IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 15, 2021, accepted November 14, 2021, date of publication December 24, 2021, date of current version January 5, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3136153

Evaluating the Performance of Machine Learning Algorithms in Gaze Gesture Recognition Systems

JIAYAO LI, SAMANTHA RAY, VIJAY RAJANNA, AND TRACY HAMMOND
Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA

Corresponding author: Samantha Ray (sjr45@tamu.edu)

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the IRB under Approval No. IRB2015-0529D, Dated August 23, 2017.

ABSTRACT Despite the utility of gaze gestures as an input method, there is a lack of guidelines available regarding how to design gaze gestures, what algorithms to use for gaze gesture recognition, and how these algorithms compare in terms of performance. To facilitate the development of applications that leverage gaze gestures, we have evaluated the performance of a combination of template-based and data-driven algorithms on two custom gesture sets that can map to user actions. Template-based algorithms had consistently high accuracies but the slowest runtimes, making them best for small gesture sets or accuracy-critical applications. Data-driven algorithms run much faster and scale better to larger gesture sets, but require more training data to achieve the accuracy of the template-based methods. The main takeaways for gesture set design are 1) gestures should have distinct forms even when performed imprecisely and 2) gestures should have clear key-points for the eyes to fixate onto.

INDEX TERMS Eye tracking, gaze gestures, gesture design, human-computer interaction, multimodal interaction.

I. INTRODUCTION

Eye gaze serves as a rich form of interaction modality—thanks to the inherent connection between a person's gaze and their attention. This relationship comes from how the brain processes visual stimuli by concentrating on focal points [1]. In other words, people focus their eyes on objects of interest in order to pay attention to them. Consequently, the gaze input can be used as a direct replacement for a mouse [2], especially in applications that rely greatly on mouse input but use limited keyboard input [3]. In fact, there are advantages in using gaze input over a traditional mouse: using eye movement is faster than using a mouse [4], and looking at an object in order to select it is notably intuitive due to the aforementioned relationship between focus and gaze [2], [3], [5].

To use gaze-assisted interaction, sensors and related equipment can be built into, or placed in front of, computer monitors to track a user's eye movements. Applications such as desktop control [6], typing [7], [8], target selection [9], entering passwords [10], [11], game control [12], [13], task prediction [10], visual analytics [14], and giving commands at a distance [15] can be performed hands-free just by tracking the user's gaze on the screen.

With gaze-assisted interaction, the user points at the target UI element with their gaze, and selection can be achieved with different methods such as a blink, dwell-time, touch, voice, or even a secondary input device like an external switch [3], [5], [9], [12]. All these selection methods have limitations related to usability and performance, and most of these issues are due to the need for accurate gaze input. Gaze gestures, on the other hand, can perform well even with inaccurate gaze input [10], but they cannot be used as a selection method. Instead, gaze gestures have the ability to be mapped to perform specific actions such as closing a window, opening a new tab, and scrolling a page. Gaze gesture-based interaction involves recognizing specific eye movement patterns, functioning as a direct analogue to touch gestures. The key advantage in using gaze gestures over the other methods of interaction lies in the fact that gaze gestures achieve high accuracy even if the system calibration is disturbed [10], [16]. Gaze gestures do not scrutinize for calibration error nor require precise pointing as they utilize relative eye positions [16]. Their main weakness, however, is that it can be challenging to distinguish gesture commands from unintentional eye movements.

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

Naturally, each interaction method comes with its respective downsides. Gaze and blink-based interactions demand a higher cognitive load than gaze gesture-based interactions and are less stable and less accurate than cursor movements [17], [18]. Gaze and dwell-based interactions use eye fixation to keep track of where the eyes focus and how long they stay there; as a consequence, dwell-based interactions are more prone to errors than gaze gesture-based interactions [12]. Gaze and touch-based systems have limitations such as camera occlusion with the user's arm, inaccuracy due to hardware limitations and the nature of eye movements, eye-hand coordination difficulties, and inability to perform multi-selection [19]. Gaze and voice-based systems generally perform worse than other interaction methods: using gaze and voice is less effective than using a traditional keyboard [20], and using gaze alone is faster and requires less coordination effort [21]. Considering these limitations with various selection methods, there exists a need to further explore the potential of gaze gesture-based interaction systems. While gaze gestures have not been used to control UI elements, gaze gestures have been used as a way to authenticate users [10], [11] and gaze typing [7], [8]. Despite various advantages, the main issue using gaze gestures for interaction is the difficulty of developing an effective gaze gesture-based system. It requires a close examination of the subtle nuances such as the uniqueness of gestures, their length, whether the gesture path is easy to remember and perform, and then it is compounded with the limitations of rapid eye movements. The effective methods for overcoming the difficulties haven't been well explored.

Hence, to support the development of gaze gesture-based interaction, this work focuses on evaluating the accuracy and time performance of five methods for gesture recognition with respect to different gesture types and training sample sizes. The five methods studied in this paper are the following: template matching (TM) [22], Pearson correlation (PC) [23], decision tree (DT) [24], backpropagation-based neural network (NN) [25], and genetic algorithm-based neural network (GA) [26]. These algorithms were trained on gaze gesture data collected via user studies. To reflect the broad range of potential users of a gaze gesture-based system, we did not set physical constraints on the participants and selected them randomly. The participants aged from 18 to 32 years old. Based on the performance evaluation of the algorithms with respect to accuracy and speed, we aim to determine which algorithms suit accuracy-centric tasks and speed-centric tasks, respectively.

Overall, the results show that the algorithms can be split into high accuracy and high-speed categories. Template matching and Pearson correlation take significantly more time to process than the decision tree algorithm and neural network-based methods. Increasing the training set size exacerbates this problem, slowing both template matching and Pearson correlation down even further. While slow, the template matching and Pearson correlation algorithms were unmatched in accuracy, having high accuracy even with few training samples. The inverse was true for the decision tree algorithm and neural network-based methods: they were consistently fast but their accuracy depended more on the training set size. Another major difference between the template-based methods (template matching and Pearson correlation) and the data-driven methods (decision tree and the neural networks) is that models of the former algorithms will have no variation in performance while the latter's will have a range of performance based on the specifics of each model's architecture and initialization. To that end, the neural network-based methods' models achieved the highest individual accuracies during evaluation. But these high-performing models are countered by models that did not generalize as well, causing the neural network-based methods' average accuracies to be lower than those of template matching and Pearson correlation. Comparing performance between gesture sets, simple gestures required less training data to achieve the same performance than more complicated gestures. These findings are supported with confusion matrices and F-measures to quantify and evaluate the performance. We believe the results from our study will assist in the development of gaze gesture-based interfaces by providing insight into algorithm choice and gesture set design.

II. RELATED WORK

Gaze-assisted interaction occupies a niche space in Human-Computer Interaction (HCI); it enables both accessible and rich interactions with a computer. People with physical impairments and disabilities rely on gaze-assisted interaction to communicate and perform basic operations on a computer [3]. Also, in the scenarios of situationally-induced impairments and disabilities, gaze-assisted interaction is crucial [27], [28]. While dwell-based selection has been majorly used in gaze-assisted interactions, using dwell has various limitations related to accuracy, performance, and usability [7], [29], [30]. Using gaze gestures for gaze-assisted interactions addresses some of the issues associated with dwell-based selection. When using gaze gestures, a user is not required to focus on the target UI element for the duration of dwell time and the chances of accidental activations are minimal. In this section, we will discuss some of the prior works that implemented gaze gestures for interactions and also discuss the applications of gaze-assisted interactions. In general, gaze-assisted interactions enable hands-free interactions, opening up interesting and engaging possibilities for computer interfaces.

A. GAZE GESTURE RECOGNITION

Delamare *et al.* [31] presented a Gaze Gesture Guiding system (G3) designed for semaphoric gaze gestures. The system allows for interacting with distant objects using a head-mounted display and an eye tracker. To execute a command, a user performs a gaze gesture by following the desired label moving along its corresponding 2D path. The $1 Recognizer [22] was used for gesture recognition on simple and complex gestures sets. The highest gesture recognition rates of 75.03% and 76.56% were achieved with simple and

complex gesture sets, respectively. Lee *et al.* [32] presented a gaze tracking system on a large-screen TV that is placed at a distance from the user. The system uses two wide-angle cameras to locate the eyes of the observer and two narrow-angle cameras to capture high-resolution eye images for tracking. The work demonstrated a multimodal interface for a shooting game where gaze tracking is used for positioning the mouse and hand gestures are used for issuing the commands. Li *et al.* [33] implemented a gaze-based real-time gesture control system on commercial tablets. The system implements both gaze estimation and gesture tracking. Gaze gesture recognition is achieved by a sliding window method which incorporates two steps: 1) direction calculation and 2) gesture extraction. The system achieved an overall gesture recognition accuracy of 82.5%. Shell *et al.* [34] presented a generic Fuzzy Transfer Learning system for gaze gesture recognition. The work demonstrated that using a gesture set from able-bodied users makes it possible to recognize gestures from participants belonging to a range of contexts in both ability and location.

B. ACCESSIBLE INTERACTIONS

Gaze-assisted text entry has been used by users with physical impairments and disabilities as their primary method of text entry. Gaze typing, a method of text entry by gaze, uses an on-screen virtual keyboard and either a dwell-based or a dwell-free selection method to enter text on a computer. The majority of gaze typing systems use dwell-time (600–1000 milliseconds) as the selection method, where the user focuses their gaze on the target character for the duration of the dwell-time [7]. Hansen *et al.* [8] conducted a comparative study and found that dwell-time selection on keys was slightly slower, had more errors, and had a higher overproduction rate than click-based selection. MacKenzie *et al.* [35] presented a dwell-free eye typing system with word and letter prediction and found that letter prediction was as good, and in some cases better than, word prediction. Kurauchi *et al.* [36] presented EyeSwipe, a dwell-free text entry system using gaze paths. EyeSwipe can achieve a typing speed of 11.7 wpm after 30 minutes of typing. Kumar *et al.* [37] presented ''TAGSwipe,'' a gaze and touch-based system mapping gaze path into words and can achieve a typing speed of 15.45 wpm.

Gaze input has also been combined with other input modalities for text entry and point-and-click interactions. Rajanna *et al.* [3] presented a gaze and foot-based framework where a user performs selections with a foot-operated wearable device. The authors found that gaze and foot-based interactions can be as good as mouse-based interactions if the interface elements dimensions are above a certain threshold. Kate *et al.* [27] designed an eye-controlled system for nonvocal patients with paralysis. It uses an eye switch controlled by the partially defective eye to select letters. The authors found that visual feedback significantly reduced the number of selection errors. Fejtová *et al.* [28] created a system called ''I4Control'' which allows individuals with special needs to make non-contact control of a personal computer through the eye or head movement.

C. USABLE PRIVACY

Usable privacy is concerned with ensuring user privacy through human-centered methods. An example issue is protecting users from shoulder surfing attacks in a secure but reasonable manner. To illustrate, a shoulder surfing attack involves the attacker observing private information by standing behind a user entering said information into an interface, e.g., a person entering their PIN number. Authentication using eye gestures has been shown as an effective method to counter shoulder surfing attacks. Using gaze gestures in authentication systems has key advantages that bolster security by enabling real-time user authentication without the need of physically entering passwords [10]. Rajanna *et al.* [10] presented a system using gaze gestures to track password patterns. The system offers a secure method for authentication as it effectively prevents shoulder surfing attacks and video analysis attacks. It protects against spoofing as it does not require the user to physically enter a password. A multimodal gaze and touch-based authentication system, ''GazeTouchPass,'' was presented by Khamis *et al.* [11] built specifically for mobile devices.

D. IMPROVED USER FEEDBACK

Eye-tracking has been used in engineering applications for tasks such as controlling an interface or conducting an inspection [38]. Eye-tracking allows users to input gestures at a distance, providing convenience when performing these tasks in the field [39]. This advantage is highlighted in scenarios where remote controlling is required due to the inability to reach or safely interact with the controls. Eye-tracking provides engineers with a flexible and efficient interaction medium to develop sophisticated control systems. A study conducted experiments where participants used eye-trackers to analyze 2D drawing, 3D CAD models, and real objects and found that remote and head-mounted eye-trackers can be efficiently used to observe behaviors of engineers when analyzing technical systems [40]. Eye-tracking provides several key benefits in medical settings as well. For example, previous work suggests that gaze data and tool motion data can be used to effectively evaluate a surgeon's surgical skills [14]. Eye-tracking can also be used as an effective training tool by providing the trainee their mentor's gaze data as visual instruction [41].

E. ENTERTAINMENT

Eye-tracking can be used to develop engaging input methods for video game control. Previous work suggests that using eye-tracking can increase the player's immersion and improve the gaming experience [13]. Gaze data enables interactions to be based on not only the user's point of focus but also on estimates of the user's head orientation [13]. The traditional dwell-time method has a speed disadvantage [36]. Studies have shown that gaze gestures are more

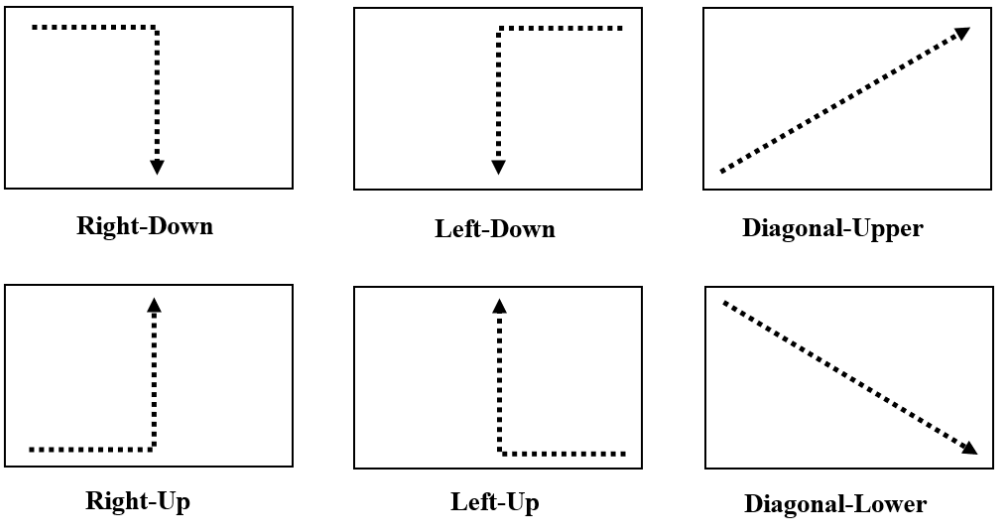


FIGURE 1. Six "regular" gestures used in Experiment 1.

suitable for issuing commands, less prone to errors, and easier to use [12], [35], [36].

From all the related work discussed on gaze-assisted interactions, it can be observed that they use dwell-based activation, specific gaze gestures, or smooth pursuit movement of the gaze to trigger dedicated actions. Applications like "TAGSwipe [37]," gaze gesture-based authentication [10], [11], gaze gesture guiding system [31], and so on did primarily use gaze gestures to express a user's intended action; however, the gestures used and the recognition method remain application-specific. This lack of a generic gesture recognition framework motivated us to explore gaze gesture design strategies, recognition algorithms, and time and performance measures of various recognition algorithms. Furthermore, this study also intends to propose a set of basic guidelines for gaze gesture design.

III. DESIGN BACKGROUND

A. GESTURE DESIGN

As mentioned, gaze gesture-based interaction has several notable advantages over gaze and dwell time-based interaction. One of the most valuable advantages comes from the fact that gaze gesture-based interaction systems can achieve high accuracy even if the calibration is a little disturbed. In other words, these systems do not require precise input nor repeated calibrations in order to be reliable [10]. Additionally, a range of interactions can be created based on a library of gaze gestures. Eyes move differently than hands or a pen, and this fact must be taken into account when designing gestures. Straight lines go hand-in-hand with how eyes move between fixation points. By contrast, curves and arcs are more difficult to produce, especially on smaller screens. Drawing a curve with one's eyes requires many fixations to create a rounded shape, increasing the necessary effort. To that same end, gestures also cannot be too complex, e.g., zigzagging gestures introduce challenges with respect to 1) the number of direction changes and 2) angle accuracy.

To provide insight into how to best design gaze gesture-based interaction systems, this work compares the recognition accuracy and time complexity of several machine learning algorithms on two different sets of eye gestures. We refer to the first set of gestures as "regular gestures" because of the various attributes associated with these gestures. Figure 1 shows the list of regular gestures. The attributes are: 1) these gestures always start from a corner of the screen, 2) for a given gesture, its path covers the entire horizontal or vertical space of the screen, 3) except for diagonal gestures, other gestures in the set do not have diagonal components in their path, and 4) importantly, for every gesture there is always a symmetric gesture in the set. The second set of gestures is called "irregular gestures," and are shown in Figure 2. Irregular gestures are any gestures that are not constrained by the rules of regular gestures. They can start anywhere on the screen, can have diagonal components in their path, and a gesture does not need to have a symmetric gesture. Both sets of gestures follow the design constraint of being distinct from every other gesture in its set, making them effective potential gestures in a real-life system where gestures must be memorable and unique.

The reason for creating two sets of gestures is to study how the constraints enforced on gesture design impact training data requirements and recognition accuracy. Our hypothesis is that stricter constraints on "regular gestures" would result in significantly unique gestures, therefore requiring minimal training data but resulting in high accuracy. Similarly, the relaxed constraints on "irregular gestures" would lead to a few overlapping features among the gestures, therefore it would require more training data to achieve high accuracy. Furthermore, relaxed constraints on "irregular gestures" would yield more gestures than the stricter constraints on "regular gestures."

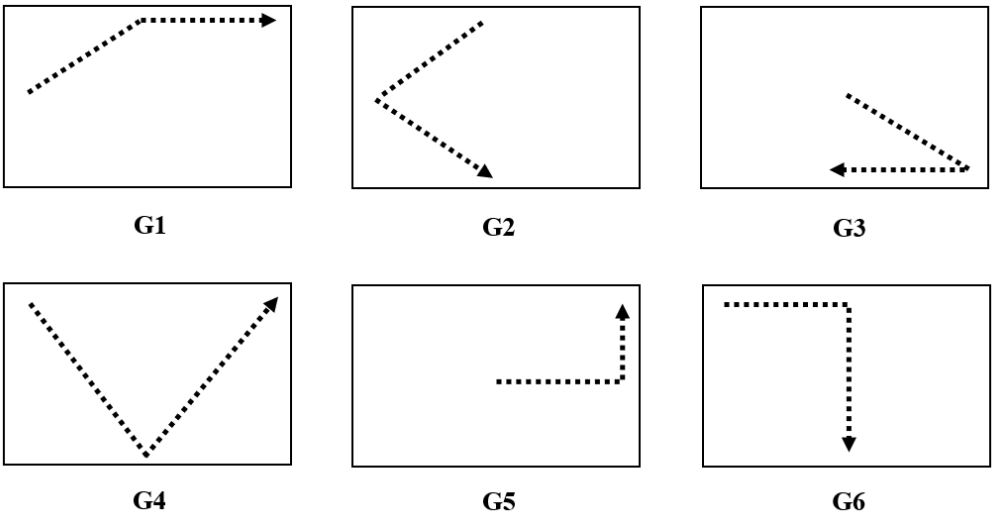


FIGURE 2. Six "irregular" gestures used in Experiment 2.

The selected algorithms consist of template matching, Pearson correlation, decision tree, and neural networks (back-propagation and genetic algorithm). The selected algorithms have been chosen because they are representatives of the most commonly used methods for gesture recognition. Template matching has demonstrated high accuracy in recognizing hand-drawn sketches. This method is able to achieve high accuracy with minimal training data and the performance is minimally impacted in the presence of high noise. Based on the similarities between hand-drawn sketches and eye gestures, this method has been chosen to be leveraged for gaze gesture recognition. With Pearson correlation, we are extending the prior work by not just using the gestures directly, but also updating each gesture through a series of pre-processing steps that help to improve the accuracy. The decision tree algorithm encompasses the simplicity and powerfulness of a classic machine learning algorithm. Our contribution comes from identifying unique features associated with these gaze gestures. The neural network algorithms have been chosen to evaluate the tradeoff between the training data requirement and the model accuracy for gesture classification. Template matching and Pearson correlation represent algorithms that, by their nature, have consistently high classification performance at the cost of nearly linear time complexity with respect to the size of the samples. Decision trees and neural networks represent data-driven methods whose performance with respect to both accuracy and speed depends on feature engineering and architecture configuration.

B. EXPERIMENT SETUP AND DATA COLLECTION

To collect data during the experiment, an eye tracker by Eye Tribe[1] was placed at the bottom of the computer screen, as shown in Figure 3. The tracker has an accuracy of 0.5° to 1° of visual angle, and a frame rate of 60 FPS. At the beginning of the study, the eye tracker was calibrated for each participant to ensure consistency in the data collection. The participants were asked to perform each of the gestures in the experiment's gesture set, repeating the same gesture until at least two quality samples were observed by the researcher administering the user study before moving on to the next gesture. To initiate a gaze gesture, the participant pressed the spacebar key on the keyboard, and gaze data was collected and plotted on the computer screen until the key was released by the participant.

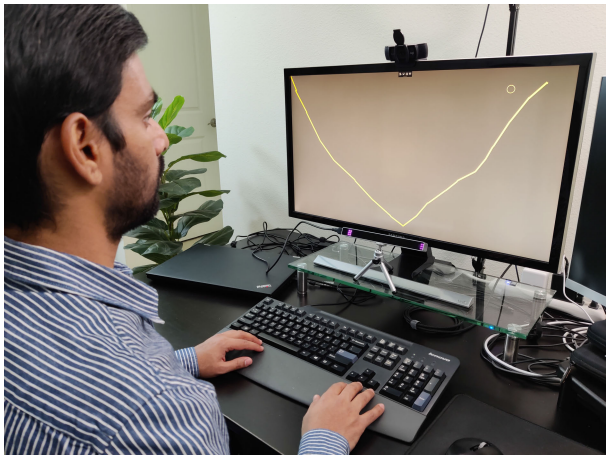


FIGURE 3. User study setup: a user is sitting in front of a computer with an eye tracker placed below the screen. The user has drawn the G4 gesture. The circle on the screen shows the current gaze location.

Eye gesture samples were collected from twenty-seven participants for each experiment. The participants consisted of undergraduate and graduate students ranging from 18 to 32 years in age. Experiment 1 had 8 female and 19 male participants with an average age of 22.11, and Experiment 2 had 10 female and 17 male participants with an average age of 23.26. During each experiment, each participant provided

[1] http://www.theeyetribe.com

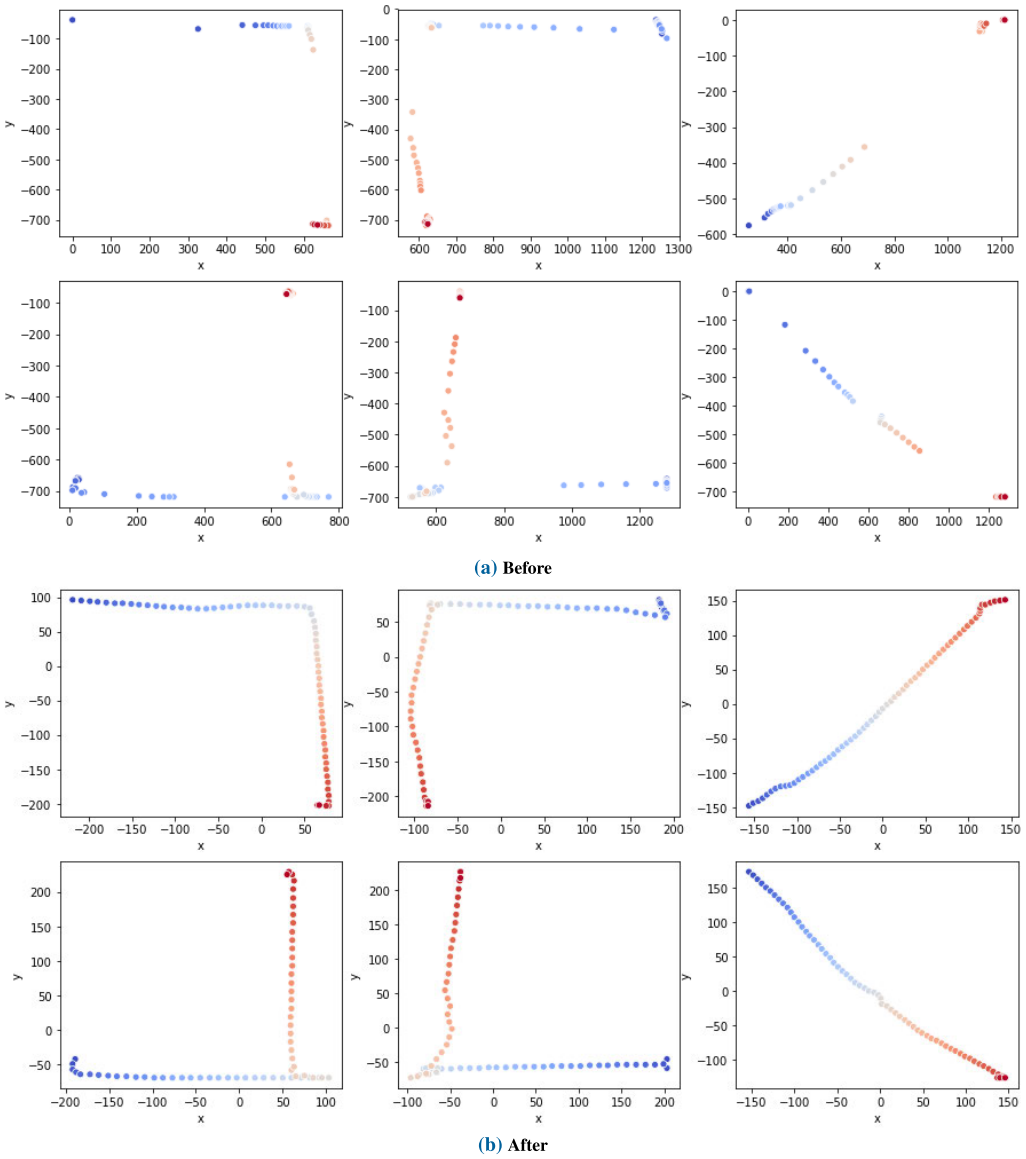


FIGURE 4. Demonstration of pre-processing on the set of regular gestures used in Experiment 1. Points transition from blue (start) to red (end) over time.

between two and four samples per gesture. At the end of the user studies, each experiment split its data into a five-participant training set and a twenty-two-participant testing set to ensure that models were evaluated on data from participants they had never previously seen. The five participants in the training set were handpicked due to the quality of their samples, making them viable templates for the template-based methods. Table 1 shows a summary of the data collection breakdown by experiment.

TABLE 1. Data collection breakdown by experiment.

	Experiment 1	Experiment 2
Total Participants	27	27
# in Training Set	5	5
# in Testing Set	22	22
# of Female	8	10
# of Male	19	17
Average Age	22.11	23.26

IV. ALGORITHMS

A. DATA PRE-PROCESSING

Although the gestures do capture the overall shapes, due to the inherent jittery nature of eye movements, they tend to be noisy, missing points, and sometimes incomplete, as shown in Figure 4a. Hence, each gesture undergoes a pre-processing phase before it is input into a recognition algorithm. We follow standard pre-processing steps in gesture recognition [22] to normalize the input gestures and ensure that the similarity calculation is consistent. The data points are resampled to be equidistant to account for variations in sampling rate, screen size, and screen resolution. This technique involves solving

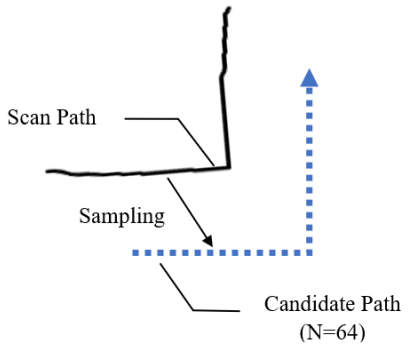


FIGURE 5. Demonstration of the scan path being scaled down to N = 64 points.

for new points along the original path and has the added effect of de-noising the gesture. The resampling process is depicted in Figure 5. Then, the resampled points are scaled to a unit square using the height and width of the bounding box to account for the shape of the screen. Lastly, the processed gesture is translated to be centered around the origin to normalize the gesture location and align it with the template. An example of the results of these processing steps is given in Figure 4b.

B. TEMPLATE-BASED ALGORITHMS

The two template-based algorithms we chose are 1) template matching and 2) Pearson correlation. In the following subsections, we will discuss each algorithm.

1) TEMPLATE MATCHING ALGORITHM

Template matching compares the input gesture against existing templates, predicting which class the input belongs to by finding the best match [22]. For eye gestures, the template matching algorithm computes the similarity based on the Euclidean distance between the input and template gestures [10]; the template with the minimum distance is the best match. This algorithm has been used in previous eye gesture recognition systems due to its high accuracy [10]. The formula for this calculation is presented in Eq. 1, calculating the total distance D between each pair of the N points in the input gesture, A, and the template gesture, B; this algorithm is illustrated in Figure 6.

$$D = \sum_{i=1}^{N} \frac{\sqrt{(A.x_i - B.x_i)^2 + (A.y_i - B.y_i)^2}}{N} \tag{1}$$

2) PEARSON CORRELATION ALGORITHM

The Pearson correlation algorithm measures the linear correlation between the input and template gestures, defined by the covariance between the input gesture, A, and the template gesture, B, divided by their respective standard deviations, σ_A and σ_B. Covariance is defined as the average product of the variances of each pair of points from their respective means, μ_A and μ_B. This algorithm has also been used in previous eye gesture recognition systems [23]. The full calculation of the

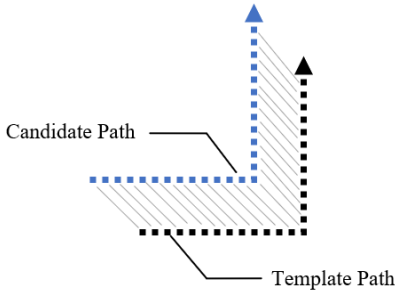


FIGURE 6. Demonstration of the candidate path being matched to the template path.

correlation coefficient C is given in Eq. 2. This calculation is performed with respect to both the x and the y axes individually; the final coefficient used as the similarity metric is the sum of these two coefficients. The template with the highest correlation coefficient is the best match.

$$\begin{aligned} C &= \frac{Cov(A, B)}{\sigma_A \cdot \sigma_B} \\ &= \frac{\sum_{i=1}^{N}(A_i - \mu_A)(B_i - \mu_B)}{\sqrt{\sum_{i=1}^{N}(A_i - \mu_A)^2}\sqrt{\sum_{i=1}^{N}(B_i - \mu_B)^2}} \end{aligned} \tag{2}$$

C. DATA-DRIVEN ALGORITHMS

Unlike template-based methods, data-driven methods, such as decision trees and neural networks, cannot take the N points directly as features. The coordinates and the timestamps of the points must be demultiplexed so that the neural network can interpret them. The model would need to know the binary state of every pixel in the canvas, i.e., the complete visual information, or be told the coordinates and timestamps of each of the N points. This practice can result in a large number of features which can increase the risk of creating a model that overfits. Instead, the data-driven algorithms can use sketch recognition-based features that characterize the gestures [24], reducing the number of features down to a much smaller number and improving the generalizability of the trained model. The data-driven models use the following eight features: 1) the x coordinate of the starting point, 2) the y coordinate of the starting point, 3) the x coordinate of the midpoint, 4) the y coordinate of the midpoint, 5) the x coordinate of the ending point, 6) the y coordinate of the ending point, 7) the total gesture length along the gesture path, and 8) the point-to-point distance between the start and end points. These features are selected to help identify unique gestures. For example, by calculating the ratio of the gesture length and the distance between the endpoints, the gesture can be predicted as a straight line if the value is close to 1.

1) DECISION TREE ALGORITHM

Decision trees classify inputs by traversing the tree based on feature values, e.g., thresholds for numeric features and categorical sorting for quantitative features. The decisions within the tree are determined by fitting to the training set, so fewer features and more training data improve generalizability by

helping prevent overfitting. The default sklearn [42] Decision Tree Classifier implementation is used: the quality of a split is determined by its Gini impurity, the best split is used at each node (in contrast to a random split), and there is no limit on the depth of the tree.

2) BACKPROPAGATION NEURAL NETWORK

Neural networks consist of layers of neurons connected by weights. Features are fed into the input layer, activating neurons based on a function of the inputs and the weights connecting to the next layer of neurons. This process propagates through the layers until the output layer is reached, determining the output of the network for the given input. Backpropagation-based neural networks update their weights based on the error between the output and the expected result, strengthening or weakening weights from the output layer backward. The sklearn [42] MLPClassifier implementation is used with a single layer of twenty nodes, the hyperbolic tangent activation function, and trained until the model converges. The weight initialization is random and the random number generator is seeded, following standard practice. A single layer is used to match the complexity of the data; the model is designed to be as lean as possible to support generalizability as is standard practice in machine learning.

3) GENETIC ALGORITHM NEURAL NETWORK

Instead of using backpropagation, neural networks can also update their weights by using a genetic algorithm. Genetic algorithms generate a population of potential solutions often referred to as chromosomes. Using a fitness function to determine the utility of the chromosome, the best performing chromosomes are selected to generate offspring. After offspring are created, some will mutate to help keep the population diverse. The algorithm converges when the generated offspring are not significantly different from the previous generation, i.e., the point where the population ceases to change.

When using a genetic algorithm to decide neural network weights, the population consists of weight matrices that are spliced together to create new architectures. This is not a common training technique, thus this algorithm was implemented from scratch in Python. Like the backpropagation neural network, the architecture used is a single layer of twenty nodes with a hyperbolic tangent activation function. To keep the population diverse, the weights for each input layer node to the hidden layer nodes will have between one and half of the weights mutated; the crossover point for offspring generation is randomly determined for each pair of chromosomes. The fitness function used is the macro-F1 score divided by the mean squared error. For training, the population size is 80, the top 20 chromosomes are used to generate offspring, and the maximum number of epochs is 200.

V. RESULTS

A. TRAINING AND EVALUATION

The algorithms were evaluated in two stages. First, all models were trained with one sample per gesture per participant in the training set (5 participants $\times$ 1 sample $\times$ 6 gestures $=$ 30 samples) and evaluated against the entirety of the testing set (data from 22 participants). Next, we used two samples per participant in the training (5 participants $\times$ 2 samples $\times$ 6 gestures $=$ 60 samples) and again evaluated the models against the entirety of the testing set. This training setup ensures that each model sees an equal number of samples per class during training. As previously stated, the gestures in the training set were handpicked to serve as a benchmark for correct classification because gaze gestures differ significantly among participants. Additionally, this setup most closely simulates a real-world deployment with respect to the size of the training data compared to the test data. The two sizes of training sets show how the performance of each algorithm changes with more training data. To account for the effects of random initialization on the algorithm performance, each of the algorithms was ran one hundred times with different random states. Naturally, the template-based methods have identical performance in each trial, but the data-driven methods can have fluctuations in performance that will factor into their relative performance.

All models are evaluated using the macro-F1 score to give equal weight to all classes. The F1-score, the harmonic mean of precision and recall (Eq. 4), gives more information about the classifier's performance than simple accuracy (true positives (TP) plus true negatives (TN) over the number of samples). Precision measures how well the classifier avoids false alarms or false positives (FP). Recall measures how well the classifier avoids missing samples of a class, i.e., has minimal false negatives (FN). Both of their formulas are given in Eq. 3. For conciseness of notation, from hereon the backpropagation-based neural network is referred to just as the neural network, and the genetic algorithm-based neural network is referred to as the genetic algorithm. These descriptions were chosen because the backpropagation-based neural network represents a basic neural network, and the genetic algorithm-based neural network is primarily defined by the genetic algorithm used to determine the network weights.

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \end{aligned} \tag{3}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FN + FP} \tag{4}$$

B. EXPERIMENT 1

1) OVERALL RESULTS

First, each algorithm was trained on one sample per participant. With six gestures and five training samples per gesture (one sample from each participant from the testing set), each algorithm was trained with 30 samples (5 participants $\times$ 1 sample $\times$ 6 gestures). For the models trained on 30 samples, the template-based algorithms had the best classification performance with a macro-F1 of 0.99. The neural network-based

TABLE 2. Average F1 score by algorithm for Experiment 1.

Alg.	30 Samples	60 Samples
TM	0.993	0.993
PC	0.993	0.993
DT	0.959	0.956
NN	0.978	0.974
GA	0.977	0.976

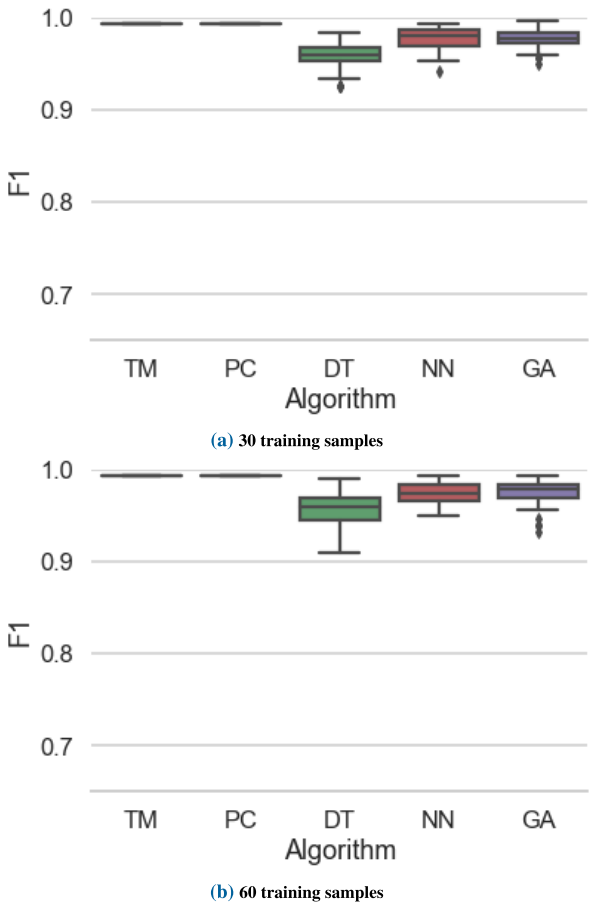
TABLE 3. Results of two-tailed t-test comparing performance results across the two training set sizes in Experiment 1.

Alg.	p-value
DT	0.014
NN	0.004
GA	0.318

methods, i.e., the genetic algorithm and the neural network, performed the best of the data-driven methods with an average macro-F1 of 0.98. While still performing well with an average macro-F1 of 0.96, the decision tree was comparatively the worst. All of the data-driven methods had relatively narrow ranges of performance. The genetic algorithm saw the maximum performance, surpassing the template-based algorithms; the decision tree saw the minimum performance. Figure 7a and Table 2 depict these results.

Next, the number of training samples per gesture was increased to two. Hence, each algorithm was trained with a total of 60 samples (5 participants $\times$ 2 samples $\times$ 6 gestures). With more training data, the template-based methods continued to have the best classification performance of 0.99. However, the data-driven methods performed minusculely worse. This small decrease implies that having more data from the five training participants caused the model to slightly overfit how those participants performed the gestures. Figure 7b and Table 2 depict these results.

While the average performance was nearly the same for each data-driven algorithm across the two training set sizes, the performance of each trial of the decision tree and neural network algorithms was significantly impacted by the increase in data. As shown in Table 3, the change in performance was statistically significant ($p < 0.05$). The genetic algorithm, on the other hand, did not experience a statistically significant change. These findings hold with the nature of each of the algorithms as the genetic algorithm uses a fitness function instead of the error to determine its weights. It is reasonable that the decision tree and neural network selected display significantly different branches and weights, respectively, with the amount of training data doubled. To enable a paired t-test, the random number generator was seeded at the beginning of the model training process to ensure that the same exact architectures are compared across the two training set sizes. As a consequence of this, the genetic algorithm started with the same initial population, therefore it is reasonable that the final architecture of each trial was similar across the two training set sizes.



(a) 30 training samples

(b) 60 training samples

FIGURE 7. F1 score range by algorithm for Experiment 1.

TABLE 4. Mean runtimes of algorithms for Experiment 1 in milliseconds.

Alg.	30 Samples	60 Samples
TM	105.0	208.8
PC	96.7	196.0
DT	0.05	0.05
NN	0.07	0.07
GA	0.02	0.02

As expected, the template-based algorithms were substantially slower than the data-driven algorithms. Template matching was the slowest with a mean runtime of 105 ms; Pearson correlation was slightly faster at 97 ms. All of the data-driven models functionally ran instantaneously, taking only nanoseconds and sometimes outpacing the clock. The genetic algorithm was the fastest followed by the decision tree and the neural network. When the number of training samples increased to 60, the template-based methods slowed down as is inherent in the algorithms' designs, taking about twice as long. All of the data-driven methods took the same amount of time. Table 4 depicts these results.

2) CONFUSION MATRICES

Template matching mislabeled a small number of the Right-Down gestures as Left-Down. The full confusion matrix is

TABLE 5. Confusion matrix for the template matching algorithm.

(a) 30 training samples

ACTUAL \ PREDICTED	RD	LD	DU	RU	LU	DL
RD	0.96	0.04				
LD		1.0				
DU			1.0			
RU				1.0		
LU					1.0	
DL						1.0

(b) 60 training samples

ACTUAL \ PREDICTED	RD	LD	DU	RU	LU	DL
RD	0.96	0.04				
LD		1.0				
DU			1.0			
RU				1.0		
LU					1.0	
DL						1.0

TABLE 6. Confusion matrix for the Pearson correlation algorithm.

(a) 30 training samples

ACTUAL \ PREDICTED	RD	LD	DU	RU	LU	DL
RD	0.96	0.04				
LD		1.0				
DU			1.0			
RU				1.0		
LU					1.0	
DL						1.0

(b) 60 training samples

ACTUAL \ PREDICTED	RD	LD	DU	RU	LU	DL
RD	0.96	0.04				
LD		1.0				
DU			1.0			
RU				1.0		
LU					1.0	
DL						1.0

TABLE 7. Confusion matrix for the decision tree algorithm.

(a) 30 training samples

ACTUAL \ PREDICTED	RD	LD	DU	RU	LU	DL
RD	0.95					0.05
LD		0.95			0.02	0.03
DU	0.02		0.94	0.03		0.01
RU			0.02	0.98		
LU				0.01	0.99	
DL	0.03					0.97

(b) 60 training samples

ACTUAL \ PREDICTED	RD	LD	DU	RU	LU	DL
RD	0.96					0.04
LD		0.96			0.01	0.03
DU	0.02		0.95	0.03		
RU			0.05	0.95		
LU			0.02		0.98	
DL	0.04					0.96

given in Table 5. Like the template matching algorithm, the Pearson correlation algorithm mislabeled a small number of the Right-Down gestures as Left-Down. The full confusion matrix is given in Table 6.

The decision tree had a high performance on most of the gestures. With 30 training samples, the worst-performing gesture was Right-Down, mistaking it mostly as the Diagonal-Lower gesture. With 60 training samples, the Right-Up gesture becomes the worst-performing gesture, being mistaken mostly as the Diagonal-Upper gesture. In general, the decision tree made a small number of mistakes, but no gesture was recognized perfectly. The full confusion matrix is given in Table 7.

Both neural network-based methods consistently recognized the Right-Up and Left-Up gestures perfectly. The neural network's worst-performing gesture was Diagonal-Lower, mistaking it for Right-Down. On the other hand, the genetic algorithm would confuse Left-Down as Left-Up. The full confusion matrix for the neural network is given in Table 8 and the full confusion matrix for the genetic algorithm is given in Table 9.

C. EXPERIMENT 2

1) OVERALL RESULTS

Similar to Experiment 1, we first measured the recognition accuracy by training each algorithm with 30 samples and with 60 samples. For the models using 30 training samples, the template-based methods still had better performance than the data-driven methods: the template-based methods had macro-F1 scores of 0.98 while data-driven methods ranged from 0.85 to 0.96. The decision tree's performance was notably poor with a wide range with an average of 0.85, making it unreliable. The neural network had the smallest range of the data-driven methods, staying consistently above 0.92. The genetic algorithm still had a decent macro-F1 at 0.95, but its performance occasionally dipped below 0.9. Figure 8a and Table 10 depict these results.

When the number of training samples increased to 60, all of the models saw an improvement in performance. The template-based methods saw a slight increase in accuracy, correctly labeling a small number of previously misclassified samples with the additional templates. All of the data-driven methods became more consistent with their performance with their range shrinking considerably. The neural network had the best improvement, almost reaching the template-based methods on average. The genetic algorithm was close behind the neural network in terms of performance with a slightly wider range and slightly lower average macro-F1 score. While still the worst performing overall, the decision tree saw significant improvement with its average performance increasing to 0.91. As seen in Experiment 1, the neural network-based methods had a maximum performance

TABLE 8. Confusion matrix for the backpropagation neural network.

(a) 30 training samples

		PREDICTED					
		RD	LD	DU	RU	LU	DL
A	RD	0.98			0.02		
C	LD	0.01	0.96			0.03	
T	DU			0.98	0.02		
U	RU				1.00		
A	LU					1.00	
L	DL	0.04			0.01		0.95

(b) 60 training samples

		PREDICTED					
		RD	LD	DU	RU	LU	DL
A	RD	0.98			0.02		
C	LD	0.02	0.94			0.04	
T	DU			0.98	0.02		
U	RU				1.00		
A	LU					1.00	
L	DL	0.06					0.94

TABLE 9. Confusion matrix for the genetic algorithm neural network.

(a) 30 training samples

		PREDICTED					
		RD	LD	DU	RU	LU	DL
A	RD	0.97			0.03		
C	LD		0.95		0.01	0.04	
T	DU		0.01	0.99	0.01		
U	RU				1.00		
A	LU					1.00	
L	DL	0.02					0.98

(b) 60 training samples

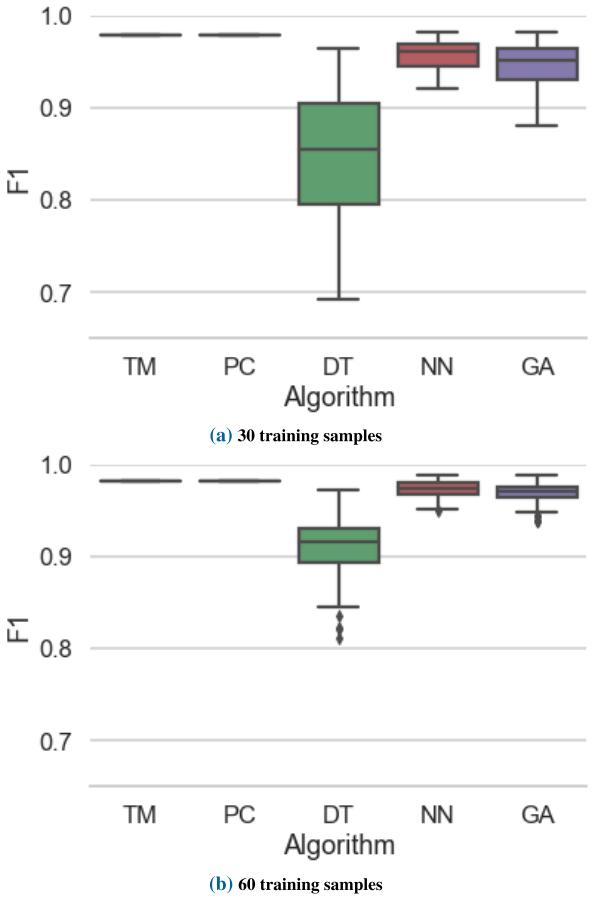
		PREDICTED					
		RD	LD	DU	RU	LU	DL
A	RD	0.97			0.02	0.01	
C	LD	0.01	0.94			0.04	
T	DU			1.00			
U	RU				1.00		
A	LU					1.00	
L	DL	0.03					0.97

TABLE 10. Average F1 score by algorithm for Experiment 2.

Alg.	30 Samples	60 Samples
TM	0.979	0.982
PC	0.979	0.982
DT	0.849	0.910
NN	0.957	0.974
GA	0.946	0.969

above the template-based methods, but the template-based methods still had the better average performance. Figure 8b and Table 10 depict these results.

As shown in Table 11, increasing the amount of training data had a statistically significant performance change for each algorithm with $p < 0.05$. These p-values in combination with the average performance increase indicate that the Experiment 2 gesture sets required more data in order to reach each model's full potential with respect to classification performance.



(a) 30 training samples

(b) 60 training samples

FIGURE 8. F1 score range by algorithm for Experiment 2.

TABLE 11. Results of two-tailed t-test comparing performance results across the two training set sizes in Experiment 2.

Alg.	p-value
DT	2.51E-15
NN	1.62E-14
GA	2.39E-14

The timing performance is consistent with that seen in Experiment 1: the template-based methods are slow relative to the data-driven methods. Template matching was the slowest with a mean runtime of 105 ms; Pearson correlation was faster at 97 ms. All of the data-driven models ran in less than a millisecond; the genetic algorithm was the fastest followed by the decision tree and then the neural network. With 60 training samples, the template-based algorithms were about 2 times slower. The data-driven methods took the same amount of time. Table 12 depicts these results.

2) CONFUSION MATRICES

Template matching misclassified only a small number of samples. The most confused gesture was G4, which was misclassified as both G1 and G5. Notably, increasing to 60 training samples did not improve the performance on G4.

TABLE 12. Mean runtimes of algorithms for Experiment 2 in milliseconds.

Alg.	30 Samples	60 Samples
TM	104.5	206.8
PC	96.8	191.3
DT	0.05	0.04
NN	0.07	0.07
GA	0.02	0.02

TABLE 13. Confusion matrix for the template matching algorithm.

(a) 30 training samples

		PREDICTED					
		G1	G2	G3	G4	G5	G6
A	G1	1.0					
C	G2		0.98				0.02
T	G3			1.0			
U	G4	0.04			0.91	0.05	
A	G5	0.02				0.98	
L	G6						1.0

(b) 60 training samples

		PREDICTED					
		G1	G2	G3	G4	G5	G6
A	G1	1.0					
C	G2		1.0				
T	G3			1.0			
U	G4	0.04			0.91	0.05	
A	G5	0.02				0.98	
L	G6						1.0

Otherwise, a sample of G2 was mistaken for G6 and a sample of G5 was mistaken for G1. The full confusion matrix is given in Table 13.

In contrast to Experiment 1, Pearson correlation did not have the exact same performance as template matching. Pearson correlation also confused G4 as both G1 and G5, but it confused more G4 samples as G5 than template matching. The only other mistake was one sample of G5 being confused for G1. Increasing to 60 training samples results in the same confusion matrix as template matching with G4 being the most confused gesture with an individual F1 score of 0.91. The full confusion matrix is given in Table 14.

The decision tree had a relatively terrible performance with functionally every class being mistaken for every other class. The worst offenders for 30 training samples are G1 and G5 with individual F1 scores of 0.82 and 0.77, respectively. G4 and G6 also had a notably low performance with individual F1 scores around 0.85. G1 was confused for both G4 and G5 most frequently while G5 was confused for G4 and G6. Increasing to 60 samples significantly improves performance, but there are still plenty of errors and individual F1 scores as low as 0.88. Like the template-based methods, G4 was the worst-performing gesture. The full confusion matrix is given in Table 15.

Although not as bad as the decision tree's performance, the neural network-based methods also had trouble recognizing G1, confusing it with G5, and G4, confusing it with G1. Increasing to 60 training samples greatly improved

TABLE 14. Confusion matrix for the Pearson correlation algorithm.

(a) 30 training samples

		PREDICTED					
		G1	G2	G3	G4	G5	G6
A	G1	1.0					
C	G2		1.0				
T	G3			1.0			
U	G4	0.04			0.89	0.7	
A	G5	0.02				0.98	
L	G6						1.0

(b) 60 training samples

		PREDICTED					
		G1	G2	G3	G4	G5	G6
A	G1	1.0					
C	G2		1.0				
T	G3			1.0			
U	G4	0.04			0.91	0.05	
A	G5	0.02				0.98	
L	G6						1.0

TABLE 15. Confusion matrix for the decision tree algorithm.

(a) 30 training samples

		PREDICTED					
		G1	G2	G3	G4	G5	G6
A	G1	0.82	0.01	0.02	0.06	0.03	0.06
C	G2	0.02	0.90	0.02	0.02	0.01	0.03
T	G3	0.01	0.02	0.91	0.03	0.01	0.02
U	G4	0.05	0.01	0.04	0.86	0.03	0.01
A	G5	0.02	0.01	0.01	0.11	0.77	0.08
L	G6	0.03	0.03	0.02	0.03	0.04	0.85

(b) 60 training samples

		PREDICTED					
		G1	G2	G3	G4	G5	G6
A	G1	0.91	0.02		0.03	0.02	0.02
C	G2	0.02	0.92	0.02	0.01	0.01	0.02
T	G3	0.01	0.03	0.90	0.02	0.01	0.03
U	G4	0.09	0.01	0.01	0.88	0.04	0.01
A	G5	0.03	0.01	0.01	0.04	0.90	0.01
L	G6		0.02	0.01	0.01	0.01	0.95

performance for every gesture except G4. While still its worst gesture, the neural network had the highest performance on G4 across all the classifiers with an F1 score of 0.92 for 30 samples and 0.93 for 60 samples. The full confusion matrix for the neural network is given in Table 16 and the full confusion matrix for the genetic algorithm is given in Table 17.

VI. DISCUSSION

A. EXPERIMENT 1

The template-based algorithms had near-perfect classification performance. Investigating the gestures they misclassified showed that the samples had a strong Left-Down diagonal (depicted in Fig. 9) which no gesture in the set has. However, it is more likely for the Left-Down gesture to tend toward that path, thus the misclassification is to be expected.

TABLE 16. Confusion matrix for the backpropagation neural network.

(a) 30 training samples

ACTUAL		PREDICTED G1	G2	G3	G4	G5	G6
A	G1	0.93				0.07	
C	G2	0.01	0.98				0.02
T	G3			1.00			
U	G4	0.06			0.92	0.02	
A	G5			0.02	0.03	0.95	
L	G6				0.02		0.98

(b) 60 training samples

ACTUAL		PREDICTED G1	G2	G3	G4	G5	G6
A	G1	0.99				0.01	
C	G2		0.99				0.01
T	G3			1.00			
U	G4	0.05			0.93	0.02	
A	G5			0.01	0.02	0.97	
L	G6				0.02		0.98

TABLE 17. Confusion matrix for the genetic algorithm neural network.

(a) 30 training samples

ACTUAL		PREDICTED G1	G2	G3	G4	G5	G6
A	G1	0.89	0.02			0.07	0.02
C	G2	0.01	0.98				0.02
T	G3			1.00			
U	G4	0.05		0.02	0.91	0.02	
A	G5			0.04	0.03	0.93	
L	G6				0.02		0.98

(b) 60 training samples

ACTUAL		PREDICTED G1	G2	G3	G4	G5	G6
A	G1	1.00					
C	G2	0.01	0.98				0.02
T	G3			1.00			
U	G4	0.06			0.91	0.03	
A	G5			0.02	0.02	0.96	
L	G6				0.02		0.98

Interestingly, the neural network generally confused gestures that follow similar paths. Diagonal-Upper and Right-Up were significantly confused in the experiment with 30 training samples. To a lesser degree, Right-Down was confused for Diagonal-Lower. If the participants gave imprecise input for the Right-Down and Right-Up gestures, e.g., if their eyes did not go straight down or up, respectively, or if the horizontal segment was at an angle, this would cause overlap in the forms of the eye gestures. With the number of training samples increased to 60, this pattern goes away as the model has more data to learn how to distinguish the similar features of these gestures.

B. EXPERIMENT 2

With 30 samples, data-driven methods had difficulty distinguishing G1 and G4 from G5. While these gestures are distinct in design and are not symmetrical, analyzing their

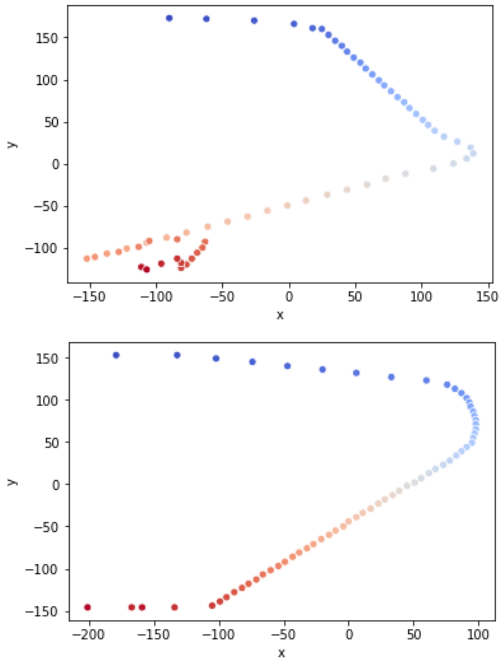


FIGURE 9. Right-Down gestures misclassified as Left-Down. Points transition from blue to red over time.

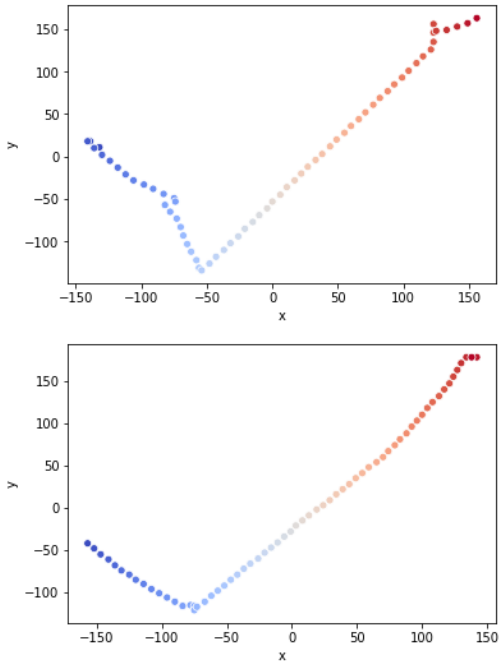


FIGURE 10. Incorrectly performed G4 gestures misclassified as G5. Points transition from blue to red over time.

appearance when performed in imprecise or deformed fashion shows why this confusion pattern occurred. As shown in Figures 10-12, the G4 gesture can look like a checkmark, i.e., it contains an acute angle and has a significant distance between the endpoints. G1, imprecise G4, and G5 generally start in the left-center region of the screen and end in the top-right corner. G1 has an obtuse angle, G4 has an acute angle,

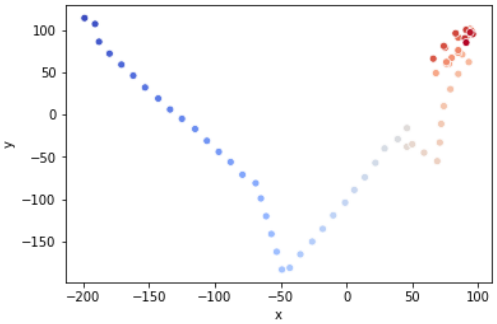


FIGURE 11. G4 gesture misclassified as G5 due to long fixation. Points transition from blue to red over time.

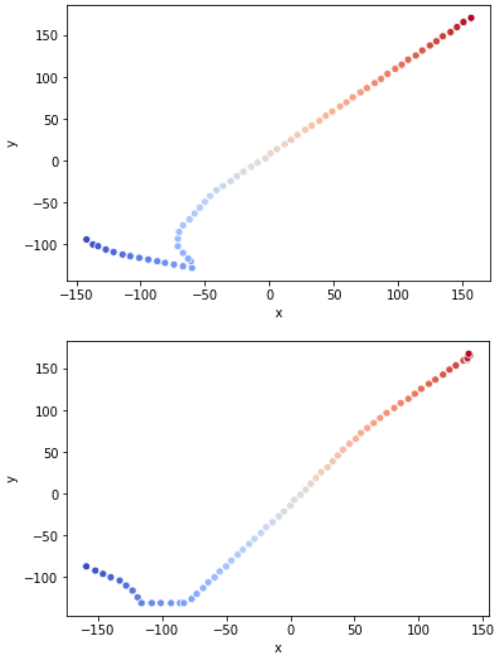


FIGURE 12. Incorrectly performed G4 gestures misclassified as G1. Points transition from blue to red over time.

and G5 has a right angle. Therefore, any samples of G5 that don't contain a sharp right angle will have an obtuse or acute angle by definition. As such, in practice, G5 can be a mirror of G1 and G4. With 60 samples, G4 was the worst-performing gesture. As it was the only gesture that the template-based methods struggled with, it implies that G4 was the hardest gesture to perform. Participants seemed to over-anticipate the Left-Down diagonal portion, not looking to the top-left corner. It is also the largest gesture, traveling over most of the screen, so participants could have been cheating the corner, so to speak, by starting the gesture too low in order to perform it faster.

C. GENERAL

Overall, our experiments show that when using the regular set of gestures, recognition accuracy is high irrespective of the algorithm used, even at a limited number of samples. Adding more training samples may not improve the performance significantly. On the other hand, with the irregular set of gestures, the accuracy improves with more training data. This is evident specifically for data-driven algorithms. In our experiments, the F1 scores for DT, NN, and GA improved with increased training data. Also, statistical tests indicate that the difference in F1 scores with 30 and 60 training samples is significant ($p < 0.05$).

The decision tree's performance is the most impacted by randomness and its training data. In almost every experiment it had the least consistent performance. Training the decision tree has the fewest constraints, so it is easy for the model to select suboptimal architectures without a lot of training data. With the context that other models outperform it with less training data, the decision tree is the least recommended classifier.

As expected, template-based methods are accurate but slow. In all but one experiment, they had the highest accuracy. They also require the least training data so they suit systems that need to avoid a cold start problem. All things considered, the neural networks performed the best. With only 30 training samples, i.e., 5 samples per gesture, their accuracy had room for improvement; increasing to 10 samples per gesture alleviated this problem. Experiment 1's better performance combined with the number of people unable to perform G4 adequately in Experiment 2 motivates that gestures need to be easy to perform, i.e., the gesture design should always be regular.

Lastly, comparing our gaze gesture recognition framework to prior works like [31]–[34], we observe that while the other recognition systems are being closely linked to an application or use case, our system generalizes gaze gesture-based interaction. It provides a platform for developers to design and evaluate the performance of various gaze gestures before incorporating them into an application. Also, developers can rank the gestures based on a single metric that combines two key factors—ease of execution and recognition accuracy.

VII. LIMITATIONS AND FUTURE WORK

We will discuss some of the primary limitations of a gaze gesture-based interaction system. First, while gaze gesture-based systems do not require precise gaze tracking or for the gestures to be a perfect match to the templates, the system still needs the gestures to retain the overall shape of the template gesture in order to work reliably. In other words, the gestures still need to be relatively accurate. Second, performing gaze gestures blindly can cause users to perform the gestures inaccurately, making recognition unreliable. To avoid this problem, the system should implement some form of feedback so that the user is aware of the path of the gesture during execution. This can be achieved by drawing a temporary overlay on the screen. Additionally, this helps the user to adjust the orientation of their head so that the gesture can be drawn as accurately as possible. Third, the screen size also influences how accurately the gestures can be recognized. As gaze moves quickly, smaller screens makes it more difficult to execute gestures that involve multiple small segments.

The screen size should be approximately 11 inches in order to maintain good accuracy [10]. Lastly, each algorithm discussed in this work has limitations that are unsuitable for certain types of systems: template-based algorithms are not suitable for time-critical systems while data-driven methods are not suitable for applications that require high accuracy.

Future work in this space involves developing applications that use both regular and irregular gesture sets and performing user studies to evaluate each application's performance, user experience, and reliability. These user studies need to collect qualitative feedback on the usability of the systems in order to establish the design implications of the gesture sets. Additionally, feedback mechanisms to aid the users in drawing gestures as accurately as possible need to be designed and evaluated. An example feedback mechanism is the aforementioned temporary transparent overlay. Lastly, an experimental variable not explored in this work is algorithm performance across varying screen sizes and environments. Generally speaking, the larger the screen size the better the performance, but it is of interest to determine how small the screen can be while still achieving acceptable performance. Similarly, it is also of interest to determine the impact of lighting conditions, head angle, and distance from the screen on gesture recognition performance.

VIII. CONCLUSION

Template-based methods are accurate and reliable, but the cost for this performance is a linear time performance with respect to the number of recognized templates. Data-driven methods, on the other hand, are consistently fast but rely on having sufficient training data to reach the template-based methods' classification performance. If building an eye gesture recognition system with a cold start, template-based methods will perform excellently with respect to accuracy and satisfactorily with respect to speed. If building a system with a large set of eye gestures, collecting data for a data-driven approach will yield better results. Based on the misclassifications from each experiment, gesture design has a key role in determining the performance of an eye gesture recognition system. Gestures that follow similar paths or have relatively-speaking complex angles caused the most confusion during classification. The more mental effort the gesture takes to perform, the more likely the user is to perform it imprecisely. Imprecise gestures generally are ambiguous to the classifier, hurting accuracy. As such, it is recommended that gestures have distinct forms even when performed imprecisely. Within that constraint, it is recommended to make gestures as simple as possible. If complex angles and/or endpoints are needed, designing unique gestures with key-points for the user to fixate onto will help them perform the gesture correctly.

REFERENCES

[1] R. J. Jacob and K. S. Karn, "Commentary on section 4—Eye tracking in human-computer interaction and usability research: Ready to deliver the promises," in *The Mind's Eye*, J. Hyönä, R. Radach, and H. Deubel, Eds. Amsterdam, The Netherlands: North-Holland, 2003, pp. 573–605.

[2] D. Fono and R. Vertegaal, "EyeWindows: Evaluation of eye-controlled zooming windows for focus selection," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 2005, pp. 151–160.

[3] V. Rajanna and T. Hammond, "GAWSCHI: Gaze-augmented, wearable-supplemented computer-human interaction," in *Proc. 9th Biennial ACM Symp. Eye Tracking Res. Appl. (ETRA)*, New York, NY, USA, 2016, pp. 233–236, doi: 10.1145/2857491.2857499.

[4] L. E. Sibert and R. J. K. Jacob, "Evaluation of eye gaze interaction," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 2000, pp. 281–288.

[5] R. J. Jacob, "What you look at is what you get: Eye movement-based interaction techniques," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 1990, pp. 11–18.

[6] E. Castellina, F. Corno, and P. Pellegrino, "Integrated speech and gaze control for realistic desktop environments," in *Proc. Symp. Eye Tracking Res. Appl. (ETRA)*, New York, NY, USA, 2008, pp. 79–82.

[7] P. Majaranta and K.-J. Räihä, "Twenty years of eye typing: Systems and design issues," in *Proc. Symp. Eye Tracking Res. Appl. (ETRA)*, New York, NY, USA, 2002, pp. 15–22, doi: 10.1145/507072.507076.

[8] J. P. Hansen, A. S. Johansen, D. W. Hansen, K. Itoh, and S. Mashino, "Command without a click: Dwell time typing by mouse and gaze selections," in *Proc. Int. Conf. Hum.-Comput. Interact. (INTERACT)*, Zürich, Switzerland, 2003, pp. 121–128.

[9] S. Stellmach and R. Dachselt, "Look & touch: Gaze-supported target acquisition," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 2012, pp. 2981–2990.

[10] V. Rajanna, A. H. Malla, R. A. Bhagat, and T. Hammond, "DyGazePass: A gaze gesture-based dynamic authentication system to counter shoulder surfing and video analysis attacks," in *Proc. IEEE 4th Int. Conf. Identity, Secur., Behav. Anal. (ISBA)*, Washington, DC, USA, Jan. 2018, pp. 1–8.

[11] M. Khamis, F. Alt, M. Hassib, E. von Zezschwitz, R. Hasholzner, and A. Bulling, "GazeTouchPass: Multimodal authentication using gaze and touch on mobile devices," in *Proc. CHI Conf. Extended Abstr. Hum. Factors Comput. Syst.*, New York, NY, USA, 2016, pp. 2156–2164.

[12] A. Hyrskykari, H. Istance, and S. Vickers, "Gaze gestures or dwell-based interaction?" in *Proc. Symp. Eye Tracking Res. Appl. (ETRA)*, New York, NY, USA, 2012, pp. 229–232.

[13] J. D. Smith and T. C. N. Graham, "Use of eye movements for video game control," in *Proc. ACM SIGCHI Int. Conf. Adv. Comput. Entertainment Technol. (ACE)*, New York, NY, USA, 2006, p. 20.

[14] N. Ahmidi, M. Ishii, G. Fichtinger, G. L. Gallia, and G. D. Hager, "An objective and automated method for assessing surgical skill in endoscopic sinus surgery using eye-tracking and tool-motion data," *Int. Forum Allergy Rhinol.*, vol. 2, no. 6, pp. 507–515, Nov. 2012.

[15] C. Hennessey and J. Fiset, "Long range eye tracking: Bringing eye tracking into the living room," in *Proc. Symp. Eye Tracking Res. Appl. (ETRA)*, New York, NY, USA, 2012, pp. 249–252.

[16] H. Drewes and A. Schmidt, "Interacting with the computer using gaze gestures," in *Proc. IFIP Conf. Hum.-Comput. Interact.* New York, NY, USA: Springer, 2007, pp. 475–488.

[17] P. Biswas and P. Langdon, "A new interaction technique involving eye gaze tracker and scanning system," in *Proc. Conf. Eye Tracking South Afr.*, New York, NY, USA, 2013, pp. 67–70.

[18] M. Adjouadi, A. Sesin, M. Ayala, and M. Cabrerizo, "Remote eye gaze tracking system as a computer interface for persons with severe motor disability," in *Proc. Int. Conf. Comput. Handicapped Persons*. New York, NY, USA: Springer, 2004, pp. 761–769.

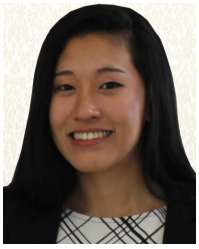
[19] K. Pfeuffer, J. Alexander, M. K. Chong, and H. Gellersen, "Gaze-touch: Combining gaze with multi-touch for interaction on the same surface," in *Proc. 27th Annu. ACM Symp. User Interface Softw. Technol.*, New York, NY, USA, 2014, pp. 509–518.

[20] T. R. Beelders and P. J. Blignaut, *The Usability of Speech and Eye Gaze as a Multimodal Interface for a Word Processor*. London, U.K.: INTECH Open Access Publisher, 2011.

[21] S. E. Brennan, X. Chen, C. A. Dickinson, M. B. Neider, and G. J. Zelinsky, "Coordinating cognition: The costs and benefits of shared gaze during collaborative search," *Cognition*, vol. 106, no. 3, pp. 1465–1477, Mar. 2008.

[22] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes," in *Proc. 20th Annu. ACM Symp. User Interface Softw. Technol. (UIST)*, New York, NY, USA, 2007, pp. 159–168.

[23] M. Vidal, A. Bulling, and H. Gellersen, "Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, New York, NY, USA, 2013, pp. 439–448.
[24] D. Rubine, "Specifying gestures by example," *SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 329–337, Jul. 1991.
[25] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.
[26] K. Sastry, D. Goldberg, and G. Kendall, "Genetic algorithms," in *Search Methodologies*, E. K. Burke and G. Kendall, Eds. Boston, MA, USA: Springer, 2005, ch. 4, pp. 97–125.
[27] J. ten Kate, E. Frietman, F. Stoel, and W. Willems, "Eye-controlled communication aids," *Med. Prog. Through Technol.*, vol. 8, no. 1, pp. 1–21, 1980.
[28] M. Fejtová, J. Fejt, P. Novák, and O. Stepankova, "System I4Control: Contactless control PC," in *Proc. Int. Conf. Intell. Eng. Syst.*, New York, NY, USA, 2006, pp. 297–302.
[29] R. J. K. Jacob, "The use of eye movements in human-computer interaction techniques: What you look at is what you get," *ACM Trans. Inf. Syst.*, vol. 9, pp. 152–169, Apr. 1991.
[30] P. Isokoski, "Text input methods for eye trackers using off-screen targets," in *Proc. Symp. Eye Tracking Res. Appl. (ETRA)*, 2000, pp. 15–21.
[31] W. Delamare, T. Han, and P. Irani, "Designing a gaze gesture guiding system," in *Proc. 19th Int. Conf. Hum.-Comput. Interact. With Mobile Devices Services*, Sep. 2017, pp. 1–13.
[32] H. Lee, S. Y. Lim, I. Lee, J. Cha, D.-C. Cho, and S. Cho, "Multi-modal user interaction method based on gaze tracking and gesture recognition," *Signal Process., Image Commun.*, vol. 28, no. 2, pp. 114–126, Feb. 2013.
[33] Y. Li, Z. Cao, and J. Wang, "Gazture: Design and implementation of a gaze based gesture control system on tablets," in *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 1–17, Sep. 2017, doi: 10.1145/3130939.
[34] J. Shell, S. Vickers, S. Coupland, and H. Istance, "Towards dynamic accessibility through soft gaze gesture recognition," in *Proc. 12th U.K. Workshop Comput. Intell. (UKCI)*, Sep. 2012, pp. 1–8.
[35] I. S. MacKenzie and X. Zhang, "Eye typing using word and letter prediction and a fixation algorithm," in *Proc. Symp. Eye Tracking Res. Appl. (ETRA)*, New York, NY, USA, 2008, pp. 55–58.
[36] A. Kurauchi, W. Feng, A. Joshi, C. Morimoto, and M. Betke, "EyeSwipe: Dwell-free text entry using gaze paths," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 2016, pp. 1952–1956, doi: 10.1145/2858036.2858335.
[37] C. Kumar, R. Hedeshy, S. MacKenzie, and S. Staab, "TAGSwipe: Touch assisted gaze swipe for text entry," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, New York, NY, USA, Jan. 2020, p. 1–12.
[38] A. T. Duchowski, "A breadth-first survey of eye-tracking applications," *Behav. Res. Methods, Instrum., Comput.*, vol. 34, no. 4, pp. 455–470, Nov. 2002.
[39] L. Stark, G. Vossius, and L. R. Young, "Predictive control of eye tracking movements," *IRE Trans. Hum. Factors Electron.*, vol. 3, pp. 52–57, Oct. 1962.
[40] S. Matthiesen, M. Meboldt, A. Ruckpaul, and M. Mussgnug, "Eye tracking, a method for engineering design research on engineers' behavior while analyzing technical systems," in *Proc. 19th Int. Conf. Eng. Design (ICED), Design Harmonies, Hum. Behav. Design (DS)*, Seoul, South Korea, vol. 7. Glasgow, Scotland: The Design Society, Aug. 2013, p. 10.
[41] A. S. A. Chetwood, K.-W. Kwok, L.-W. Sun, G. P. Mylonas, J. Clark, A. Darzi, and G.-Z. Yang, "Collaborative eye tracking: A potential training tool in laparoscopic surgery," *Surgical Endoscopy*, vol. 26, no. 7, pp. 2003–2009, Jul. 2012.
[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.



JIAYAO LI received the B.S. degree double major in computer science and mechanical engineering from Texas A&M University, in 2018. She also minored in mathematics and cybersecurity. She was a member of the Sketch Recognition Laboratory and was advised by Dr. Tracy Hammond. She currently works as a Software Engineer at Goldman Sachs in Manhattan, New York. Prior to working with Goldman Sachs, she worked as a Professional Engineering Intern at Walt Disney World, Orlando, FL, USA. Her research interests include human–computer interaction and machine learning and will pursue her advanced degree in the future.

SAMANTHA RAY received the B.S. degree in computer engineering with minors in cybersecurity and Spanish from Texas A&M University, in 2018, where she is currently pursuing the Ph.D. degree with the Sketch Recognition Laboratory under Director Tracy Hammond. She is also an Intelligent User Interface Researcher with the Sketch Recognition Laboratory under Director Tracy Hammond at Texas A&M University. She also works on sketch recognition systems that aid students in STEM fields develop their visual communication and spatial skills with instructor-like feedback. Other projects she has worked on include the human activity recognition (HAR) system for promoting healthy habits by recognizing the performance of activities of daily living (ADLs). Her research interests include human-AI collaboration and developing systems that understand human behavior.

VIJAY RAJANNA received the Ph.D. degree in computer science from Texas A&M University and was advised by Dr. Tracy Hammond. He was a member of the Sketch Recognition Laboratory, where he was involved in the development of sketch, gesture, and activity recognition systems. He is a Human–Computer Interaction and Machine Learning Researcher. He currently works at Sensel as a Senior Research Engineer. His dissertation work focused on the development of gaze-assisted, multi-modal, and accessible interaction methods for addressing impairments and disabilities. Also, using various machine learning models, he performed predictive analytics built on eye movements data applied to the domains of security, education, and economics.

TRACY HAMMOND received the B.S. degree in applied mathematics, the B.A. degree in mathematics, the M.S. degree in computer science, and the M.A. degree in anthropology from Columbia University, and she received the FTO (Finance Technology Option) degree and the Ph.D. degree in computer science from MIT. She is currently the Director of the Sketch Recognition Laboratory and a Professor with the Department of Computer Science and Engineering, Texas A&M University. She is an International Leader in sketch recognition and human-computer interaction research. Her sketch recognition research has been funded by NSF, DARPA, Google, Microsoft, and many others, totaling over 9 million dollars in peer-reviewed funding.