Identifying Lines and Interpreting Vertical Jumps in Eye Tracking Studies of Reading Text and Code

Mor Shamy Dror G. Feitelson

Department of Computer Science
The Hebrew University of Jerusalem, 91904 Jerusalem, Israel

***Abstract*—Eye tracking studies have shown that reading code, in contradistinction to reading text, includes many vertical jumps. As different lines of code may have very different functions (e.g. variable definition, flow control, or computation) it is important to accurately identify the lines being read. We design experiments that require a specific line of text to be scrutinized. Using the distribution of gazes around this line, we then calculate how the precision with which we can identify the line being read depends on the font size and spacing. The results indicate that, even after correcting for systematic bias, unnaturally large fonts and spacing may be required for reliable line identification.**

Interestingly, during the experiments the participants also repeatedly re-checked their task and that they are looking at the correct line, leading to vertical jumps similar to those observed when reading code. This suggests that observed reading patterns may be "inefficient", in the sense that participants feel the need to repeat actions beyond the minimal number apparently required for the task. This may have implications regarding the interpretation of reading patterns. In particular, reading does not reflect only the extraction of information from the text or code. Rather, reading patterns may also reflect other types of activities, such as getting a general orientation, and searching for specific locations in the context of performing a particular task.

***Index Terms*—Eye tracking, reading order, behavior model**

1. Introduction

Eye trackers are increasingly being used by researchers in software engineering to gain insights into what developers—or experiment participants—are actually doing and thinking [3, 19, 33, 42, 47, 48]. A recurring result is that code, unlike natural language texts, is not read in a largely linear fashion. Rather, participants seem to repeatedly scan the code, and to jump from one place to another.

When analyzing reading behavior, it is important to correctly identify exactly what part of the code the participants are looking at at each instant. We set out to perform a methodological study of how accurately this can be done. Specifically, we were interested in the trade-off between realism (trying to be as close as possible to normal working conditions, with many coding elements crowded on the screen) and accuracy (which can be improved by using unrealistically large fonts and spacing). We did this by designing experiments that require participants to focus on only one specific line of text. As we know exactly where they are supposed to be looking, we can analyze the distribution of gazes around this target, and calculate the probability of misidentifying it.

Interestingly, these simple experiments led to reading patterns similar to those observed for code. In particular, the participants often jumped from the target line to the question specifying what they were supposed to look for, and back again. In other words, their reading appeared to be inefficient, repeatedly focusing on things that they had just focused on previously (similarly to the repeated scanning sometimes seen in code reading [13, 20, 53]). This suggests that the patterns observed when people perform a task can be inherently different from the patterns they use when reading. When reading, one needs to go over the whole text, which is presented in the order in which it should be read. So reading text is largely linear in "story order". But to perform a given task *which is different from just reading the information in the text* other orders may be used. In particular, the order may reflect the need to *orient* oneself in the text, and to *search* for the parts which are relevant for the task. In our case this sometimes included repeated verification of the line involved in the task and the precise details of the task. We also observe that participants probably use their peripheral vision to home in on targets with visual cues.

We therefore suggest "task order" reading, specifically including orientation and search activities, to generalize the "story order" and "execution order" previously suggested by Busjahn et al. [8]. In addition, we suggest that simple experiments such as ours can be embedded into larger experimental settings, both as a simple recalibration to compensate for drift during an experiment, and to provide objective exclusion criteria for participants who achieve low accuracy.

2. Background and Motivation

Eye tracking has been used in reading research for many years. Extremely detailed information about reading is now known, including the distribution of saccade lengths (the distance the eye moves from one fixation to the next), the perceptual span (how much can be seen in a single fixation, typically about 8 letters), and the prevalence of regressions (looking back at previously read text, around 10–15% of saccades) [6, 39, 40]. Importantly, such attributes of eye movement change to reflect the difficulty of the text being read [41].

Eye tracking was first used to study code reading by Crosby and Stelovsky [13]. One of their findings was that reading code often involved multiple scans of the code. In addition, they found wide differences between participants—for example, employing different numbers of scans. Both findings indicate a departure from reading in a largely linear order, as one may expect for, say, a newspaper story. Subsequent studies have also found that reading code is different from reading text [5,

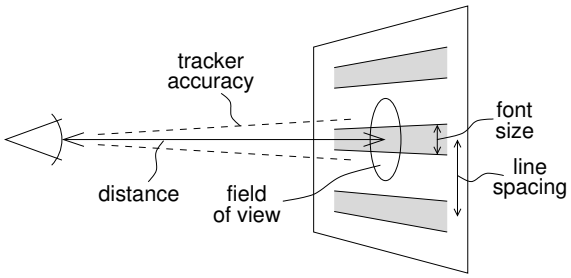

Fig. 1. Factors affecting the ability to correctly identify the line of text being read.

10, 43]. For one thing, reading code is slower, perhaps due to its complexity. In addition, different code elements receive different levels of attention, depending on their function and location [36]: for example, more emphasis is given to identifiers relative to keywords [9].

Significant research has been focused on the issue of reading order. Regressions in reading natural-language text typically involve the previously fixated word, and rarely go to previous lines [6]. But practically all studies of code reading have noticed vertical scans of the code as identified by Crosby and Stelovsky. According to Uwano et al., such scans are a prerequisite for efficient finding of locations of interest in the code [53]. Busjahn et al. found that the reading order also depends on experience: reading by experts is less linear than reading by novices [8]. This result was replicated by Peitek et al. [35]. In addition to scans, a whole catalog of reading patterns has been identified, including jumps back to previously read parts of the code, and jumps forward to preview what is coming [10, 20]. Sharafi et al. show that different reading patterns are used in distinct phases of performing a task, for example when looking for relevant code locations, learning about the code, and editing it [46].

It is important to note that in computer program code different lines often have completely different functions: a variable declaration, an assignment, a branching instruction, a function header, etc. Identifying exactly which line is being read is therefore extremely important in order to understand what the reader is doing. The ability to distinguish between reading of adjacent lines in eye tracking studies depends on the interaction of four important factors (Fig. 1) [32]:

1) The angular discrimination of the eye tracker. Most trackers claim an accuracy of 0.5–1°.
2) The distance to the screen, which translates the angular discrimination into an uncertainty regarding the location on the screen.
3) The characteristics of the target line, and especially the line spacing used (which might relate to the font size). This dictates the differences in screen locations that must be discriminated.
4) The vertical deviation between the center of the target line and where the eye is actually focused. The high-resolution center of the field of view, corresponding to the fovea in the eye's retina, is about 2°, which may span more than one line [39, 51]. It is therefore not really necessary to focus exactly on the target line to read it. In addition, deviations may arise from small movements and drift during a fixation [39].

In addition, measurements always include noise. Data from reading 25-line pages of text, which were shown line-by-line to obtain a ground truth for the assignment of fixations to lines, indicates that the combined view paths exhibit erratic behavior and many overlaps [7].

To improve tracking accuracy, experimental recommendations typically suggest that the font used be "large enough" to provide good discrimination, say around a size of 18 pt [47]; one recent study on the parsing of URLs even used a font size of 64(!) [38]. But even 18 pt is larger than what developers usually use in routine work. This reduces the realism of the study, and also limits the amount of code that can be displayed in one screen.

If the font and spacing are not large enough, studies will exhibit data quality problems [16]. These sometimes cause uncertain assignments of fixations to lines, leading to apparent fluctuations between adjacent lines (see, for example, [5, 20, 53]). The detrimental effects of such problems have been recognized previously also in the context of reading natural language texts, and a number of studies have specifically considered the issue of assigning fixations to lines [7, 11, 30, 49, 58]. However, the developed heuristics are based, at least to some degree, on the assumption that the text consists of lines that are read sequentially. This assumption is often not valid in the context of reading code.

Our initial goal was to study this problem more systematically. We set out to design experiments in which we can measure the vertical distribution of gaze points, as measured by an eye tracker, around a known target. This would enable a precise calculation of the probability to misinterpret the line being looked at as a function of the font size and line spacing. But the results also included unanticipated reading patterns with vertical jumps. These allow us to make some observations on why reading code may be different from reading normal text.

3. Line Focus Experiments

As noted above, under normal viewing conditions the vertical resolution may not be adequate to correctly and consistently identify the line being read. This obviously depends on the font size and line spacing. Our experiments were designed to systematically study this effect in practice. The approach was to design tasks that require experiment participants to focus on one specific line in the text. We then collect data on how they find this line, and on how focused they appear to be, given the experimental apparatus and the parameters of displaying the text. We do this in a real experimental setting, as opposed to using an artificial eye [16], as our goal is to characterize what can be achieved in practice and not to assess an eye tracking apparatus.

3.1. Texts and Tasks

Eight English texts were used. They come from different categories, such as songs ("Yesterday"), plays ("All the world's a stage") and short descriptions of nature or leaders around the world. All texts were 10 or 11 lines long.

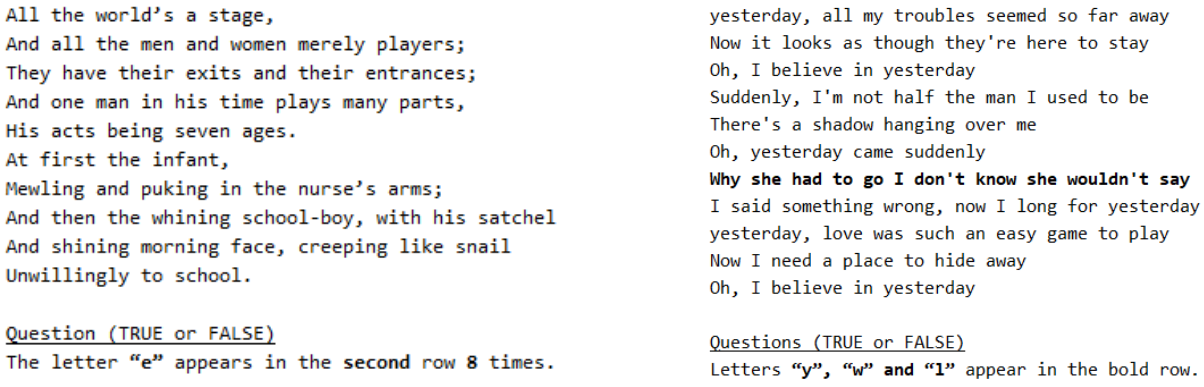

Fig. 2. Examples of texts and questions used in the experiment, as shown to participants (except for scale). Note that lines are not numbered.

The texts were displayed in different ways:

- Font size – four texts differed in their font sizes: 10 pt, 13 pt, 16 pt, and 20 pt. All had a standard line spacing of 1.2.
- Spacing – four texts with a standard font size of 13 pt differed in their spacing: 1, 1.2, 1.5, and 2.
- Text and background colors – either black lettering on a white background or the other way around. A dark background is considered better for eye tracking, and is not uncommon also in programming environments.

The first two variations form the heart of our study. Given that we know where participants are supposed to look, we can see what fraction of fixations is indeed assigned to the correct line, and how many are mis-assigned to adjacent lines.

Two tasks were used to make participants focus on a specific line:

- Count the number of times a given letter appears in the line.
- Verify whether a set of three letters all appear in the line.

The question appeared after the text. A couple of examples are given in Fig. 2.

We decided not to use questions regarding the meaning of the texts, even if they are about information contained in a specific line. The reason was that participants might still read other lines, thinking they may be relevant. In addition, participants may answer from memory without even looking at the target line. This is not expected to happen for questions about letters that appear in the line.

The target line was specified in either of two ways:

- A line number. We chose either the second line or the seventh line of the text. The reason for choosing them was that the second line can be identified at a glance, but the seventh most probably requires counting. We avoided the first and last lines so that adjacent lines would exist in both directions. Note that the text lines were not numbered—the participants had to count to find the seventh line, but could probably guess the second line.
- Using visual cues to identify the line. Two alternative cues were used: being indented, or being set in **boldface**. The goal was to see whether participants would be able to home in on the target line rapidly (presumably using their peripheral vision) without scanning all the lines.

TABLE I
COMBINATIONS OF PARAMETERS USED IN THE EXPERIMENTAL PLAN.

No.	Text	Size	Space	target Line	target Indication	Look for
1	Shakespeare quote	10 pt	1.2	2	number	e × 8
2	Yesterday (Beatles)	13 pt	1.2	7	bold	y, w, l
3	Facts about giraffes	16 pt	1.2	7	indent	f × 4
4	History of Yemen	20 pt	1.2	7	number	n, p, k
5	Fidel Castro bio	13 pt	1	2	number	a × 4
6	Facts about stars	13 pt	1.2	6	indent	s, c, b
7	Bear in Vancouver	13 pt	1.5	7	bold	e × 6
8	Facts about sharks	13 pt	2	7	number	w, g, d

To enable fair comparisons, the target line indicated by the visual cues was the sixth or seventh line, similar to the case of counting down to the seventh line—that is around $\frac{2}{3}$ of the way down the text. But of course the participants did not know that the targets are always in the same area. This was further obscured by the two texts where the target was the second line.

The experimental plan with the combinations that were used is described in Table I. To reduce variability all the experiment participants received the same 8 texts with the same attributes, so this is not a full factorial design. In particular, we do not consider the text itself to be a relevant factor. However, the order of the texts was randomized so as to avoid systematic fatigue and learning effects. Each participant was presented with one of the coloring options: either all texts appeared in black on a white background, or the other way around.

3.2. Experimental Setup

The experiments were conducted using a Gazepoint GP3 eye tracker operating at 60 Hz. This is a remote eye tracker [31], which is typically placed below the display used in the experiment. It employs an infra-red light source and a video camera focused on the face of the experiment participant. By analyzing the reflection of the light source from the cornea of the participant's eyes the tracker can estimate where the participant is looking. This analysis is parameterized by a calibration procedure executed before the experiment begins, in which the participant is asked to look at various points on the screen (typically 5 or 9 points).

Fig. 3. Experimental setup. The eye tracker is positioned below the display.

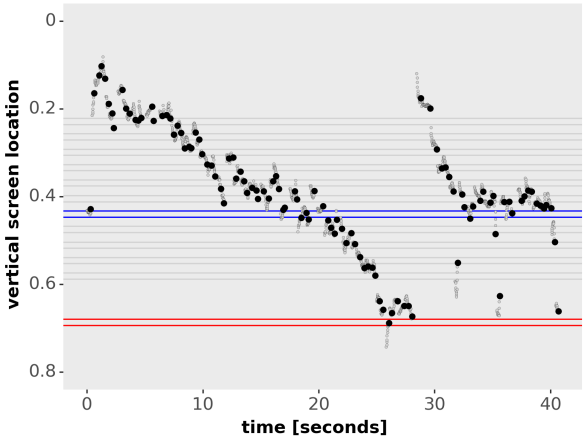

Fig. 4. Example of participant gaze results, showing vertical position (in screen dimension from 0 at the top to 1 at the bottom) as a function of time. Raw gaze data is in gray, and estimated fixations in black. The locations of lines of text are indicated by pairs of lines, the bottom one being the baseline and the other indicating the x-height (that is, the top of a lowercase 'x'). Ascenders (as in 'f', 'h', etc.) go above the x-height, and descenders (as in 'g' or 'y') hang below the baseline. The sequence of fixations shows the reading of the text (indicated by the gray lines), followed by reading the question (red lines), counting the text lines from the top, and focusing on the target line (the seventh line, in blue). [participant 8, text 4 – 20 pt at 1.2 spacing]

While the GP3 is one of the most low-cost remote eye trackers available today, a recent study shows that its performance is comparable to that of more expensive eye trackers [50]; previous work has also found no appreciable advantage to more costly trackers [49].

In our setup the eye tracker was positioned just below a 24" Dell flatpanel P2419H LED-based display with a resolution of 1920×1080 pixels. The texts were presented at the center of the screen, in fullscreen mode. There were no other distracting elements on the screen. This setup is shown in Fig. 3.

Data exported from the tracker included (X,Y) locations, pupil diameter, and validity for each eye. A separate output contained fixations computed from the raw data by the Gazepoint software.

The experiment was conducted at a desk situated far from the window, at a right angle, and behind an open-space partition, to reduce ambient light from outside and prevent it from interfering with the eye tracking. We checked the option of performing the experiments in the evening when it was dark outside, but saw no significant effect. Participants were seated in a wheel-less chair to reduce movement.

3.3. Experiment Execution

The participants were 17 students from the Hebrew University, 11 females and 6 males. The average age was 24.5 years (SD = 1.54). They were paid 20 Shekels for their participation. The whole procedure (including obtaining consent, explanations, and calibration) took about 15–20 minutes.

The experimenter initially gave a general overview about the experiment and the eye tracker. Participants were told that the experiment is about testing the reliability of the eye tracker device.

On average, participants sat at a distance of 69 cm from the screen (range of 55–85 cm). After confirming that they were sitting comfortably, they were asked to perform a nine-points calibration. We decided in advance that only results of participants who achieved a perfect score (9/9) in both eyes, in two tries, will be considered. Luckily, all 17 participants met this criterion.

After the calibration, the eight texts appeared in a random order one after another. Participants were instructed to read the text and then to answer the question. Answers were given verbally, so no physical touch with mouse or keyboard was needed. After an answer was received, the experimenter moved to the next text.

One participant was disqualified because she moved toward the screen to read better during the experiment. Two of the results of another participant were disqualified for the same reason.

4. Data Cleaning and Adjustment

The collected data required some cleaning and adjustments before it could be used. All our analysis is based on the average of data from the left and right eyes [17]. As noted above, fixations were computed by the tracker software. We observed no difference between the color options, so the following presentation is based on both together and we do not distinguish between them.

We plotted the gaze and fixation data relative to the lines of text, using time for the horizontal axis and the screen Y coordinate as the vertical axis. In this display the screen X coordinate is not shown, which matches our focus on the vertical dimension. An example is shown in Fig. 4.

Initial observations revealed that some data was extremely noisy, and no period of focus on the target line could be identified (Fig. 5). Both authors independently reviewed the graphs depicting each participant/text combination, and judged which were unusable. There was complete agreement on excluding one participant altogether. Apparently this participant had narrow eyes so the pupils were partly obscured, and the tracker could not identify them reliably; perhaps she was squinting. In addition, six specific texts of other participants were also excluded. While this exclusion rate is high, it is not

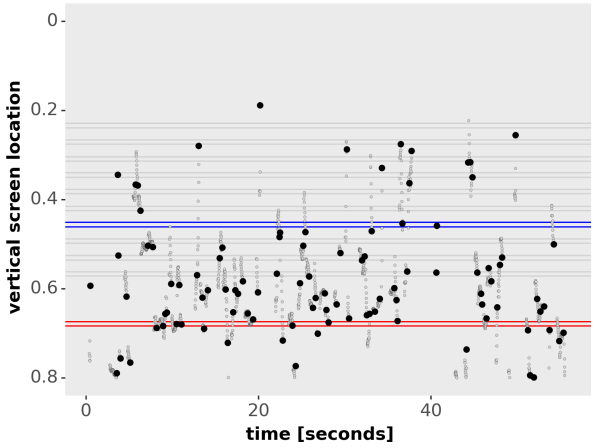

Fig. 5. Example of noisy results that were excluded, as no focus on the target line could be identified. [participant 9, text 8 – 13 pt at 2.0 spacing]

unprecedented, and much higher exclusion rates of up to 60% have been reported in eye tracking studies [32].

The heart of the experiment is to measure the distribution of gazes when the participants are supposed to be focusing on the target line. But participants also spend time reading the question and finding the target line. To identify the time period when they are focused on the target line we recruited an external judge. This judge had previously participated in the experiment, so she had first-hand experience with it and understood the meaning of focusing on the target line. The first author and this judge independently viewed the graphs of all texts for all participants, and noted when they thought the participant focused on the target line. In most cases they agreed to within 3 seconds, which we consider to be an immaterial difference. There were 11 cases of larger disagreements; in these cases the second author acted as arbiter.

Given the identification of periods in which it was believed that the participants were focused on the target line, it was obvious that many of the observations suffered from systematic bias. For most participants, the fixations appeared to be located above the target line, sometimes by a considerable margin. For a few, the fixations appeared below the line. We note that this is not unusual, and such bias has also been noticed in the context of developing the iTrace tool for eye tracking when using an IDE [45]. Palmer and Sharif have suggested a methodology for the automatic correction of such a bias, based on creating clusters of fixations, and trying to adjust them so that they fit likely areas of interest in the stimuli [34]. Other algorithms have also been proposed [11, 49, 58].

As we know exactly where the participants are supposed to be looking, we can use a simple and effective strategy. For each displayed text, we find the median height of all the fixations in the focus period, and compute the deviation of this height from the center of the target line—similar in principle to the task-embedded calibration suggested by Pi and Shi [37]. We then average these deviations for all the texts displayed to each participant. This produces a participant-specific correction that can be applied to all texts. An example is shown in Fig. 6.

We elected to make the correction per participant, because it was obvious that different participants had different biases, which could be the result of their individual physiology or posture. The alternative would be to make a separate correction for each text, but it is less obvious why there could be different biases for different texts displayed one after the other using the same device in the same session.

After correcting the systematic bias, we can finally study the distribution of fixations around the target line.

5. Results

Our main goal from the outset was to characterize the distribution of fixations around the target line. We want to know, during the periods when the experiment's participants are focused on the target lines, what fraction of the fixations indeed identify the target line.

5.1. Deviations from the Focus

The vertical distribution of fixations during the focus period can be visualized using an ECDF (empirical cumulative distribution function) of the screen Y coordinate of these fixations, as shown in Fig. 7. The first panel shows all the fixations in the experiment and the identified focus period (delimited by the two vertical lines). The second panel shows the ECDF. As one can see, many of the fixations are indeed distributed around the target line, although a few are actually closer to or on adjacent lines. In addition, a few deviate to the question line. It seems that the participant took a couple of quick looks at the question during the focus period, to make sure he is counting the right thing.

This behavior is not unique to this example. Given the simplicity of the questions (count appearances of a letter, or verify that three letters appear at least once) we expected participants to focus exclusively on the target line to answer them. But surprisingly, in many cases they interrupted the execution of the task to re-check the question. In a few cases they also re-checked the line they were looking at. As a result the distribution of fixations could be bimodal or stretched, in a way that does not represent focusing on the target line.

To counter such behavior, in the following analysis we exclude fixations to locations below the bottommost line. This filters out instances of glancing down at the question. We return to this behavior in Sect. 5.3 below.

5.2. Distribution of Focus

Given the data on fixations on the target line, we can analyze the effect of font size and spacing on our ability to identify the line being read. We start by defining a fixation during the focus time as "indicative" if it can be assigned to the target line, namely if the fixation's Y coordinate is closer to the center of the target line than to any other line. In other words, indicative fixations are those that fall between two imaginary lines: one half way between the center of the target line and the center of the line above it, and the other half way between the center of the target line and the center of the line below it.

We use this definition to find the fraction of fixations that are indicative as a function of the text parameters. The results are shown in Fig. 8. Interestingly, the fraction of indicative

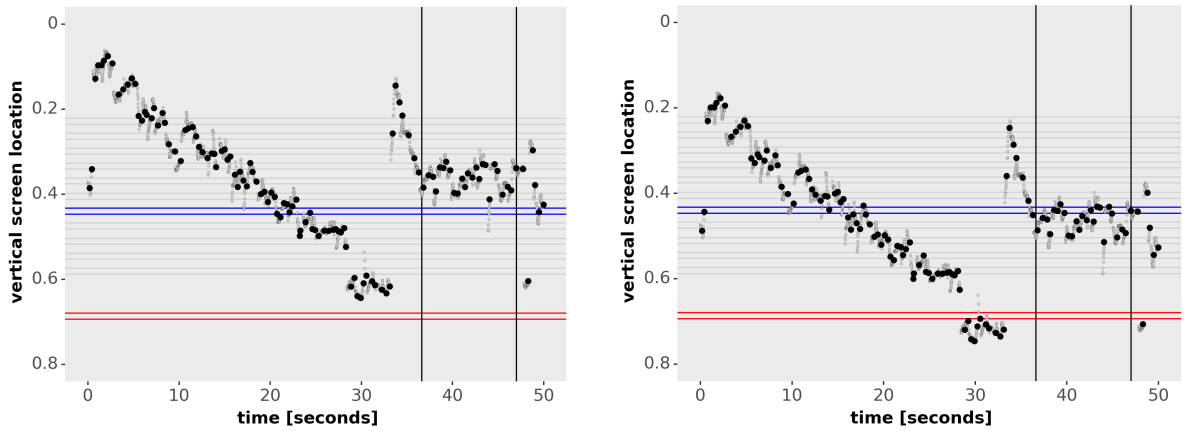

Fig. 6. Example of correction of bias (left: original; right: after correction of shifting down by 0.1024). The identified focus period is marked by vertical lines. Note that the correction is based on an average of all the texts viewed by the participant, so it may be sub-optimal for any specific text. [participant 6, text 4 – 20 pt at 1.2 spacing]

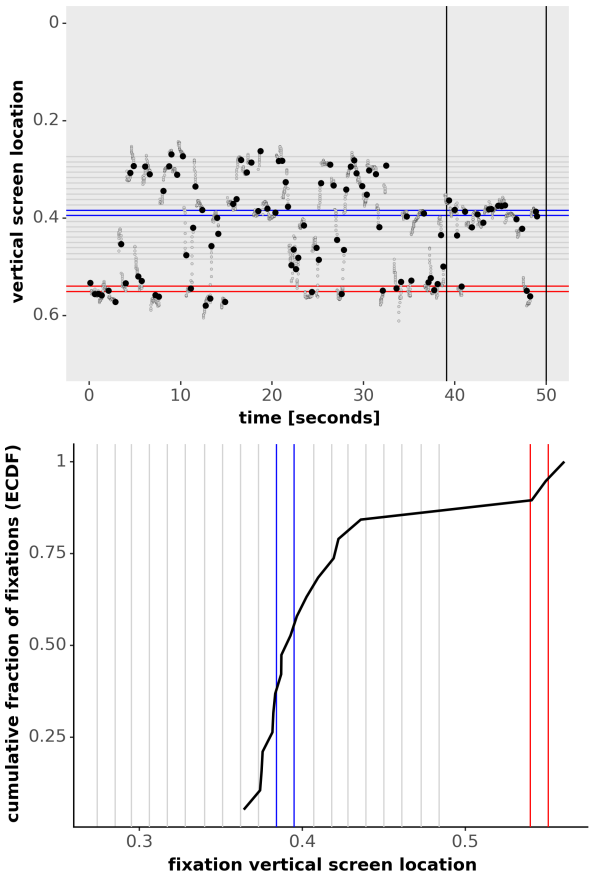

Fig. 7. Example of the distribution of fixation heights during the focus period (seconds 39–50). ECDF: empirical cumulative distribution function. [participant 13, text 6 – 13 pt at 1.2 spacing]

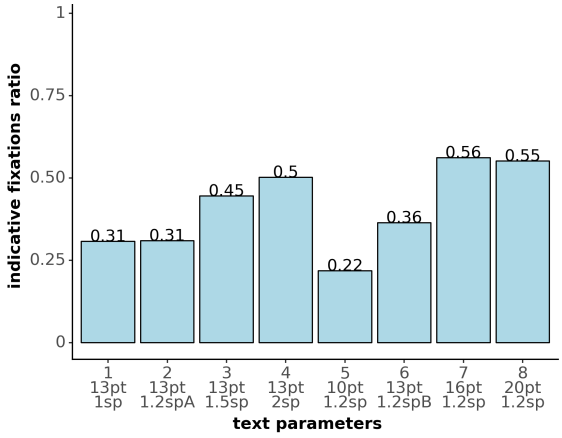

Fig. 8. Effect of display parameters on how many of the fixations are identified as focusing on the target line (indicative fixations).

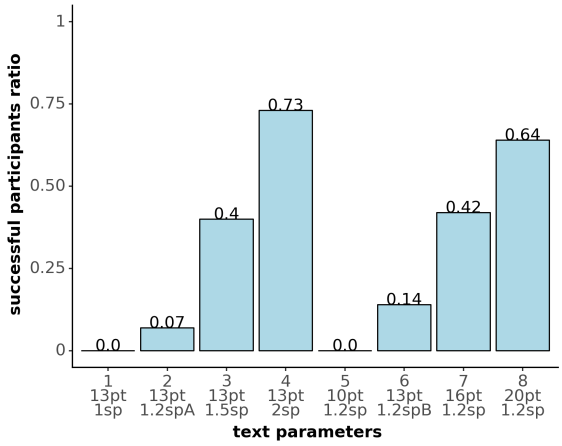

Fig. 9. Effect of display parameters on how many participants had at least 50% of their fixations on the target line (indicative fixations).

fixations does not exceed 56% even for the most permissive settings checked—double space or a 20 pt font.

The implication is that the fixations are distributed quite widely around the target line. Looking at the plots depicting the fixations in each case, we find that some of these deviations represent random glances at other places or an uncorrected systematic bias. But others just reflect a wide distribution which includes adjacent lines.

As these problems were more typical of some experiment participants than others, we perform a second analysis in

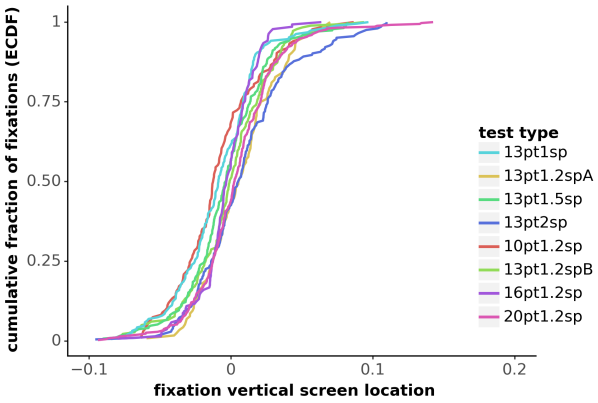

Fig. 10. Distributions of deviations from the target line based on all participants.

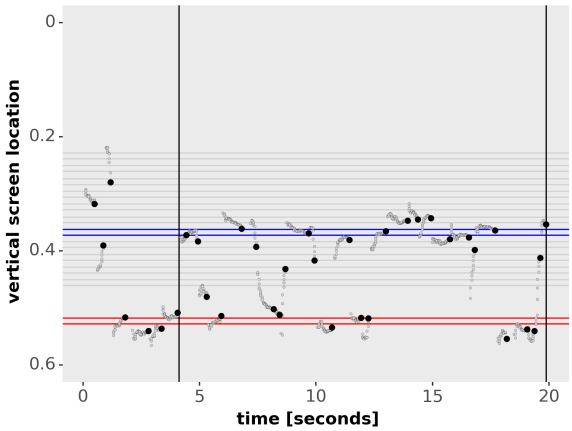

Fig. 11. Example of multiple regressions to the question line. [participant 12, text 2 – 13 pt at 1.2 spacing]

which we find the fraction of experimental participants for whom the majority of fixations are indicative (i.e. at least 50%). The results are shown in Fig. 9. As we can see, when the font or the spacing are too small *none* of the experiment's participants meet this criterion. Using near-conventional viewing conditions (text size of 13 pt with 1.2 spacing) only about 10% of the participants meet it. The majority of experimental participants meet the criterion only with the largest font (20 pt) or the largest spacing (double space).

These poor results are in line with those of Špakov et al. [49], which is the only other analysis of this kind of which we are aware. In their study all the participants had 90% correct identification only when the screen was divided into just 2 horizontal bands. With 10 bands 77 pixels high each this dropped to around a third of the participants.

Another way to look at the results is to check whether the distribution of fixations depends on the text parameters. In other words, when the font and spacing are large, do participants "allow themselves" to be more lax, and let their eyes wander farther away from the center of the target line?

To check this we create combined ECDFs of the fixation Y values across the participants. Thus we get 8 lines, one for each experimental setting, each of which includes all of the results for all participants. The results are shown in Fig. 10. These distributions are all centered on the target line, which is expected given that we correct for bias. But in addition, the distributions are all rather similar to each other in terms of spread. This leads to the conclusion that the distribution of gazes is not affected by the font and spacing settings.

5.3. Regressions to the Question

Looking back at text you have already read is called a "regression" in eye tracking reading studies. For normal text this may refer to lines above the current one. In our experiments, a common form of regression was looking again at the question. As the question appeared at the bottom of the screen, these are regressions to a line *below* the current line.

Jumping back to the question was most probably done for verification, and may reflect the limits of working memory capacity [12]. We identified these regressions as sequences of consecutive fixations that are below the bottommost line.

TABLE II
REGRESSIONS TO THE QUESTION LINE DURING FOCUSING ON THE TARGET LINE. (SEM=STANDARD ERROR OF THE MEAN)

Question type	n	Regressions avg±sem
Count letter appearances	54	0.44±0.08
Verify 3 letters exist	58	1.50±0.25

For example, at the extreme right end of Fig. 11 there are 3 consecutive fixations on the question, that count as one regression. In the experiments, such regressions occurred in exactly half of the cases. In some cases not one but several such regressions occurred (up to 5, Fig. 11).

Interestingly, a clear distinction was found between question types (Tab. II). When participants needed to check the presence of three different letters, there were more than 3 times as many regressions on average. This corroborates the conjecture that the regressions are related to a need to verify that the correct letters are being counted.

5.4. Finding the Target Line

In many cases it appeared that participants found the target line effortlessly: They read the question, and immediately moved to the target line—presumably being able to identify it using their peripheral vision [21]. But when directed to the seventh line by number, as opposed to by a visual cue, the fixation pattern usually indicated that they moved to the first line and then counted the lines. However, the number of fixations on the way was typically less than 7: The field of view was large enough to count without fixating on every line.

Also, similarly to the regressions to the question, in many cases the participants counted more than once (Fig. 12). Fig. 13 summarizes the average number of times that the lines above target line were scanned. Recall that when visual cues were used (target line set in boldface or indented) the target line was also the 6th or 7th line, so this is a fair comparison. Cases where the target line was the second line are excluded, because it was impossible to decide whether any counting took place. Counting to the seventh line was done at least once and often more, leading to an average of 1.24 times. Indented lines

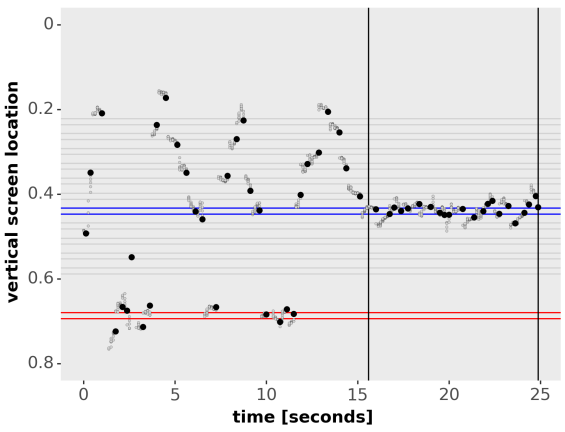

Fig. 12. Example of counting lines multiple times, presumably to ascertain the target line. [participant 13, text 4 – 20 pt at 1.2 spacing]

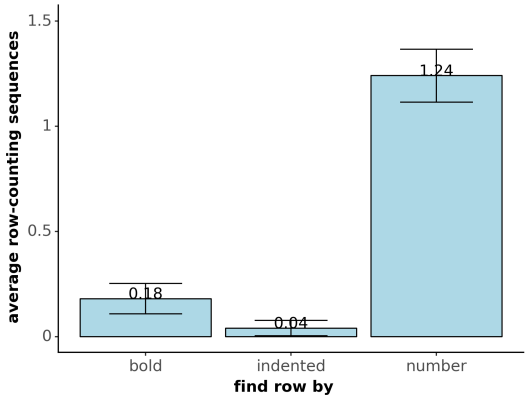

Fig. 13. Participants counted lines on average more than once when they needed to get to the seventh line, but seldom scanned previous lines when the target line was identified by a visual cue. Error bars indicate standard error of the mean.

TABLE III
ANGLE BETWEEN ADJACENT LINES FOR DIFFERENT EXPERIMENTAL SETUPS. LINE PITCH WAS MEASURED DIRECTLY ON THE SCREEN.

Size	Space	Line pitch	° at distance 60cm	75cm
10 pt	1.2	0.49cm	0.47°	0.37°
13 pt	1.2	0.66cm	0.63°	0.50°
16 pt	1.2	0.77cm	0.73°	0.59°
20 pt	1.2	1.04cm	0.99°	0.79°
13 pt	1.0	0.54cm	0.52°	0.42°
13 pt	1.5	0.82cm	0.79°	0.63°
13 pt	2.0	1.10cm	1.05°	0.84°

were apparently the easiest to identity, and there was only one (questionable) case of looking at previous lines.

6. Discussion

6.1. Assigning Fixations to Text

We set out to measure the accuracy with which fixations can be assigned to lines of text. The results reported in Sect. 5.2 are rather disappointing: in most cases the discrimination between adjacent lines was quite poor. This was not so unexpected, as quantified in Table III. We chose to study font sizes and line spacings that are not too far removed from those used in conventional work conditions. But with these settings the

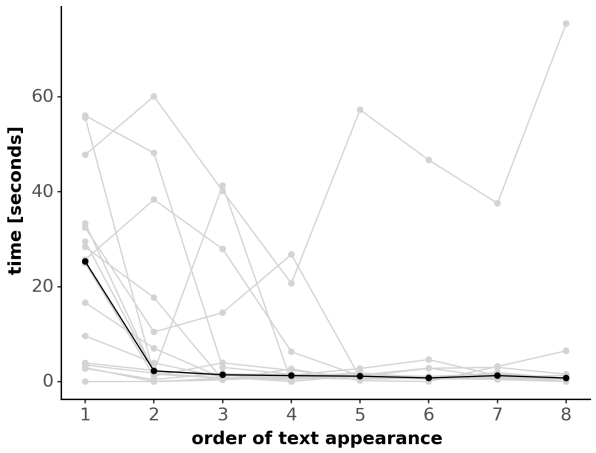

Fig. 14. Time till participants started to read the question, as a function of text serial number. Each line represents one participant in the experiment. The dark line is the median.

pitch of the lines on the screen was in the range of 0.5–1.1 cm. When viewed at a distance of 60 cm, the difference in viewing angle of adjacent lines ranged from 0.47° to 1.05°. Such angles are at the resolution limit of the eye tracker, and fall within the field of view of the eye. Therefore it is not surprising that they are hard or impossible to distinguish.

Given that the resolution of eye trackers is around 1° [50], and the coverage of the fovea is about 2° [39, 51], the limiting factor is the fovea. To be completely sure which line is being looked at, the separation between the lines should be at least 2°. At a distance of 60 cm this means a pitch of 2.1 cm on the screen, which can be achieved with a 20 pt font at a spacing of 2.6—more than double space. At a distance of 75 cm a pitch of 2.6 cm is required, implying a triple-spaced 20 pt font. Placing the experiment's participants closer to the screen reduces the requirements, but they are still quite far from what developers normally use in their work.

In an experiment like ours, where the only relevant target is one predefined line, this is not a problem. But in a reading study where the line being read needs to be identified without ambiguity it is. In some cases it may be possible to identify the line by using the horizontal dimension, provided adjacent lines have different indentations and lengths. However, given our result that counting lines required fewer fixations than the number of lines, one cannot in general exclude the possibility that adjacent lines can be read during the same fixation.

6.2. Human Behavior

Our experiment was not designed to study human behavior. However, the participants did behave in diverse and interesting ways. Two contrasting forms of behavior were easily discerned: avoiding unnecessary work, and repeating work unnecessarily.

The results concerning avoiding unnecessary work are shown in Fig. 14. The instructions given to the participants were to initially read the texts, and then to read the question and figure out the answer. But they soon learned that they do not really need to read the whole text in order to answer the questions. So they started to skip the text and go straight

to the question. As shown in Fig. 14, initially nearly all the participants read or at least skimmed the text. From the second text, half already skipped it. After the 4th text, only one participant continued to read the text each time.

This result resonates with the literature on code comprehension. While reading and understanding code takes up the majority of developers' time [29, 57], there have been many observations implying that they may not really "read" the code in the conventional sense of the word. Instead, they employ the "as-needed" comprehension strategy of Littman et al. [27], use the opportunistic approach of Letovsky [25], or perform fact finding activities as suggested by LaToza et al. [24]. Rather than trying to understand the code, they may look for shortcuts which enable them to complete the task without investing the effort required to achieve actual understanding [26, 44].

The results concerning unnecessary work were described above in Sect. 5.3 and 5.4. The term "unnecessary" is of course subjective. We use it because of the extreme simplicity of our tasks: they require only to remember the line number and 1 to 3 letters. So we expected that participants would exhibit an efficient viewing pattern, where they read the question, find the target line, and focus on it. Instead some of them returned to the question multiple times, or counted the lines multiple times. Presumably this was done to ascertain the line number, and to ascertain the letters that need to be looked for. This behavior is somewhat surprising given the simplicity of the tasks, as noted above. Nevertheless, some of our participants needed to reassure themselves during the execution of the tasks that they were doing the right thing.

6.3. Reading Order

The phenomena discussed above have an effect on reading order. In the context of reading code, Busjahn et al. suggested two possible reading orders: "story order" (top-to-bottom, left-to-right) and "execution order" (tracing the execution of the program, including function calls and loops) [8]. Importantly, these possible orders were studied in the context of code comprehension tasks, where participants need to summarize the code or determine the value of a variable.

When performing code comprehension research we need to be punctilious regarding the distinction between comprehension in the sense of understanding *what* the code does (it's functionality), and comprehension in the sense of understanding *how* the code does it, that is, the code's structure and the inter-relationships between different parts of the code. Reading patterns for these diverse goals may be completely different. Academic research often emphasizes the first, e.g. in studies of code summarization [1, 42]. But in real-life work developers are typically more interested in the second, e.g. when performing code maintenance tasks.

As we see it, both "story order" and "execution order" are just simple and easy-to-define special cases of what we may call "task order": the reading order needed to perform a given task. If the task is syntactic, for example to find all the instances of a certain variable name (what Binkley et al. call a "where's Waldo" problem [4]) "story order" may be used. If the task is to trace the execution of the code and record the values assigned to a certain variable, the code will be read in "execution order", including re-reading loop bodies multiple times the way they would be executed.

But it is questionable whether either "story order" or "execution order" reflect developer behavior when performing other tasks. When we read a story for fun, we most probably indeed read it in "story order"—from beginning to end, passing through all the words on the way. When we read a news story we might do the same, or we might skim some paragraphs that seem less interesting [15]. But we rarely just read a piece of code, let alone a whole module or system. Usually we have some specific goal in mind, like fixing a bug or adding a feature. This requires a different approach: we first need to understand the structure of the code, and then to use this understanding to find the relevant location to perform the task [53]. Consequently, the reading order need not be directly related to the way the code is written or executed, and in particular, need not be linear.

Trying to characterize "task order" in general may be futile, as different tasks may require very different reading patterns. However, many tasks probably include the following three components (foreshadowed by [46, 53]):

- *Orient* yourself to get a feel for the structure of the code. This may involve scanning the visible code.
- *Search* for the specific part of the code you need to work on to complete the task. This may include skipping complete blocks of the code based on using peripheral vision to identify beacons.
- *Focus* on the relevant code to complete the task. This may combine repeated reading as one learns the details of the code, and then modifying it by performing editing operations, which leads to different reading patterns [46].

What we saw in our experiment is apparently this sort of orient-search-focus task order, for the task of counting letters in a line of text. At a minimum, the participants in the experiment needed to scan the page to identify and read the question, search for and find the target line, and then focus on the target line to count the desired letters. The same conceptualization can be applied when interpreting observed patterns of reading code.

But our experiments also showed that this "task order" may be tainted by personal conditions and attitudes. For example, some people may be confident in their actions, while others are unsure about how to approach the task. People may also require constant validation during their work. This may explain the actions of our experimental participants who regressed to the question to re-check the letters, or re-counted the lines leading to the target line.

Moreover, in some cases the task may be too hard or ill-defined, and participants may be unable to cope. This may lead to unorderly reading with many skips and reversals. Such reading patterns have indeed been observed in the past, where they were called "thrashing" or "fumbling" [10, 20, 46].

Due to such differences in approach and attitude, the specific reading patterns employed by different developers may be quite different. However, it may still be fruitful to look for commonalities between these reading patterns. This is because the commonalities can be expected to reflect the core activities

that need to be performed to complete the task—namely, those that constitute the actual "task order".

At the same time, we should stay mindful of the fact that variations are expected to exist. And indeed, one of the repeated results in the code eye tracking literature is the diversity of scan paths of different experimental participants performing the same task (e.g. [13, 20, 53]). These variations are also interesting, as they reflect the differences between participants. Studying the differences can help uncover the effects of knowledge, experience, and attitude. This includes both minor variations, such as random small regressions to ascertain the last word read, and major variations, such as either initially scanning the whole text or else not doing so.

6.4. Navigation and Peripheral Vision

In our experiment, once the question was read, the participants needed to find the relevant line. The results show a difference between counting and using visual cues. It appears that visual cues can be noted using peripheral vision, and there is no need to fixate on each line to check them.

The implication for code is that the importance of indentation and color-coding keywords is not as a direct aid for comprehension, but that they provide beacons for navigation. Crosby and Stelovsky frown upon the practice of printing keywords in boldface, saying that "keywords are the least observed portions of a program's text" [13]. But this misses the point that clear visual identification of the keywords allows them to be identified at a glance, and helps developers to easily focus on the code *between* these keywords. Bauer et al. claim that indentation has no effect on gaze pattern [2]. However, they consider only aggregate metrics such as fixation duration, fixation rate, and saccade amplitude, and did not analyze gaze paths. We need to design experiments that involve navigation, and specifically navigation that can be aided by the code's block structure and indentation, to really see whether indentation has an effect. Results by Talsma et al. indicate that, at least for beginning students, highlighting the block structure of code helps them focus their attention and leads to more linear reading [52].

6.5. Additional Uses of the Experiment

While the experiment was designed to measure the accuracy of identifying the line being read, it turns out that it can actually have wider uses as a component in other studies.

One use is for validation. It is very common to assume that the output of an eye tracker and the fixation locations computed from it are valid. But such data may suffer from inaccuracies and systematic bias. As a result, counting fixations in a predefined area of interest may include fixations that actually represent looking at something else, and vice versa (called "gaze uncertainty" by Wang et al. [56]). Embedding a simple experiment like ours as a step in a larger experiment can provide independent testimony that the data is indeed valid (or not).

Another use is for recalibration. Given that our experiment requires the participants to focus on one specific line, performing this task in the context of a larger experiment can be used to check the calibration and measure the bias between the eye tracker output and the actual target. This can be used to compensate for drift during an experiment without having to resort to a disruptive full recalibration—essentially a realization of Hornof and Halverson's "required fixation location" [18].

Finally, it should be noted that the accuracy of line identification can depend not only on the physical attributes of the stimuli (font size and line spacing) but also on the experimental participant. For some participants, certain sizes can be adequate, while for others a larger size may be needed. Thus focus experiments like ours can also be used as a criterion for excluding certain participants. By requiring the ratio of fixations that are assigned to the target line correctly to be above a certain threshold, the criterion becomes both quantifiable and directly relevant to reducing threats to the validity of the actual study.

7. Threats to Validity

As we set out to characterize the accuracy of eye tracking in the vertical dimension, a major threat is whether our setup is optimal and representative. One specific concern is that we use a low-cost eye tracker. While this tracker is apparently not inferior to more expensive models [50], it is still desirable to replicate the experiment with a range of trackers and experimental conditions.

A related concern is dealing with drift and the need for recalibration. Our experiments were not very long, so we decided to settle for correction for systematic bias at the level of individual experimental participants. But this issue too deserves additional research, including the option of using the knowledge of the target line to perform the adjustment per experiment.

Some specific elements in the experiments also suffer from threats. For example, when we study the distribution of the focus around the target line (Sect. 5.2), there is a risk that some of the fixations represent intentional glances at other locations in the text. We excluded glances at the question using the technical criterion of being below the bottom line of text, but there is no easy way to identify other such glances. So the results concerning the fraction of "indicative" fixations may not correctly reflect all what the participants were doing.

Another major threat is posed by the need to identify the focus period, when the participants are looking at the target line. We tried to mitigate this by devising a protocol involving both authors and an external judge. However, the judgment of when the focus period starts and ends is still subjective.

Finally, our experimental materials were based on regular text and not code, and on completely synthetic tasks. Using text to learn about code has been done before (e.g. by Marter et al. [28]). In our case it is justified by the fact that using normal text and such tasks is actually appropriate for the study of the distribution of gazes around a target line. And it enabled us to make observations and theorize about the causes of reading patterns. However, focusing on lines of code may be different due to variability in length and indentation; text, in contradistinction, tends to have a "block shape" with lines that are all similar to each other. Also, our conjectures regarding reading patterns need to be validated in the context

of code comprehension, using carefully designed experiments with more realistic tasks, and also by soliciting intent from the participants.

In addition, contemporary code development is done using integrated development environments (IDEs), which affects the way code is accessed. This adds a whole new level of complexity to code reading studies [45], similar to the difference between online and physical reading of news [15]. This is a major undertaking which we leave to future work.

8. Conclusions

Eye tracking studies of code reading are promoted as a window to the mind of the reader (e.g. [3, 13]). They are thought to be useful for identifying mental models used for code comprehension—and specifically whether a top-down or bottom-up approach is being used [1, 3, 9]. Our work indicates that such interpretations should be done with care. We note that analyzing what readers are doing requires a precise identification of the lines they are focusing on, because adjacent code lines may have completely different functions. But this is difficult to achieve with reasonably sized fonts. Moreover, we observe that reading patterns may reflect human behaviors such as the need to reassure yourself in what you are doing, and may not necessarily reflect more significant cognitive processes.

In practical terms, this paper makes two immediate contributions. The first is to show that conventional settings of font sizes and line spacing lead to very small angles between adjacent lines, which do not allow for adequate identification of which line is being read. A font size of at least 20 pt and a wide spacing are apparently needed in order to guarantee that most fixations are assigned correctly, at least for our setup. However, additional research is needed to derive recommendations that are suitable for different eye tracking devices. The second is to devise a simple experiment that can be embedded into larger experimental frameworks. This can be used for two purposes: to augment the conventional calibration procedure and correct accumulated bias, and to provide objective exclusion criteria for participants.

In future work, we intend to design experiments on actual code reading. These experiments will employ different tasks, which require different levels of comprehension [14], and use code with different levels of readability—e.g. employing variables with concise abbreviated names or long verbose names. Such experiments will expose whether the reading patterns change depending on the task and the code characteristics. We will also debrief the participants after they perform the task, and ask them to explicitly explain their reading pattern as reflected in the scanpath generated by the eye tracker. Independently, we will also analyze the scanpath data to identify different phases of perception, such as exploratory scanning vs. focused study [54, 55]. Such analysis can use Krejtz et al.'s K coefficient, which is based on the ratio of fixation durations to saccade lengths [22, 23].

Experimental Materials

The full experimental materials (texts used in the experiment, raw results from the eye tracker, and analyses) are available from Zenodo at DOI 10.5281/zenodo.7464051.

Acknowledgments

This research was supported by the ISRAEL SCIENCE FOUNDATION (grant no. 832/18).

References

[1] N. J. Abid, J. I. Maletic, and B. Sharif, "*Using developer eye movements to externalize the mental model used in code summarization tasks*". In 11th *Symp. Eye Tracking Res. & Apps.*, art. 13, Jun 2019, DOI: 10.1145/3314111.3319834.

[2] J. Bauer, J. Siegmund, N. Peitek, J. C. Hofmeister, and S. Apel, "*Indentation: Simply a matter of style or support for program comprehension?*" In 27th *Intl. Conf. Program Comprehension*, pp. 154–164, May 2019, DOI: 10.1109/ICPC.2019.00033.

[3] R. Bednarik and M. Tukiainen, "*An eye-tracking methodology for characterizing porgram comprehension processes*". In 4th *Symp. Eye Tracking Res. & Apps.*, pp. 125–132, Mar 2006, DOI: 10.1145/1117309.1117356.

[4] D. Binkley, M. Davis, D. Lawrie, J. I. Maletic, C. Morrell, and B. Sharif, "*The impact of identifier style on effort and comprehension*". *Empirical Softw. Eng.* **18(2)**, pp. 219–276, Apr 2013, DOI: 10.1007/s10664-012-9201-4.

[5] T. Blascheck and B. Sharif, "*Visually analyzing eye movements on natural language texts and source code snippets*". In 11th *Symp. Eye Tracking Res. & Apps.*, art. 14, Jun 2019, DOI: 10.1145/3314111.3319917.

[6] R. W. Booth and U. W. Weger, "*The function of regressions in reading: Backward eye movements allow rereading*". *Memory & Cognition* **41(1)**, pp. 82–97, Jan 2013, DOI: 10.3758/s13421-012-0244-y.

[7] S. Bottos and B. Balasingam, "*Tracking the progression of reading through eye-gaze measurements*". In 22nd *Intl. Conf. Information Fusion*, Jul 2019, DOI: 10.23919/FUSION43075.2019.9011436.

[8] T. Busjahn, R. Bednarik, A. Begel, M. Crosby, J. H. Paterson, C. Schulte, B. Sharif, and S. Tamm, "*Eye movements in code reading: Relaxing the linear order*". In 23rd *Intl. Conf. Program Comprehension*, pp. 255–265, May 2015, DOI: 10.1109/ICPC.2015.36.

[9] T. Busjahn, C. Schulte, and A. Busjahn, "*Analysis of code reading to gain more insight in program comprehension*". In 11th *Koli Calling Intl. Conf. Computing Education Res.*, pp. 1–9, Nov 2011, DOI: 10.1145/2094131.2094133.

[10] T. Busjahn, C. Schulte, B. Sharif, Simon, A. Begel, M. Hansen, R. Bednarik, P. Orlov, P. Ihantola, G. Shchekotova, and M. Antropova, "*Eye tracking in computing education*". In 10th *Intl. Computing Education Research Conf.*, pp. 3–10, Aug 2014, DOI: 10.1145/2632320.2632344.

[11] J. W. Carr, V. N. Pescuma, M. Furlan, M. Ktori, and D. Crepaldi, "*Algorithms for the automated correction of vertical drift in eye-tracking data*". *Behavior Res. Meth.* **54(1)**, pp. 287–310, Feb 2022, DOI: 10.3758/s13428-021-01554-0.

[12] N. Cowan, "*The magical mystery four: How is working memory capacity limited, and why?*" *Current Directions Psychological Sci.* **19(1)**, pp. 51–57, Feb 2010, DOI: 10.1177/0963721409359277.

[13] M. E. Crosby and J. Stelovsky, "*How do we read algorithms? a case study*". *Computer* **23(1)**, pp. 24–35, Jan 1990, DOI: 10.1109/2.48797.

[14] D. G. Feitelson, "*Considerations and pitfalls for reducing threats to the validity of controlled experiments on code comprehension*". *Empirical Softw. Eng.* **27(6)**, art. 123, Nov 2022, DOI: 10.1007/s10664-022-10160-3.

[15] K. Holmqvist, J. Holsanova, M. Barthelson, and D. Lundqvist, "*Reading or scanning? a study of newspaper and net paper reading*". In *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyönä, R. Radach, and H. Deubel (eds.), chap. 30, pp. 657–670, North Holland, 2003, DOI: 10.1016/B978-044451020-4/50035-9.

[16] K. Holmqvist, M. Nyström, and F. Mulvey, "*Eye tracker data quality: What it is and how to measure it*". In *Symp. Eye Tracking Res. & Apps.*, pp. 45–52, Mar 2012, DOI: 10.1145/2168556.2168563.

[17] I. T. C. Hooge, G. A. Holleman, N. C. Haukes, and R. S. Hessels, "*Gaze tracking accuracy in humans: One eye is sometimes better than two*". *Behavior Res. Meth.* **51(6)**, pp. 2712–2721, Dec 2019, DOI: 10.3758/s13428-018-1135-3.

[18] A. J. Hornof and T. Halverson, "*Cleaning up systematic error in eye-tracking data by using required fixation locations*". *Behavior Research Methods, Instruments, & Comput.* **34(4)**, pp. 592–604, Nov 2002, DOI: 10.3758/BF03195487.

[19] Y. Huang, K. Leach, Z. Sharafi, N. McKay, T. Santander, and W. Weimer, "*Biases and differences in code review using medical imaging and eye-tracking: Genders, humans, and machines*". In 28th *ESEC/FSE*, pp. 456–468, Nov 2020, DOI: 10.1145/3368089.3409681.

[20] A. Jbara and D. G. Feitelson, "*How programmers read regular code: A controlled experiment using eye tracking*". *Empirical Softw. Eng.* **22(3)**, pp. 1440–1477, Jun 2017, DOI: 10.1007/s10664-016-9477-x.

[21] M. Kean and A. Lambert, "*Orienting of visual attention based on peripheral information*". In *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyönä, R. Radach, and H. Deubel (eds.), chap. 2, pp. 27–47, North Holland, 2003, DOI: 10.1016/B978-044451020-4/50003-7.

[22] K. Krejtz, A. Çöltekin, A. T. Duchowski, and A. Niedzielska, "*Using coefficient K to distinguish ambient/focal visual attention during cartographic tasks*". *J. Eye Movement Res.* **10(2)**, art. 3, 2017, DOI: 10.16910/jemr.10.2.3.

[23] K. Krejtz, A. Duchowski, I. Krejtz, A. Szarkowska, and A. Kopacz, "*Discerning ambient/focal attention with coefficient K*". *ACM Trans. Applied Perception* **13(3)**, art. 11, May 2016, DOI: 10.1145/2896452.

[24] T. D. LaToza, D. Garlan, J. D. Herbsleb, and B. A. Myers, "*Program comprehension as fact finding*". In 6th *ESEC-FSE*, pp. 361–370, Sep 2007, DOI: 10.1145/1287624.1287675.

[25] S. Letovsky, "*Cognitive processes in program comprehension*". *J. Syst. & Softw.* **7(4)**, pp. 325–339, Dec 1987, DOI: 10.1016/0164-1212(87)90032-X.

[26] O. Levy and D. G. Feitelson, "*Understanding large-scale software systems — structure and flows*". *Empirical Softw. Eng.* **26(3)**, art. 48, May 2021, DOI: 10.1007/s10664-021-09938-8.

[27] D. C. Littman, J. Pinto, S. Letovsky, and E. Soloway, "*Mental models and software maintenance*". *J. Syst. & Softw.* **7(4)**, pp. 341–355, Dec 1987, DOI: 10.1016/0164-1212(87)90033-1.

[28] T. Marter, P. Babucke, P. Lembken, and S. Hanenberg, "*Lightweight programming experiments without programmers and programs: An example study on the effect of similarity and number of object identifiers on the readability of source code using natural texts*". In *Onward!*, pp. 1–14, Oct 2016, DOI: 10.1145/2986012.2986020.

[29] R. Minelli, A. Mocci, and M. Lanza, "*I know what you did last summer: An investigation of how developers spend their time*". In 23rd *Intl. Conf. Program Comprehension*, pp. 25–35, May 2015, DOI: 10.1109/ICPC.2015.12.

[30] A. Mishra, M. Carl, and P. Bhattacharya, "*A heuristic-based approach for systematic error correction of gaze data for reading*". In 1st *Workshop Eye-Tracking & Natural Language Processing*, pp. 71–79, Dec 2012.

[31] F. B. Narcizo, J. E. R. de Queiroz, and H. M. Gomes, "*Remote eye tracking systems: Technologies and applications*". In 26th *Conf. Graphics, Patterns and Images*, pp. 15–22, Aug 2013, DOI: 10.1109/SIBGRAPI-T.2013.8. (tutorial).

[32] M. Nyström, R. Andersson, K. Holmqvist, and J. van der Weijer, "*The influence of calibration method and eye physiology on eyetracking data quality*". *Behavior Res. Meth.* **45(1)**, pp. 272–288, Mar 2013, DOI: 10.3758/s13428-012-0247-4.

[33] U. Obaidellah, M. Al Haek, and P. C.-H. Cheng, "*A survey on the usage of eye-tracking in computer programming*". *ACM Comput. Surv.* **51(1)**, art. 5, Jan 2018, DOI: 10.1145/3145904.

[34] C. Palmer and B. Sharif, "*Towards automating fixation correction for source code*". In 9th *Symp. Eye Tracking Res. & Apps.*, pp. 65–68, Mar 2016, DOI: 10.1145/2857491.2857544.

[35] N. Peitek, J. Siegmund, and S. Apel, "*What drives the reading order of programmers? an eye tracking study*". In 28th *Intl. Conf. Program Comprehension*, pp. 342–353, Oct 2020, DOI: 10.1145/3387904.3389279.

[36] C. S. Peterson, N. J. Abid, C. A. Bryant, J. I. Maletic, and B. Sharif, "*Factore influencing dwell time during source code reading: A large-scale replication experiment*". In 11th *Symp. Eye Tracking Res. & Apps.*, art. 38, Jun 2019, DOI: 10.1145/3314111.3319833.

[37] J. Pi and B. E. Shi, "*Task-embedded online eye-tracker calibration for improving robustness to head motion*". In 11th *Symp. Eye Tracking Res. & Apps.*, art. 8, Jun 2019, DOI: 10.1145/3314111.3319845.

[38] N. Ramkumar, V. Kothari, C. Mills, R. Koppel, J. Blythe, S. Smith, and A. L. Kun, "*Eyes on URLs: Relating visual behavior to safety decisions*". In *Symp. Eye Tracking Res. & Apps.*, art. 19, Jun 2020, DOI: 10.1145/3379155.3391328.

[39] K. Rayner, "*Eye movements in reading and information processing: 20 years of research*". *Psychological Bulletin* **124(3)**, pp. 372–422, Nov 1998, DOI: 10.1037/0033-2909.124.3.372.

[40] K. Rayner, "*Eye movements and attention in reading, scene perception, and visual search*". *Quarterly J. Experimental Psychology* **62(8)**, pp. 1457–1506, Aug 2009, DOI: 10.1080/17470210902816461.

[41] K. Rayner, K. H. Chace, T. J. Slattery, and J. Ashby, "*Eye movements as reflections of comprehension processes in reading*". *Sci. Stud. Reading* **10(3)**, pp. 241–255, 2006, DOI: 10.1207/s1532799xssr1003_3.

[42] P. Rodeghero, C. Liu, P. W. McBurney, and C. McMillan, "*An eye-tracking study of Java programmers and application to source code summarization*". *IEEE Trans. Softw. Eng.* **41(11)**, pp. 1038–1054, Nov 2015, DOI: 10.1109/TSE.2015.2442238.

[43] P. Rodeghero and C. McMillan, "*An empirical study on the patterns of eye movement during summarization tasks*". In *Intl. Symp. Empirical Softw. Eng. & Measurement*, pp. 11–20, Oct 2015, DOI: 10.1109/ESEM.2015.7321188.

[44] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, "*How do professional developers comprehend software?*" In 34th *Intl. Conf. Softw. Eng.*, pp. 255–265, Jun 2012, DOI: 10.1109/ICSE.2012.6227188.

[45] T. R. Shaffer, J. L. Wise, B. M. Walters, S. C. Müller, M. Falcone, and B. Sharif, "*iTrace: Enabling eye tracking on software artifacts within the IDE to support software engineering tasks*". In *ESEC/FSE*, pp. 954–957, Aug 2015, DOI: 10.1145/2786805.2803188.

[46] Z. Sharafi, I. Bertram, M. Flanagan, and W. Weimer, "*Eyes on code: A study on developers' code navigation strategies*". *IEEE Trans. Softw. Eng.* **48(5)**, pp. 1692–1704, May 2022, DOI: 10.1109/TSE.2020.3032064.

[47] Z. Sharafi, B. Sharif, Y.-G. Guéhéneuc, A. Begel, R. Bednarik, and M. Crosby, "*A practical guide on conducting eye tracking studies in software engineering*". *Empirical Softw. Eng.* **25(5)**, pp. 3128–3174, Sep 2020, DOI: 10.1007/s10664-020-09829-4.

[48] Z. Sharafi, Z. Soh, and Y.-G. Guéhéneuc, "*A systematic literature review on the usage of eye-tracking in software engineering*". *Inf. & Softw. Tech.* **67**, pp. 79–107, Nov 2015, DOI: 10.1016/j.infsof.2015.06.008.

[49] O. Špakov, H. Istance, A. Hyrskykari, H. Siirtola, and K.-J. Räihä, "*Improving the performance of eye trackers with limited spatial accuracy and low sampling rates for reading analysis by heuristic fixation-to-word mapping*". *Behavior Res. Meth.* **51(6)**, pp. 2661–2687, Dec 2019, DOI: 10.3758/s13428-018-1120-x.

[50] L. Spitzer and S. Mueller, "*Using a test battery to compare three remote, video-based eye trackers*". In *Symp. Eye Tracking Res. & Apps.*, art. 28, Jun 2022, DOI: 10.1145/3517031.3529644.

[51] H. Strasburger, I. Rentschler, and M. Jüttner, "*Peripheral vision and pattern recognition: A review*". *J. Vision* **11(5)**, art. 13, Dec 2011, DOI: 10.1167/11.5.13.

[52] R. Talsma, E. Barendsen, and S. Smetsers, "*Analyzing the influence of block highlighting on beginning programmers' reading behavior usig eye tracking*". In 9th *Comput. Sci. Education Research Conf.*, art. 9, Oct 2020, DOI: 10.1145/3442481.3442505.

[53] H. Uwano, M. Nakamura, A. Monden, and K.-i. Matsumoto, "*Analyzing individual performance of source code review using reviewers' eye movement*". In *Symp. Eye Tracking Res. & Apps.*, pp. 133–140, Mar 2006, DOI: 10.1145/1117309.1117357.

[54] B. M. Velichkovsky, M. Joos, J. R. Helmert, and S. Pannasch, "*Two visual systems and their eye movements: Evidence from static and dynamic scene perception*". In 27th *Proc. Ann. Meeting Cognitive Science Soc.*, pp. 2283–2288, 2005.

[55] B. M. Velichkovsky, A. N. Korosteleva, S. Pannasch, J. R. H. V. A. Orlov, M. G. Sharaev, B. B. Velichkovsky, and V. L. Ushakov, "*Two visual systems and their eye movements: A fixation-based event-related experiment with ultrafast fMRI reconciles competing views*". *Modern Technologies in Medicine* **11(4)**, pp. 7–16, 2019, DOI: 10.17691/stm2019.11.4.01.

[56] Y. Wang, M. Koch, M. Bâce, D. Weiskopf, and A. Bulling, "*Impact of gaze uncertainty on AOIs in information visualizations*". In *Symp. Eye Tracking Res. & Apps.*, art. 60, Jun 2022, DOI: 10.1145/3517031.3531166.

[57] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "*Measuring program comprehension: A large-scale field study with professionals*". *IEEE Trans. Softw. Eng.* **44(10)**, pp. 951–976, Oct 2018, DOI: 10.1109/TSE.2017.2734091.

[58] A. Yamaya, G. Topić, and A. Aizawa, "*Vertical error correction using classification of transitions between sequential reading segments*". *J. Inf. Proc.* **25**, pp. 100–106, 2017, DOI: 10.2197/ipsjjip.25.100.