# Combining Low and Mid-Level Gaze Features for Desktop Activity Recognition

NAMRATA SRIVASTAVA, The University of Melbourne, Australia
JOSHUA NEWN, The University of Melbourne, Australia
EDUARDO VELLOSO, The University of Melbourne, Australia

Human activity recognition (HAR) is an important research area due to its potential for building context-aware interactive systems. Though movement-based activity recognition is an established area of research, recognising sedentary activities remains an open research question. Previous works have explored eye-based activity recognition as a potential approach for this challenge, focusing on statistical measures derived from eye movement properties—low-level gaze features—or some knowledge of the Areas-of-Interest (AOI) of the stimulus—high-level gaze features. In this paper, we extend this body of work by employing the addition of *mid-level gaze features*; features that add a level of abstraction over low-level features with some knowledge of the activity, but not of the stimulus. We evaluated our approach on a dataset collected from 24 participants performing eight desktop computing activities. We trained a classifier extending 26 low-level features derived from existing literature with the addition of 24 novel candidate mid-level gaze features. Our results show an overall classification performance of 0.72 ($F_1$-Score), with up to 4% increase in accuracy when adding our mid-level gaze features. Finally, we discuss the implications of combining low- and mid-level gaze features, as well as the future directions for eye-based activity recognition.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**; *Human computer interaction (HCI)*;

Additional Key Words and Phrases: Eye tracking, activity recognition, gaze features

**ACM Reference Format:**
Namrata Srivastava, Joshua Newn, and Eduardo Velloso. 2018. Combining Low and Mid-Level Gaze Features for Desktop Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 189 (December 2018), 27 pages. https://doi.org/10.1145/3287067

## 1 INTRODUCTION

Human activity recognition (HAR) has received much attention over the past few decades as the ability to identify and understand human activities has many immediate applications for quantifying human behaviours in areas such as surveillance, healthcare, education, as well as for building context-aware interactive systems in HCI and Ubicomp [3, 6]. Sensors used to capture activity information range from vision-based sensors (e.g. *RGB cameras* [56]) to wearable sensors (e.g. *accelerometers* [30]). These approaches have shown success in domains including fitness [12, 50], surveillance [1], rehabilitation [39], and affect [25, 51]. In contrast, recognising sedentary activities with minimal physical movements cannot leverage the same sensors and approaches; requiring new ways of thinking about activity recognition.

Authors' addresses: Namrata Srivastava, The University of Melbourne, Australia, namrata.srivastava@unimelb.edu.au; Joshua Newn, The University of Melbourne, Australia, joshua.newn@unimelb.edu.au; Eduardo Velloso, The University of Melbourne, Australia, eduardo.velloso@unimelb.edu.au.


2474-9567/2018/12-ART189 $15.00
https://doi.org/10.1145/3287067

In modern life, increasing amounts of time are spent using computers for work, education, and leisure. Recent surveys estimate that American adults spend between 9-10 hours in front of screens [31] and similar figures have been found for other industrialised countries like Australia [33] and the UK [40]. Time-tracking applications (e.g. *RescueTime*[1], *ManicTime*[2]) can monitor the amount of time a user spends using an application to improve productivity with the aim of encouraging a more sustainable work-life balance. However, these applications are limited in their ability, as they can only track *which* application is active, but not *how* it is being used. For example, in a software engineering context, such time-tracking applications can determine whether a user is programming by the fact that an Integrated Development Environment (IDE) is active but are unable to discriminate between finer-grained programming tasks such as *writing* or *debugging* a section of code.

Researchers in the field have proposed novel approaches for overcoming the limitations of existing approaches. For instance, Du et al. [13]'s approach uses audio-based (or acoustic) features generated from keyboard input to recognise sedentary activities. However, such approaches are highly dependent on the user explicitly providing input to the system and therefore not applicable to a broad range of common desktop activities (e.g. watching a video, reading a news article). Hence, the ability to perform activity recognition 'with a finer grain' and on a wide range of desktop-based activities remains an open and active challenge for the research community, as evidenced by the recent *Eye Movements in Programming* workshop series [3]. Overcoming such challenges will allow us to design computing systems that can proactively monitor daily activities and can either assist users with their daily tasks or encourage them to follow a healthy lifestyle. As described by Bulling et al. [6]'s, several real-world domains such as industrial sector, educational sector, sports and health sector can clearly benefit from activity recognition. We elaborate on the implications of our work in our discussion on future directions (Section 7.3).

Eye tracking provides a potential solution to this problem, as the relationship between action and visual attention can give us an insight into the activity being performed and the context of the user [10, 29]. For instance, reading has a recognisable left-to-right saccadic pattern [44], distinct from the pattern of concentrated fixations at the centre of screen elicited by first-person shooter games [53]. These changes in eye movement patterns are apparent, for instance, in the mean size and direction of saccades, or in the mean duration of fixations, which vary from activity to activity [29]. We consider these statistical measures to be *low-level* gaze features as they can be computed directly from the eye movement data for any activity or stimulus. Drawing from this principle, researchers have explored *eye-based activity recognition*; demonstrating the potential of machine learning classifiers that recognise activities based solely on eye movement measures [9, 28].

Whereas *low-level gaze features* are versatile and easy to compute, they are vulnerable to overfitting. This happens because certain eye movements are linked to visual characteristics of the specific stimulus used during training (e.g. layout, colour, salience) rather than the activity itself. In contrast, *high-level gaze features* abstract from this by considering the Areas-of-Interest (AoI) in the interface. Examples of such features include transitions between AoI's and time spent in a given AoI. The downside of these is that they require previous knowledge of the interface design.

In this paper, we propose the use of a level of abstraction in-between low- and high-level gaze features. We call such features '*mid-level gaze features*'. Mid-level gaze features do not require knowledge of the design of the interface itself but build on intuitions about the kinds of eye movements likely to arise during an activity. Intuitively, any person can make inferences about the type of activity the user is engaged in by observing gaze plots. For example, reading patterns in Western languages usually, present themselves as a series of short saccades from left to right with a long saccade from right to left at line breaks. We leverage this intuition to identify atomic components that are characteristic of a given activity in the time series data, hypothesising that using them as features might improve classification results (see Figure 1).

[1] http://www.rescuetime.com/
[2] http://www.manictime.com/
[3] http://emipws.org/

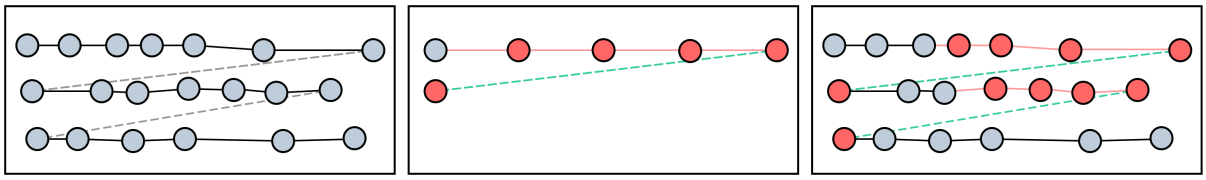

Fig. 1. Detecting mid-level gaze features from gaze plots. Left: Gaze plot of a user reading an English language text. Centre: Example of a mid-level gaze feature designed for detecting reading behaviours (*medium-line*). Right: Mid-level gaze features detected in the reading pattern.

To explore this idea, we first collected participants' eye movements (*N=24*) using a desktop screen-mounted eye tracker in a lab-based setting while they performed eight different activities consisting of general desktop activities and software engineering activities. We opted to collect a wide range of activities with a high number of participants using a non-intrusive eye tracker in response to the limitations found in the overall eye-based activity recognition literature (detailed in Section 3). We then extracted a total of 50 features to train our machine learning classifier. These features included two types of candidate mid-level gaze features (*shape-based* and *distance-based*), with the remaining being low-level gaze features either drawn from or inspired by related work. Our results show the distinct correlation between eye movement patterns and the activities that elicited them, suggesting a strong motivation for using the eye movements themselves as an indicator of activity. Further, the combination of our candidate mid-level gaze features has shown to have overall performance gain on recognising activities but varies from activity to activity. Lastly, we discuss the implications of our improved approach for activity recognition using the combination of low- and mid-level gaze features.

## 2 RELATED WORK

In this section, we focus on two key areas of literature: (1) prior work on eye-based activity recognition and (2) prior work on gaze pattern analysis that has motivated us to explore a higher level of gaze features.

### 2.1 Eye-Based Activity Recognition

Eye-based activity recognition is a topic that has been gaining interest in the fields of Human-Computer Interaction (HCI) and Ubiquitous Computing (UbiComp), with a majority of studies focusing on recognising reading behaviours and associated cognitive processes. Bulling et al. [8, 9]'s seminal work demonstrated the use of eye tracking as a promising modality for activity recognition; obtaining a 76.5% precision and 70.5% recall in their classification. In their work, participants (*N=8*) performed five activities in an office environment: reading a printed paper, watching a video, copying a text, taking handwritten notes, and browsing the web. Not only do eye movements show distinct patterns when different activities are performed, but can differ on a variety of factors; making it a compelling input to explore for activity recognition. For instance, an earlier work by Bulling et al. [7] showed that *context* can have an influence on a specific activity. Their work investigated the importance of gaze features in the prediction of reading activities in different contexts such as sitting, standing, walking indoors and outdoors, or riding a tram. Their straightforward analysis using a string matching algorithm achieved a recognition rate of 80.2%. Further, Kunze et al. [28] findings show that gaze patterns differ from stimuli to stimuli for a single activity. Their approach using eye-movement features based on fixation- and saccade-based features obtained a recognition performance of 74% from data collected on eight participants as they read five different Japanese document types. Therefore, it becomes an open question whether the models used in previous work (where only a single stimulus is presented per activity) predicted the result based on the gaze patterns themselves or were biased towards the specific stimuli employed in these studies.

Furthermore, we have found a wide variety of methods used in existing eye-based activity recognition literature. For classification, supervised machine learning algorithms have been widely used (e.g. Support Vector Machine (SVM) [9]; Hidden Markov Models (HMM) [9]); J48 decision tree classifier [28]). So far only Steil and Bulling [48] has proposed an unsupervised approach; where it was used to discover users' everyday activities in outdoor locations. We also note that the gaze features selected for classification in previous work tend to vary from one work to another. From a machine learning perspective, feature selection plays an important role,r and it is crucial to select a good subset of features to train the model, as the addition of irrelevant and redundant features can suffer from the *curse of dimensionality* or commonly known the problem of overfitting when trained with a small sample size [21]. As expected, all existing works on eye-based activity recognition have commonly employed gaze features derived from fixations and saccades, with a subset employing features derived from eye blinks (e.g. [24]) and head motion (e.g. [48]). The use of pupil diameter has also been proposed by Bulling et al. [9] as part of future work. However, the ability to elicit certain types of gaze data to be used for feature extraction is dependent on the hardware used for eye tracking itself, for example, obtaining pupil diameter can only be from eye trackers that use optical methods (e.g. video-based).

More importantly, the hardware used for eye tracking influences the mobility of participants and its applicability in everyday scenarios (i.e. head-mounted or screen-mounted). A significant portion of work has employed electrooculography (EOG) for eye-based activity recognition [7, 9, 10, 27], which requires electrodes to be attached onto the users' face while performing activities. Such devices can be classed as intrusive (or invasive) systems as they come into contact with the skin [14]. As a result, products that integrate an array of sensors into everyday wearable devices are beginning to emerge; a prime example is the JINS MEME eyeglasses[4]. Ishimaru et al. [19] employed the JINS MEME and was able to distinguish between natural and controlled reading; achieving an accuracy of 73.8% using 7 participants. Whereas using electrode-based eye trackers are well-suited for situations where the user requires six degrees of freedom or where calibration is not possible (as they do not require a direct line of sight to the pupil), they are limited in that they are usually not accurate enough to precisely estimate the point-of-regard. Further, the use of use electrodes can make these devices uncomfortable to be worn for extended periods and the fact that they also require good skin contact to work.

In contrast to electrode-based eye trackers, Kunze et al. [28] used head-mounted eye-tracker to recognise different kinds of Japanese literature in natural environments—office, coffee shop, home, library and lecture hall. Kiefer et al. [24] used a similar eye-tracker to predict six map-based activities using 220 features extracted from fixations and saccades; obtained a 78% accuracy with their 17 participants dataset. One advantage of head-mounted eye trackers is that they allow mobility, free head-movement and are comparatively less intrusive than traditional electrode-based eye trackers. However, their form factor still affects participants behaviour and their mobility can interfere with the quality of the data and can further result in difficulty in the analysis of the data due to changing field of view of the video feed.

Other approaches using multiple sensors have also been proposed to overcome the limitations of intrusive eye trackers. Ogaki et al. [41] combined eye-marker recorder (EMR) with a video-camera to predict five common activities like reading text, watching a video, writing, copying and browsing. In another study, Ishimaru et al. [20] combined eye-blink features with head motion features to perform activity recognition (82% accuracy with 8 participants). Although the combination increased recognition accuracy compared to works that have employed the use of a single sensor, the processing time was significantly high.

## 2.2 Gaze Patterns during Human Activity

Since Yarbus [55]'s seminal work on how eye movement patterns differ depending upon the task given to the participant, studies in recent decades have described the role of eye-movements in reading and information processing (see Rayner [44] for a review). For example, it is now clear that gaze patterns during natural reading

[4]https://jins-meme.com/en/

are different than while typing text. While reading English text, the mean fixation duration lasts approximately 200-250 milliseconds with a mean saccade length of 7-9 letter spaces. On the other hand, while typing, the mean fixation duration tends to increase to 400 milliseconds with smaller saccades (approximately four letters in length). There is also substantial variation within the same activity depending on the goal the user is trying to accomplish. For instance, eye movements during visual search are different than during natural reading, where fixations were longer when the same stimuli were used for both activities [44].

Understanding how software engineers perform software engineering activities has become an interest in recent years, which had led to researchers to employ eye tracking to gain insights (see Obaidellah et al. [38] for a survey). Software engineering activities also exhibit unique eye movement behaviours. Sharif et al. [46] showed that during debugging, programmers would often perform a scan of the entire code in an attempt to understand it before attempting to find and fix any bugs in the code. Their data also shows that after completing a scan, a programmer will jump vertically through the source code. Further, Busjahn et al. [11] demonstrated that in code reading, the mean fixation length is 100 milliseconds larger than natural text reading. Therefore, these findings indicate that software engineering activities, such as reading and debugging code, are not only fundamentally different from one another but can be differentiated from other behaviours such as natural text reading.

The overall literature suggests a strong relationship between eye movements and common everyday activities, and that activity recognition can be performed using eye movement data. Moreover, the existing body of work provides evidence that gaze features and context plays an important role in identifying everyday activities; which makes it compelling to explore the development of eye-based activity recognition systems. However, the results of existing work are difficult to generalise for three reasons. First, the results due to their small sample sizes (up to 17 participants) with multivariate data may suffer from the problem of overfitting [43]. Second, low-level features of eye movement have typically been extracted for classification, such as statistical measures of fixations and saccades, which can easily be affected by the design of the stimuli. Third, in works that compared a range of activities, a small number of activities are usually compared and has the tendency to being context-dependent (e.g. office activities). To do this effectively, we set out a research agenda in this work to further explore the types of gaze-based features, feature extraction methods and classification methods for eye-based activity recognition. Furthermore, we test the feasibility of using non-invasive eye trackers for data collection such as the use of screen-mounted eye trackers that have not yet been used for activity recognition to our knowledge.

## 3 DATA COLLECTION

For our data collection, we factored in two main limitations found in our review of the literature. First, we collected a dataset with 24 participants as they performed eight activities, with three variants of each activity (except for the browsing activity) to ensure a wider variety of behaviours to minimise the effect of the stimulus on the classification. In previous works, each activity was elicited with a single stimulus per activity (e.g. all participants read the same piece of text in the 'reading' condition or watched the same video in the 'watching' condition). As demonstrated by Kunze et al. [28], eye movement patterns can differ depending on the stimulus (e.g. textbook *versus* novel), even though the same activity of reading was performed. This raises the question of whether the classifiers in previous works recognised the activities or the stimuli. Moreover, using a small sample size with a large number of features can often result in misleading performance as the machine learning classifier often 'overfits' the data [43]. This substantially hinders the generalisability of these results.

Second, we examine the feasibility of using non-intrusive eye trackers i.e. screen-mounted eye trackers to perform eye-based activity recognition due to its unconstrained nature to allow for the tracking of natural gaze behaviour. We found that previous works have typically employed intrusive methods such as electrooculography

(EOG) in which electrodes are placed on the face around the eyes to measure the differences in eye movement [9] or require a user to wear a pair of wearable eye trackers [28]. We believe these limits natural movement, but more importantly, EOG signals are prone to drifting (slow signal changes mostly unrelated to gaze coordinates), which may result in an offset between the measured gaze point and the actual point of gaze, even with successful calibration [35]. In recent years, low-cost or integrated eye trackers have become increasingly ubiquitous, especially for the home environment [52], and therefore important to test its feasibility to perform activity recognition.

### 3.1 Activities

Each participant performed five *common desktop activities* and three *software engineering activities*, giving us a total of eight activities in our activity set. We derived our set of desktop-based activities from the combination of activities found in published research on activity recognition (e.g. [9, 13, 36]). We then selected the activities based on a number of considerations: (1) sustainable for a period of time (e.g. $\tilde{5}$ minutes), (2) applicability to desktop-based activities, (3) reflective of real-world use, and (4) varying level of interaction and input; whereby some having no input at all (e.g. watching a video) to constant input (e.g. writing code). Based on the aforementioned considerations, we devised an initial set of common desktop-based activities from Bulling et al. [9],—Read, Watch, and Browse. We derived the Play activity from Du et al. [13], which compared chatting, coding, writing documents, and playing games). The Search activity was derived from the list of tasks explored by Lorigo et al. [34] where they asked the participants to search for ten questions on the internet—five of which were homepage-searches, and other five were informational searches. However, to deeply understand the search and evaluation behaviour of participants we have only included informational questions for our Search activity.

Further, we considered software engineering activities as a common desktop activity. As described in Section 2.2, researchers have explored eye movements in such activities and demonstrated that there are unique differences in their patterns (e.g. [11, 38, 46]). While Du et al. [13] have only considered 'coding' activity as whole, we considered the three coding-based activities—Interpret, Debug, and Write as separate activities. We used Microsoft Visual Studio as our IDE, and these activities were written (at a beginner-intermediate level) in C# but also included a Python variant for participants who preferred it. Only the Watch activity did not require the participant to interact with the keyboard or mouse. All stimuli, except the Watch activity-set were presented for 5 minutes, even if the participants were unable to complete their task, they were shown the next stimuli. The stimuli were presented in English, but in certain tasks (e.g. Search), participants were allowed to access content in other languages if they wished. For the remainder of this section, we detail the nature of the activities and the motivations of inclusion based on existing eye tracking literature. Figure 2 shows examples of three activities (Play, Read and Browse) overlaid with gaze plots for a single participant; showing differences in gaze patterns for each stimulus variant used.

**Read:** In Bulling et al. [9] and Kunze et al. [28]'s work, the authors collected reading behaviours while reading on paper. However, taking into account the recent trend towards digital reading [32], we collected screen-based reading behaviours instead. We asked participants to read a piece of text on a computer and then summarise it to the researcher. We have chosen this approach instead of quizzing participants to elicit natural reading behaviours (e.g. skimming) instead of focused reading. For the three sets of activities, we decided to include three different reading materials in English—an excerpt from a book, an article, and a short story. The text layout of these materials differed, where the article was written in paragraphs of more approximately 5-6 lines, while the book contained a mixed of speech and narration. As eye movements patterns can differ depending on the difficulty of the text [44], we differed the stimulus significantly in their fonts, spacing and paragraph structure (see Figure 2).

**Watch:** Similar to Browse, the Watch activity was performed the same way as in previous studies [9, 20]. The participant was asked to watch a short video played in full screen and then to summarise it to the researcher. Each video was between 5-6 minutes long and consist of a different number of main characters to add variation in the activity-set. In short, we have chosen a black-and-white animated short movie with two main characters, an animated short movie

with only three characters and a short independent film. Further, all videos selected was deemed to be interesting as well inspirational in their nature; sufficient to hold the attention of the participants throughout the activity.


Fig. 2. Examples of Activities and Corresponding Gaze Patterns for Play, Read and Browse activities. As shown, different types of stimulus for the same activity results in different gaze patterns, and therefore it is important to vary the stimulus across activities to prevent overfitting the features to the classifier.

**Browse:** We asked participants to browse the internet freely using the web browser similarly to prior work (e.g. [9, 41]). In this activity, participants opted not to log into websites that required a login, particularly personal

social media and instead chosen to browse public news websites and blogs that they frequently visit. Interestingly, we observed that participants who spoke English as a second language gravitated toward reading the news in their first language, which presents us with a large variety of gaze patterns. In Figure 2, Browse-Centre is in the Persian language, which webpage layout is flipped to a right-to-left format, giving us right-to-left saccades.

**Search:** Unlike previous studies, where the search task consisted of visual inspection of the scene [18], we asked participants to search for answers using a search engine from a list of predefined questions. Depending on the query, some answers were immediately provided by the search engine while others required more effort. We cleared the browsing history before every session so that all participants started from the same baseline. Our motivation to add this new category of activity was to understand how different the eye-gaze patterns in focused search are compared to natural reading or free-browsing.

**Play:** Participants were presented with simple free to play online game with an explanation of the rules. We chose three different games, one requiring the participant to look ahead (Classic Mario), one requiring the player to follow an object (Pong variant) and one where the character was affixed to the middle of the screen and is required to look in all directions they needed to navigate the character for survival (Agario). The participant was instructed to play using the keyboard and mouse and received a short training ($\tilde{1}$ minute) to get used to the controls if requested. Figure 2 shows the three different games chosen along with clear differences in gaze patterns for each game.

**Interpret:** Participants were presented with three short function implementations and were asked to predict the output of a code snippet. We provided three functions with increasing difficulty, by increasing the number of variables and loops.

**Debug:** Participants were presented with a function implementation that had multiple bugs and the expected output of the function. They were then asked to identify and fix any bugs they find and to confirm that their fix worked by executing the code. They were free to debug the code in any way they deemed appropriate, but without consulting any resources outside the IDE (e.g. a search engine, which would confound with the Search activity). We provided the expected output of the code and varied the type of bugs present in the code (e.g. syntax, logic, arithmetic).

**Write:** Participants were asked to implement three functions in increasing order of complexity. Examples include: printing the product of a set of numbers, printing the first ten numbers in the Fibonacci sequence, and sorting a set of numbers in ascending order using bubble sort algorithm.

### 3.2 Participants & Procedure

We collected data from 24 participants (16M/8F) from the same University, consisting of postgraduate students and research staff, aged 24 to 48 (M=29.8). We recruited a larger sample size than previous studies to push the generalisability of our approach. Participants were fluent English speakers and proficient in either C# or Python programming languages. This was a requirement to ensure that they were able to complete the software engineering activities. The majority self-rated their programming ability as 'intermediate' (N=13), followed by 'advanced' (N=8) while the remainder as 'beginner' (N=3).

Before starting the session, participants were asked to sign a consent form and to fill in a short demographics questionnaire. We recorded our eye movement data using a Tobii Pro X2-30 eye tracker[5] (mounted on a 24-inch monitor) and the Tobii Pro Studio software. We seated the participant in a comfortable position and adjusted the chair so that the participant was approximately 60 cm away from the screen. We then performed the manufacturer's default 9-point calibration. Once the participant was successfully calibrated, the participant was free to move their head and upper body throughout the experiment; afforded by the tracking robustness of using a screen-based IR-illuminated eye tracker. Depending on the activity, the participant was instructed to interact with the keyboard and/or mouse before commencing and reminded to focus their attention on the activity at hand. All stimuli were pre-loaded on the computer and half of the participants performed the software engineering activities first. We

[5]https://www.tobiipro.com/product-listing/tobii-pro-x2-30/

randomised the order of each set but kept the activity groupings intact, meaning that participants performed all the software engineer activities together first then all the common desktop activities together after or the other way around, but in a random order in each group. Each session lasted approximately 60 minutes.

## 4 FEATURES

As mentioned in Section 2, feature extraction and selection play an important role in a system's ability to make correct classifications. In this section, we describe the differences between the three levels of gaze-based features used in our work based on their characteristics. Figure 3 below illustrates the differences between each category of features.

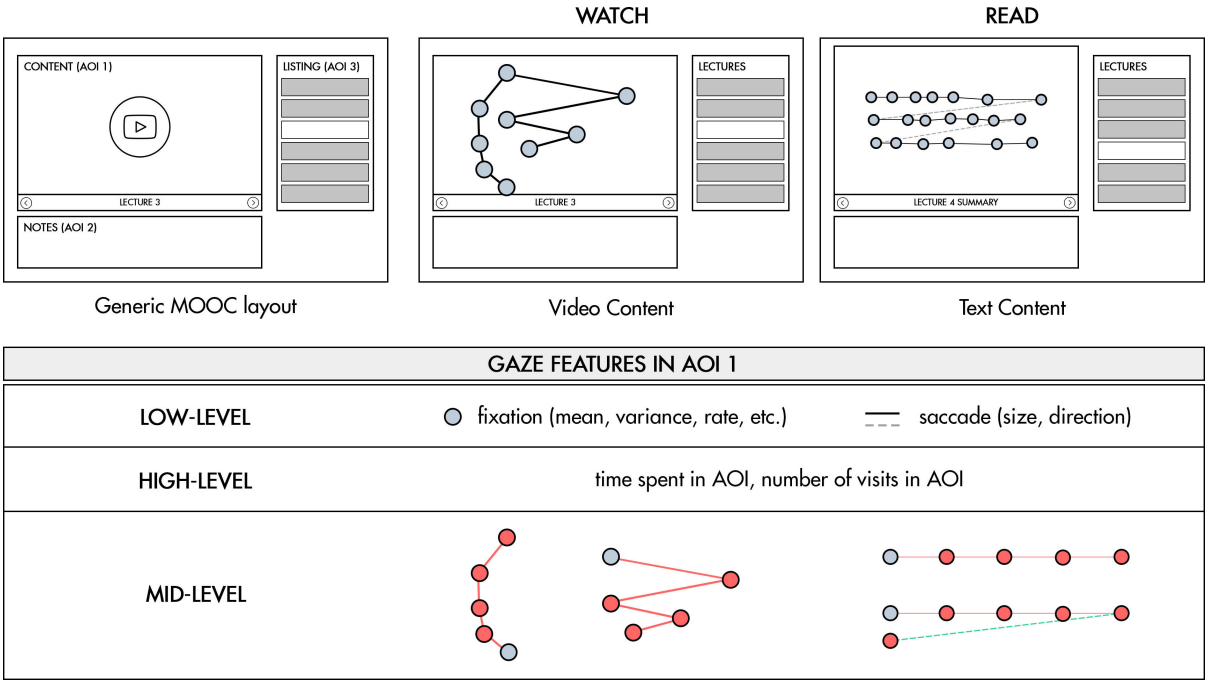


Fig. 3. Feature Levels. The figure shows that different activities can be performed in the same area-of-interest (AOI) on the user interface. Low-level features are derived from raw gaze coordinates, and are mostly fixation and saccade-based features. High-level features summarises low-level features such as the total fixation duration in each AOI to form a feature. This requires the knowledge of the interface, and may not be generalised to a broad range of activities. Our proposed mid-level gaze features combines fixations and saccades into patterns, and their counts are used as features for classification.

**Low-Level**: Low-level features are computed from eye tracking raw gaze coordinates $(x, y)$, and eye behaviours such as eye blinks, or at a physiological level (e.g. pupil diameter). Such features are easy to compute but at the same time vulnerable to overfitting when used for classification. This happens because certain eye movements are linked to visual characteristics of the specific stimulus used during training (e.g. layout, colour, salience) rather than the activity itself. We provide a more detailed description of low-level gaze features in Section 4.1 below, including features that we have employed in our study.

**Mid-Level**: Features at this level are derived from fundamental gaze behaviours such as direction, and therefore require knowledge of the sequence of low-level gaze features. For example, reading patterns in Western languages usually present themselves as a series of short saccades from left to right with a long

saccade from right to left at line breaks (as shown in Figure 3). Intuitively, any person can make inferences about the type of activity the user is engaged in by observing gaze plots. We leverage this intuition to identify atomic components that are characteristic of a given activity in the time series data, hypothesising that using them as features might improve classification results (see Figure 1). Mid-level gaze features do not require knowledge of the design of the interface itself, but build on intuitions about the kinds of eye movements likely to arise during an activity.

**High-Level**: These features consider the Areas-of-Interest (AoI) of the interfaces and require either manual or automatic determination of the bounds of these AoI's. Examples of such features include transitions between AoI's and time spent in a given AoI. We have decided not to include high-level gaze features in our analysis and comparisons, as these features are stimuli-dependant and require previous knowledge of the interface design and therefore not applicable to all contexts due to changing interface designs across activities.

We extracted from our dataset a total of 50 features (summarised in Table 1). As critiqued by Bednarik et al. [4], there is still a lack of standard metrics in eye-movement research, and that finding the best feature set often requires exploratory analysis. For this reason, we have not only cast a wide net by including features for the purposes of possible higher classification accuracy but have also begun to explore variations in such existing features. For simplicity, we have classified features used in prior work (e.g. [24]) that were computed using the positions and duration of fixations and saccadic movements as *low-level gaze features*.

Furthermore, our work aims to explore a new feature set based on the shapes and patterns of these low-level gaze features. The idea is inspired by Bulling et al.'s work, where they define an activity "*using a grammar built upon an alphabet of basic eye movement atoms*" [9]. Whilst our approach does not define activities using a grammar, it uses this idea whereby there are fundamental eye movement 'atoms' that are representative of different activities. We used these atoms in order to generate more features from existing fixation- and saccade-based features. We call these atoms *mid-level gaze features*. For the remainder of the section, we describe in detail the features we have employed in our work, and how they were conceptualised.

### 4.1 Low-level Gaze Features

Low-level gaze features capture spatiotemporal characteristics of the eye movements and is centred around two fundamental eye movements—*fixations* and *saccades*. We summarise these features in Figure 4 and describe each of them below. We have chosen not to include features derived from pupil data and blink rate in our study, as these features are strongly influenced by extraneous factors such as the interface design and the difficulty level of the activities. Existing works have shown that with increase in task difficulty, change in pupil diameter increases [15] and blink frequency decreases [2]. For instance, for the same activity such as playing a video game, two participants can have different levels of cognitive load depending on their experience with the game. Moreover, to accurately measure the small changes in pupil diameter, we require eye-trackers with high sampling rates (500 Hz or over) with restraint movement, which may not be available to the average consumer, and unsuitable for everyday interaction if users need to be restrained. Furthermore, we selected a set of common low-level gaze features drawn from related work, which consists of features based on horizontal ($x$) and vertical gaze coordinates ($y$), i.e. fixations- and saccade-based features only for the purposes of our study.

*Fixation-based Features.* Based on the basic properties of fixations, we selected five initial features: the mean, variance and standard deviation of fixation duration, followed by fixation rate and fixation slope. These features have shown to be important for accurate classification in previous studies [9, 28]. In order to capture how close fixations are in a space-dimension, we introduce a new feature called *fixation dispersion area* based on fixation dispersion feature commonly found in eye tracking literature (e.g. [5, 37]). The fixation dispersion feature used in prior work takes the mean of all fixations and finds the distance from the mean to all the fixation points. Our modified feature (illustrated in Figure 4), encloses dispersed fixation points in a rectangle. We chose a threshold

Table 1. List of all the 50 features used for our approach.

| *Category* | *Sub-Category* | *Features* | *N* |
|---|---|---|---|
| shape | string | string-up, string-right, string-down, string-left | 4 |
| | line | medium-line, long-line | 2 |
| | compare | compare-left-right, compare-right-left<br>compare-up-down, compare-down-up | 4 |
| | scan | scan-right-left, scan-left-right, scan-up-down, scan-down-up<br>scan-right-up, scan-up-right, scan-left-up, scan-up-left<br>scan-right-down, scan-down-right, scan-left-down, scan-down-right | 12 |
| distance | return patterns | r-pattern | 1 |
| | elsewhere patterns | e-pattern | 1 |
| saccade | size | sacc-mean, sacc-var, sacc-sd | 3 |
| | direction | follow-saccade, neighbouring-saccade, opposite-saccade,<br>sacc-right, sacc-left, sacc-up, sacc-up-down,<br>sacc-up-right, sacc-down-right, sacc-up-left, sacc-down-left | 11 |
| fixation | duration | fix-mean, fix-var, fix-sd | 3 |
| | rate | fix-rate | 1 |
| | slope | fix-slope | 1 |
| | dispersion area | fix-disp-area | 1 |
| | count | brief-fixation, hold-fixation | 2 |
| | distraction | distract-right, distract-left, distract-up, distract-down | 4 |

of 75% for this feature, where the rectangle takes into 75% of dispersed fixation points and ignores the remainder to prevent outliers from skewing the rectangle.

In prior work, blink durations have been used as a feature to determine if a participant has disengaged from the task (e.g. [24]). We have taken a simpler approach to determine disengagement i.e. if a participant has looked away from the screen. As eye trackers can track beyond the bounds screen, we are not only able to determine whether a participant has looked away from the screen but also in which direction. Hence, we counted the number of fixations that occurs outside the active display in four directions (up, down, left, right) as features that we call *distractions*. Lastly, we added two features based on fixation duration—the number of brief fixations (< 500 milliseconds) and the number of long fixations (> 500 milliseconds). In total, we selected a total of 12 fixation-based features for our classifier.

*Saccade-Based Features.* Similar to fixation-based features, we selected initial features from basic statistical properties: the mean, variance and standard deviation of saccade length. Next, we included the saccade counts in 8 directions (defined as 45-degree sectors) as features (illustrated in Figure 4). Based on the cardinal direction features, we computed three additional features: follow direction count, neighbouring direction count and opposite direction count. The majority of these features were inspired by Kiefer et al. [24]'s work.

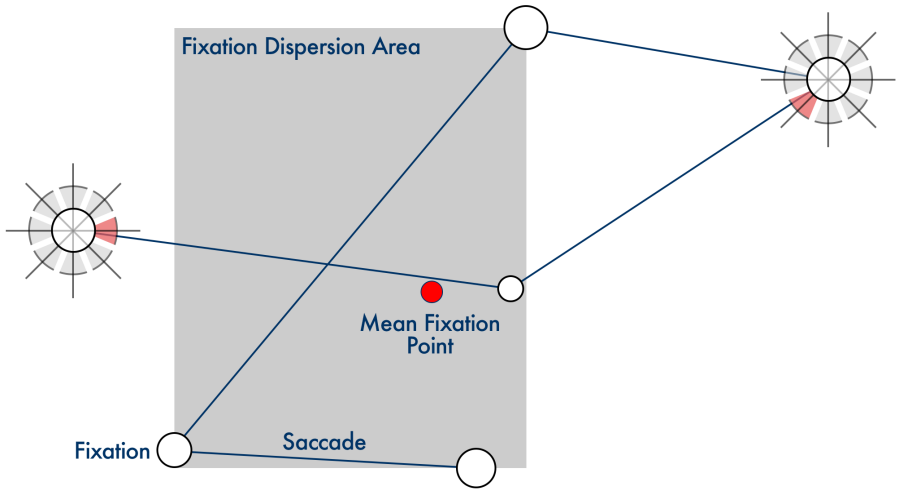


Fig. 4. Low-level gaze features.

## 4.2 Mid-level Gaze Features

As described in Section 1, the proposed category of *mid-level gaze features* was built upon our intuitions considering the different patterns that arise while performing an activity. During pilot testing, we observed that activities were different from each other based on their recorded saccade lengths. For example, the Read activity consists of a large number of short saccades in the right direction followed by a long left saccade, whereas Browse and Search activities contain short saccades in both left and right directions. Similarly, we also observed distinct patterns while considering saccade directions. For instance, a large number of saccades in the sequence of *up-down-up-down* directions were observed in the Debug activity, as the participant continuously compared the content in the source code window with the output window at the bottom in the IDE. Likewise, the spread of the fixation data also gave us insights about the activities. For example, the Interpret activity requires the user to focus on a small area of the screen, creating a cluster of fixations around the code blocks; whereas for activities such as Watch and Search, the fixations were distributed throughout the screen.

In order to build these patterns, we employed a character encoding methodology similar to Bulling et al. [9]'s wordbook. We first converted the eye-movement sequence into a series of saccade atoms. An atom is defined as a combination of two letters where the first letter denotes the saccade type (short, medium or long) and the second letter denotes the direction of its occurrence (left, right, up or down). For the fixation data, we calculated the distance between every three consecutive fixations and based on their measured distance; we extracted fixation-based features. Based on these encodings, we found that we can group the characters into patterns which can relate to the visually distinguishable patterns discovered during pilot testing. We categorised them into two primary types of pattern features—*shape-based pattern features* and *distance-based pattern features*. The shape-based pattern features are based on saccade encoding, while distance-based pattern features are generated using consecutive fixations. The number of occurrences of all the mid-level gaze patterns—*shape-based pattern features* and *distance-based pattern features*—were used as a feature to train the classifier.

*Shape-Based Pattern Features.* Figure 5 above summarises the 22 shape-based patterns and their encodings used in this paper. Shape-based patterns consist of atoms that correlate with gaze length and gaze direction. Considering the

different patterns which can be created, we further classified them as *string-patterns*, *line-patterns*, *compare-patterns* and *scan-patterns*. Based on our observations in pilot testing, we found that different shapes capture different aspects of eye-movement activity. For instance, we found that *string-patterns* help us to determine if there was a sequence of consecutive saccades in the same direction. As for *line-patterns*, they are used to help us to detect any consecutive saccades in the right direction, followed by a left medium or long saccade. Patterns used for comparison *compare-patterns* are indicative when a participant is performing any comparison between two points, for instance, when a participant is looking between two parts of the screen repeatedly. Further, to make these patterns interface-independent, we have considered saccades in both horizontal and vertical direction. We also found that during activities such as writing code (Write), the participant continuously reviews their writing. The *scan-patterns* can help us to identify such patterns not only in horizontal and vertical directions but also in corner directions.

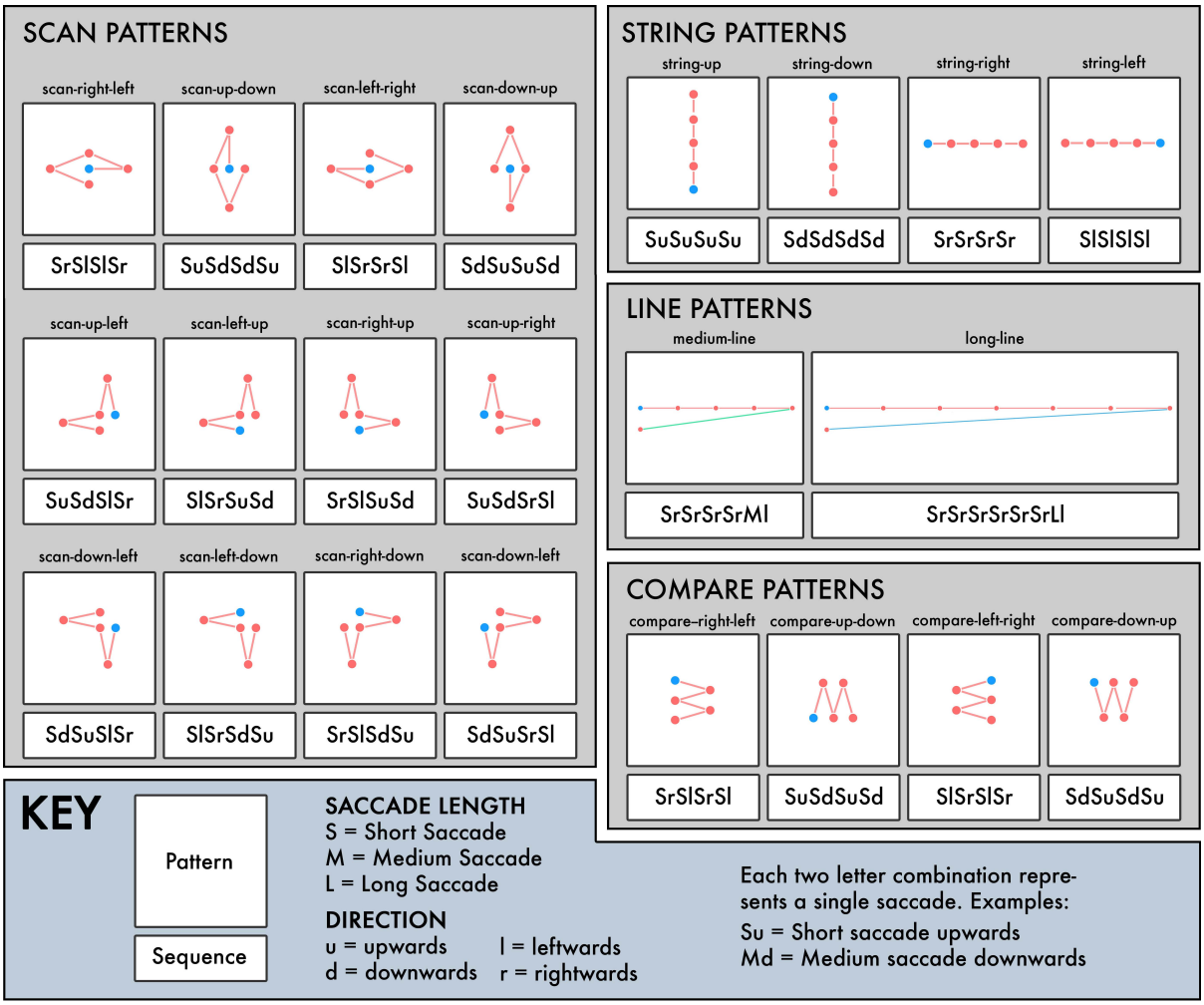


Fig. 5. Shape-Based Pattern Features.

*Distance-Based Pattern Features.* We propose another category of mid-level gaze features in this work, what we call 'distance-based pattern features'. Unlike shape-based patterns, we based these features on the sequence of fixations occurring together within a well-defined distance. They can be further classified as *return patterns* and *elsewhere patterns*. With reference to Figure 6, let $f_i$, $f_{i+1}$ and $f_{i+2}$ be three consecutive fixations, such that $f_i$ and $f_{i+1}$ are at least $r$ distance away. Then, the next fixation $f_{i+2}$ is known as a *return pattern* when it is less than $r$ distance away from the initial fixation ($f_i$). However, if $f_{i+2}$ is outside $r$ distance away from other fixations, then it is known as elsewhere pattern. Based on visual inspection, we set the value of $r$ as 300 pixels in our work. In our pilot testing, we observed that *elsewhere-patterns* were indicative of activities in which participants' fixations were distributed throughout the screen, for example, in the Watch and Search) activities. Whereas *return patterns* identify activities in which they were required to revisit a section on screen, e.g. in the Play activity, the participant continuously tracks its move as well. Similar patterns can also be observed in the code-based activities, such as Debug and Interpret where the participant revisits blocks of code to attain deeper understanding.

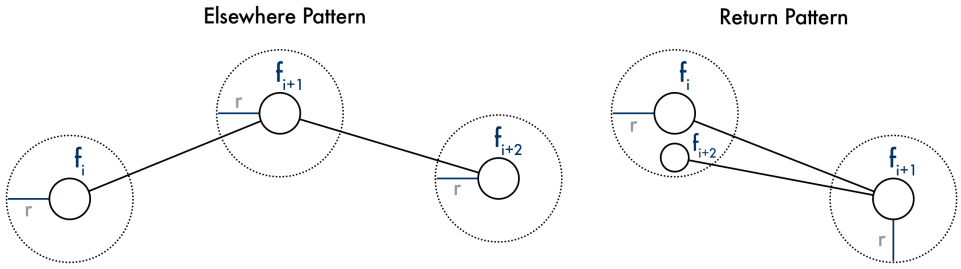


Fig. 6. Distance-Based Pattern Features

## 5 METHODOLOGY

In this section, we describe our step-by-step method for classifying everyday desktop activities. First, in the preprocessing step, we convert the raw gaze data into a list of fixations and saccades using a fixation filter. Next, we extract a list of candidate features; a combined total of 50 low- and mid-level gaze features (described in Section 4). Further, in the classification step, we build and test different classifiers based on these features. We then measure the performance of machine learning classifiers by comparing the original activity label with the predicted activity label. Our method from the preprocessing step to feature extraction step is illustrated in Figure 7, while the following subsections give a detailed explanation of each of these steps. Additionally, we added the algorithm for our method in Appendix A.

### 5.1 Preprocessing

The raw gaze data is a time-series data, comprising of horizontal ($x$) and vertical ($y$) gaze coordinates and a timestamp. Although the eye tracker's proprietary software (Tobii Pro Studio[6]) was able to automatically process the raw gaze data into fixations and saccades, we opted to implement Olsson [42]'s Fixation filter to gain control over two fixation parameters—*average windows size* and *peak threshold*. Using a custom built tool, we can visualise the fixations and saccades by adjusting the parameters. We configured the filter with an average window size of 7 samples and a peak threshold of 25 pixels, and subsequently extracted the saccades between every pair of consecutive fixations.

[6]https://www.tobiipro.com/product-listing/tobii-pro-studio/

## 5.2 Feature Extraction

The most common features used for eye-based activity recognition are low-level gaze features [19, 22]. These features depend on the spatial-tempo coordinates of the saccades and fixations. However, to record the interaction between these basic eye-movement features, additional features were generated by encoding saccades into a sequence of characters according to their direction and length as used in previous studies [7, 9, 24, 41]. An example of such an approach is to build a wordbook using n-grams matching and computing a histogram of each of the n-gram patterns as used by Bulling et al. [9].

Instead of employing Bulling et al. [9]'s approach of building a wordbook using n-grams, our approach was to create a table that consists of all mid-level gaze patterns that we have described in the previous section. This was inspired by Bulling et al. [7]'s work where the authors detected reading activity in transit by matching a prototype string that represented a typical reading sequence ("Lrrrrrrrr") over the encoded saccade string, character by character. With knowledge of this pattern, the Levenshtein distance between the saccade string and prototype string was calculated for each step. Finally, the classification between 'reading' and 'not reading' was done by applying a threshold on the Levenshtein distance vector.

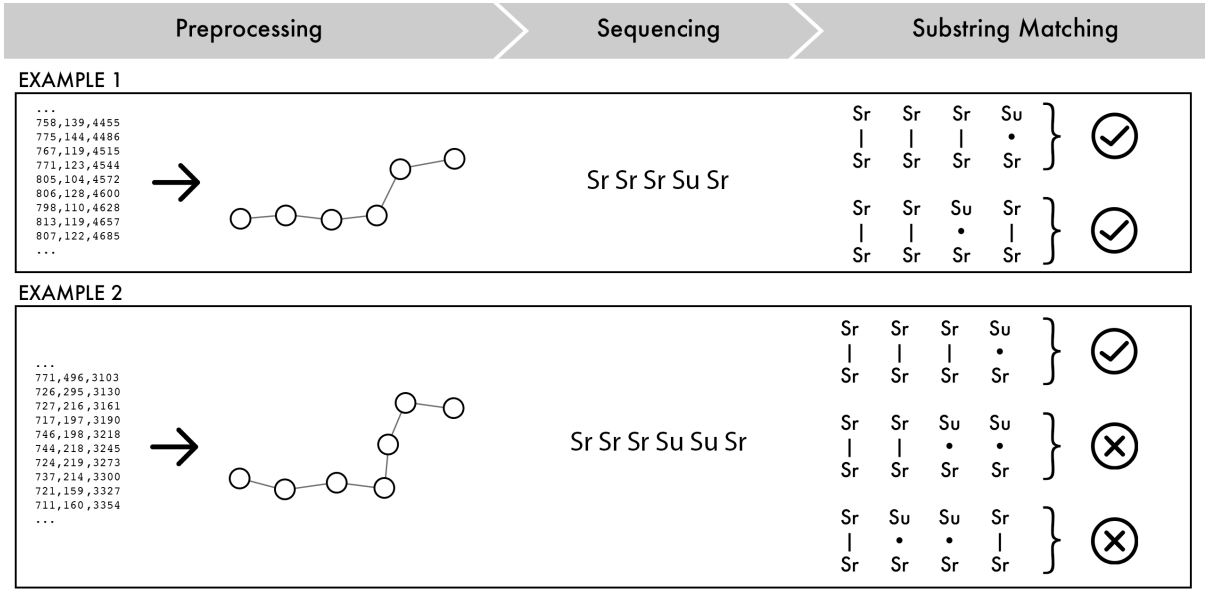


Fig. 7. Feature Extraction. We first process raw gaze data into fixation and saccades. We then encode saccades into a sequence of strings, each representing the saccade length and direction. Here, we are searching for four short saccades in the right direction (SrSrSrSr). Using the Smith-Waterman algorithm [47] with a threshold of one replacement, we obtain two matches in the first example and only one match in the second example ( | denotes an exact match, • denotes a replacement ).

In our work, we extend this idea by firstly identifying patterns of behaviours that are intuitively linked to a certain activity based on saccade encoding in 4-cardinal directions and three length sizes (short, medium or long). We define these patterns as *shape-based patterns* and are presented in Figure 5. We next stored these patterns in a lookup table, and derived our mid-level gaze features $F_p$ by counting the number of matches of the patterns ($p$) in the eye-movement sequence using Smith-Waterman local alignment algorithm [47]; an algorithm typically used for detecting DNA or protein sequences. The advantage of using a local alignment method for string matching helped

us to handle small gaze errors by allowing one-replacement during string-matching. For instance, as shown in Figure 7, after encoding the eye movement raw data as sequence-string, we first match it with the prototype string ("SrSrSrSr") using local alignment method. The algorithm instead of performing character by character matching returns a list of possible alignments generated by either deleting, inserting or substituting a single character. The algorithm then applies a threshold $T_p$ to allow only one substitution for each locally aligned sub-sequences. This threshold defines how tolerant is string-alignment towards gaze errors in the raw data. Similarly, we calculated the distance between three consecutive fixations and based on their distance; we further counted the number of occurrences of *return-patterns* and *elsewhere-patterns*. The number of occurrences of both the *shape-based patterns* and *distance-based patterns* were used as mid-level gaze features to train our classifiers.

Finally, we extracted low- and mid-level features from the fixation and saccades based on different time windows. The time window has a significant effect on the number of fixations and saccades available for feature extraction. The smaller the time window, the more fine-grained the recognition can be, but less information it will have to make a robust decision. Previous research has used different times windows over which they extract the features, from 20 seconds [24] to 30 seconds [9] and up to 60 seconds [28]. For our study, we generated data for 15-second increments until 150 seconds. To further analyse the effects of different time window on classifier performance, we trained our classifiers on each of the time windows with a window step of 1 second.

### 5.3 Classification

To understand, the importance and differences between our selected features based on classifier performance, we first divided the feature dataset into two feature sets:

**Low-Only**: Low-level gaze features only.
**Low+Mid**: Low-level combined with mid-level gaze features.

We trained a set of three classifiers: Support Vector Machines (SVM), K-Nearest Neighbour (K-NN) and Random Forest on our whole dataset consisting of 24 participants. The classifiers were trained for both feature sets generated in each time window. Further, to evaluate the performance of our model, we used a k-fold cross validation as compared to leave-one-out cross-validation (LOOCV) used in previous studies [9, 28]. LOOCV has a large variance in comparison to k-fold CV because training sets in LOOCV have more overlap [17]. For this reason, we performed a k-fold cross validation ($k = 4$) across all participants, such that in each cross-validation fold, all the data from a participant is either in training set or testing set. In our case as an example, a value of $k = 4$ meant that 18 participants' data was used for training and the remaining 6 participants' data for testing. This was repeated across all the folds.

Following, we performed *parameter tuning* for finding the set of optimal parameters for each of the three classifiers. For the SVM classifier, we fixed the two hyper-parameters $C = 10$ and $gamma = 0.01$ with RBF kernel. Similarly, the random forest classifier was tuned using 1000 trees and seven variables for splitting at each node. The K-NN classifier was trained with $k = 10$ neighbours. Similar to Bulling et al. [9], we used $F_1$-Score to evaluate the performance of each of the classifiers. The $F_1$-Score is defined as $F_1 = 2 \times (precision \times recall)/(precision + recall)$ and corresponds to the harmonic mean of Precision ($TP/TP+FP$) and Recall ($TP/TP+FN$) scores, where TP, FP, TN and FN are number of true-positives, false-positives, true-negatives and false-negatives respectively [45].

## 6 RESULTS

The primary aim of this paper is to explore the opportunity of incorporating mid-level gaze features to perform activity recognition on everyday desktop-based activities, and how the combination of low- and mid-level gaze features can potentially lead to greater recognition performance. In the following subsection (Section 6.1), we first describe how the performance of the classifiers improved when mid-level gaze features were used in combination with low-level gaze features. Following, we then focus our attention on the role of eye-movement features in classifying individual activities using confusion-matrices and importance graphs in Section 6.2.

## 6.1 Classifier Performance

*6.1.1 Performance across All Activities.* We first compared the performance across all the activities for different time windows by evaluating its classification performance using three classifiers—SVM, K-NN and Random Forest. Table 2 shows the performance of the classifiers using the $F_1$-Score measure for both feature sets for comparison. The K-NN classifier performed poorly for both feature sets. The reason being it is a 'lazy learner', and does not learn a discriminative function, but rather " memorises" the training data as noted by Duda et al. [16]. As our dataset is highly complex and multi-dimensional, the K-NN classifier proved not suitable for our purpose. Similarly, the SVM classifier performed well, but the results were highly dependent on the two hyper-parameters—$C$ and *gamma*. Due to high variability in results for these parameters, we have chosen to explore our results using the Random Forest classifier for the remainder of our analysis.

Table 2. Classification Results. $F_1$-score for three classifiers (SVM, K-NN and Random Forest) for both the feature sets for each time window.

| Time Window (s) | SVM | | K-NN | | Random Forest | |
|---|---|---|---|---|---|---|
| | *Low-Only* | *Low+Mid* | *Low-Only* | *Low+Mid* | *Low-Only* | *Low+Mid* |
| 15 | 0.53 | 0.51 | 0.43 | 0.41 | 0.53 | 0.54 |
| 30 | 0.59 | 0.58 | 0.49 | 0.47 | 0.60 | 0.61 |
| 45 | 0.61 | 0.62 | 0.53 | 0.50 | 0.64 | 0.66 |
| 60 | 0.64 | 0.66 | 0.54 | 0.55 | 0.63 | 0.66 |
| 75 | 0.66 | 0.66 | 0.58 | 0.55 | 0.69 | 0.70 |
| 90 | 0.65 | 0.68 | 0.58 | 0.59 | 0.66 | 0.69 |
| 105 | 0.67 | 0.68 | 0.60 | 0.60 | 0.70 | **0.72** |
| 120 | 0.68 | 0.68 | 0.61 | 0.60 | 0.69 | 0.72 |
| 135 | 0.68 | 0.70 | 0.63 | 0.61 | 0.68 | 0.73 |
| 150 | 0.69 | 0.71 | 0.63 | 0.60 | 0.69 | 0.73 |

Random forests are ensemble classifiers, and unlike SVM and K-NN, they are not dependent on their hyper-parameters. With highly complex multi-dimensional dataset such as ours, they were able to achieve a maximum of 0.70 $F_1$- score with low-level features only (105 second time window). Moreover, we can see an improvement in the scores when mid-level gaze features were used as shown in Figure 8. The figure shows the values of $F_1$-Score over different time window for both the feature sets (Low-Only and Low+Mid) whereby we can see an improvement of 1% - 4% in the performance of the classifier by incorporating mid-level gaze features.

Figure 8 also shows that the length of the time window greatly influences the performance of the Random Forest classifier. When using a short time window (i.e. between 15-60 seconds), the random forest classifier was able to achieve a maximum $F_1$-Score of 0.63 using the Low-Only feature set and 0.66 with the Low+Mid feature set. One possible explanation for this low performance can be related to the fact that the time window defines the number of samples used for generating the features. Some of our low-level gaze features (e.g. fixation dispersion area, saccades direction features) and all mid-level gaze features depended on the interaction between the fixations and saccades in the time window. Therefore, the greater the time window, the higher the number of interactions that are captured through the features definition. We decided to choose the 105 seconds as a representative time-window for further analysis, as after this time-window, the $F_1$-Score appeared to have plateaued and no longer increased. Moreover, we believe a window-size of this size ($\tilde{2}$ minutes) was fine-grained enough for the time scale of the activities in which we are interested.

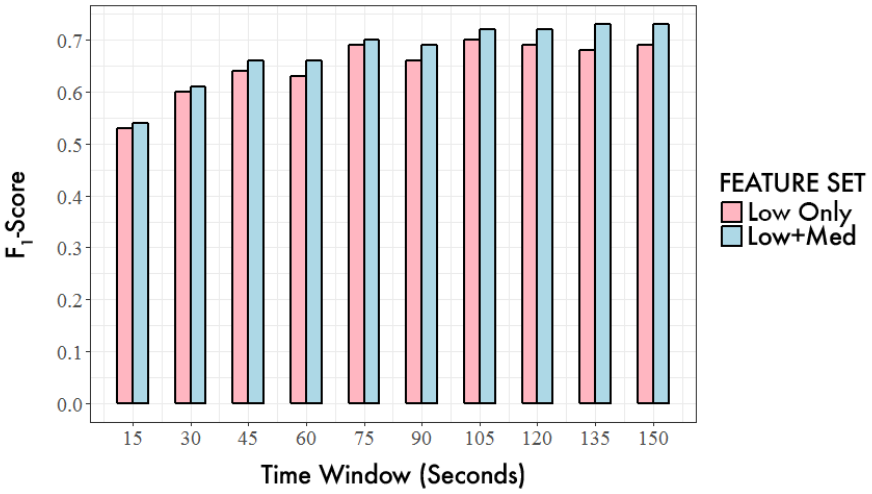


Fig. 8. Random Forest Classification. $F_1$-Score for Random Forest classifier for each time window..

*6.1.2 Performance for Each Activity.* We were also interested in understanding the effect of the combination of features on the performance of individual activities. Further, we also wanted to find out if our novel mid-level gaze features were able to increase the performance of the classification. As described earlier, we selected a representative time window of 105 seconds and trained a Random Forest classifier. The input dataset consists of 43748 samples of 26 low-level gaze features and 24 mid-level gaze features, containing 8 class labels, where each class represents on activity. Table 3 shows that our dataset is balanced between the 8 activities. Training a classifier with a balanced dataset helps us to avoid misleadingly high accuracies due to multi-class imbalance problems (see Wang and Yao [54] for further discussion).

Table 3. Class distribution. Number of samples and proportion of each class in the dataset.

| | Browse | Debug | Interpret | Play | Read | Search | Watch | Write | **Total** |
|---|---|---|---|---|---|---|---|---|---|
| Samples | 4812 | 5362 | 5392 | 5261 | 5045 | 5791 | 5860 | 6224 | **43748** |
| Proportion (%) | 11 | 12.2 | 12.3 | 12 | 12 | 13.2 | 13.3 | 14 | **100** |

Table 4 contains a brief overview of the performance of the two feature sets in predicting each activity by comparing their accuracy and $F_1$-Scores. As shown, the Read activity performed the best in both models in terms of both accuracy and $F_1$-Score, while the Debug and Interpret performed the lowest. There was an overall increase between 1% to 6% in performance across the activities (except for Play) by using the combination of low- and mid-level gaze features. For example, the Search activity was predicted with an accuracy of 85% using low-level gaze features, which further increased to 91% when combined with mid-level gaze features. A similar kind of improvement was observed in $F_1$-Scores for all the activities except for the Interpret activity.

Table 4. Classification Results for each activity. Accuracy and $F_1$-Scores for each activity class.

| Activity Class | Accuracy (%) | | $F_1$-Score | |
|---|---|---|---|---|
| | *Low-Only* | *Low+Mid* | *Low-Only* | *Low+Mid* |
| Browse | 58 | 61 (+3) | 0.63 | 0.67 (+0.04) |
| Debug | 48 | 49 (+1) | 0.50 | 0.51 (+0.01) |
| Interpret | 51 | 52 (+1) | 0.53 | 0.51 (-0.02) |
| Play | 68 | 67 (-1) | 0.72 | 0.75 (+0.03) |
| Read | 94 | 95 (+1) | 0.96 | 0.96 (+0.00) |
| Search | 85 | 91 (+6) | 0.83 | 0.87 (+0.04) |
| Watch | 86 | 91 (+5) | 0.80 | 0.82 (+0.02) |
| Write | 66 | 69 (+3) | 0.66 | 0.70 (+0.03) |

## 6.2 Feature Importance

*6.2.1 Confusion Matrices.* Figure 9 shows normalised confusion matrices for both feature sets. The diagonal of the matrices represent the proportion of correctly classified samples. Our classifier worked best for detecting the Read activity, followed by Browse (around 4-5% of the time) for both the feature sets. The classifier was also able to predict Search and Watch activities with over 85% accuracy. However, as the coding activities were found to be similar to one another, they were often misclassified among themselves. For example, the Debug activity was misclassified as Interpret 24% and 28% in the feature sets respectively. This resulted in the reduction of $F_1$-Score as previously reported (Section 6.1.2). This can be supported by the fact that while 'debugging' the code, we sometimes are required to 'interpret' the code as well. Overall, the additional mid-level gaze features reduced the misclassification rate for each activity. As an example, the Watch activity was misclassified more often when only low-level gaze features were used (e.g. with Debug and Write). Similarly, the Search activity was predicted as Browse 13% of the time with low-level gaze features, which was decreased to 8% when mid-level gaze features were added.

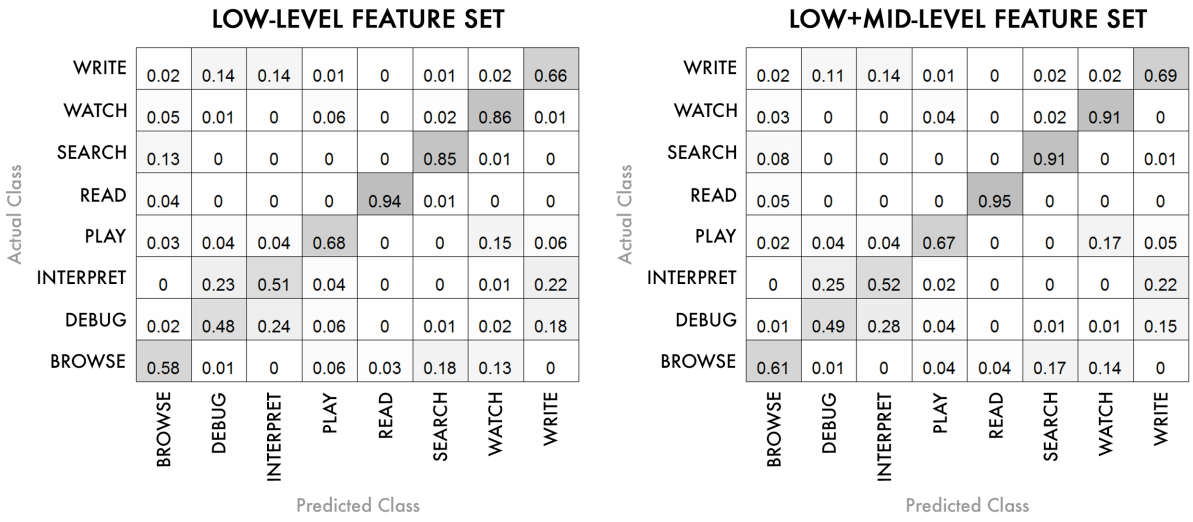


Fig. 9. Confusion Matrices for Low-level feature set and Low+Mid level feature set.

*6.2.2 Importance Table.* Table 5 lists the top-ten features according to Random Forest importance score when predicting the eight desktop-based activities using gaze features. The top-10 most important features consisted of 4 mid-level gaze features and 6 low-level gaze features with *fixation dispersion area* having the highest importance. The three saccadic direction features—*number of saccades in right direction*, *number of long-lines* and *number of strings in right direction*—are based on horizontal saccades (i.e. right direction) and received high importance. Fundamental low-level gaze features such as *fixation duration* and *saccade size* also scored high in importance, in line with prior work (e.g. [9, 27]).

Table 5. Important Features. Top ten important features across all activities based on Random Forest feature importance score.

| *Features* | *Set* | *Score* |
|---|---|---|
| Fixation dispersion area | low-level | 100.0 |
| Number of saccades in right direction | low-level | 93.1 |
| Number of long-lines | mid-level | 84.1 |
| Number of elsewhere fixations | mid-level | 62.5 |
| Number of brief fixations | low-level | 60.8 |
| Number of return fixations | mid-level | 48.3 |
| Saccade size (mean) | low-level | 47.4 |
| Fixation duration (mean) | low-level | 45.3 |
| Fixation rate | low-level | 44.7 |
| Number of strings in right direction | mid-level | 43.9 |

To further understand *why* these features predicted our desktop-based activities with high accuracy, we performed an exploratory data analysis (EDA) for our top-4 features. We concluded that the *fixation dispersion area* was helpful in distinguishing activities where the participants were asked to focus on the outcome of the task, in particular, the software engineering activities (Interpret, Debug and Write). Similarly, a large number of horizontal saccades and long lines were characteristic of the Read activity. Moreover, during free-viewing activities like Watch, Play and Browse; the *number of elsewhere fixations* were high i.e. participants tended to fixate all over the screen.

## 7 DISCUSSION

### 7.1 Overall Recognition Performance

The results on our 24-participant dataset demonstrate a strong relationship between eye-movements and everyday desktop-based activities. We found that our feature set of combined low-level gaze features and candidate mid-level gaze features (Low+Mid) sensed by an unobtrusive eye tracker were predictive of eight different activities, consisting of five common desktop activities and three software engineering activities. Our Random Forest classifier achieved a high performance using the combined feature set ($F_1$-Score = 0.72). Out of the eight activities, we were able to recognize three activities—Read, Search and Watch with high recognition performance with 91% accuracy ($F_1$-Score > 0.82). When compared to using our low-level gaze feature set (Low-Only), we were still able to recognise the activities with comparable performance ($F_1$-Score = 0.70). We believe that this performance is likely due to the nature of low-level gaze features we selected, as they were able to capture the relationship between the fixations and saccades to an extent. Specifically, we found that eight low-level gaze features were based on the interaction between the consecutive fixations and saccades (e.g. fixation dispersion area); eight features were based on saccades direction (which direction a person is looking), and two were related to the duration of the fixations (brief and long fixation). Consequently, this gave us a performance score higher than expected using only low-level gaze features. For instance, we were able to achieve classify the Read activity with a high accuracy of 94% ($F_1$-Score

= 0.96), which is a great improvement in the results reported as compared to previous work [9, 20, 28]. Additionally, we were able to increase this already high accuracy to 95% when adding our mid-level gaze features to the feature set. Closer inspection revealed that the remaining 5% of activities were misclassified with Browse, as browsing is structurally diverse, and reading can be a sub-activity within the task (e.g. browsing involves reading). These findings are consistent with previous studies, which showed that reading is the easiest to recognise as it includes characteristics sequences of several consecutive eye-movement [7, 9]. This fact gave us confidence in our results.

The addition of our candidate mid-level gaze features to the model showed an increase of 1-6% in classification performance as compared to using only low-level gaze features. Similar to the Read activity, Watch and Search activities showed similar high performance scores and obtained over 85% accuracy ($F_1$-Scores > 0.80), with an improvement of 5% and 6% when adding mid-level gaze features. We found that for activity such as Search, there were few misclassification with Browse (8%), though none with Read. The findings suggest that while searching for information, the user scans through the content (similarly to browsing), but does not thoroughly read the content. This is also evident by large number of *long-lines* found in the Read activity but not present in the Search activity. Similarly, for activity such as Watch, which is considered an unstructured activity concentrated over a relatively small field of view [9], average small *fixation-dispersion-area* and fewer *long-lines* were observed. Moreover, due to the changing visual information in every video frame, fixations were distributed all over the screen which is evident by large *number of elsewhere* fixations.

The Browse and Play activities pose a larger challenge due to the wide variety of eye movement patterns involved, and thus it is difficult to separate relevant patterns from distractions [9]. However, with our approach, we were able to detect Browse with 61% accuracy ($F_1$-Score = 0.67) and Play with 67% accuracy ($F_1$-Score = 0.75). We found that is more difficult to classify software engineering activities, specifically, the classifier struggled to distinguish Debug from Interpret. Though this is substantially higher than the chance value of 0.125, it is not high enough for an interactive system. We found that both activities contain similar eye movement patterns, and therefore were misclassified between themselves. A possible explanation is that while debugging or interpreting the output of the code, the user iterates over each line of the code to understand whether it is working. However, we found that when the participant scans each line, it generates a gaze path which is different than reading a text, searching information or browsing the web. This can be confirmed by the low (0-1%) confusion rate with Read, Search and Browse activity as shown in Figure 9. Moreover, the classifier showed a low performance in predicting Write activities ($F_1$-Score = 0.7), and falsely predicted them as Debug or Interpret 11-14% of times. Further inspection revealed that there were a large number of distraction fixations (*distract-up, distract-down, distract-left, distract-right*) for the code-based activities. These distractions can be related to either thinking, typing or disengagement; however, it is hard to distinguish.

Overall, we showed that the addition of mid-level gaze features was beneficial for recognition performance, increasing the already good recognition performance from our selected low-level gaze features. In essence, mid-level gaze features to capture the relationship between the segments of the screen, and are more closely related to the type of activity being performed than using only low-level gaze features. Our promising results demonstrate the potential of combining low- and mid-level gaze features for eye-based activity recognition for desktop-based activities as they play a prominent role in classifying. The results also suggest that we should rethink the use of features at all levels. In our work, we introduce enhancements on existing low-level features instead of relying on traditional gaze features such as *fixation-duration, saccade-duration, fixation-dispersion, saccade-length*. This gave us a high base performance, and therefore it is important to look at different abstractions, in particular, features that capture the interaction between these traditional features. Lastly, our results show that activity recognition itself is a challenging process as activities can contain a multitude of behaviours and that sub-activities can be contained in the activities; encouraging us to explore further into the research area.

## 7.2 Limitations

In this section, we declare a number of limitations of our work despite our best efforts to minimise them. First, the dataset was not entirely naturalistic. We believe that users behave differently when being observed in a lab setting as compared to when they are in the comfort of their own homes. During our data collection, we attempted to minimise the fact that the researcher is observing by having the researcher sit adjacent to the participant where they are out of sight. However, the mere presence of the researcher may have caused the participant to act differently when performing the activities as they were aware that they were being monitored. Then again, collecting naturalistic eye movements is often an ambitious goal that is certainly worth pursuing but also introduced challenges in the labelling of the dataset [10]. Moreover, they were asked to complete all the tasks within a fixed time-duration (5 minutes); irrespective of their interest or abilities. The time pressure and continuous monitoring may have affected the gaze patterns, for instance, for the reading activity, all the participant performed focused reading, in order to be able to explain the summary to the researcher. However, with no monitoring, we may have obtained different reading patterns such as skimming or scanning patterns.

Second, the shape-based and distance-based pattern features used for activity recognition were based on fixed thresholds during their extraction step. For shape-based patterns features, we have considered arbitrary values based on observation, such as a fixed length string ($l = 4$) and only in four cardinal directions, but it was possible to consider other directions as well (see Section 7.3.1). Further, we have only considered the counts for our mid-level gaze features, and have not taken into account the interaction patterns and spatial or temporal considerations i.e. when and where feature occurred during the stimulus. Third, we have also used a large window size of 105 *s* for extracting features and training our model. Although large window-size helped us to track user-behaviour habits, however for finer-grained activities such as momentary distractions within a longer activity, a shorter window size would be necessary. Additionally, we have only used three variants of each activity and assumed them as the representative examples of the whole activity. We note that is would be difficult to select a definite set of activities and as noted in related work section, there are a number of factors such a context or medium that may affect the types of gaze behaviours when presented with a stimuli. Fourth, we did not include a NULL/VOID class activity as a baseline as used in previous work (e.g. [9, 41]); therefore its effects on our model remains unknown. Lastly, we have not generalised our approach to a similar eye-tracking dataset that was collected on a different population and experimental setup (e.g. display size, eye tracker, etc.). In addition, we have not compared our performance to other approaches such as Bulling et al. [9]'s feature extraction and classification approach.

## 7.3 Future Directions

We propose two future directions for our work. The immediate direction is the continued exploration of gaze features at the mid-level by making improvements, while the long-term direction is to apply our approach to potential real-work applications which we outline in the second part of this section.

*7.3.1 Improving Mid-level Gaze Features.* The addition of our proposed set of candidate mid-level gaze features demonstrated an overall performance increase when combined with low-level gaze features for eye-based activity recognition. Upon reflecting on the features, we believe further improvements can be made. First, most of the shape-based features were derived by considering four cardinal directions (left, right, up and down). Hence, if the saccade appeared in the upper-right region, it will be either categorised as *saccade-up* or *saccade-right*. However, for activities where there are a wide variety of gaze patterns involved (e.g. Watch and Play), the saccades occurring in diagonal directions may give us more interesting insights about how people perceive those activities. Therefore, we can further consider saccades in all the eight directions in our future work, adding four new string-based pattern features—*string-up-right, string up-left, string-down-left* and *string-down-right*. Secondly, the *line-patterns* were derived from the use of English-based stimulus. However, languages such as Arabic, Japanese have reading directions different than English. To make our features generalisable, we plan to expand the *line-patterns* by

including patterns in all the directions. For instance, the pattern containing short saccades in the down direction followed by long saccade in up direction('SdSdSdSdLu') can be added to detect Japanese reading style. Lastly, the threshold used for defining types of saccade length (short, medium, long) was based on visual inspection. Similarly, for distinguishing features as *return patterns* or *elsewhere patterns*, a fixed threshold was used. However, these thresholds are dependent on the display properties of the monitor and the distance between the monitor and the participant. Therefore, computing optimal or dynamic thresholds can potentially improve our recognition performance. These are just some improvements we propose, and there are still many aspects to explore such as the addition of new features. However, the caveat of adding more features is that they might reduce classification performance, and therefore it's important to test their overall performance.

*7.3.2 Real-World Applications.* The findings in this paper present an opportunity to apply eye-based activity recognition in real-world applications. We consider three broad areas of applications: *e-learning*, *gaze-based interaction systems* and *quantified-self applications*. In e-learning, the implementation of eye-based activity recognition can enable us to monitor different activities such as Read, Watch and Write. This will enable the design of better learning systems that can adapt to the experience of the student. The interface of such applications tends to be rich. For example, the video can contain text and an area for the student to take down notes on the same screen. To be able to detect patterns will help practitioners gain an understanding of the student capabilities. Further, with gaze data, we can also identify incurred task difficulty and underlying cognitive states while the students performed the learning activity.

There is a growing interest in using eye tracking as an implicit input in interaction systems (i.e. gaze-based context-aware interactive systems) in recent years. For example, *NUIA Productivity +*[7] aims to increase productivity at the computer workstation, by replacing mouse and keyboard interaction by gaze. The application uses the combination of gaze-control and artificial intelligence based software platform which identifies activities such as scrolling longer text, selecting a link and allows more comfortable reading using gaze-click and gaze-scrolling. We also can implement eye-based activity recognition more implicitly for quantified-self as proposed by Kunze et al. [26] and the motivations behind the development of JINS MEME eyeglasses[8]. Such systems can proactively monitor daily activities and can either assist users with their daily tasks or encourage them to follow a healthy lifestyle. With our approach, we can start to consider a wider range of daily activities. The recent implementation of eye tracking in mixed-reality devices such as head-mounted displays further presents opportunities for eye-based activity recognition (e.g. *VR* [23], *AR* [49]). In a mixed-reality environment, we can easily manipulate or argument the environment to provide context and this combined with the ability to detect recognise their activity through their eye movements will enable us to explore scenarios that we were not able to explore previously. These are just some example of applications areas that can benefit from using of eye-based activity recognition.

## 8 CONCLUSIONS

In this paper, we explored the use of *mid-level gaze features*, which is a level of abstraction between low- and high-level gaze features. We collected a dataset of eye movement data containing 24 participants as they performed eight desktop-based activities, including programming activities. We obtain an overall high recognition performance score ($F_1$-Score = 0.72) using the combination of low- and mid features using a Random Forest classifier, showing a performance increase of up to 4% increase in accuracy. There was also an overall improvement in individual performance. Our results also showed an improvement in classifying the activities when we compared the low-level gaze features only with the combination; suggesting the further exploration of gaze features at the mid-level to be a potential approach for advancing the state of the art eye-based activity recognition. We have then discussed the implications of our work and present its future directions. All in all, this work provides the groundwork for

[7] https://4tiitoo.com/en/
[8] https://jins-meme.com/en/

its exploration and towards building a robust generalisable model for activity recognition, and a call to explore feature combinations to perform improved eye-based activity recognition.

## ACKNOWLEDGMENTS

Dr Eduardo Velloso is the recipient of an Australian Research Council Discovery Early Career Award (Project Number.: DE180100315) funded by the Australian Government. Namrata Srivastava and Joshua Newn are both supported by scholarships under the Australian Commonwealth Government Research Training Program. Lastly, we acknowledge Phillip McKenna for his initial contributions to the project, particularly with the software development for preprocessing and feature extraction.

## A SUPPLEMENTARY MATERIALS

### A.1 Feature Extraction Algorithm

**ALGORITHM 1:** Mid-level gaze feature extraction algorithm

**Input:** List of fixations and saccades given by $F = (t_{start}, t_{end}, f_x, f_y)$ and $Sacc = (x_{start}, y_{start}, x_{end}, y_{end})$
**Output:** The number of occurrence of shape-based patterns.

```
Saccade_sequence = ConvertSaccadesToAtoms(Sacc);
Shape_features = {StringTypes, LineTypes, CompareTypes, ScanTypes};
for each feature F in Shape_features do
    for each pattern P in feature F do
        Count[P] = 0;
        Score[P] = FindLocalMatches(P, Saccade_sequence);
        if (Score[P] > Threshold then
            Count[P] = Count[P] + 1;
        end
    end
end
ReturnPatterns = 0 ;
ElsewherePatterns = 0 ;
for each fixation F_i do
    if (distance(F_i,F_i+1) > 300) then
        if (distance(F_i+1,F_i+2) < 300) || (distance(F_i,F_i+2) < 300) then
            ReturnPatterns = ReturnPatterns + 1;
        else if (distance(F_i+1,F_i+2) > 300) || (distance(F_i,F_i+2) > 300) then
            ElsewherePatterns = ElsewherePatterns + 1;
    end
end
Distance_features = {ReturnPatterns,ElsewherePatterns};
Function FindLocalMatches (pattern,sequence)
    X = pattern ;
    Y = sequence ;
    DeltaTable = new Table(match-score, mismatch-score, gap-penalty) ;
    Score = Smith_Waterman(X,Y,DeltaTable) ;
    return Score
end
```

```
Function ConvertSaccadesToAtoms (Saccades)
    Saccade_sequence = [] ;
    for each saccade S in Saccades do
        Sacc_direction = FindDirection(S);
        Sacc_length = FindLength(S);
        if (Sacc_length < 200) then
            Sacc_type = S ;
        else if (Sacc_length < 600) then
            Sacc_type = M;
        else
            Sacc_type = L;
        Saccade_atom = Concat(Saccade_type,Saccade_direction);
        Saccade_sequence = Concat(Saccade_sequence,Saccade_atom) ;
    end
    return Saccade_sequence;
end
```

## REFERENCES


[1] Umut Akdemir, Pavan Turaga, and Rama Chellappa. 2008. An Ontology Based Approach for Activity Recognition from Video. In *Proceedings of the 16th ACM International Conference on Multimedia (MM '08)*. ACM, New York, NY, USA, 709–712. https://doi.org/10.1145/1459359.1459466

[2] Oliver Amft, Florian Wahl, Shoya Ishimaru, and Kai Kunze. 2015. Making regular eyeglasses smart. *IEEE Pervasive Computing* 14, 3 (2015), 32–43.

[3] Ong Chin Ann and Lau Bee Theng. 2014. Human activity recognition: A review. In *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*. 389–393. https://doi.org/10.1109/ICCSCE.2014.7072750

[4] Roman Bednarik, Hana Vrzakova, and Michal Hradis. 2012. What Do You Want to Do Next: A Novel Approach for Intent Prediction in Gaze-based Interaction. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. ACM, New York, NY, USA, 83–90. https://doi.org/10.1145/2168556.2168569

[5] Pieter Blignaut. 2009. Fixation Identification: The Optimum Threshold for a Dispersion Algorithm. *Attention, Perception, & Psychophysics* 71, 4 (01 May 2009), 881–895. https://doi.org/10.3758/APP.71.4.881

[6] Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors. *ACM Comput. Surv.* 46, 3, Article 33 (Jan. 2014), 33 pages. https://doi.org/10.1145/2499621

[7] Andreas Bulling, Jamie A. Ward, Hans Gellersen, and Gerhard Tröster. 2008. Robust recognition of reading activity in transit using wearable electrooculography. In *International Conference on Pervasive Computing*. Springer, 19–37.

[8] Andreas Bulling, Jamie A. Ward, Hans Gellersen, and Gerhard Tröster. 2009. Eye Movement Analysis for Activity Recognition. In *Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp '09)*. ACM, New York, NY, USA, 41–50. https://doi.org/10.1145/1620545.1620552

[9] Andreas Bulling, Jamie A. Ward, Hans Gellersen, and Gerhard Troster. 2011. Eye Movement Analysis for Activity Recognition Using Electrooculography. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 4 (April 2011), 741–753. https://doi.org/10.1109/TPAMI.2010.86

[10] Andreas Bulling, Christian Weichel, and Hans Gellersen. 2013. EyeContext: Recognition of High-level Contextual Cues from Human Visual Behaviour. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 305–308. https://doi.org/10.1145/2470654.2470697

[11] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. 2011. Analysis of Code Reading to Gain More Insight in Program Comprehension. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research (Koli Calling '11)*. ACM, New York, NY, USA, 1–9. https://doi.org/10.1145/2094131.2094133

[12] Keng-Hao Chang, Mike Y. Chen, and John Canny. 2007. Tracking Free-weight Exercises. In *Proceedings of the 9th International Conference on Ubiquitous Computing (UbiComp '07)*. Springer-Verlag, Berlin, Heidelberg, 19–37. http://dl.acm.org/citation.cfm?id=1771592.1771594

[13] He Du, Zhiwen Yu, Dong Xiao, Zhu Wang, Qi Han, and Bin Guo. 2017. Sensing Keyboard Input for Computer Activity Recognition with a Smartphone. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and*

*Proceedings of the 2017 ACM International Symposium on Wearable Computers (UbiComp '17)*. ACM, New York, NY, USA, 25–28. https://doi.org/10.1145/3123024.3123150

[14] Andrew T. Duchowski. 2007. *Eye Tracking Methodology: Theory and Practice*. Springer-Verlag, Berlin, Heidelberg.

[15] Andrew T Duchowski, Krzysztof Krejtz, Izabela Krejtz, Cezary Biele, Anna Niedzielska, Peter Kiefer, Martin Raubal, and Ioannis Giannopoulos. 2018. The Index of Pupillary Activity: Measuring Cognitive Load vis-à-vis Task Difficulty with Pupil Oscillation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 282.

[16] Richard O Duda, Peter E Hart, and David G Stork. 2012. *Pattern Classification*. John Wiley & Sons.

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*. Vol. 1. Springer series in statistics New York.

[18] John M. Henderson, James R. Brockmole, Monica S. Castelhano, and Michael Mack. 2007. Chapter 25 - Visual saliency does not account for eye movements during visual search in real-world scenes. In *Eye Movements*, Roger P.G. Van Gompel, Martin H. Fischer, Wayne S. Murray, and Robin L. Hill (Eds.). Elsevier, Oxford, 537 – III. https://doi.org/10.1016/B978-008044980-7/50027-6

[19] Shoya Ishimaru, Kensuke Hoshika, Kai Kunze, Koichi Kise, and Andreas Dengel. 2017. Towards Reading Trackers in the Wild: Detecting Reading Activities by EOG Glasses and Deep Neural Networks. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers (UbiComp '17)*. ACM, New York, NY, USA, 704–711. https://doi.org/10.1145/3123024.3129271

[20] Shoya Ishimaru, Kai Kunze, Koichi Kise, Jens Weppner, Andreas Dengel, Paul Lukowicz, and Andreas Bulling. 2014. In the Blink of an Eye: Combining Head Motion and Eye Blink Frequency for Activity Recognition with Google Glass. In *Proceedings of the 5th Augmented Human International Conference (AH '14)*. ACM, New York, NY, USA, Article 15, 4 pages. https://doi.org/10.1145/2582051.2582066

[21] Anil Jain and Douglas Zongker. 1997. Feature Selection: Evaluation, Application, and Small Sample Performance. *IEEE transactions on pattern analysis and machine intelligence* 19, 2 (1997), 153–158.

[22] Shian-Ru Ke, Hoang Le Uyen Thuc, Yong-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, and Kyoung-Ho Choi. 2013. A Review on Video-Based Human Activity Recognition. *Computers* 2, 2 (2013), 88–131. https://doi.org/10.3390/computers2020088

[23] Mohamed Khamis, Carl Oechsner, Florian Alt, and Andreas Bulling. 2018. VRpursuits: Interaction in Virtual Reality Using Smooth Pursuit Eye Movements. In *Proceedings of the 2018 International Conference on Advanced Visual Interfaces (AVI '18)*. ACM, New York, NY, USA, Article 18, 8 pages. https://doi.org/10.1145/3206505.3206522

[24] Peter Kiefer, Ioannis Giannopoulos, and Martin Raubal. 2013. Using Eye Movements to Recognize Activities on Cartographic Maps. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'13)*. ACM, New York, NY, USA, 488–491. https://doi.org/10.1145/2525314.2525467

[25] Andrea Kleinsmith and Nadia Bianchi-Berthouze. 2013. Affective Body Expression Perception and Recognition: A Survey. *IEEE Trans. Affect. Comput.* 4, 1 (Jan. 2013), 15–33. https://doi.org/10.1109/T-AFFC.2012.16

[26] K. Kunze, M. Iwamura, K. Kise, S. Uchida, and S. Omachi. 2013. Activity Recognition for the Mind: Toward a Cognitive "Quantified Self". *Computer* 46, 10 (October 2013), 105–108. https://doi.org/10.1109/MC.2013.339

[27] Kai Kunze, Yuki Shiga, Shoya Ishimaru, and Koichi Kise. 2013. Reading Activity Recognition Using an Off-the-Shelf EEG – Detecting Reading Activities and Distinguishing Genres of Documents. In *2013 12th International Conference on Document Analysis and Recognition*. 96–100. https://doi.org/10.1109/ICDAR.2013.27

[28] Kai Kunze, Yuzuko Utsumi, Yuki Shiga, Koichi Kise, and Andreas Bulling. 2013. I Know What You Are Reading: Recognition of Document Types Using Mobile Eye Tracking. In *Proceedings of the 2013 International Symposium on Wearable Computers (ISWC '13)*. ACM, New York, NY, USA, 113–116. https://doi.org/10.1145/2493988.2494354

[29] Michael Land and Benjamin Tatler. 2009. *Looking and Acting: Vision and Eye Movements in Natural Behaviour*. Oxford University Press.

[30] O. D. Lara and M. A. Labrador. 2013. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys Tutorials* 15, 3 (Third 2013), 1192–1209. https://doi.org/10.1109/SURV.2012.110112.00192

[31] AR Lauricella, DP Cingel, L Beaudoin-Ryan, MB Robb, M Saphir, and EA Wartella. 2016. The Common Sense Census: Plugged-In Parents of Tweens and Teens. *San Francisco, CA: Common Sense Media* (2016).

[32] Ziming Liu. 2005. Reading Behavior in the Digital Environment: Changes in Reading Behavior Over the Past Ten Years. *Journal of Documentation* 61, 6 (2005), 700–712. https://doi.org/10.1108/00220410510632040 arXiv:https://doi.org/10.1108/00220410510632040

[33] Chris Lonergan. 2017. Screen Time. *Sydney, Australia, Lonergan Research* (2017).

[34] Lori Lorigo, Maya Haridasan, Hrönn Brynjarsdóttir, Ling Xia, Thorsten Joachims, Geri Gay, Laura Granka, Fabio Pellacini, and Bing Pan. 2008. Eye tracking and Online Search: Lessons Learned and Challenges Ahead. *Journal of the American Society for Information Science and Technology* 59, 7 (2008), 1041–1052.

[35] Päivi Majaranta and Andreas Bulling. 2014. *Eye Tracking and Eye-Based Human–Computer Interaction*. Springer London, London, 39–65. https://doi.org/10.1007/978-1-4471-6392-3_3

[36] Andrea Mannini, Stephen S Intille, Mary Rosenberger, Angelo M Sabatini, and William Haskell. 2013. Activity Recognition using a Single Accelerometer Placed at the Wrist or Ankle. *Medicine and Science in Sports and Exercise* 45, 11 (2013), 2193.

[37] Marcus Nyström and Kenneth Holmqvist. 2010. An Adaptive Algorithm for Fixation, Saccade, and Glissade detection in Eyetracking Data. *Behavior Research Methods* 42, 1 (01 Feb 2010), 188–204. https://doi.org/10.3758/BRM.42.1.188

[38] Unaizah Obaidellah, Mohammed Al Haek, and Peter C.-H. Cheng. 2018. A Survey on the Usage of Eye-Tracking in Computer Programming. *ACM Comput. Surv.* 51, 1, Article 5 (Jan. 2018), 58 pages. https://doi.org/10.1145/3145904

[39] Megan K. O'Brien, Chaithanya K. Mummidisetty, Xiao Bo, Christian Poellabauer, and Arun Jayaraman. 2017. Quantifying Community Mobility After Stroke Using Mobile Phone Technology. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers (UbiComp '17)*. ACM, New York, NY, USA, 161–164. https://doi.org/10.1145/3123024.3123085

[40] Ofcom. 2017. Adult's Media Use and Attitudes. *Report* (2017).

[41] Keisuke Ogaki, Kris M Kitani, Yusuke Sugano, and Yoichi Sato. 2012. Coupling Eye-motion and Ego-motion Features for First-person Activity Recognition. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 1–7. https://doi.org/10.1109/CVPRW.2012.6239188

[42] Pontus Olsson. 2007. Real-time and Offline Filters for Eye Tracking.

[43] Sarunas J Raudys and Anil K. Jain. 1991. Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 3 (1991), 252–264.

[44] Keith Rayner. 1998. Eye movements in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin* (1998), 372–422.

[45] C. J. Van Rijsbergen. 1979. *Information Retrieval* (2nd ed.). Butterworth-Heinemann, Newton, MA, USA.

[46] Bonita Sharif, Michael Falcone, and Jonathan I. Maletic. 2012. An Eye-tracking Study on the Role of Scan Time in Finding Source Code Defects. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. ACM, New York, NY, USA, 381–384. https://doi.org/10.1145/2168556.2168642

[47] T.F. Smith and M.S. Waterman. 1981. Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147, 1 (1981), 195 – 197. https://doi.org/10.1016/0022-2836(81)90087-5

[48] Julian Steil and Andreas Bulling. 2015. Discovery of Everyday Human Activities from Long-term Visual Behaviour Using Topic Models. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. ACM, New York, NY, USA, 75–85. https://doi.org/10.1145/2750858.2807520

[49] Hidde van der Meulen, Andrew L. Kun, and Orit Shaer. 2017. What Are We Missing?: Adding Eye-Tracking to the HoloLens to Improve Gaze Estimation Accuracy. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 396–400. https://doi.org/10.1145/3132272.3132278

[50] Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2011. Towards Qualitative Assessment of Weight Lifting Exercises Using Body-worn Sensors. In *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*. ACM, New York, NY, USA, 587–588. https://doi.org/10.1145/2030112.2030226

[51] Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2013. AutoBAP: Automatic Coding of Body Action and Posture Units from Wearable Sensors. In *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII '13)*. IEEE Computer Society, Washington, DC, USA, 135–140. https://doi.org/10.1109/ACII.2013.29

[52] Eduardo Velloso and Marcus Carter. 2016. The Emergence of EyePlay: A Survey of Eye Interaction in Games. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '16)*. ACM, New York, NY, USA, 171–185. https://doi.org/10.1145/2967934.2968084

[53] Eduardo Velloso, Amy Fleming, Jason Alexander, and Hans Gellersen. 2015. Gaze-Supported Gaming: MAGIC Techniques for First Person Shooters. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '15)*. ACM, New York, NY, USA, 343–347. https://doi.org/10.1145/2793107.2793137

[54] Shuo Wang and Xin Yao. 2012. Multiclass Imbalance Problems: Analysis and Potential Solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42, 4 (Aug 2012), 1119–1130. https://doi.org/10.1109/TSMCB.2012.2187280

[55] Alfred L. Yarbus. 1967. *Eye Movements During Perception of Complex Objects*. Springer US, Boston, MA, 171–211. https://doi.org/10.1007/978-1-4899-5379-7_8

[56] Yang Zhao, Zicheng Liu, Lu Yang, and Hong Cheng. 2012. Combing RGB and Depth Map Features for Human Activity Recognition. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*. 1–4.

Received May 2018; revised August 2018; accepted October 2018