot et al., 2018). Gaze point of regard position data was the only feature used, and stimuli were static images that did not elicit smooth pursuits.

Modeling of Output Probabilities

Pekkanen and Lappi and Coutrot et al. both transformed the features used so that they followed a roughly Gaussian distribution (Pekkanen and Lappi, 2017, Coutrot et al., 2018). The output probabilities for multiple features could then be modeled as multivariate Gaussian distributions, allowing the model to capture interaction between the features. Zhu et al. used a single feature, also modeled using a Gaussian distribution, for each pass of their hierarchical HMM (Zhu et al., 2020).

CHAPTER 3

METHODS

§ 3.1 Data Collection

§ 3.1.1 Apparatus

The objective of this research is to develop an eye movement segmentation method to analyze gaze point of regard data measured in the transverse plane in a robotic virtual reality environment (KINARM End-Point Lab, KINARM, Kingston, Ontario, Canada).

The KINARM robot, shown in Figure Figure 3.1, allows researchers to design visuomotor tasks where hand movements take place in the same visual workspace as eye movements: Stimuli are presented on a horizontal display, and reaching movements are performed by manipulating a handle whose movement is constrained to the transverse plane. The two dimensional point of regard and three dimensional gaze direction vector are recorded by the integrated EyeLink 1000 eye tracker (SR Research Ltd., Ottawa, Ontario, Canada) at 500 Hz.

The main components of the KINARM experimental setup integrated with eye-tracking are as follows:

- Remote monocular eye-tracker positioned in back of the robot
- Illuminator positioned in back of the robot to light the face for eye-tracking
- Force-sensing handle which moves freely in the transverse plane
- Horizontal display showing the stimulus and a cursor to indicate the handle position
- Locking chair to keep the participant in place during data collection

Figure 3.1: A research participant seated at the KINARM End-Point Lab. Reprinted from Barany, D. A., Gómez-Granados, A., Schrayer, M., Cutts, S. A., Singh, T. (2020). Perceptual decisions about object shape bias visuomotor coordination during rapid interception movements. Journal of neurophysiology, 123(6), 2235-2248.

- Robot computer to control the display and handle force actuation of the robot
- GUI computer to run the user interface to the robot and collect output data
- Gaze computer to control the eye-tracker and run the eye-tracking user interface

In contrast to experiments where the hand controls a mouse or joystick to manipulate a cursor on a screen in the frontal plane, the KINARM End-Point setup integrated with eye tracking allows for more lifelike investigation of the relationship between eye- and hand-tracking of visual targets. This technology allows us to conduct behavioral research on brain function in patients with neurological and neurodegenerative disorders as well as differences in visuomotor processing between younger and older adults in a controlled yet realistic environment. However, eye tracking in a horizontal plane presents unique challenges to eye movement segmentation and data analysis. A specialized method is needed to automatically segment eye-tracking data recorded on the KINARM into fixations, saccades, and smooth pursuits.

§ 3.1.2 Task Design

To generate a corpus of suitable eye-tracking data to train and test the experimental models, a task was developed for the robot in Simulink/MATLAB (version R2015a Service Pack 1) using the KINARM Task Development Kit (KINARM, Kingston, Ontario, Canada). The task stimulus consists of a single 1 cm white circle, which the participant was instructed to track with their eyes throughout the duration of each trial. The hand tracking and force sensing features of the KINARM were not used in the task, and participants were instructed to keep their hands relaxed away from the robot manipulandum. The task consists of 100 trials divided into four blocks of 25 trials, each about 5-30 seconds in duration, with a 1000 ms inter-trail break with no stimulus between each trial. Participants took a two minute break between each block and were offered the option to take a longer break if their eyes were fatigued.

Target Trajectory Types

In order to train a model to reliably detect gaze events in a variety of speeds and directions. Five trajectory types were used:

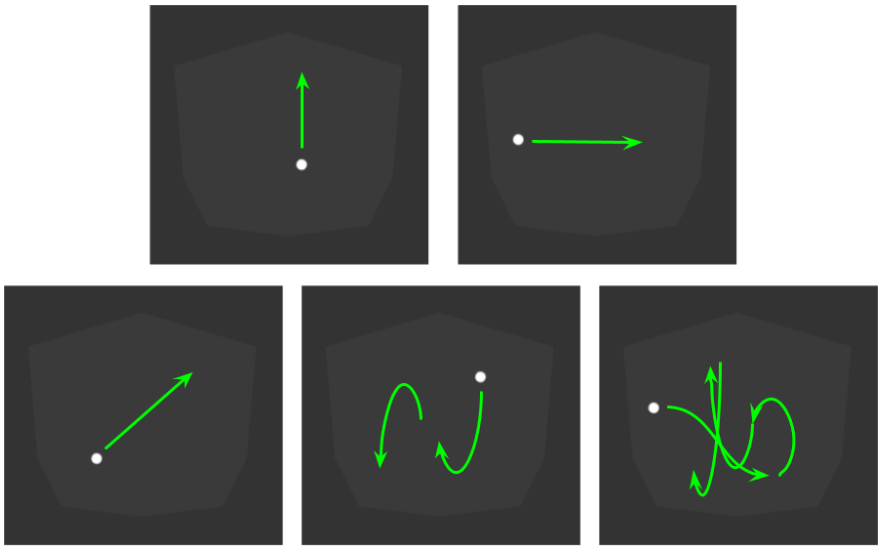


Figure 3.2: Sample trajectories of each type: Vertical, horizontal, diagonal, sinusoidal, and pseudorandom.

1. Horizontal
2. Vertical
3. Diagonal, with a variety of slopes
4. Sinusoidal, with a variety of periods and amplitudes
5. Pseudorandom, following one of the pseudorandom trajectories developed in Danion and Flanagan, 2018 to inhibit the participant from predicting the target's path.

For each trajectory type, trajectory start and end positions within the workspace were varied. Trajectories of all types were included in each block of 25 trials in a random order. Samples of each trajectory type are shown in Figure 3.2.

Trial Description

Each trial begins with a mandatory fixation: The target appears stationary at the start position, and the trial does not begin until the recorded gaze point of regard remains within a logical radius of 5 cm for 200 ms. This gaze control ensures that the task does not proceed for too long if the gaze signal is lost. After the mandatory fixation period, the target moves along a predefined trajectory. Every 1000 ms, an event is randomly generated (3.3):

- With probability 1/3 a static event, designed to elicit a fixation, is generated and the target remains stationary until the next non-static event is generated
- With probability 1/3 a continuous event, designed to elicit a smooth pursuit, is generated and the target moves continuously along the trajectory
- With probability 1/3 a disjoint event, designed to elicit a saccade, is generated and the target jumps ahead along the trajectory, to the same position it would reach if traveling along the trajectory for 1000 ms at speed. Another event is generated immediately after to determine the behavior of the target following the jump. If multiple disjoint events are generated consecutively, they are chained together and the target jumps ahead by 1000 ms per disjoint event.

The trial ends once the entire trajectory has been traversed twice. Because of variability in the number of static events, which extend the time required to traverse the trajectory; and disjoint events, which diminish the time required to traverse the trajectory; the total duration of the trials varies from approximately 5-25 seconds.

§ 3.1.3 Participants

Twelve right handed participants (7 female, ages 18-23) with no history of visual, neurological, or skeleto-muscular problems were recruited for the study. Participants were given a brief description of the overall purpose of the laboratory's research but were naive to the goals of the study.

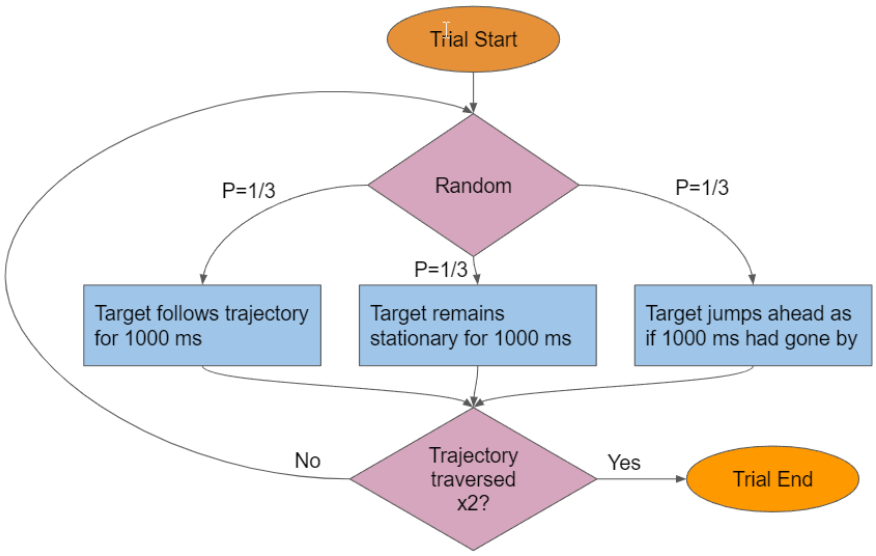


Figure 3.3: Process for determining the target movement during a trial.

§ 3.1.4 Procedure

Participants trained on the task using a slide presentation incorporating still images and a video of the task generated in Dexterit-E Explorer, the KINARM data analysis software. Participants then completed a practice block of ten trials. Prior to starting the task, the gaze tracker was calibrated to the participant. As many attempts as needed were made to achieve an eye tracking calibration with a maximum error of less than two degrees for all calibration points. Each participant completed four blocks of twenty-five trials each, split evenly between the five trajectory types, in a random order. Participants rested for two minutes or as needed between each block and were allowed additional breaks as needed to prevent fatigue. One participant had to stop after the first block because of poor eye tracking data quality.

§ 3.2 Data Labeling

To produce enough labeled data to support the modeling process, data files from the first six participants were manually labeled by at least two trained reviewers each. The manual review process allowed for the data to be checked for quality before inclusion in the model development process. Of the 425 trials completed by the first five participants, 259 had to be excluded from the dataset due to poor eye-tracking

data quality. After the eye-tracking calibration procedure was revised, eye-tracking data quality improved, and very little data had to be excluded from the remaining participants.

§ 3.2.1 Manual Labeling Procedure

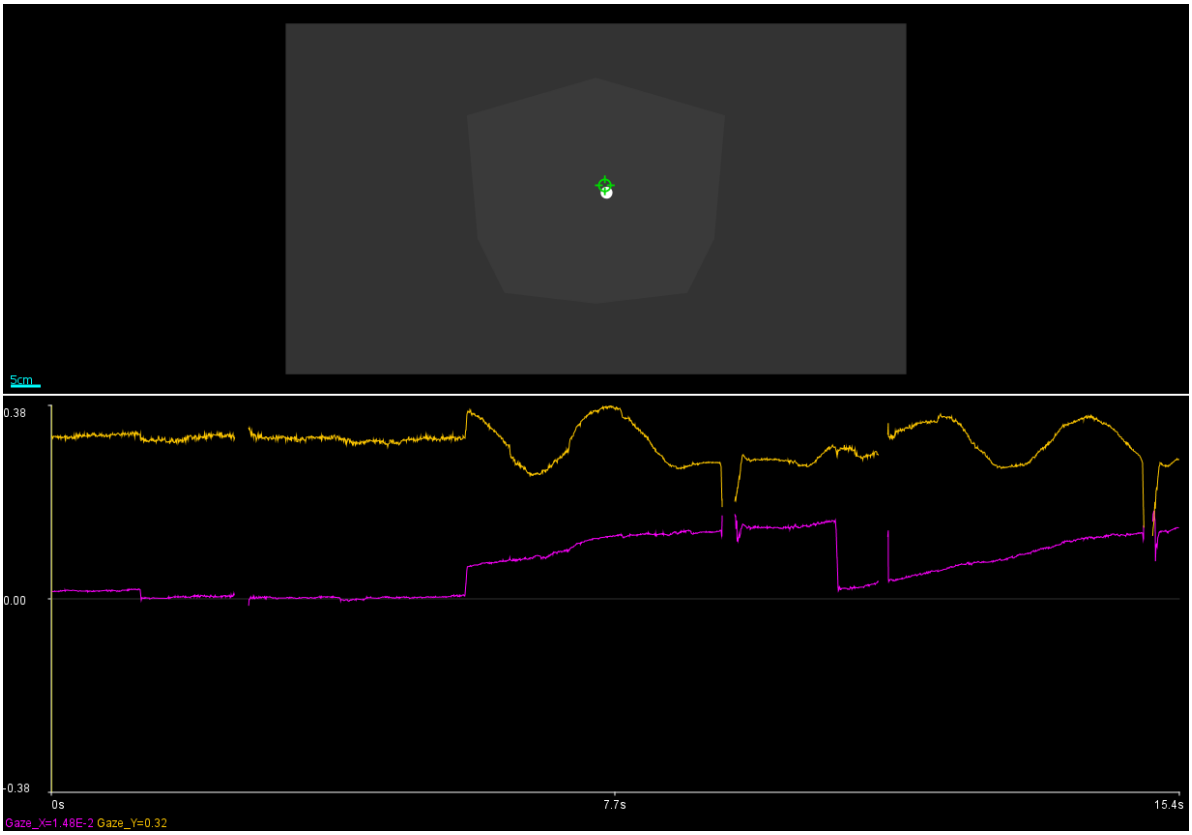
Seven coders were trained to identify and label gaze events based on plots of the gaze X and Y coordinates on the two dimensional plane of the KINARM robot's virtual reality display, playback of the stimulus and recorded gaze point of regard positions during the trial, and known properties of the different eye movement types. The labeler's view of a sample trial in Dexterit-E Explorer is shown in Figure 3.7a. After the first pass at manual labeling, each labeled data file was reviewed at least twice by a different trained labeler. To aid the review process, the gaze signal was plotted with existing labels to quickly check for precision and accuracy. A plot of the sample trial with labeles included is shown in Figure 3.7b.

§ 3.2.2 Gaze Event Label Classes

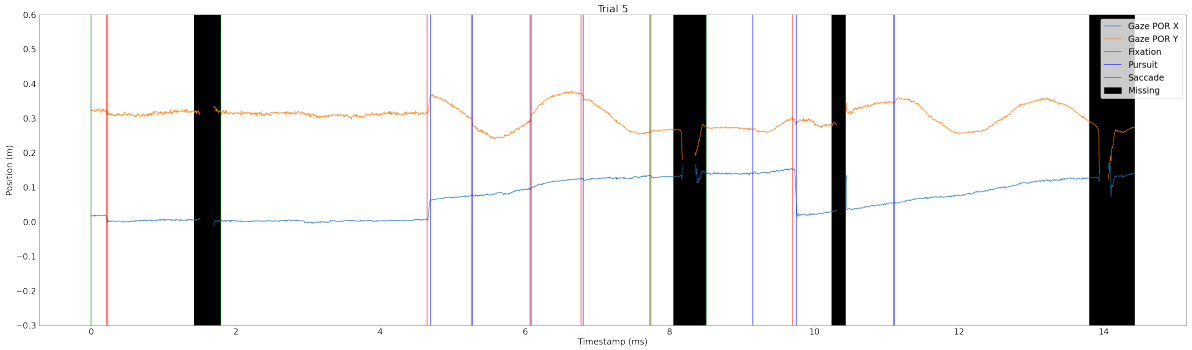
During the labeling process, pursuits and saccades with and without vergence were distinguished by using separate labels. Pursuits and saccades that involved change in the Y coordinate of the gaze point of regard signal were labeled as "with vergence," while those restricted to the X coordinate were labeled as "without vergence." Blink onsets and offsets were labeled, providing a sufficient margin around each blink to avoid mislabeling parts of the blink as another gaze event. Finally, periods of either missing data or data too noisy to reliably be labeled was labeled as "missing." In all, there were seven gaze event label classes: Fixation, pursuit without vergence, pursuit with vergence, saccade without vergence, saccade with vergence, blink, and missing. A simplified coding that ignored vergence was derived from the original coding, with five classes: Fixation, pursuit, saccade, blink, and missing.

§ 3.2.3 Challenges in the Manual Labeling Process

The process of manually recording gaze event onsets and offsets based on the gaze point of regard X and Y signals is very labor intensive and prone to human error. Differences in the scaling of gaze position signal plots made it difficult to identify small saccades, with low amplitudes and duration as short as 10 ms, in some cases. Additionally, in noisy data it is difficult to distinguish these small saccades from surrounding



(a) A gaze data labeler's view of a single trial. The green crosshairs indicate the gaze point of regard.



(b) The same trial, plotted along with gaze event labels recorded by a trained manual labeler.

Figure 3.4: The manual gaze data labeling process

noise in the signal and to identify exact event onset and offset times. Each labeled trial was included in the dataset only after three rounds of review to ensure accurate labeling of gaze events.

§ 3.2.4 Decision to Include or Exclude Data

Eye tracking data quality for the first five participants was mixed. In many trials, the gaze signal was mostly or completely absent as the eye tracker could not pick up either the head target or the pupil. In others, the signal was so noisy that it could not be manually labeled. Gaze labelers were instructed to mark sections of noisy or missing data within an otherwise usable trial as "Missing," and these sections were excluded from the dataset. Labelers were instructed to skip trials that contained little salvageable data. Based on this instruction, all trials that were within a labeled data file (corresponding to a block of 25 trials) but left unlabeled by the coder were presumed unusable and left out of the dataset.

Trials from unlabeled data files were evaluated for excessive noise separately, during the piecewise linear segmentation step.

§ 3.3 Data Preprocessing

Based on the hand-coding of the gaze events, a gaze event label was applied to each data point in each labeled trial. Because the robot records data at 1000 Hz while the gaze tracker records data at 500 Hz, the data was downsampled by a factor of two to remove all duplicate gaze data entries. If a trial contained blinks or periods of missing data, it was divided into smaller subsequences excluding the unusable blinks and missing data.

§ 3.3.1 Foveal Visual Radius

The fovea comprises 5 degrees of the visual field. The gaze direction vector output from the eye tracker was used to compute the approximate foveal visual radius given the X and Y coordinates of the gaze point of regard; that is, the radius of the circle on the transverse stimulus plane captured by the fovea. The

formula for calculating the estimated foveal visual radius (FVR) in the two dimensional target plane is:

$$FVR = (\frac{\tan{(\tan^{-1}(-\frac{\sqrt{x^2+y^2}}{z}) + 2.5)}}{-\frac{\sqrt{x^2+y^2}}{z}} - 1) * \sqrt{x^2+y^2}$$

where x, y, and z are the components of the gaze vector output by the eye tracker. The estimated foveal visual radius can be used to distinguish eye movements; for example, a fixation has ended when the gaze point of regard changes by a distance greater than the foveal visual radius.

§ 3.3.2 Blink Detection

The blink detection and interpolation methods developed in Singh et al., 2016 and Larsson et al., 2013 were applied to the dataset with some modifications. The following steps were used to process blinks:

1. Periods of missing eye tracking data were located. If a gap in the eye tracker signal was shorter than 700 ms, blink interpolation was attempted.

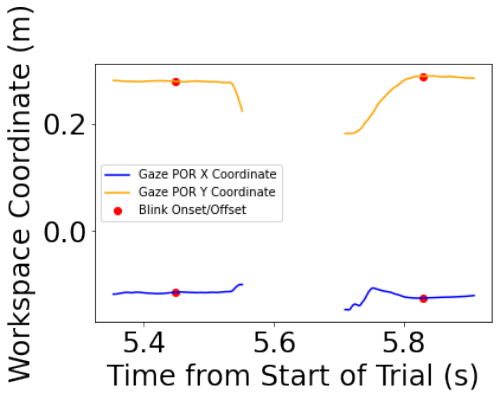
2. The gaze point of regard X and Y coordinate signals surrounding the gap in the data were filtered using a low pass Butterworth filter with a cutoff frequency of 10 Hz.

3. The first local minima in the gaze point of regard Y signal before and after the gap in the data (red dots in Figure 3.5a) were taken to be the boundary of the blink.

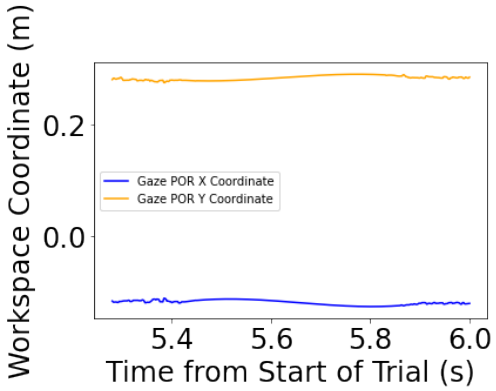
4. The filtered values of the gaze point of regard X and Y coordinate signals within a margin of 30 ms before and after the blink were used to perform a cubic spline interpolation of the signals during the blink, and the missing or compromised signal during the blink was replaced by the interpolated values as seen in Figure 3.5b.

Relabeling Labeled Data After Blink Correction

Relabeling labeled trials after blink detection and interpolation posed a challenge because the labels immediately before and after each blink alone were not sufficient to estimate the label that should comprise the



(a) Before interpolation



(b) After interpolation

Figure 3.5: A gaze point of regard trajectory before and after interpolating missing values during a blink.

blink itself. For example, in most cases, blinks take place during a fixation and the gaze point of regard returns close to its position before blink onset after the blink. However, blinks can also accompany saccades and pursuits. In cases where the blink is accompanied by a change in the gaze point of regard exceeding the foveal visual radius (FVR), it can be challenging to determine whether the blink should be labeled as a saccade or a pursuit, especially given the wide margins around the blink required for interpolation.

Without an existing eye movement classification method, labeling interpolated blinks requires expert knowledge, and so blink interpolation was not applied to labeled trials. Instead, in labeled trials blinks were treated as equivalent to periods of missing eye tracking data values caused by other factors such as glare or poor calibration. These missing values were excluded from the labeled dataset. Because labelers were trained to mark blinks and overly noisy or missing values in each trial with clear margins, the labeled dataset remained clean.

§ 3.3.3 Splitting into Subsequences

Following the identification of sections of missing data, each trial was split into subsequences along the lines of the margins of the missing sections. Subsequences shorter than 250 ms were discarded.

§ 3.3.4 Filtering

Filtering was applied to the gaze point of regard and gaze vector component signals based on methods used in Singh et al., 2016:

1. First, a low pass Butterworth filter with a cutoff frequency of 10 Hz was applied to all gaze signals
2. Second, a Savitzky-Golay filter with order 4 and window size 21 was applied to all gaze signals

§ 3.3.5 Gaze Angular Velocity Computation

The gaze angular velocity was computed by calculating the magnitude of the angular displacement from the previous observation at each time point using the following formula:

$$\Delta G_i = \arccos(\vec{G_i} \cdot \vec{G_{i-1}})$$

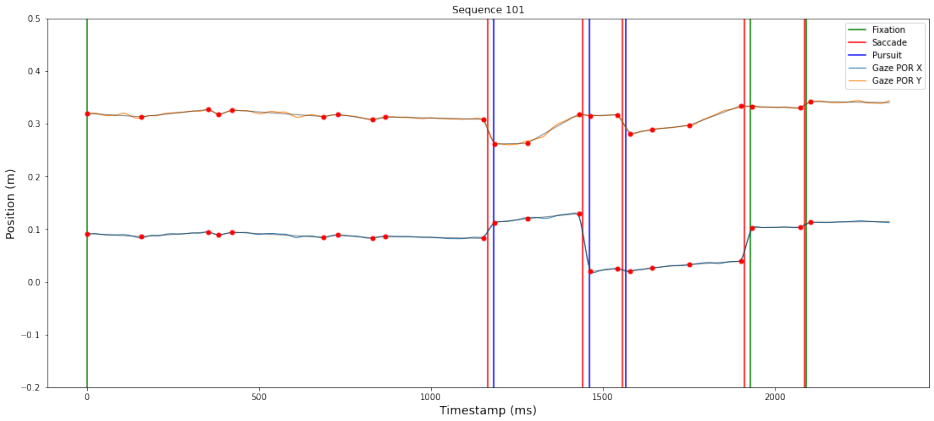
where $\vec{G}_i$ is the gaze vector, output by the eye tracker and then filtered as described above, at time point i. The computed angular velocity signal ΔG was filtered again by a low pass Butterworth filter with cutoff frequency 10 Hz.

§ 3.4 Piecewise Linear Segmentation

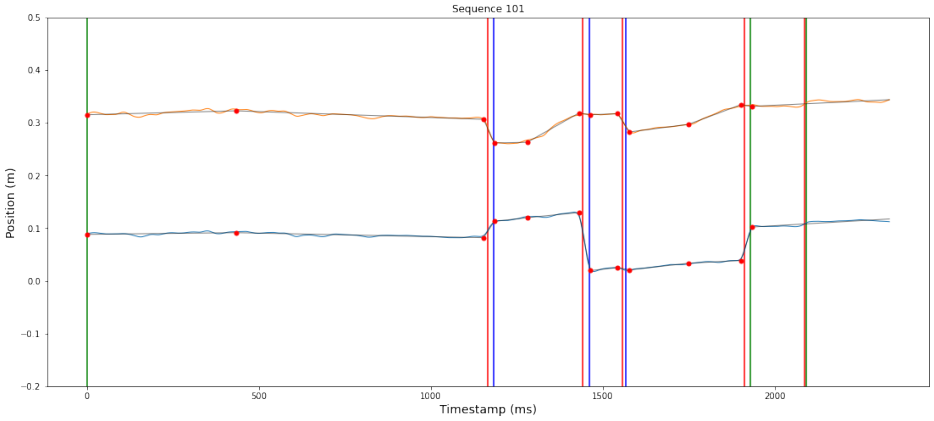
The processed gaze data cannot be modeled effectively as a Markov process taking every time point (representing a two millisecond interval) as an element of a sequence. Because of the high sampling rate, transition probabilities between the hidden states corresponding to different gaze events (e.g. fixation-to-pursuit or saccade-to-fixation) are miniscule compared to the probabilities of self-transition (e.g. the probability that the gaze will still be in a fixation at the next observation, two milliseconds later, given that it is currently in a fixation). A hidden Markov model that treats each two millisecond time step as a separate observation, then, tends to simply predict the same state value for every time point in the sequence, never transitioning out of the start state. This problem cannot be mitigated by downsampling the sequence because self-transition probabilities still dominate, and the high data sampling rate is needed for precision because small saccades can have duration as short as ten milliseconds.

§ 3.4.1 Segmented Linear Regression

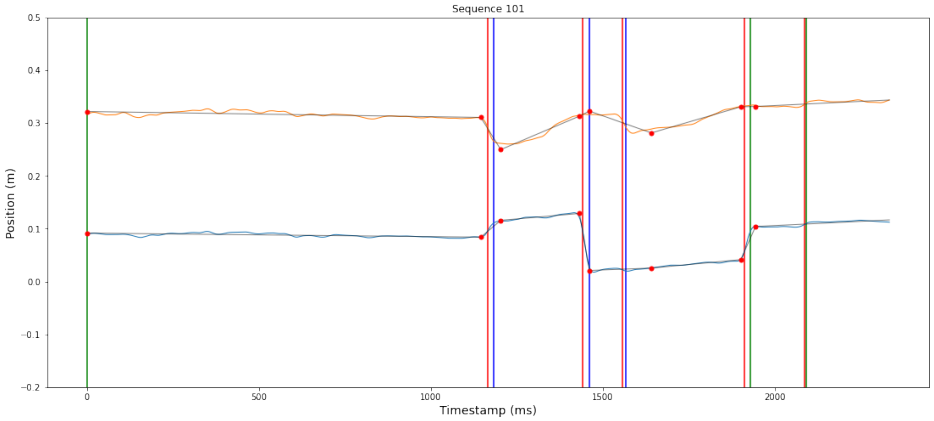
The chosen solution to the problem of high self-transition probabilities is segmented linear regression. In this paradigm, the naive segmented linear reconstruction (NSLR) method described in Pekkanen and Lappi, 2017 was applied to automatically generate a variable length sequence of variable length linear segments that closely approximate the gaze point of regard X and Y coordinate signals. The segment division points are the same for both coordinates. The result is a piecewise linear function approximating the gaze signal. The estimated noise level in the gaze signals can be specified as a parameter of the NSLR algorithm. Figure 3.6 shows an example of a sequence plotted alongside its NSLR reconstructions generated with three different estimated noise parameters.



(a)



(b)



(c)

Figure 3.6: The same sequence plotted with its manual labels and segmented linear reconstructions generated using three different estimated noise parameter values.

§ 3.4.2 Postprocessing and Labeling of Segmented Linear Reconstructions

After generating NSLR reconstructions of every labeled and unlabeled sequence in the dataset, the first and last segment of each reconstruction were discarded. Because the first and last segments were adjacent either to the start or end of the trial or to an area of missing gaze data values, the durations of these segments did not accurately reflect the durations of their corresponding eye movements. Trials from labeled data files that were noisy or contained a lot of missing values were eliminated from the dataset during preprocessing, before NSLR reconstruction. Because unlabeled data files were not hand reviewed, many still contained some noisy trials. However, noisy sequences from unlabeled data files could be automatically identified and removed from the dataset after NSLR reconstruction because they contained an excessive number of segments.

Labeling NSLR Reconstructions

After generating NSLR reconstructions of labeled sequences, each segment was assigned a label based on the original label timestamps for the sequence. Three methods were tested for labeling NSLR reconstructions:

1. If at least 10% of the time points in a segment are labeled as saccade, then label the segment as a saccade. Otherwise, label the segment according to the plurality of its time points' labels (either fixation or pursuit). The 10% threshold was chosen because imprecise manual labeling of saccade onsets and offsets sometimes led to saccade segments that did not contain a plurality of time points labeled as saccade.

2. Iterate over the list of gaze events in the sequence in order of timestamp. Assign each event to the segment starting at the nearest segment division point that has not already been labeled and does not violate the order of the gaze events. Then, in a second pass, assign any unlabled segment the same label as its predecessor.

After manually examining several NSLR reconstructions labeled using both methods, the first option was found to be a more reliable method of labeling reconstructions.

§ 3.4.3 Feature Calculations for Segmented Linear Reconstructions

Because the individual segments resulting from NSLR reconstruction correspond to sequence elements in the hidden Markov model, features had to be calculated for each segment:

1. D: Segment duration (s)
2. S: Cartesian speed of the gaze point of regard (m/s)
3. V: Average angular velocity (deg/s)
4. A: Angle between the segment and its predecessor (deg), a feature used in Pekkanen and Lappi's NSLR-HMM implementation (Pekkanen and Lappi, 2017)

§ 3.5 Hidden Markov Models

To find an appropriate model for the gaze sequences, hidden Markov models were fit to the labeled NSLR reconstruction data. The models were applied to the eye movement classification problem as described in 2.2.4. Four factors defined each hidden Markov model:

1. State space: Either the original state space enumerated in 3.2.2, incorporating vergence (five states), or the simplified state space without vergence (three states).
2. Distribution type: Output probabilities for each feature could be modeled as either dependent or independent
3. Learning type: Either supervised or semi-supervised learning
4. Feature set: Models were trained using each nonempty subset of the five features listed in 3.4.3.

In all, $2 * 2 * 2 * (2^4 - 1) - (2 * 2 * 4)$=104 models were tested (since the distinction between dependent and independent features is meaningless for single-feature models). The open source Python package pomegranate (Schreiber, 2018) was used for all hidden Markov models.

§ 3.5.1 Modeling the Starting and Transition Probabilities

For each state space (unsimplified and simplified), the start state probabilities and transition probabilities were calculated based on all start states and transitions in the labeled NSLR reconstructions. Transition matrices for each state space are shown in 3.7.

§ 3.5.2 Modeling the Output Probabilities

To facilitate modeling of the output probabilities for each feature set, transformations were applied to the data. The cosine of the angle between segments was Fisher transformed, and all other features were log transformed. The resulting feature data could be modeled effectively for each state using normal distributions for independent feature models or multivariate Gaussian distributions for dependent feature models.

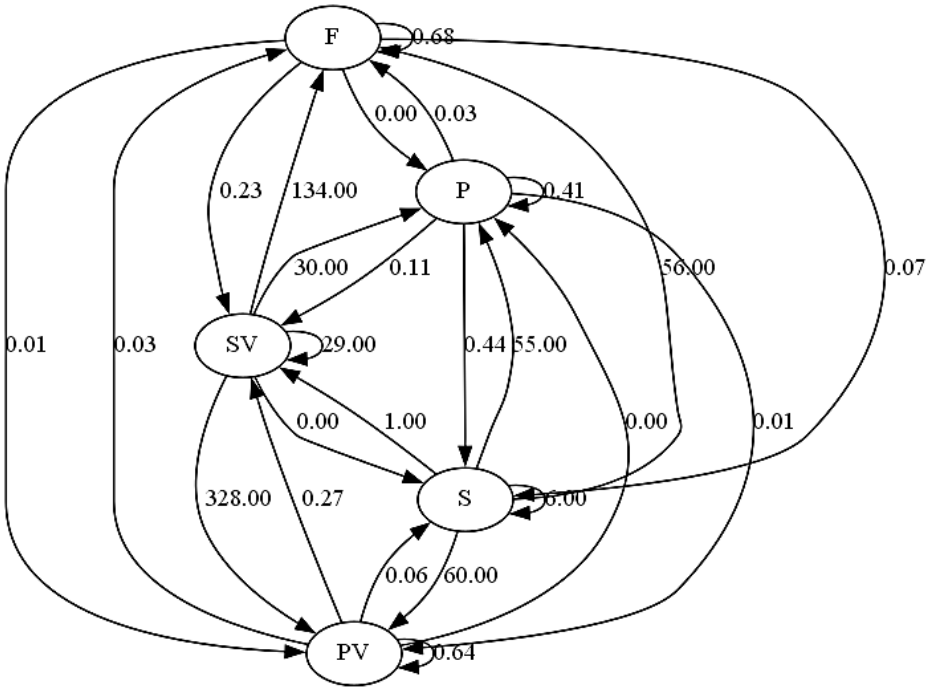
Output distributions for each state and feature set combination were fit to the set of feature values for each segment coded as the given state across all labeled NSLR reconstructions. The data for each state-feature combination contained outliers due to error in the labeling and processing pipelines. To correct for this problem when modeling the output probabilities, observations below the 1st percentile or above the 99th percentile for each state-feature combination were excluded when fitting the output probability distributions.

§ 3.5.3 Supervised Learning

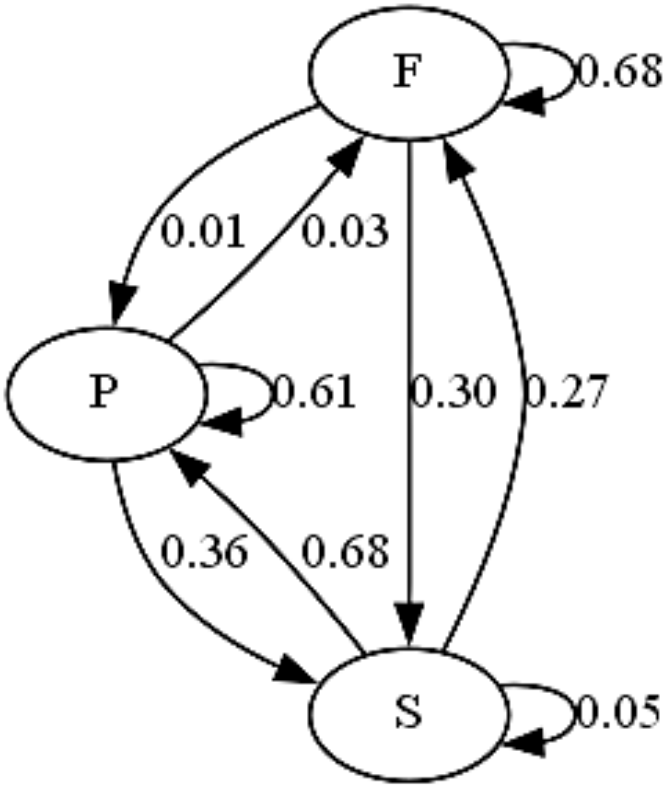
In half of the models, only the labeled data was considered when modeling the eye movement process. The start state probabilities, transition probabilities, and output probabilities were determined as described in 3.5.1 and 3.5.2.

§ 3.5.4 Semi-Supervised Learning

In the other half of the models, unlabeled data was used to enhance the underlying model of the eye movement process. The model was first initialized as in supervised learning. Then, an unsupervised hidden Markov model is fit using the Baum-Welch algorithm on the unlabeled data for the same size state



(a) Full state space.



(b) Simplified state space.

Figure 3.7: Transition matrices for each state space

space, and resulting model statistics are used to adjust the output distributions of the original supervised model (Schreiber, 2018).

CHAPTER 4

RESULTS

§ 4.1 Unconsolidated NSLR Reconstruction Dataset

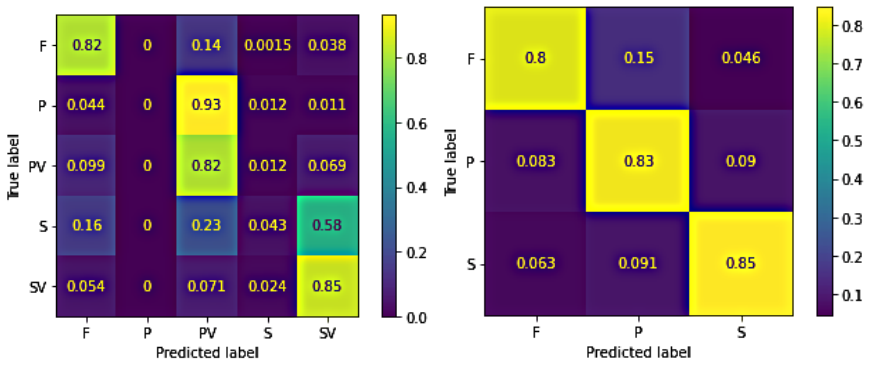
The results of the hidden Markov model experiments on the NSLR reconstruction dataset with no further processing or consolidation of segments are presented in table 4.1 for supervised models in the simplified state space, table 4.2 for supervised models in the full state space, table 4.3 for semi-supervised models in the simplified state space, and table 4.4 for semi-supervised models in the full state space. Table 4.5 summarizes the best-performing models for each state space and learning type.

§ 4.1.1 Supervised Model Results

In both state spaces, the supervised single feature HMM using only the Cartesian speed was the most accurate model, at 82% accuracy for the simplified state space (three states) and 72% accuracy for the full state space (five states). For the top performing model in the simplified state space, precision and recall for each state are within a few percentage points of each other for each other, indicating that the classifier does not perform substantially better at identifying any one gaze event compared to the remaining events (4.1).

Insufficient State Predictions in the Full State Space

Many of the supervised hidden Markov models operating in the full state space did not predict even a single occurrence of either pursuit without vergence or saccade without vergence in any of the test sequences, even though these two events made up 5.24% and 7.20% of the labeled dataset, respectively. Consequently, the precision, recall, and F1 score values for these states are zero for many of the classifiers.



(a) Full state space (b) Simplified state space

Figure 4.1: Confusion matrices for the best performing models in each state space.

The failure to predict these events may be attributed to low transition probabilities from other states to pursuit without vergence or saccade without vergence and to a low total number of samples.

§ 4.1.2 Semi-supervised Model Results

The semi-supervised hidden Markov models failed to achieve more than 50% accuracy across all variations. As in the supervised hidden Markov model experiments, models using a single feature performed better than models using two or more features. The highest performing semi-supervised single feature model achieved 50% accuracy while the highest performing semi-supervised multi-feature model achieved 44% accuracy.

Output Distribution Estimates for Semi-Supervised Models

The semi-supervised learning process using Baum-Welch training resulted in output distributions very different from the output distributions calculated in supervised learning. For example, the mean log segment duration for the fixation state is approximately -1.25 in supervised models but over 500 in semi-supervised models. Instead of enhancing the model by forming a more complete representation of the output distributions, semi-supervised learning appeared to converge to a degenerate solution.

Table 4.1: Supervised Model Classification Metrics for the Simplified State Space
(D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity)

Features		**Accuracy**	**Precision**			**Recall**			**F1 Score**		
			F	P	S	F	P	S	F	P	S
D		.73	.67	.73	.77	.53	.76	.88	.59	.74	.82
S		.82	.81	.83	.81	.80	.82	.84	.80	.83	.82
A		.62	.64	.64	.45	.75	.83	.16	.69	.72	.24
V		.67	.61	.62	.82	.32	.82	.82	.41	.70	.82
D,S	Independent	.42	.39	.50	.37	.42	.46	.38	.40	.48	.37
	Dependent	.41	.30	.44	.37	.09	.69	.32	.14	.54	.34
D,A	Independent	.32	.30	.28	.43	.87	.00	.22	.44	.00	.29
	Dependent	.32	.30	.51	.44	.87	.00	.21	.44	.01	.29
D,V	Independent	.29	.29	.0	.39	.90	.0	.10	.43	.0	.13
	Dependent	.27	.27	.0	.28	.57	.0	.39	.36	.0	.33
S,A	Independent	.35	.39	.45	.30	.52	.11	.54	.44	.18	.39
	Dependent	.34	.37	.48	.30	.53	.11	.52	.43	.17	.38
S,V	Independent	.32	.31	.44	.33	.58	.01	.52	.39	.02	.39
	Dependent	.31	.29	.36	.14	.83	.16	.00	.43	.16	.01
A,V	Independent	.39	.33	.49	.30	.46	.47	.20	.38	.48	.24
	Dependent	.39	.34	.50	.31	.46	.45	.24	.38	.47	.26
D,S,A	Independent	.36	.32	.65	.41	.78	.08	.34	.45	.15	.37
	Dependent	.41	.30	.45	.40	.19	.62	.33	.23	.52	.36
D,S,V	Independent	.33	.30	.0	.45	.84	.0	.29	.44	.0	.34
	Dependent	.36	.29	.45	.32	.54	.47	.00	.38	.46	.01
D,A,V	Independent	.30	.28	.2	.51	.91	.00	.11	.43	.00	.16
	Dependent	.26	.25	.20	.28	.59	.00	.32	.35	.00	.30
S,A,V	Independent	.31	.35	.38	.29	.53	.01	.54	.40	.03	.37
	Dependent	.29	.29	.40	.27	.95	.03	.01	.44	.05	.02
D,S,A,V	Independent	.29	.28	.2	.36	.84	.00	.17	.42	.00	.21
	Dependent	.40	.33	.49	.3	.61	.52	.00	.42	.50	.00

Table 4.2: Supervised Model Classification Metrics for the Full State Space
(D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity)

Features		**Accuracy**	**Precision**					**Recall**				
			F	P	PV	S	SV	F	P	PV	S	SV
D		.62	.65	.0	.62	.0	.60	.57	.0	.73	.0	.88
S		.72	.79	.0	.71	.22	.68	.81	.0	.81	.04	.85
A		.56	.61	.0	.55	.0	.35	.78	.0	.83	.0	.10
V		.56	.57	.0	.52	.33	.65	.37	.0	.77	.02	.78
D,S	Independent	.38	.39	.0	.44	.08	.29	.43	.0	.48	.01	.34
D,S	Dependent	.36	.30	.0	.39	.06	.30	.09	.0	.73	.00	.30
D,A	Independent	.31	.30	.0	.21	.0	.39	.92	.0	.00	.0	.20
D,A	Dependent	.31	.30	.0	.21	.0	.38	.91	.0	.00	.0	.19
D,V	Independent	.28	.29	.0	.0	.0	.25	.89	.0	.0	.0	.13
D,V	Dependent	.25	.27	.0	.0	.0	.23	.53	.0	.0	.0	.48
S,A	Independent	.29	.37	.0	.39	.0	.21	.53	.0	.11	.0	.49
S,A	Dependent	.30	.37	.0	.41	.0	.21	.55	.0	.10	.0	.48
S,V	Independent	.28	.31	.0	.34	.0	.24	.57	.0	.01	.0	.52
S,V	Dependent	.30	.28	.0	.30	.0	.13	.70	.0	.25	.0	.00
A,V	Independent	.32	.34	.03	.40	.00	.22	.44	.05	.39	.00	.19
A,V	Dependent	.32	.33	.03	.42	.0	.22	.44	.04	.40	.0	.22
D,S,A	Independent	.32	.32	.06	.52	.08	.31	.80	.00	.06	.02	.26
D,S,A	Dependent	.35	.30	.0	.39	.03	.30	.20	.0	.62	.01	.27
D,S,V	Independent	.30	.30	.0	.2	.26	.31	.82	.0	.00	.01	.29
D,S,V	Dependent	.34	.29	.0	.40	.0	.38	.49	.0	.53	.0	.00
D,A,V	Independent	.29	.28	.0	.0	.2	.36	.89	.0	.0	.00	.14
D,A,V	Dependent	.25	.26	.0	.08	.0	.24	.57	.0	.00	.0	.41
S,A,V	Independent	.26	.35	.0	.36	.0	.21	.49	.0	.02	.0	.57
S,A,V	Dependent	.30	.29	.0	.39	.26	.22	.94	.0	.04	.01	.01
D,S,A,V	Independent	.28	.29	.0	.1	.0	.23	.84	.0	.00	.0	.17
D,S,A,V	Dependent	.37	.32	.08	.43	.0	.26	.59	.00	.53	.0	.00

Features		**F1 Score**				
		F	P	PV	S	SV
D		.60	.0	.67	.0	.71
S		.80	.0	.76	.07	.75
A		.68	.0	.66	.0	.16
V		.44	.0	.62	.05	.71
D,S	Independent	.40	.0	.46	.03	.31
D,S	Dependent	.13	.0	.51	.01	.30
D,A	Independent	.45	.0	.00	.0	.26
D,A	Dependent	.45	.0	.00	.0	.25
D,V	Independent	.43	.0	.0	.0	.15
D,V	Dependent	.36	.0	.0	.0	.31
S,A	Independent	.43	.0	.17	.0	.29
S,A	Dependent	.44	.0	.17	.0	.29
S,V	Independent	.39	.0	.02	.0	.31
S,V	Dependent	.39	.0	.25	.0	.00
A,V	Independent	.37	.03	.39	.00	.20
A,V	Dependent	.37	.03	.40	.0	.21
D,S,A	Independent	.45	.00	.12	.03	.28
D,S,A	Dependent	.24	.0	.48	.01	.28
D,S,V	Independent	.44	.0	.00	.02	.29
D,S,V	Dependent	.36	.0	.45	.0	.01
D,A,V	Independent	.43	.0	.0	.00	.18
D,A,V	Dependent	.36	.0	.00	.0	.31
S,A,V	Independent	.38	.0	.04	.0	.31
S,A,V	Dependent	.45	.0	.08	.02	.02
D,S,A,V	Independent	.43	.0	.00	.0	.19
D,S,A,V	Dependent	.42	.01	.47	.0	.01

Table 4.3: Semi-Supervised Model Classification Metrics for the Simplified State Space (D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity)

Features		**Accuracy**	**Precision**			**Recall**			**F1 Score**		
			F	P	S	F	P	S	F	P	S
D		.44	.90	.44	.21	.07	.98	.01	.13	.61	.02
S		.50	.40	.67	.85	.90	.43	.19	.55	.51	.31
A		.49	.41	.60	.32	.61	.65	.14	.48	.61	.19
V		.41	.32	.43	.14	.00	.93	.04	.01	.59	.06
D,S	Independent	.29	.31	.31	.24	.21	.20	.52	.23	.17	.30
D,S	Dependent	.25	.26	.16	.16	.78	.03	.02	.40	.05	.04
D,A	Independent	.42	.34	.43	.31	.05	.91	.06	.08	.59	.10
D,A	Dependent	.42	.36	.43	.32	.05	.89	.07	.09	.58	.11
D,V	Independent	.44	.39	.45	.22	.12	.95	.00	.18	.61	.01
D,V	Dependent	.43	.59	.43	.24	.07	.91	.05	.13	.59	.08
S,A	Independent	.43	.41	.45	.28	.21	.82	.06	.28	.58	.09
S,A	Dependent	.44	.40	.45	.29	.27	.80	.06	.32	.58	.10
S,V	Independent	.41	.19	.46	.34	.16	.72	.20	.10	.54	.23
S,V	Dependent	.29	.29	.33	.14	.96	.02	.01	.45	.04	.03
A,V	Independent	.43	.39	.45	.18	.17	.87	.02	.24	.59	.04
A,V	Dependent	.42	.37	.45	.22	.16	.84	.05	.22	.58	.08
D,S,A	Independent	.42	.31	.43	.26	.03	.92	.04	.05	.59	.07
D,S,A	Dependent	.27	.28	.32	.24	.44	.13	.30	.33	.19	.25
D,S,V	Independent	.44	.50	.44	.28	.09	.96	.02	.15	.60	.03
D,S,V	Dependent	.26	.27	.18	.23	.80	.04	.03	.41	.07	.05
D,A,V	Independent	.44	.51	.43	.16	.09	.95	.00	.16	.60	.01
D,A,V	Dependent	.35	.36	.46	.23	.17	.46	.36	.23	.46	.28
S,A,V	Independent	.34	.22	.43	.25	.23	.44	.30	.17	.41	.25
S,A,V	Dependent	.33	.25	.41	.23	.41	.36	.19	.26	.28	.19
D,S,A,V	Independent	.34	.05	.42	.19	.09	.52	.32	.07	.46	.24
D,S,A,V	Dependent	.34	.0	.42	.28	.0	.44	.57	.0	.43	.37

Table 4.4: Semi-Supervised Model Classification Metrics for the Full State Space
(D=Duration, S=Cartesian Speed, A=Angle with Previous Segment, V=Average Angular Velocity)

Features		**Accuracy**	**Precision**					**Recall**				
			F	P	PV	S	SV	F	P	PV	S	SV
D		.36	.85	.01	.42	.0	.15	.09	.01	.87	.0	.04
S		.50	.43	.0	.61	.03	.76	.87	.0	.53	.01	.21
A		.51	.66	.0	.52	.11	.28	.60	.0	.81	.07	.10
V		.38	.2	.0	.38	.0	.19	.00	.0	.98	.0	.05
D,S	Independent	.08	.34	.06	.21	.02	.09	.10	.84	.01	.04	.01
D,S	Dependent	.24	.27	.00	.25	.04	.15	.78	.00	.02	.01	.03
D,A	Independent	.35	.30	.0	.37	.0	.14	.12	.0	.81	.0	.04
D,A	Dependent	.36	.38	.02	.37	.05	.25	.04	.01	.89	.00	.06
D,V	Independent	.38	.38	.0	.41	.02	.14	.11	.0	.89	.01	.03
D,V	Dependent	.10	.60	.06	.24	.03	.34	.07	.81	.10	.04	.01
S,A	Independent	.30	.4	.07	.37	.09	.25	.00	.05	.74	.22	.04
S,A	Dependent	.32	.31	.05	.37	.10	.20	.04	.08	.76	.15	.03
S,V	Independent	.39	.16	.0	.39	.0	.41	.03	.0	.95	.0	.12
S,V	Dependent	.18	.37	.05	.24	.01	.05	.49	.49	.02	.01	.00
A,V	Independent	.18	.38	.07	.33	.04	.15	.15	.67	.27	.01	.01
A,V	Dependent	.18	.37	.06	.31	.06	.10	.14	.56	.28	.02	.01
D,S,A	Independent	.15	.35	.06	.34	.05	.26	.03	.58	.28	.08	.01
D,S,A	Dependent	.28	.28	.0	.34	.05	.19	.37	.0	.34	.01	.19
D,S,V	Independent	.38	.44	.0	.40	.0	.14	.10	.0	.91	.0	.05
D,S,V	Dependent	.24	.26	.0	.16	.00	.19	.73	.0	.04	.00	.03
D,A,V	Independent	.38	.46	.0	.39	.02	.18	.12	.0	.90	.00	.03
D,A,V	Dependent	.35	.50	.05	.38	.05	.19	.09	.03	.80	.05	.05
S,A,V	Independent	.37	.30	.0	.40	.00	.22	.30	.0	.72	.00	.07
S,A,V	Dependent	.12	.40	.06	.37	.04	.13	.18	.81	.07	.06	.00
D,S,A,V	Independent	.08	.0	.05	.23	.0	.0	.0	.94	.07	.0	.0
D,S,A,V	Dependent	.22	.0	.01	.22	.03	.07	.0	.08	.47	.29	.10

Features		F1 Score				
		F	P	PV	S	SV
D		.16	.01	.56	.0	.06
S		.58	.0	.56	.02	.32
A		.62	.0	.64	.09	.15
V		.00	.0	.55	.0	.08
D,S	Independent	.16	.11	.02	.03	.02
D,S	Dependent	.40	.00	.04	.01	.04
D,A	Independent	.16	.0	.51	.0	.07
D,A	Dependent	.07	.01	.52	.00	.10
D,V	Independent	.17	.0	.56	.01	.05
D,V	Dependent	.12	.11	.13	.03	.03
S,A	Independent	.00	.06	.49	.12	.06
S,A	Dependent	.08	.04	.49	.12	.05
S,V	Independent	.05	.0	.56	.0	.17
S,V	Dependent	.40	.09	.04	.01	.01
A,V	Independent	.22	.12	.30	.02	.02
A,V	Dependent	.21	.10	.29	.03	.02
D,S,A	Independent	.05	.10	.30	.06	.03
D,S,A	Dependent	.30	.0	.34	.01	.16
D,S,V	Independent	.17	.0	.55	.0	.07
D,S,V	Dependent	.39	.0	.06	.00	.05
D,A,V	Independent	.20	.0	.55	.00	.04
D,A,V	Dependent	.15	.04	.52	.04	.07
S,A,V	Independent	.25	.0	.51	.00	.10
S,A,V	Dependent	.24	.12	.10	.05	.01
D,S,A,V	Independent	.0	.10	.11	.0	.0
D,S,A,V	Dependent	.0	.02	.30	.06	.07

Table 4.5: Summary of Most Accurate Classifiers by Category

State Space	Learning Type	Features	Accuracy
Simplified	Supervised	S	.82
Full	Supervised	S	0.72
Simplified	Semi-Supervised	S	0.50
Full	Semi-Supervised	A	0.51

CHAPTER 5

DISCUSSION

This study comprises a proof of concept for the use of hidden Markov models to model eye movement sequences and automatically classify eye movements in virtual reality and robotic environments and in applications where smooth pursuit identification is critical. Though the classification accuracy scores achieved in this study were not sufficient to automate the eye movement classification process for research purposes, they provides a new benchmark. The corpus of gaze data generated for this study was designed specifically for use to train eye movement classification models and can be used in further research in this domain. This chapter describes some pitfalls of the presented gaze data processing pipeline and suggests some possible strategies to mitigate them, as well as future directions of research.

§ 5.1 Limitations

§ 5.1.1 Manual Labeling of the Training and Testing Dataset

The progress of this study was hindered by errors and ambiguity in the manual data labeling process. The process of manually classifying eye movements is very labor intensive. In the dataset used for this study, labeling the eye movement onsets and offsets took approximately one to two minutes per second of gaze data. The resulting labeled dataset was riddled with inaccuracies even after several hours of training for the labelers, and three rounds of review were required to correct the mistakes. When gaze data is noisy, the manual labeling process becomes even more difficult and unreliable. A rigorous training and review process for manual labeling with more stringent criteria for each gaze event is needed to generate a larger corpus of labeled gaze data for further model development and testing.

§ 5.1.2 Output Probability Modeling Limitations

The output probability distributions for each feature tested in this study were all modeled as either multivariate or independent Gaussians. However the transformed features are not exactly normally distributed. In particular, values at the extremes of the Gaussian distributions, with low probability values, should really have probability zero since they are often biologically impossible. If the calculated likelihood of state that is physiologically impossible is small but nonzero, it is possible for the low output probabilities to be outweighed by a high transition probability, leading to an incorrect hidden state prediction. A larger dataset and improved raw sequence segmentation methods will allow for more detailed analysis of the true output probability distribution shapes for each feature. Custom probability distributions, perhaps bounded at the limits of physiological possibility for each state-feature pair, can then be developed to improve the prediction of true eye movement sequences.

§ 5.2 Future Directions

§ 5.2.1 Segmentation of Gaze Data Sequences

A weakness of the naive segmented linear regression (NSLR) approach Pekkanen and Lappi, 2017 is that the noise estimate is held constant throughout a sequence, whereas in real gaze data the noise levels often fluctuate throughout a trial. A modification to the NSLR algorithm to allow different noise levels between segments in the same sequence may lead to a regression that better reflects the physiology of eye movement. Downstream, this improvement to the NSLR algorithm may lead to a more realistic transition model of gaze events. Without the inflated self-transition probabilities that were prevalent in the tested hidden Markov models, the model may better represent the likelihood of transitioning from one eye movement state to another.

§ 5.2.2 Validation

To validate its robustness to varying gaze dataset characteristics, the classification method should be validated on other labeled gaze datasets from other KINARM tasks, including tasks with different base rates and transition probabilities between eye movements, tasks involving concurrent hand and eye movement,

and tasks completed by participants of different ages and with visuomotor pathologies such as cerebral palsy and Parkinson's disease.

§ 5.2.3 Extension to Related Domains

Once a model with stable performance is developed to classify eye movements in the KINARM End Point Lab environment, the same technique may be adapted to eye movement classification in other virtual reality and robotic environments. The utility of classifying eye movements recorded in the same workspace as body movement will extend beyond the neuroscience lab as augmented reality and virtual reality solutions are developed for practical training, therapeutic uses, assistive technology, and other applications.

§ 5.3 Conclusion

The novel contribution of this work is combining previously successful automatic eye movement classification methods using piecewise linear segmentation and hidden Markov models to the new domain of robotic and virtual reality environments, achieving substantially higher classification accuracy for fixations and smooth pursuits than existing methods. These results were achieved in part by learning transition probabilities from the dataset, rather than setting them to a pre-determined value. This work also includes innovations in hidden Markov model design for this application, such as changes to the features used, varied output probability modeling methods, and the use of semi-supervised learning to take advantage of unlabeled gaze datasets. We hope this new approach to eye movement segmentation for this type of gaze data will inspire further innovation to achieve higher accuracy and further automate gaze data processing in neuroscience research.

BIBLIOGRAPHY

Bahill, A. T., Clark, M. R., & Stark, L. (1975). Glissades—eye movements generated by mismatched components of the saccadic motoneuronal control signal. *Mathematical Biosciences*, *26*(3-4), 303–318.

Barany, D. A., Gómez-Granados, A., Schrayer, M., Cutts, S. A., & Singh, T. (2020). Perceptual decisions about object shape bias visuomotor coordination during rapid interception movements. *Journal of neurophysiology*, *123*(6), 2235–2248.

Bergstrom, J. R., & Schall, A. (2014). *Eye tracking in user experience design*. Elsevier.

Boghen, D., Troost, B., Daroff, R., Dell'Osso, L., & Birkett, J. (1974). Velocity characteristics of normal human saccades. *Investigative Ophthalmology & Visual Science*, *13*(8), 619–623.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Coutrot, A., Hsiao, J. H., & Chan, A. B. (2018). Scanpath modeling and classification with hidden markov models. *Behavior research methods*, *50*(1), 362–379.

Danion, F. R., & Flanagan, J. R. (2018). Different gaze strategies during eye versus hand tracking of a moving target. *Scientific reports*, *8*(1), 1–9.

De Brouwer, S., Missal, M., Barnes, G., & Lefèvre, P. (2002). Quantitative analysis of catch-up saccades during sustained pursuit. *Journal of neurophysiology*, *87*(4), 1772–1780.

Duchowski, A. T. (2017). *Eye tracking methodology: Theory and practice*. Springer.

Fletcher, W. A., & Sharpe, J. A. (1988). Smooth pursuit dysfunction in alzheimer's disease. *Neurology*, *38*(2), 272–272.

Gómez-Granados, A., Barany, D. A., Schrayer, M., Kurtzer, I. L., Bonnet, C. T., & Singh, T. (2021). Age-related deficits in rapid visuomotor decision-making. *Journal of neurophysiology*, *126*(5), 1592–1603.

Helmchen, C., Pohlmann, J., Trillenberg, P., Lencer, R., Graf, J., & Sprenger, A. (2012). Role of anticipation and prediction in smooth pursuit eye movement control in parkinson's disease. *Movement Disorders*, *27*(8), 1012–1018.

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Van de Weijer, J. (2011). *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford.

Holsanova, J., Rahm, H., & Holmqvist, K. (2006). Entry points and reading paths on newspaper spreads: Comparing a semiotic analysis with eye-tracking measurements. *Visual communication*, *5*(1), 65–93.

Jarodzka, H., & Brand-Gruwel, S. (2017). Tracking the reading eye: Towards a model of real-world reading.

Jones, M. W., Obregón, M., Kelly, M. L., & Branigan, H. P. (2008). Elucidating the component processes involved in dyslexic and non-dyslexic reading fluency: An eye-tracking study. *Cognition*, *109*(3), 389–407.

Komogortsev, O. V., & Karpov, A. (2013). Automated classification and scoring of smooth pursuit eye movements in the presence of fixations and saccades. *Behavior research methods*, *45*(1), 203–215.

Krauzlis, R. J. (2004). Recasting the smooth pursuit eye movement system. *Journal of neurophysiology*, *91*(2), 591–603.

Larsson, L., Nyström, M., & Stridh, M. (2013). Detection of saccades and postsaccadic oscillations in the presence of smooth pursuit. *IEEE Transactions on biomedical engineering*, *60*(9), 2484–2493.

Latash, M. L. (2008). *Neurophysiological basis of movement*. Human Kinetics.

Meyer, C. H., Lasker, A. G., & Robinson, D. A. (1985). The upper limit of human smooth pursuit velocity. *Vision research*, *25*(4), 561–563.

Niehorster, D. C., Siu, W. W., & Li, L. (2015). Manual tracking enhances smooth pursuit eye movements. *Journal of Vision*, *15*(15), 11–11.

O'Driscoll, G. A., & Callahan, B. L. (2008). Smooth pursuit in schizophrenia: A meta-analytic review of research since 1993. *Brain and cognition*, *68*(3), 359–370.

Pekkanen, J., & Lappi, O. (2017). A new and general approach to signal denoising and eye movement classification based on segmented linear regression. *Scientific reports*, *7*(1), 1–13.

Santini, T., Fuhl, W., Kübler, T., & Kasneci, E. (2016). Bayesian identification of fixations, saccades, and smooth pursuits. *Eye Tracking Research and Applications Symposium (ETRA)*, *14*, 163–170. https://doi.org/10.1145/2857491.2857512

Schreiber, J. (2018). Pomegranate: Fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, *18*(164), 1–6.

Scott, S. H. (1999). Apparatus for measuring and perturbing shoulder and elbow joint positions and torques during reaching. *Journal of neuroscience methods*, *89*(2), 119–127.

Sharpe, J. A., & Sylvester, T. O. (1978). Effect of aging on horizontal smooth pursuit. *Investigative Ophthalmology & Visual Science*, *17*(5), 465–468.

Singh, T., Perry, C. M., & Herter, T. M. (2016). A geometric method for computing ocular kinematics and classifying gaze events using monocular remote eye tracking in a robotic environment. *Journal of neuroengineering and rehabilitation*, *13*(1), 1–17.

Toates, F. (1974). Vergence eye movements. *Documenta Ophthalmologica*, *37*(1), 153–214.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*, 5998–6008.

Young, L. R., & Sheena, D. (1975). Eye-movement measurement techniques. *American Psychologist*, *30*(3), 315.

Zhu, Y., Yan, Y., & Komogortsev, O. (2020). Hierarchical hmm for eye movement classification, 544–554.