Investigating the Feasibility of Conducting Webcam-Based Eye-Tracking Studies in Code Comprehension

Zihan Fang, Robert Wallace, Zachary Karas, *Graduate Student Member, IEEE*, Toby Li, Collin McMillan, and Yu Huang

***Abstract*—Researchers in Software Engineering (SE) often use onsite screen-mounted eye-tracking experiments to investigate programmers' visual attention patterns in various programming activities. The pandemic and the difficulty of recruiting many participants, especially those with special expertise in SE, have hastened the shift towards conducting eye-tracking studies offsite, which use integrated webcams to track participants' gaze in natural settings. This study compares the efficacy of a webcam-based eye tracker to a research-focused screen-mounted eye tracker in code comprehension tasks. We conducted onsite experiments with 49 participants, each using both types of eye trackers simultaneously to assess the webcam-based eye tracker's capability to capture visual patterns at general, semantic, and token levels and detect individual differences. Additionally, we conducted offsite experiments with 10 participants to supplement the findings. Our findings indicate that while the webcam-based eye tracker effectively captures programmers' semantic comprehension, but faces challenges in accurately identifying cognitive patterns at a more detailed token level in onsite settings. Furthermore, the elevated noise observed in real-world offsite conditions significantly limits the tracker's reliability for drawing accurate conclusions. Participants also encountered challenges with calibration and task initiation, highlighting areas for improvement in conducting webcam-based eye-tracking studies offsite in the future. This study investigates the feasibility of webcam-based eye-tracking studies in SE, offers insights to enhance the accuracy of webcam-based eye-tracking in programming potentially, and provides guidelines for future webcam-based eye-tracking study designs.**

***Index Terms*—Webcam-based eye tracking, code comprehension.**

I. Introduction

RESEARCHERS in Software Engineering (SE) often conduct human eye-tracking studies to understand how programmers visually process code, gain insights into their cognitive processes, and understand the challenges behind [1], [2], [3], [4]. Eye-tracking offers a high-resolution record of visual behavior, serving as an effective measure for cognitive processes [1]. It has enabled SE researchers to examine programmers' code reading pattern [5], [6], revealing insights into their strategies and thought processes during code comprehension [7], biases during code review [8], and differences between expert and novice programmers [9], [10]. Despite its utility, eye-tracking is often limited to laboratory settings (i.e., onsite settings) and highly controlled environments due to its high expense and delicate setup procedures [1]. Furthermore, onsite eye-tracking studies may be limited by the availability of eligible participants, who may be constrained by personal or geographical factors from participating in studies that require specialized equipment and controlled settings. These limitations can affect the diversity and representativeness of the study sample, potentially impacting the generalizability of the findings, which is particularly critical when the study aims to examine expertise or demographics. For example, recruiting a diverse sample of programmers from various cultural backgrounds for onsite studies, aimed at understanding their approaches to coding tasks, can be challenging. Such limitations also affect the statistical power in eye-tracking studies, and ultimately the ability of researchers to confidently explain underlying behavioral phenomena and reject false positives [11]. Additionally, onsite experiments can be criticized for lacking realism due to their highly controlled settings [12]. Although onsite eye tracking is widely recognized for its high accuracy [13], these factors emphasize its limitations and highlight the need for alternative paradigms.

The pandemic has accelerated the adoption of remote modalities for work, education [14], and eye-tracking research [15]. Advances in technology now enable remote (i.e., offsite) eye-tracking studies using standard webcams [16], allowing data collection in naturalistic settings such as participants' homes or other preferred environments [17]. Some studies in non-programming domains have compared the performance of webcam eye trackers and screen-mounted eye trackers [16], [18], [19], and found that webcam-based eye trackers offer commensurate performance but with slightly lower raw data quality. Though webcam-based eye-tracking has been evaluated in other fields, its feasibility for programming activities in SE remains uncertain, especially since studies in areas such as

Received 24 June 2025; revised 13 November 2025; accepted 26 December 2025. Date of publication 6 January 2026; date of current version 17 March 2026. Recommended for acceptance by J. Wang. *(Corresponding author: Zihan Fang.)*

Zihan Fang, Zachary Karas, and Yu Huang are with the Department of Computer Science, Vanderbilt University, Nashville, TN 37235, USA (e-mail: zihan.fang@vanderbilt.edu).

Robert Wallace, Toby Li, and Collin McMillan are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA.

Digital Object Identifier 10.1109/TSE.2026.3651615

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

web searching often focus on broader interests that may not address the needs of programming-related research in certain areas on the screen [5], [16]. In addition, due to the complexity of programs, SE researchers always conduct eye-tracking studies focusing on different levels of program understanding [1]. For instance, studies on how programmers interpret individual keywords or operators of a program require more accurate data focusing on a limited area (e.g., individual tokens in a program) [20]. Studying how programmers tackle a programming program on the logic level requires understanding semantics [21], which is more detailed than general web searching but does not need extreme accuracy (i.e., compared to token- or character-level attention). When investigating IDE navigation behaviors [22], [23], the reduced accuracy of webcam-based eye trackers may not greatly impact overall findings, as they focus on broader patterns. In addition, individual differences may also lead to different visual patterns revealed by previous studies in SE [24]. Therefore, in this study, we specifically designed analysis at the *general*, *semantic*, and *token* levels as well as *individual difference* analysis to shed light on the feasibility of webcam-based eye-tracking study for different aspects of program understanding commonly explored in SE. To our knowledge, these differences have not been evaluated in the context of SE experiments. A deeper understanding of the strengths and weaknesses of eye-tracking studies using a webcam will enable SE researchers to make appropriate design decisions for their specific research questions.

In this paper, we present eye-tracking studies conducted in both onsite ($n = 49$) and offsite ($n = 10$) settings. For the onsite study, participants' gaze was recorded concurrently using a webcam-based eye tracker (WebGazer) and a high-accuracy screen-mounted tracker (Tobii Pro Fusion), which served as the reference baseline [25]. WebGazer is software that uses a standard RGB webcam to infer on-screen gaze, whereas Tobii Pro Fusion is dedicated screen-mounted hardware with stereo cameras for precise eye tracking. In the offsite study, only the webcam-based eye tracker was used to characterize the noise expected in real-world offsite eye-tracking data. We specifically chose code comprehension as the programming task because it is fundamental to nearly all programming activities. To the best of our knowledge, this is one of the largest eye-tracking studies on programming activities in SE to date. We evaluated the feasibility of webcam-based eye-tracking studies, focusing on their reliability in observing programmers' visual behaviors. To provide a comprehensive exploration, we specifically demonstrated the investigation regarding three different levels of area of interest (AOI) analysis, as well as individual differences: *1) General level: overall visual patterns (e.g., fixations); 2) Semantic level: tokens grouped into a single semantic AOI representing a concrete program structure with a functional role (e.g., the tokens in* `myFunc(arg1, arg2)` *form a Function Call semantic AOI); 3) Token level: the smallest elements in a program.* We also collect participants' perceptions of self-setup for the webcam-based eye-tracking. In onsite settings, we found that webcam-based eye tracking is generally reliable at the semantic AOI level but less accurate at the token level, occasionally producing false positives in analyses of individual differences. The substantially higher noise observed in offsite conditions further limits the tracker's reliability for drawing accurate conclusions. Additionally, participants primarily reported difficulties with calibration and initiating tasks. Consequently, we proposed suggestions to enhance webcam-based eye-tracker accuracy, along with guidelines for future webcam-based eye-tracking studies. The main contributions are as follows:

- The first investigation of the feasibility of using webcam-based eye tracking in code comprehension.
- A comprehensive exploration of feasibility across three levels of analysis: general, semantic, and token, along with individual differences.
- First study provides guidelines for conducting webcam-based eye-tracking studies in code comprehension.
- First study provides pairwise datasets for future improvements to the webcam-based eye trackers.

A comprehensive understanding of webcam-based eye-tracker performance in code comprehension can enhance the feasibility of webcam-based eye-tracking studies in SE and provide guidelines for future studies. This improvement will streamline participant engagement, enhance their experience in webcam-based studies, and broaden the pool of potential participants.

II. Related Work

A. Code Comprehension in Software Engineering

In software engineering (SE), code comprehension is crucial, impacting the effectiveness of the software development process [24]. It involves understanding a program's structure, functionality, and intent, essential for efficient maintenance, debugging, and enhancement of software systems [26]. With the complexity of modern software systems, code comprehension is a focus for both researchers and practitioners [27]. Several factors, including code structure complexity and readability, as well as individual factors like expertise and gender, can influence code comprehension [28], [29], [30]. Research in SE has also explored the cognitive processes involved in comprehending code, such as pattern recognition, problem-solving strategies, and memory recall. These studies have revealed that code comprehension is not merely about reading and understanding syntax but also involves higher-level cognitive activities [31]. To tackle code comprehension challenges, researchers have used approaches such as eye-tracking [32], code annotations [33], and think-aloud protocols [34]. However, there is still no study comparing code comprehension patterns under different eye-tracking settings, which would provide researchers with more reliable insights into how programmers comprehend programs, particularly those unable to attend onsite studies.

B. Screen-Mounted and Webcam-Based Eye-tracking Studies

Various webcam-based eye-trackers have been designed to accommodate the requirements of offsite eye-tracking studies [16], such as WebGazer [16], TurkerGaze [35], SearchGazer [36]. One study found that an open-source webcam-based eye tracker had higher accuracy than a screen-mounted eye tracker, despite a higher error rate [37], suggesting it as a potential alternative for those needing a mouse input substitute with financial

constraints. Another study with more subjects found a moderate correlation between gaze features from webcam-based systems and traditional eye trackers [38], suggesting that webcam-based systems can accurately model user engagement and comprehension similar to screen-mounted eye trackers. Moreover, research has examined webcam-based and screen-mounted eye-tracking in UX studies [39], finding that subtle unconscious behaviors such as eye and hand movements may slightly affect their performance. Expanding on this, Hutt et al. [40] show early signs of predictive validity for the webcam-based eye-tracker, demonstrating its accuracy in predicting unrelated thinking and reading comprehension from a single calibration. Their findings reveal a positive correlation between predicted rates of task-unrelated thought and comprehension scores, while also noting the influence of factors like lighting and wearing glasses on performance. In addition, previous studies (e.g., [19], [41]) also examined bigger components and analyzed data quality or the technique of webcam-based eye tracking (e.g., [19], [42]) in non-programming tasks. However, it is unclear if webcam-based eye trackers can be comparable with screen-mounted eye trackers in capturing detailed eye movement data for programming tasks. Thus, it is crucial to evaluate their accuracy in tracking eye movements as programmers comprehend code to assess their feasibility for detailed programming studies. In comparison, our study evaluated different levels of accuracy regarding different research questions SE researchers care about for code comprehension tasks to determine the webcam's applicability and, more importantly, understand in what scenarios it is appropriate or not to use a webcam to allow offsite recruitment.

C. Offsite and Onsite Eye-Tracking Studies

A shift towards exploring usability in offsite and onsite eye-tracking studies is noted [43]. Previous research emphasizes challenges in onsite studies, including participant scheduling and equipment acquisition [44]. Nadj et al. discuss the pandemic's impact on offsite eye-tracking experiments, highlighting increased reliance on naturalistic environments [45]. Comparative eye-tracking studies have analyzed website navigation behaviors in both onsite and offsite settings, showing a high degree of similarity [46]. However, challenges in offsite eye-tracking studies, such as differences in participant setups and potential distractions, complicate ensuring data quality [43]. These findings underscore the evolving nature of usability assessments with eye-tracking, emphasizing the importance of understanding offsite and onsite methodologies to enhance the user experience and guide future research. However, a guideline is still needed for conducting offsite and onsite eye-tracking studies on code comprehension tasks. In this study, we set guidelines and best practices for designing effective studies in these contexts, addressing challenges posed by code-related tasks, such as requiring a greater level of precision [28]. By bridging this gap, we intend to enhance the methodological rigor of eye-tracking studies in programming, ultimately advancing the understanding of user interactions with programming activities in both offsite and onsite environments.

III. Experiment

To evaluate the feasibility of webcam-based eye tracking for programming tasks, we compared a webcam-based eye tracker (WebGazer) with a screen-mounted device (Tobii Fusion Pro) in an onsite study. Participants completed 20 C++ code comprehension tasks while their gaze data were simultaneously recorded using both tools. In addition, we conducted an offsite study that fully simulated a remote setting using only the webcam-based eye tracker, allowing us to further assess its practicality in real-world remote scenarios. We structured our research around the following research questions:

- **RQ1:** Does the raw data collected from a webcam-based eye tracker align with that of a high-accuracy eye tracker in programming tasks?
- **RQ2:** Do the visual patterns yielded from webcam-based eye tracking in code comprehension align with screen-mounted eye tracking?
- **RQ3:** Is webcam-based eye tracking feasible and accurate in an unconstrained, real-world offsite setting?
- **RQ4:** What are participants' perceptions of self-setting up the webcam-based eye-tracking?

Specifically, RQ1 evaluates whether raw gaze data from the webcam-based tracker can serve as a viable substitute for data from a high-accuracy device in programming contexts. For RQ2, we assess whether higher-level visual patterns derived from the raw gaze data (e.g., cognitive load, transitions among code) yield consistent conclusions about how participants read code across the two trackers. RQ3 quantifies noise in unconstrained offsite to complement the findings, and RQ4 investigates setup experiences and perceptions to guide future offsite study design for code comprehension.

A. Participant Recruitment

In the onsite study, all participants were recruited from the same CS3 course at Vanderbilt University via a combination of forum posts and an in-class presentation. To be qualified for the experiment, participants must be at least 18 years old, have taken Data Structures or the equivalent courses (the local university's Data Structure course is taught using C++), and have no history of epileptic seizures [47]. In total, we recruited 49 undergraduate students for the onsite study, who received 0.5 extra credits upon completing the experiment. The average participant age was 20.4, with 23 identifying as women and an average of 2.08 years of C++ coding experience.

In the offsite study, we recruited 10 programmers through Prolific, a web platform that supports screening and recruiting research participants from the UK and USA[1]. Each participant was compensated at an hourly rate of $22.50 for completing the study. The participants had an average age of 34.3, all identified as men, and all reported an educational background in computer science. On average, they had 2.0 years of experience with C++. Table I provides a summary of the demographics of the participants.

[1]https://www.prolific.com/

TABLE I
DEMOGRAPHIC INFORMATION OF PARTICIPANTS

Demographic	Category	Participants
Onsite Experiment (n = 49)		
Gender	Men	26
	Women	23
C++ Experience	1 yr	10
	2 yrs	28
	$\geq$3 yrs	11
Offsite Experiment (n = 10)		
Gender	Men	10
	Women	0
C++ Experience	1 yr	4
	2 yrs	2
	$\geq$3 yrs	4

B. Eye Tracking Study Setup

We concurrently collected eye-tracking data using a webcam-based tracker (WebGazer) and a research-grade screen-mounted tracker (Tobii Pro Fusion). WebGazer infers on-screen gaze using a standard RGB webcam and is commonly used in offsite eye-tracking studies. In contrast, Tobii Pro Fusion performs hardware-assisted tracking with dedicated screen-mounted stereo cameras, offering greater robustness and precision for controlled onsite use.

Webcam-Based Eye Tracking. WebGazer, an online tracking software, uses the built-in RGB laptop webcam to estimate the real-time 2D gaze position on the screen [16]. We chose it for its ability to estimate eye-gaze positions across various settings [16]. Another reason is that WebGazer has been widely used in previous studies across different research fields [48], [49], [50]. Compared to Tobii EyeX[2], WebGazer showed a mean error of 130 pixels, corresponding to an average visual angle error of 4.17° in prior web search tasks, as reported in a smaller-scale study [16]. In the onsite study, WebGazer operated at approximately 40 Hz on a laptop integrated with our interface. In the offsite study, it also ran at around 40 Hz, embedded in the same interface but executed on participants' laptops.

Screen-Mounted Eye Tracking. In the onsite study, to validate the performance of WebGazer in code comprehension tasks in a large-scale setting, we mounted a Tobii Pro Fusion eye tracker at 60 Hz [47] on the same laptop as the webcam-based tracker. Tobii Pro Fusion uses dedicated screen-mounted hardware with two cameras to capture stereo images of both eyes, enabling robust and precise measurements of gaze, 3D eye position, and pupil diameter[3]. Moreover, this high-end commercial eye tracker has been thoroughly tested in industrial labs with 100 participants in a setup and test environment representative of normal usage, demonstrating its high accuracy with 0.04° error under optimal conditions [25].

[2]First-generation eye tracker from Tobii with high accuracy.

[3]https://www.tobii.com/products/eye-trackers/screen-based/tobii-pro-fusion#specifications

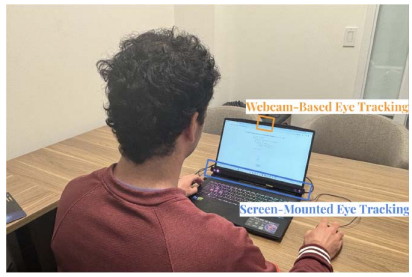


Fig. 1. Onsite experiment setup.

To enhance data quality for both eye trackers, we instructed participants to wear contact lenses instead of glasses when applicable. An example of the onsite experimental setup is presented in Fig. 1. Both WebGazer and Tobii provide x and y coordinates to indicate the participants' observed screen locations. To improve comparability between WebGazer and Tobii data, we standardized all timestamps using Tobii's timestamps.

C. Study Materials

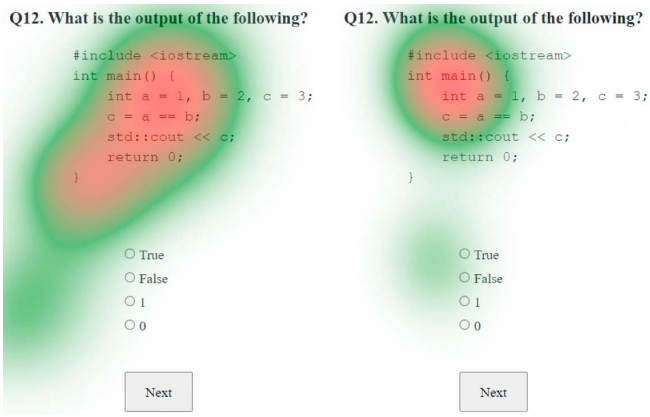
In this section, we detail the materials used, including the setup instructions for the participants, the code comprehension tasks, and the survey and interview questions.

Webcam Set-up Instruction: To support participants in independently setting up the webcam-based eye tracker for both the offsite and onsite experiments, we provided detailed instructions. Participants were instructed to start by initiating the `app.py` file in the webcam directory on the experiment laptop. The participants were then instructed to open a Web browser, navigate to the provided link to access the tasks, and maximize the window for improved accuracy. The instructions also directed participants to calibrate the webcam-based eye tracker to achieve at least 70% accuracy, with immediate feedback and the option to recalibrate if necessary. Participants were also required to recalibrate the tool after each break. Finally, instructions were provided on how to properly terminate the tasks upon completion.

Code Comprehension Task: In this study, we have 20 stimuli comprising multiple-choice code comprehension tasks written in C++. The tasks cover C++ syntax, data structures, and run-time analysis based on the guideline from official C++ tutorial[4] and online educational materials[5]. Overall, the average number of lines per stimulus is 8.3 (SD = 2.7), with trivial lines containing only curly brackets excluded to focus participants' attention on meaningful code. Task selection followed best practices in software engineering, considering participant fatigue and hardware and software constraints [1]. We developed a local task interface presenting C++ comprehension tasks without syntax highlighting at font size 17 following the standard coding style. Participants had to choose at least one option from four to proceed to the next task. All tasks followed a question-code-choices format. Fig. 2(a) and 2(b) are C++ stimuli with gaze point heatmaps, which will be discussed in Sec. IV-A.

[4]https://cplusplus.com/doc/tutorial/

[5]https://www.geeksforgeeks.org/structures-in-cpp/



(a) General heatmap of gaze points using WebGazer

(b) General heatmap of gaze points using Tobii Fusion Pro

Fig. 2. Heatmaps of gaze points using different eye trackers.

Surveys and Interviews: The pre-task survey is designed to collect demographic information, including age, gender, educational background, and participants' experience with programming in C++. The post-task survey consists of 15 questions, combining Likert-scale items and short open-ended questions. This survey is intended to collect participants' subjective experiences and perceptions of setting up the eye-tracking study independently. Specifically, it evaluates their comfort with the webcam-based eye tracker and the overall experimental setup, as well as the clarity of the instructions provided. The open-ended questions are specifically designed to gather participants' feedback and recommend enhancements for future studies or researchers. In addition to the surveys, we conducted semi-structured interviews to gain deeper insights into participants' experiences with self-setting up the study. The interview focused on the specifics of the webcam setup instructions, exploring the challenges participants faced during the setup process and gathering their suggestions for enhancing the setup experience. The semi-structured format allowed for flexibility in the discussion, enabling participants to share thoughts and experiences that might not be fully captured through the structured surveys.

D. Experiment Protocol

For the onsite study (RQ1, RQ2), we used an MSI Pulse 15 B13VFK laptop featuring a 15.6-inch display with a resolution of 1920 $\times$ 1080 pixels. The device was equipped with a 720p HD webcam positioned approximately 60 cm from the participant. All onsite experiments were conducted on this laptop, running Windows 11 and using Google Chrome in a maximized window. The sessions took place in an office environment with natural lighting. The study was approved by the IRB at Vanderbilt University. Prospective participants first electronically completed a consent form along with a prescreening survey to assess their eligibility. Before the experiment, each eligible participant had to complete a pre-test with three data structure questions on traversal methods, achieving an average accuracy of 66.7%. At the experiment's beginning, we required participants to complete an electronic pre-task survey and read a paper-based instruction for webcam setup. A researcher then assisted them with Tobii calibration using Tobii Pro Eye Tracker Manager before starting the tasks. To minimize experimental limitations and observer bias [51], the researcher left the room during the experiments, simulating a more independent environment similar to a fully offsite eye-tracking study. Participants started the tasks by running a Python script for a locally hosted web task integrated with WebGazer and could refer to the instructions during the experiment. Tasks were randomized, with participants completing 12 stimuli, and taking a break without a time limit. After the break, the researcher recalibrated the Tobii, and participants recalibrated WebGazer as instructed before completing the remaining tasks. At the experiment's end, we required participants to complete a post-task survey administered on Google Forms and a short semi-structured interview, which lasted about ten minutes in total (RQ4). The primary aim of this study is to assess how closely WebGazer's data quality and resulting conclusions align with those obtained from the high-precision Tobii tracker.

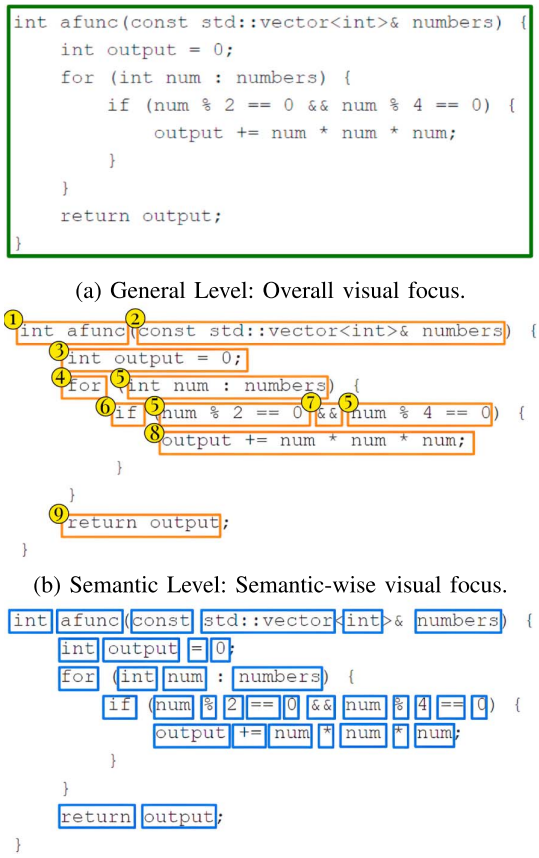
For the offsite study (RQ3), the study protocol remained consistent with the onsite setup in terms of tasks, interfaces, and use of WebGazer. However, participants used their own computer hardware, and we did not employ the Tobii eye tracker for simultaneous data recording. The hardware varied considerably, with most participants using internal webcams on laptops. Due to privacy and technical limitations, we were unable to collect detailed hardware specifications beyond self-reported information. In addition, offsite participants were also required to complete the post-task survey and participate in follow-up interviews (RQ4). Our offsite study quantifies the noise introduced by uncontrolled, real-world remote environments, thereby complementing the onsite results that do not reveal these effects.

E. Data Analysis

In this section, we detail the preprocessing and analysis methods for the eye-tracking, survey, and interview data.

1) Eye tracking Data Analysis: SE researchers often conducted eye-tracking studies to investigate different levels of program understanding, ranging from specific operators and code semantic structures to IDE navigation. To thoroughly compare the compatibility of the two eye trackers and the feasibility of the webcam-based eye tracker under various contexts, we broke down the problem into three levels: *General level, Semantic level, and Token level.*

- *General Level:* At the general level, we explored the feasibility by concentrating on overall visual patterns across the entire stimuli, without reference to problem content, such as specific tokens or semantics. This is shown in Fig. 3(a). Comparison at this level serves as a basic understanding of the performance of each eye tracker in capturing more general viewing behavior, which is essential for evaluating their effectiveness in a broader area. Furthermore, comparing the performance of the two eye trackers at this level



(a) General Level: Overall visual focus.

(b) Semantic Level: Semantic-wise visual focus.

(c) Token Level: Token-wise visual focus.

Fig. 3. Comparison of different level visual focus. The numbers correspond to the semantics described in Table III.

can demonstrate the consistency of the two setups, as well as highlight any calibration issues or drifts[6] in the eye-tracking data from an overall perspective.

- *Semantic Level:* Semantics in programming refer to the meaning of program statements and constructs, and they are important in programming research for examining cognitive processes [52], [53]. At the semantic level, we focus on visual patterns within semantic AOIs, which are structure-level groupings of tokens with a functional role (e.g., `return output` forms a *Return* semantic AOI). Each semantic AOI comprises multiple tokens. In other words, only gaze points that fall on any of the tokens associated with these semantic AOIs will be included in further analysis. This enables us to assess the consistency of two eye trackers at a more detailed level of program understanding compared to the general level. Drawing on previous work on C++ semantics [54], [55] and an introductory C++ textbook [56], we defined 12 semantics to classify tokens based on their functionality and characteristics (see Table III) and used them to form semantic AOIs for later analysis. To address potential ambiguities arising from multiple labels for a single semantic, we used simple C++ programs, no more complex than CS2-level questions, to simplify classification. We distinguished semantic AOIs based on their occurrence in different program positions. Fig. 3(b) showed the semantic-level visual focus, and the numbers displayed correspond to the semantics described in Table III.
- *Token Level:* Tokens, the smallest meaningful elements in programming languages, are often used to understand how programmers comprehend specific syntax in a program [57], [58], [59]. Thus, comparing the two eye trackers at the token level evaluates WebGazer's feasibility for a more specific analysis of how programmers process individual code components (e.g., `int`, `0`, and `%` are all tokens). In other words, only the gaze points that land on the token will be included in the analysis. Fig. 3(c) shows the token-wise visual focus. Since both eye trackers generate gaze data as x and y screen coordinates, we computed the bounding boxes for each token on the screen to identify each token in the program following best practices from previous studies [60]. We then mapped participants' gaze data to C++ code by surrounding each token with bounding boxes, with a final manual review by researchers to ensure accuracy. To distinguish between identical tokens with distinct meanings that appear multiple times in the same task, we added the occurrence of the token as a suffix.

Next, we evaluated the onsite and offsite studies using different metrics, aligned with their respective purposes: (i) **accuracy** (raw data quality and code comprehension patterns) and (ii) **noise**. A summary is shown in Table II.

Accuracy Analysis. For RQ1 and RQ2, we employed 7 measures at multiple levels to assess WebGazer's accuracy performance relative to Tobii, examining two key aspects: raw data quality and code comprehension patterns. Following are the specific descriptions of each metric:

- *Euclidean Distance:* The Euclidean distance is used to evaluate the spatial distance between gaze points captured by two eye trackers on the screen. Following the method from a prior study [16], the eye-tracking data from both eye trackers were processed and aggregated into 10-millisecond bins based on the same timestamp (see Sec III-B). The distance was computed as the average distance between WebGazer and Tobii for the equivalent bin. For different levels of analysis, we focused only on the gaze points within the areas of interest, as shown in Fig. 3(a)–3(c).
- *Element Coverage:* Element coverage is the proportion of elements successfully tracked by eye trackers [5]. Comparing element coverage evaluates the comprehensiveness and consistency of data captured by two eye trackers in code comprehension tasks. Similar element coverage between the eye trackers suggests equal reliability in the comprehensiveness of eye-tracking data and its ability to represent the user's cognitive processes completely. We used z-tests to verify the difference in element coverage rates.
- *Fixations Counts and Fixation Durations:* Fixations are indicators of a person's attention focus and are widely used

[6]Drifts are slow, irregular, smooth-motion eye movements that occur during attempted fixation

TABLE II
SUMMARY OF THE PERFORMANCE METRICS AND DEFINITIONS USED TO EVALUATE DATA QUALITY, CODE COMPREHENSION PATTERNS IN THE ONSITE EXPERIMENT, AND NOISE IN THE OFFSITE EXPERIMENT, ALONG WITH THEIR CORRESPONDING LEVELS OF ANALYSIS

Aspects	Metrics	Definition	Analysis Level
Raw Data Quality	Euclidean Distance	Distance between two gaze points identified by WebGazer and Tobii	G, S, T
	Element Coverage	Fraction of elements identified among all elements in the program	S, T
Code Comprehension Patterns	Fixation Counts	Number of fixations	G, S, T
	Fixation Durations	Average duration of fixations	G, S, T
	Cognitive Load	Combination of fixation counts and durations	G, S, T
	Element Weight	Importance of an element within a scan path	S, T
	Regression Rate	Fraction of looking back to the previous element	S, T
	Edit Distance	Minimum operations required to transform sequence A to B	S, T
Noise	Mean Gaze Distance	Mean euclidean distance between gaze points sequentially	G
	Mean Fixation Distance	Mean euclidean distance between fixations sequentially	G
	Mean Regression Count	Mean number of regressions	T
	Mean Token Distance	Mean distance between detected tokens sequentially	T
	Same Line Token Fixation Rate	Percentage of fixated tokens that were on the same line as the last	T

*G: General Level; S: Semantic Level; T: Token Level.

in eye-tracking studies [61]. Following prior research, we classified each gaze point as a saccade[7] using the Velocity-Threshold Identification (I-VT) algorithm [62]; otherwise, we considered it a fixation. The same procedure was used for both eye trackers. We then counted the identified fixations and calculated their average duration. The combination of two measures defines participants' cognitive load. At the semantic level, we aggregated fixation counts and durations to determine the total count and duration for each semantic. This evaluates if the webcam-based eye tracker can accurately and consistently capture programmers' cognitive processes and load.

- *Edit Distance:* Scan path, a common method in eye-tracking, shows the sequence of fixation points in a person's gaze during task comprehension [1]. In this study, we evaluated two types of scan paths: one for tokens and the other for semantics. We employed Edit Distance [63] to assess the similarity of two scan paths, which evaluates whether the attention transitions of programmers revealed by two eye trackers are similar.
- *Element Weight:* Inspired by Term Frequency-Inverse Document Frequency (TF-IDF), we treated the entire scan path sequence as a document, with each element's weight determined by its TF-IDF score [64], reflecting both its frequency in the scan path (term frequency) and uniqueness across the scan path (inverse document frequency). We calculated the element weight for each semantic and token using a consistent rule, allowing us to understand the elements significant in programmers' code comprehension. The range is from 0 to 1, where 1 indicates a higher element weight. The Mann-Whitney U test was used to compare weight rankings between two eye trackers.
- *Regression Rate:* The regression rate measures how often participants go back to earlier parts of the code, which may indicate difficulty in comprehension [1]. Following previous research, a regression is identified when a participant's gaze moves from a later token to an earlier token in the code segment being analyzed [65]. This rule applies to semantic analysis as well. By comparing regression rates, we can assess if WebGazer captures gaze transition patterns as effectively as Tobii, particularly in revisiting earlier code segments. Z-tests were employed to determine the extent of the difference.

Noise Analysis. In RQ3, we employed five noise metrics to assess and compare the noise levels in eye-tracking data obtained from Tobii, WebGazer (onsite), and WebGazer (offsite). The absolute value of these metrics does not necessarily indicate high or low noise – the purpose is to compare the WebGazer data to Tobii, which we consider a reference. Following are the specific descriptions of each metric:

- *Mean Gaze Distance:* We computed the mean Euclidean distance between each gaze point and the subsequent point. This calculation is based on every individual gaze data point, not just fixations. At each timestamp t, a gaze point is represented by screen coordinates (x, y). We measured the Euclidean distance to the coordinates at timestamp $t + 1$. The "mean gaze distance" refers to the average of all such distances across the entire study. This metric was calculated only at a general level of analysis.
- *Mean Fixation Distance:* The mean Euclidean distance between each fixation and the subsequent fixation. Unlike mean gaze distance, this metric is computed after identifying fixations from the raw gaze data, and then measuring the distance between successive fixations. This metric was computed only at a general level.
- *Mean Regression Count:* This measure is similar to the regression rate discussed in the accuracy section above, but it uses the raw count of regressions rather than a percentage. Noisy samplers may detect fewer regressions because both the original and regression fixation may not be quite at the correct location. This metric was computed only at the token level.

[7]Saccades are fast ocular movements, generally occurring when the gaze is reoriented to a new target.

- *Mean Token Distance:* The mean Euclidean distance between each token that receives a fixation (i.e., is viewed) and the subsequent token that is also fixated. This metric is similar to mean gaze or fixation distance, but computed at the token level.
- *Same Line Token Fixation Rate:* The percent of fixations on tokens that occur on the same line as the previous fixation on a token. A much lower same line token fixation rate than a reference may indicate higher levels of noise. This metric was computed only at the token level.

Consequently, Euclidean distance quantifies how far webcam-collected gaze deviates from expected locations, while element coverage tests whether the webcam covers the key items that Tobii captures. To assess how closely the webcam infers code comprehension patterns compared to Tobii, we use fixation counts and durations and summarize their intensity as cognitive load; we compare the order of attention via scanpath edit distance, highlight importance with element weight, and capture backtracking with regression rate, which adds directional information that edit distance alone cannot. Since offsite sessions occur in heterogeneous environments, we further quantify the noise level: mean gaze distance reflects frame-to-frame jitter in raw gaze, and mean fixation distance and mean token distance indicate how fixation detection and token identification, respectively, are affected by jitter arising from environmental confounds. The same-line token fixation rate (locality across lines) and the mean regression count (backward re-inspections) indicate how error detection and corrective search during code comprehension are influenced by these confounds. We do not include semantic-level noise because semantic AOIs aggregate many tokens and vary substantially in size, making estimates sensitive to AOI design and liable to conflate measurement noise with genuine behavioral differences.

2) Interview and Survey Analysis: In both the onsite and offsite studies, we administered surveys and conducted interviews to gather feedback from programmers on setting up WebGazer using the provided instructions (see Sec. III-C). In total, we conducted 4.1 hours of semi-structured interviews with all participants. Thematic analysis using ATLAS.ti[8] was applied to interview transcripts. Initially, twelve categories were identified by one researcher, and open codes were generated based on recurring statements from participants in the interview, resulting in four meaningful themes. Theoretical saturation was reached at the 25th participant.

Additionally, 56 valid survey responses were collected from participants in both the onsite and offsite experiments for quantitative analysis. To ensure robust data interpretation, each response was meticulously cleaned and coded before analysis. Descriptive statistics such as mean and standard deviation were calculated to summarize the central tendencies and variability of responses. For Likert-scale questions, we calculated frequency distributions to understand the proportion of responses for each scale point. Furthermore, we also analyzed the relationships between different demographic variables (such as age and programming experience) and survey responses using inferential statistics. Chi-square tests were used for categorical data to determine if there were significant associations between demographic categories and responses. However, no significant differences were found. More details are discussed in Sec. IV-D.

[8]https://atlasti.com/

IV. Result

In this section, we present a comparative analysis of webcam-based (WebGazer) and screen-mounted (Tobii) eye trackers in an onsite experiment, focusing on data quality and code comprehension patterns, supplemented by participant feedback on the webcam-based study. For the offsite study, we analyze the noise introduced in the gaze data compared to the onsite setting.

A. RQ1: Does the Raw Data Collected From a Webcam-Based Eye Tracker Align With That of a High-Accuracy Eye Tracker in Programming Tasks?

1) General Level: In webcam-based eye tracking studies, raw data quality is paramount, as it determines the accuracy of gaze point capture, which is vital for precise analysis in tasks that demand meticulous attention, such as code comprehension. At the general analysis level, WebGazer's performance is incomparable to Tobii's in terms of accuracy, as demonstrated by the great discrepancy in raw data, with WebGazer exhibiting an average distance of 323.6 pixels (SD = 39.9) across 20 stimuli measured by Euclidean distance. This distance, notably larger than the 134-pixel discrepancy between WebGazer and Tobii EyeX reported in a previous study [16], raises concerns about WebGazer's reliability in capturing gaze for detailed analysis. Individual stimulus analysis further revealed relatively low data quality for WebGazer, with Euclidean distances ranging from 275.1 to 466.0 pixels. To explore whether a similar conclusion can be drawn specifically on the code area, we conducted a focused analysis to gain more precise insights.

2) Semantic Level: At the semantic level, WebGazer performs comparably to Tobii, as evidenced by two accuracy metrics, and demonstrates relatively higher accuracy in capturing gaze points related to code semantics.

First, gaze points collected by WebGazer were much closer to those by Tobii measured by Euclidean distance, with a lower average distance of 127.3 pixels (SD = 37.5) across 20 stimuli, ranging from 64.5 to 212.7 pixels. Second, WebGazer demonstrated a comparable capability to Tobii in capturing the semantics of the program focused on by participants, as measured by element coverage. Specifically, WebGazer demonstrated a slightly higher element coverage rate of 98.4% compared to Tobii's 93.4%. A z-test indicated no statistically significant difference, implying both eye trackers provide similar gaze data on participants' focus on program semantics. Additionally, the distribution of semantics for each stimulus identified by WebGazer and Tobii remains similar, further indicating the comparable raw data quality at the semantic level, as confirmed by the Kolmogorov-Smirnov (K-S) test.

3) Token Level: We further evaluated the raw data quality of WebGazer at the token level using the same measures to ascertain whether WebGazer can sustain a comparable ability to

Tobii in a more granular analysis. While we found WebGazer's performance decreased compared to the semantic level, it still performed better than the general level.

First, gaze points collected by the two eye trackers at similar time points were still proximate, with an average Euclidean distance of 192.5 pixels (SD = 55.6). While this distance exceeded that observed at the semantic level, it remains better than the results obtained when considering all gaze points on the stimuli. Upon examining each stimulus individually, we found that the range of average distance varied from 60.8 to 312.0 pixels. However, in terms of token capture, WebGazer reported that participants observed more tokens when resolving tasks compared to Tobii. This inconsistency persists across all stimuli individually, as the distribution of tokens identified by WebGazer and Tobii is significantly different, as evidenced by the K-S test ($p < 0.001$). Specifically, Tobii captured programmers' attention on only 24.6% of all tokens in a program, whereas WebGazer recorded a higher percentage of tokens viewed by programmers, at 38.9%. Z-test confirmed this 14.3% difference is significant ($p < 0.05$). These results suggest that WebGazer may not have good raw data quality about specific tokens within a program's semantic structure, possibly due to its reliance on a webcam for eye tracking, which may lack the precision of specialized eye-tracking hardware such as Tobii. Furthermore, WebGazer's algorithm in filtering noise and identifying tokens may also contribute to this discrepancy, which is discussed in Sec. VI.

> WebGazer exhibits performance similar to Tobii in capturing gaze points across semantic AOI, such as collecting spatially proximate gaze points and identifying relevant semantics focused on by participants. However, its performance at the general level and token level is not comparable to Tobii's.

B. RQ2: Do the Visual Patterns Yielded From Webcam-Based Eye Tracking in Code Comprehension Align With Screen-Mounted Eye Tracking?

1) General Level: Expanding on raw data quality, we evaluated whether WebGazer can capture comparable code comprehension patterns with Tobii. At a general level, we found a significant difference in capturing participants' comprehension patterns between WebGazer and Tobii. WebGazer indicated that participants exerted more effort on each task compared to Tobii, as reflected in the combined fixation counts and durations (96.9 seconds vs. 63.1 seconds). This raises a concern that WebGazer's algorithm might not effectively differentiate saccades from valid fixations. Upon further examination, Tobii reported an average of 161.7 fixations per stimulus, with each fixation lasting about 0.39 seconds. In contrast, WebGazer recorded an average of 334 fixations per stimulus, lasting 0.29 seconds per fixation. An in-depth analysis of individual stimuli showed notable differences in the distribution of fixation counts and durations between the two eye trackers, as confirmed by the K-S test. Fig. 2 depicts general heatmaps illustrating participants' gaze points using two eye trackers.

2) Semantic Level: While the general behavior differs between the two eye trackers, we further evaluated the consistency of code comprehension behavior regarding semantics, building on the promising preliminary results for their raw data quality.

Surprisingly, despite there are differences in code comprehension data, the high-level conclusions drawn from it on programmers' cognitive load are comparable between WebGazer and Tobii. Specifically, we found that the cognitive load on semantics, as measured by the two eye trackers, is comparable, even though there are differences in the exact number of fixation counts and average duration underlying the cognitive load. WebGazer indicated that programmers allocated an average duration of 0.15 seconds to each semantic, while Tobii reported an average of 0.34 seconds, based on a combination of fixation counts and durations. More specifically, WebGazer showed an average of 2.1 fixations per semantic, lasting 0.07 seconds each, whereas Tobii recorded an average of 6.7 fixations, each lasting 0.05 seconds. When analyzing individual stimuli, only half showed a similar distribution of fixation counts, confirmed by the K-S test. Moreover, only a quarter of the stimuli demonstrated comparable distribution for average fixation duration. Despite these differences, the overall ranking of cognitive load across semantics was consistent between WebGazer and Tobii. This suggests general agreement on which semantics demand the most cognitive attention from participants, as supported by the Mann–Whitney U test ($p = 0.12$). WebGazer identified *Assignment*, *Variable Declaration*, and *Argument* as the top semantics, while Tobii similarly highlighted *Assignment* and *Variable Declaration* as well as *Loop Body*.

Additionally, regarding scan paths, WebGazer and Tobii still exhibit notable differences in straight scan path data, but they yield similar high-level conclusions, as corroborated by edit distance, element weight, and regression. On average, our analysis revealed an average edit distance of 177.5 between the semantic scan paths generated by WebGazer and Tobii. This indicates that aligning WebGazer's scan path (224.7 semantics in length) with Tobii's requires approximately 177.5 operations, suggesting significant differences in the order participants focus on semantics. Looking at individual cases, we observed varied edit distances, with the lowest at 160.1 operations for a scan path of 224.7 semantics in length and the highest at 171.3 operations for a scan path of 113.1 semantics in length. This further indicates that the discrepancy in scan paths is largely inconsistent across all tasks. However, the element importance concluded from the scan paths between WebGazer and Tobii is similar according to the Mann–Whitney U test ($p = 0.76$). Specifically, WebGazer emphasized *Conditional Body*, *Constructor Call*, and *Function Call* as consistently important, whereas Tobii identified *Conditional Body*, *Assignment*, and *Function Call* as important on average across all scan paths. Table III summarizes the overall element weight ranking from two eye trackers. Furthermore, the conclusions about the likelihood that programmers backtrack during their comprehension of program semantics, as indicated by the regression rate, were similar in the scan path analysis from both WebGazer and Tobii.

TABLE III
COMPARISON OF ELEMENT IMPORTANCE RANKING BETWEEN WEBGAZER AND TOBII AND ITS FREQUENCY IN PROGRAMS

Semantic	WebGazer	Tobii	Frequency
Conditional Body[8]	1	1	84
Assignment	7	2	122
Function Call	3	3	77
Loop Body[4]	4	4	73
Parameter[2]	8	5	48
Variable Declaration[3]	5	6	122
Constructor Call	2	7	4
Argument[5]	10	8	122
Function Definition[1]	9	9	59
Class Definition	6	10	5
Operator[7]	11	11	1
Return[9]	13	12	30
Conditional Statement[6]	12	13	38

* 1-9 indicate the corresponding semantic in Fig. 3(b).

Our findings indicate that Tobii reported an average regression rate of 41.0% (SD = 22), while WebGazer reported a regression rate of 46% (SD = 21) across all programmers. A z-test confirmed that this difference is not statistically significant, further supporting their consistency. However, upon examining individual programmers, we found that 9 out of 49 did not exhibit similar regression patterns between WebGazer and Tobii, with WebGazer consistently reporting a higher regression rate than Tobii. This implies that while WebGazer can generally perform well, it can be inconsistent.

3) Token Level: At the token level, WebGazer's performance was less promising than at the semantic level, showing inconsistent code comprehension patterns and occasionally leading to different conclusions compared to Tobii. Stated differently, while WebGazer sometimes reached similar conclusions to Tobii in detailed analysis, its performance was not as consistent.

First, we observed significant differences in cognitive load captured by WebGazer and Tobii at the token level, including variation in fixation counts and duration. WebGazer reported an average of 1.7 fixations per token with an average duration of 0.05 seconds, whereas Tobii reported 4.4 fixations with an average duration of 0.03 seconds across all stimuli. When analyzing the cognitive load per token on each stimulus, the two eye trackers still exhibited significant disagreement. Only 8 out of 20 stimuli showed a similar distribution of fixation counts, and 2 out of 20 stimuli showed a similar distribution of average duration, as determined by the K-S test.

Similar results were observed in the token scan paths, where the two eye trackers yielded largely different scan paths. The conclusions drawn from this scan path data regarding code comprehension behavior are also different. The average edit distance is 206.5 over a scan path of 193.1 tokens in length. When considering each stimulus individually, the edit distance ranges from 138.6 operations over a scan path of 157.6 tokens in length to 182.0 operations over a scan path of 113.1 tokens in length. This implies that the order in which programmers focus on tokens, as captured by the two eye trackers, differs substantially across all tasks. Apart from differences in the raw scan path data, our analysis also revealed persistent differences in the token importance concluded from participants' scan path between WebGazer and Tobii, as confirmed by the Mann–Whitney U test. However, when examining each stimulus separately, we observed that 16 out of 20 stimuli showed similar results regarding the element weight of tokens, suggesting WebGazer might have the potential to provide comparable detailed analysis results, although its reliability may vary across different cases. Additionally, differences in regression rates concluded from participants' scan path were still observed significantly between Tobii and WebGazer. Tobii showed participants have an average regression rate of 36.1% (SD = 19), while WebGazer reported an average regression rate of 42.4% (SD = 16) among the 49 programmers, a statistically significant difference confirmed by a z-test. However, when examining individual participants, WebGazer produced regression rates comparable to those of Tobii for 21 out of the 49 programmers. This suggests that, although WebGazer's results are less consistent, it has the potential to deliver detailed analysis outcomes similar to those of Tobii.

4) Gender Influence: Multiple previous eye-tracking studies have reported that individual factors (i.e., gender) may also contribute to different code comprehension patterns [8], [29], [66], such as different fixation patterns, and disparities in cognitive strategies during code reading. Thus, we investigated whether WebGazer can identify gender influence in code comprehension patterns that are revealed by Tobii. In general, both WebGazer and Tobii found no difference in cognitive load between women and men. At the semantic level, WebGazer generally aligned with Tobii in identifying gender differences; however, it may occasionally produce false positives when detecting specific patterns. Both Tobii and WebGazer agreed that there is no significant difference between genders in cognitive load on semantics, regression rates, element weights within scan paths or fixation counts. However, WebGazer found a significant difference in fixation durations, with women averaging 0.064 seconds and men averaging 0.074 seconds per semantic ($p = 0.001$), a distinction not observed by Tobii. Similar findings were observed at the token level, but WebGazer produced more false positive conclusions at this level of detail. While Tobii and WebGazer aligned on cognitive load for each token and element weight between genders, WebGazer indicated a notable disparity in fixation durations ($p = 0.03$) and regression rates ($p < 0.001$) between genders, which was not supported by Tobii. Additionally, a comparison was conducted to assess whether gender could, in turn, affect the accuracy of WebGazer. Analysis indicated that gender did not have a significant effect on WebGazer's accuracy, as verified by each measure. A summary of the performance of WebGazer compared to Tobii has been summarized in Table IV.

> While there are distribution disparities in code comprehension data, WebGazer and Tobii still yield similar high-level conclusions. Although WebGazer sometimes generates inaccurate code comprehension conclusions across genders, its overall results are consistent with Tobii's.

TABLE IV
EVALUATION OF WEBGAZER'S PERFORMANCE USING VARIOUS MEASURES AT DIFFERENT LEVELS. ✓ INDICATES THE SUPERIOR PERFORMANCE OF WEBGAZER COMPARED TO TOBII, WHILE ✗ DENOTES A LESS FAVORABLE PERFORMANCE

Aspects	Measures	Token	Semantic
Raw Data Quality	*Euclidean Distance	✗	✓
	Element Coverage	✗	✓
Code Comprehension Pattern	Fixation Counts & Duration	✗	✗
	Cognitive Load	✗	✓
	Element Weight	✗	✓
	*Edit Distance	✗	✗
	Regression Rate	✗	✓
Gender Influence	Fixation Counts	✓	✓
	Fixation Duration	✗	✗
	Cognitive Load	✓	✓
	Element Weight	✓	✓
	Regression Rate	✗	✓

* is not applicable for statistical tests for distribution comparison purposes.

TABLE V
EVALUATION OF WEBGAZER'S PERFORMANCE IN OFFSITE AND ONSITE EXPERIMENTS USING NOISE MEASURES AT BOTH GENERAL AND TOKEN LEVELS

Level	Metric	Tobii	WebGazer	
			Onsite	Offsite
General	Mean Gaze Distance	23.71	64.29	201.79
	Mean Fixation Distance	16.08	38.89	228.53
Token	Mean Regression Count	1028.96	763.33	42.98
	Mean Token Distance	103.29	138.50	177.96
	Same Line Token Fixation Rate	82%	83%	43%

C. RQ3: Is Webcam-Based Eye Tracking Feasible and Accurate in an Unconstrained, Real-World Offsite Setting?

Our findings in RQ1 and RQ2 suggest that WebGazer performs reasonably well in onsite studies for semantic-focused code comprehension tasks. However, its accuracy declines when evaluated using general gaze metrics and token-level analysis. To further investigate this, we assessed the extent of noise introduced at both the general gaze level and the token level in onsite and offsite settings. Our results show that offsite WebGazer produces noisier eye-tracking data compared to both onsite WebGazer and Tobii. In contrast, onsite WebGazer produces noises that are comparable to those of Tobii. Table V summarizes the noise measurements for onsite WebGazer, offsite WebGazer, and Tobii. Specifically, mean gaze distance for offsite WebGazer was 160.30 pixels, compared to 64.29 for onsite WebGazer and 23.71 for Tobii. Similarly, mean fixation distance was 158.07 pixels offsite, nearly four times higher than onsite (38.89) and about ten times greater than Tobii (16.08), reflecting great dispersion in raw gaze and fixation data in the offsite condition. At the token level, the mean regression count was 1028.96 for Tobii and 763.33 for onsite WebGazer. Still, it dropped sharply to 44.86 for offsite WebGazer, suggesting that elevated noise likely disrupted the detection of both initial and regressive fixations. Mean token distance further highlights the increased dispersion in offsite data. In terms of same line token fixation rate, Tobii and onsite WebGazer both achieved high rates (82% and 83%, respectively), while offsite WebGazer dropped significantly to 32%, probably due to vertical noise in the offsite WebGazer data making it seem as though participants are reading above or below a line they are scanning.

Thus, while WebGazer's noise levels are reasonably comparable to Tobii's in controlled onsite settings, its accuracy degrades significantly in uncontrolled, offsite environments, where variability in hardware, lighting, and user behavior may introduce substantial measurement noise.

> Compared to onsite use, WebGazer can introduce increased noise in offsite experiments, underscoring the importance of caution when employing webcam-based eye trackers for offsite code comprehension studies.

D. RQ4: What Are Participants' Perceptions of Self-Setting Up The Webcam-Based Eye-Tracking?

1) Survey: The experimental setup was generally perceived as conducive to concentration, with most of the onsite participants (33 out of 47) and almost all the offsite participants (9 out of 10) reporting minimal distractions. The setup instructions effectively guided participants through the process, with all participants reporting high confidence in independently configuring the webcam-based eye tracker. They also found the instructions clear and easy to follow. The quality of the instructions was rated highly by participants, with onsite participants giving an average score of 4.7 out of 5, and offsite participants rating it 4.2 out of 5. Additionally, 45 out of 47 onsite participants and 9 out of 10 offsite participants indicated they would recommend the instructions to future researchers, reflecting strong overall endorsement. However, a potential area for improvement lies in the format of the instructions. A small minority (4 onsite participants and 1 offsite participant) reported that they did not find the instructions particularly user-friendly. In addition, 61.7% of participants did not feel fully engaged because the motivation for the tasks and the study's goals were not clearly explained.

2) Interview: During the interviews, both onsite and offsite participants shared additional insights about their experiences and offered suggestions for improving the self-setup process of the study. Concerning the setup instruction, six participants recommended enhancing the formatting by grouping sequential steps on the same page, which they believed would reduce confusion and improve the ease of following the instructions, particularly for complex processes. Additionally, a significant majority (89.4%) emphasized the effectiveness of visual aids in the instructions in enhancing their comprehension of the steps involved.

Overall, 73.5% of participants reported a positive experience. However, a primary challenge identified was the initial software setup, especially when starting the tasks. Given that 18 participants were unfamiliar with this process and preferred using an IDE to run the script, simplifying the setup process, such as by publishing the tasks online, may be more user-friendly and facilitate participation. Another challenge identified was the calibration process, where 17 participants struggled to achieve satisfactory results and were uncertain how to improve their calibration scores. Suggestions for improvement include refining the calibration process to enhance user-friendliness and reduce the likelihood of errors. Additionally, the instructions could be augmented with more detailed guidance on this aspect. Additionally, two participants reported that the flashing of the webcam was distracting, while four others found it challenging and tiring to remain stationary for extended periods during the code comprehension task. Addressing these concerns could also enhance the overall comfort and concentration of participants in the experiment.

> The setup instructions were helpful, with participants recommending them for future webcam-based eye-tracking research, albeit with format improvements. Most participants reported a positive experience, but challenges were reported in initiating tasks, calibration, distractions, and discomfort due to prolonged sitting.

V. Limitations and Threats to Validity

A. Experiment Setup

In our onsite study, we aimed to directly compare the data obtained from Tobii and WebGazer by requiring participants to complete the task in a controlled experimental room using the same laptop. This approach allowed us to minimize variations in the experimental conditions and ensure a consistent comparison between data captured by two eye trackers. However, it is important to acknowledge that this experimental setup does not accurately reflect the conditions of a real-world offsite study. In a typical offsite study, participants would use their own devices in various environments, leading to a wide range of variables that could affect the data, such as different screen sizes, lighting conditions, and levels of distractions. As a result, our findings in RQ1 and RQ2 may not be directly applicable to offsite eye-tracking studies conducted in more naturalistic settings. To mitigate potential limitations, we opted for a laptop with a standard webcam. Moreover, participants were instructed to complete all tasks independently, without any interruptions from researchers. In addition, we performed an independent offsite study involving ten participants to assess the extent of noise introduced in a real-world remote eye-tracking experiment.

B. Generalization

All participants recruited in the onsite study had a similar level of experience with C++ programming, ranging from 1 to 3 years. This homogeneous group may not accurately reflect the diverse eye-tracking patterns of novice or advanced programmers, as previous studies have shown differences in code comprehension patterns between experts and novices [10]. Additionally, the narrow age range of our participants could limit the generalizability of our findings to a wider population, as age-related factors may impact the collected eye-tracking data [67]. To mitigate these limitations, the offsite study incorporated a broader participant pool, including individuals with more varied programming experience and age.

Additionally, our tasks involved relatively simple C++ programs, which may not capture the complexity and cognitive demands of real-world software development. In industrial settings and on more complex tasks with experienced developers, the webcam's lower spatial accuracy at the token level may become more consequential, especially for dense, unfamiliar APIs or multi-file algorithms where brief, overlapping fixations span many small code elements. Under these conditions, token-level AOIs can amplify alignment errors (e.g., boundary slips, rapid saccades across symbols), potentially biasing inferences about attention allocation to identifiers, operators, or control-flow markers. Future work should recruit professional developers, replicate on larger API-heavy codebases across multiple languages, and expand datasets to strengthen evaluation and improve the robustness of webcam-based eye tracking.

VI. Discussion

A. Onsite vs. Offsite Eye-Tracking Studies

Off-site eye tracking expands accessibility and convenience, enabling broader and more diverse participation and thereby improving the generalizability of findings in eye-tracking research [68]. However, our results indicate that webcam-based trackers commonly used offsite can compromise data accuracy and may yield different conclusions than on-site studies that use specialized, screen-mounted hardware.

In our experience, WebGazer's spatial error was smaller at the semantic level than at the token level (127.3 pixels vs. 192.5 pixels). It may be because larger AOIs are inherently more tolerant of spatial noise and lower sampling rates, so semantic AOIs yield more stable inferences about code-comprehension patterns (e.g., regressions, element weights, and cognitive load within specific semantic regions). Mechanistically, this pattern reflects known limitations of webcam tracking: lower sampling rates and regression-based, monocular gaze estimation introduce more spatial jitter and drift than higher-rate, stereo, screen-mounted trackers. Consequently, threshold-based event detection can overcount fixations and/or shorten their durations. These artifacts are attenuated when AOIs are larger (semantic level) but amplified at token granularity. Accordingly, token-level conclusions from webcam data should be interpreted with caution, and semantic-level analyses are generally more reliable with webcam-based trackers. However, it is worth noting that off-site variability (e.g., heterogeneous cameras, lighting, and head-pose changes) further increases noise, as shown in Sec. IV-C. Thus, absent substantial advances in webcam methods, screen-mounted eye trackers remain the preferred choice when fine spatial precision is required.

Nevertheless, onsite studies are limited by geographical constraints and require more setup and resources. Therefore, when choosing between offsite and onsite eye-tracking studies in programming research, the study's goals, required data precision, and available resources should be considered and tradeoff. Based on our findings, we recommend that eye-tracking research in SE may be more suitable for offsite tracking when focusing on high-level program aspects and requiring a large participant pool. However, to minimize data quality issues, it is essential to provide explicit instructions regarding participants' hardware and environmental setup. Our offsite study revealed that in the absence of such guidance, such as specifying the type of laptop to use or the environment in which to complete the experiment, significant noise was introduced into the eye-tracking data. Therefore, while offsite deployment can be practical, researchers must exercise caution and implement appropriate controls to ensure data validity. Furthermore, for a detailed understanding of programmers' focus on specific program parts, conducting an onsite eye-tracking study is still advised. Thus, there is still considerable room for improvement in webcam-based eye tracking technology, particularly to enhance its reliability in remote and uncontrolled settings. Advancing this technology would not only increase convenience by removing the need for participants to visit a physical lab but also improve accessibility by making it easier for individuals with disabilities to participate in eye-tracking studies focused on code comprehension. To support continued development and research, we will make our eye-tracking dataset publicly available as an open-source resource.

B. Challenges in Webcam-Based Eye-Tracking Experiments and Solutions

To mitigate these issues in remote eye-tracking studies, researchers should establish clear lighting and camera placement requirements and record basic hardware and environmental metadata to enable covariate control. They should also predefine and enforce exclusion criteria based on calibration quality and environmental thresholds. Furthermore, subgroup comparisons (e.g., by gender or experience) based solely on webcam data should be interpreted with caution, given the false positives observed with webcam-based eye trackers (see Sec. IV-B) and the additional noise across participants and contexts in offsite settings, which can further increase the risk of spurious differences.

In addition, pre-study setup and calibration can impose the greatest burden on remote participants. Technical prerequisites (e.g., configuring terminals or IDEs) and multi-step procedures increase drop-offs. Streamlining the workflow, such as hosting tasks and data collection online, consolidating setup steps on a single page, adding visual aids, and using simpler or self-calibrating trackers, can help reduce burden and improve data quality. Clear, explicit guidance on how lighting, camera placement, and posture affect eye-tracking accuracy is also beneficial. We observed that participants who understood these factors adhered more closely to the protocol (see Sec. IV-D), which in turn improved data quality, particularly in offsite settings.

Guidelines for improving data quality. In summary, our guidelines for improving data quality in offsite studies are: (1) set explicit environment requirements (e.g., lighting and camera-placement); (2) consolidate steps to simplify study setup and reduce tool switching; (3) prefer simpler or self-calibrating webcam-based eye trackers; (4) provide one-page visual setup guide for onboarding, and include brief explanations of factors that affect accuracy so participants are aware.

C. Improving Participant Experience

Building on our findings, participants struggled to get started on tasks, encountered notable friction during calibration, felt uncertain about the study's goals and their role, and reported physical discomfort from prolonged sitting. To address these barriers and support sustained, satisfying participation in webcam-based eye-tracking studies, we recommend practices that actively foster motivation, increase comfort, and improve understanding. Motivation can be increased with light-touch gamification that signals progress without pressure: for example, a clear progress bar and brief "you're good to go" confirmations that make repetitive steps feel purposeful. In the participant-facing UI, use brief, just-in-time prompts and simple visual cues that signal progress and clarify next steps to sustain motivation. Comfort should be supported both physically and cognitively. Physically, design sessions to avoid prolonged stillness, encourage posture resets, and schedule short breaks between tasks; cognitively, maintain a consistent interface, minimize context switching, and decompose multi-step procedures into smaller, guided actions. Moreover, provide small, empowering controls, such as pausing between tasks or adjusting camera framing with a live preview, to enhance comfort and a sense of control. Finally, well-informed participants are more likely to trust the research process and follow procedures accurately. To support this, provide clear explanations during self-setup and task execution. These explanations should cover what each step involves, why it is necessary, and how the data will be used. It is also important to communicate the overall purpose of the study, how gaze data will be processed and stored, and what information will or will not be inferred from their participation.

Guidelines for improving participants' experience. In summary, our guidelines for improving participants' experience in offsite webcam-based eye tracking studies are: (1) use lightweight progress cues and timely UI prompts to motivate participants; (2) design for physical comfort (e.g., allow posture resets, include scheduled breaks); (3) reduce cognitive load through consistent interfaces and step-by-step guidance; (4) ensure transparency about the study's purpose, data use, storage, and what will not be inferred.

D. Algorithm Improvement

In addition to researcher interventions in webcam-based eye-tracking studies, advances in eye-tracking technology remain essential. In our study, we found that WebGazer showed comparable accuracy to Tobii in capturing semantics during code comprehension tasks. This indicates that WebGazer has the potential to reveal how programmers interact and comprehend

the code. However, it still struggled with identifying specific tokens and did not align well with Tobii's results overall. Moreover, in our offsite experiment, even with participants following the instructions, the data still contained significant noise. Its performance also appeared to be largely affected by variability in participants' environments and hardware configurations. To enhance WebGazer's performance, future work could focus on refining its algorithm to better capture detailed code syntax, possibly by integrating natural language processing and machine learning methods. Additionally, developing a more advanced calibration method tailored to programming-related tasks may also improve gaze tracking accuracy, as traditional methods may not address the unique eye movement patterns associated with reading and understanding code. To improve reliability in diverse, uncontrolled environments, future work could also explore real-time adaptation mechanisms that respond to varying lighting conditions, screen sizes, and camera qualities—for example, through dynamic brightness normalization and adaptive face tracking. Furthermore, incorporating hardware-aware calibration models that adjust to individual device configurations may help mitigate errors introduced by low-resolution webcams or inconsistent display setups. By enhancing WebGazer's performance, we can provide SE researchers with a reliable tool for conducting webcam-based eye-tracking studies in programming.

VII. Conclusion

The pandemic and the challenge of recruiting a large number of participants with specialized SE expertise have accelerated the transition to conducting eye-tracking studies remotely. Webcam-based eye trackers present a convenient alternative for conducting offsite eye-tracking research and enabling programmers to participate in code comprehension studies from remote locations. However, a critical concern is whether these webcam-based eye trackers can produce conclusions comparable to those derived from high-accuracy research-focused eye trackers. Our study indicates that the webcam-based eye tracker WebGazer performs reliably in capturing code comprehension patterns at the semantic level in onsite settings. However, it struggles to accurately detect gaze at the token level. Additionally, webcam-based trackers may occasionally produce false positives when identifying individual differences. The elevated noise levels in offsite settings further undermine the reliability of webcam-based trackers for drawing precise conclusions. Participants also commonly experienced challenges with calibration and initiating tasks. In this study, we also provide recommendations for future research directions, including developing and enhancing algorithms for webcam-based eye tracking to better capture the nuances of visual patterns in programming. Additionally, we propose guidelines for conducting offsite webcam-based eye-tracking studies in SE, tailored to various research objectives.

Data Availability

Related data is available via our online appendix: https://anonymous.4open.science/r/webcam-6C3B/

References

[1] 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.*, vol. 25, pp. 3128–3174, Sep. 2020.

[2] A. Poole and L. J. Ball, "Eye tracking in HCI and usability research," in *Encyclopedia of Human Computer Interaction.* IGI Global Scientific Publishing, 2006, pp. 211–219.

[3] R. J. Jacob and K. S. Karn, "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises," in *The Mind's Eye.* North-Holland: Elsevier, 2003, pp. 573–605.

[4] P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 390–401.

[5] T. Busjahn, et al., "Eye movements in code reading: Relaxing the linear order," in *Proc. IEEE 23rd Int. Conf. Program Comprehension*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 255–265.

[6] M. E. Crosby, J. Scholtz, and S. Wiedenbeck, "The roles beacons play in comprehension for novice and expert programmers," in *Proc. PPIG*, 2002, p. 5.

[7] W. Maalej, R. Tiarks, T. Roehm, and R. Koschke, "On the comprehension of program comprehension," *ACM Trans. Softw. Eng. Methodol.*, vol. 23, no. 4, pp. 1–37, 2014.

[8] 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 *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2020, pp. 456–468.

[9] N. J. Abid, B. Sharif, N. Dragan, H. Alrasheed, and J. I. Maletic, "Developer reading behavior while summarizing java methods: Size and context matters," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. (ICSE)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 384–395.

[10] S. Jessup, S. M. Willis, G. Alarcon, and M. Lee, "Using eye-tracking data to compare differences in code comprehension and code perceptions between expert and novice programmers," in *Proc. 54th Hawaii Int. Conf. Syst. Sci. (HICSS)*, 2021, pp. 114–123

[11] E. Dumas-Mallet, K. S. Button, T. Boraud, F. Gonon, and M. R. Munafò, "Low statistical power in biomedical science: A review of three human research domains," *R. Soc. Open Sci.*, vol. 4, no. 2, 2017, Art. no. 160254.

[12] L. Berkowitz and E. Donnerstein, "External validity is more than skin deep: Some answers to criticisms of laboratory experiments," *Amer. Psychol.*, vol. 37, no. 3, p. 245, 1982.

[13] A. George and A. Routray, "Fast and accurate algorithm for eye localisation for gaze tracking in low-resolution images," *IET Comput. Vis.*, vol. 10, no. 7, pp. 660–669, 2016.

[14] G. Oliveira, J. Grenha Teixeira, A. Torres, and C. Morais, "An exploratory study on the emergency remote education experience of higher education students and teachers during the covid-19 pandemic," *Brit. J. Educ. Technol.*, vol. 52, no. 4, pp. 1357–1376, 2021.

[15] B. Eschman, J. T. Todd, A. Sarafraz, E. V. Edgar, V. Petrulla, and M. McNew, "Remote data collection during a pandemic: A new approach for assessing and coding multisensory attention skills in infants and young children," *Front. Psychol.*, vol. 12, 2022, Art. no. 731618.

[16] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, and J. Hays, "Webgazer: Scalable webcam eye tracking using user interactions," in *Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI)*. AAAI Press, 2016, pp. 3839–3845.

[17] J. J. MacInnes, S. Iqbal, J. Pearson, and E. N. Johnson, "Wearable eye-tracking for research: Automated dynamic gaze mapping and accuracy/precision comparisons across devices," *BioRxiv*, p. 299925, 2018.

[18] T. Anthonijsz, "Are high-end and webcam eye trackers comparable?," B.S. thesis, Utrecht University, Utrecht, The Netherlands, 2021.

[19] L. Burton, W. Albert, and M. Flynn, "A comparison of the performance of webcam vs. infrared eye tracking technology," *Proc. Human Factors Ergonom. Soc. Annu. Meeting*, vol. 58, no. 1, pp. 1437–1441, 2014.

[20] Z. Sharafi, Z. Soh, and Y.-G. Guéhéneuc, "A systematic literature review on the usage of eye-tracking in software engineering," *Inf. Softw. Technol.*, vol. 67, pp. 79–107, 2015.

[21] T. Busjahn, C. Schulte, and A. Busjahn, "Analysis of code reading to gain more insight in program comprehension," in *Proc. 11th Koli Calling Int. Conf. Comput. Educ. Res.*, 2011, pp. 1–9.

[22] Z. Sharafi, I. Bertram, M. Flanagan, and W. Weimer, "Eyes on code: A study on developers' code navigation strategies," *IEEE Trans. Softw. Eng.*, vol. 48, no. 5, pp. 1692–1704, May 2020.

[23] X. Li, W. Liu, H. Liu, J. Xu, and W. Cheng, "Task-oriented analysis on debugging process based on eye movements and ide interactions," in *Proc. 16th Int. Conf. Comput. Sci. & Educ. (ICCSE)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 379–384.

[24] C. S. Yu, C. Treude, and M. Aniche, "Comprehending test code: An empirical study," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Piscataway, NJ, USA: IEEE Press, Sep. 2019, pp. 501–512.

[25] Tobii, "Data quality resource center," 2024. Accessed: Feb. 21, 2024. [Online]. Available: https://www.tobii.com/resource-center/data-quality

[26] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, "How do professional developers comprehend software?," in *Proc. 34th Int. Conf. Softw. Eng. (ICSE)*, Piscataway, NJ, USA: IEEE Press, 2012, pp. 255–265.

[27] M.-A. Storey, "Theories, tools and research methods in program comprehension: past, present and future," *Softw. Qual. J.*, vol. 14, pp. 187–208, Sep. 2006.

[28] J. Johnson, S. Lubo, N. Yedla, J. Aponte, and B. Sharif, "An empirical study assessing source code readability in comprehension," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 513–523.

[29] Z. Sharafi, Z. Soh, Y.-G. Guéhéneuc, and G. Antoniol, "Women and men—Different but equal: On the impact of identifier style on source code reading," in *Proc. 20th IEEE Int. Conf. Program Comprehension (ICPC)*, Piscataway, NJ, USA: IEEE Press, 2012, pp. 27–36.

[30] P. L. Li, A. J. Ko, and J. Zhu, "What makes a great software engineer?," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, vol. 1, 2015, pp. 700–710.

[31] M. Muñoz Barón, M. Wyrich, and S. Wagner, "An empirical validation of cognitive complexity as a measure of source code understandability," in *Proc. 14th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, 2020, pp. 1–12.

[32] R. Bednarik and M. Tukiainen, "An eye-tracking methodology for characterizing program comprehension processes," in *Proc. Symp. Eye Tracking Res. Appl.*, 2006, pp. 125–132.

[33] M. Adeli, N. Nelson, S. Chattopadhyay, H. Coffey, A. Henley, and A. Sarma, "Supporting code comprehension via annotations: Right information at the right time and place," in *Proc. IEEE Symp. Vis. Lang. Human-Centric Comput. (VL/HCC)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–10.

[34] D. Gopstein, A.-L. Fayard, S. Apel, and J. Cappos, "Thinking aloud about confusing code: A qualitative investigation of program comprehension and atoms of confusion," in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2020, pp. 605–616.

[35] P. Xu, K. A. Ehinger, Y. Zhang, A. Finkelstein, S. R. Kulkarni, and J. Xiao, "TurkerGaze: Crowdsourcing saliency with webcam based eye tracking," 2015, *arXiv:1504.06755.*

[36] A. Papoutsaki, J. Laskey, and J. Huang, "SearchGazer: Webcam eye tracking for remote studies of web search," in *Proc. Conf. Human Information Interact. Retrieval*, 2017, pp. 17–26.

[37] H. Skovsgaard, J. S. Agustin, S. A. Johansen, J. P. Hansen, and M. Tall, "Evaluation of a remote webcam-based eye tracker," in *Proc. 1st Conf. Novel Gaze-Controlled Appl.*, 2011, pp. 1–4.

[38] S. Hutt and S. K. D'Mello, "Evaluating calibration-free webcam-based eye tracking for gaze-based user modeling," in *Proc. Int. Conf. Multimodal Interact.*, 2022, pp. 224–235.

[39] S. A. Stone and C. S. Chapman, "Unconscious frustration: Dynamically assessing user experience using eye and mouse tracking," in *Proc. ACM Human-Comput. Interact.*, vol. 7, no. ETRA, 2023, pp. 1–17.

[40] S. Hutt, A. Wong, A. Papoutsaki, R. S. Baker, J. I. Gold, and C. Mills, "Webcam-based eye tracking to detect mind wandering and comprehension errors," *Behavior Res. Methods*, pp. 1–17, Jan. 2024.

[41] K. Wisiecka et al., "Comparison of webcam and remote eye tracking," in *Proc. Symp. Eye Tracking Res. Appl.*, 2022, pp. 1–7.

[42] C. Zheng and T. Usagawa, "A rapid webcam-based eye tracking method for human computer interaction," in *Proc Int. Conf. Control, Automat. Inf. Sci. (ICCAIS)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 133–136.

[43] A. Bánki, M. de Eccher, L. Falschlehner, S. Hoehl, and G. Markova, "Comparing online webcam-and laboratory-based eye-tracking for the assessment of infants' audio-visual synchrony perception," *Front. Psychol.*, vol. 12, 2022, Art. no. 733933.

[44] N. B. Mogilever et al., "Expedition cognition: A review and prospective of subterranean neuroscience with spaceflight applications," *Front. Hum. Neurosci.*, vol. 12, p. 407, 2018.

[45] M. Nadj, A. Maedche, and C. Schieder, "The effect of interactive analytical dashboard features on situation awareness and task performance," *Decis. Support Syst.*, vol. 135, 2020, Art. no. 113322.

[46] A. S. Alghamdi, A. Al-Badi, R. Alroobaea, and P. Mayhew, "A comparative study of synchronous and asynchronous remote usability testing methods," *Int. Rev. Basic Appl. Sci.*, vol. 1, no. 3, pp. 61–97, 2013.

[47] "Tobii pro fusion user manual," Tobii Technology. Accessed: Feb. 18, 2024. [Online]. Available: https://go.tobii.com/tobii-pro-fusion-user-manual

[48] X. Yang and I. Krajbich, "Webcam-based online eye-tracking for behavioral research," *Judgment Decis. Making*, vol. 16, no. 6, pp. 1485–1505, 2021.

[49] A. Gudi, X. Li, and J. van Gemert, "Efficiency in real-time webcam gaze tracking," in *Proc. Comput. Vis. ECCV 2020 Workshops*, Glasgow, U.K. New York, NY, USA: Springer, 2020, pp. 529–543.

[50] T. Robal, Y. Zhao, C. Lofi, and C. Hauff, "Webcam-based attention tracking in online learning: A feasibility study," in *Proc. 23rd Int. Conf. Intell. User Interfaces*, 2018, pp. 189–197.

[51] J. Cesario, "What can experimental studies of bias tell us about real-world group disparities?," *Behavioral Brain Sci.*, vol. 45, 2022, Art. no. e66.

[52] J. Siegmund et al., "Understanding understanding source code with functional magnetic resonance imaging," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 378–389.

[53] Y. Zhang et al., "Eyetrans: Merging human and machine attention for neural code summarization," 2024, *arXiv:2402.14096.*

[54] C. Wallace, "The semantics of the C++ programming language," in *Specification and Validation Methods*, E. Börger, Ed. Oxford: Oxford Univ. Press, 1995, pp. 131–164.

[55] D. S. Rosenblum and A. L. Wolf, "Representing semantically analyzed C++ code with reprise," in *Proc. C++ Conf.*, 1991, pp. 119–134.

[56] B. Stroustrup, *Programming: Principles and Practice Using C++*. Pearson Educ., Addison-Wesley Professional, 2014.

[57] J. Li, Y. Zhang, Z. Karas, C. McMillan, K. Leach, and Y. Huang, "Do machines and humans focus on similar code? Exploring explainability of large language models in code summarization," 2024, *arXiv:2402.14182.*

[58] A. Bansal et al., "Modeling programmer attention as scanpath prediction," in *2023 38th IEEE/ACM Int. Conf. Automat. Softw. Eng. (ASE)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1732–1736.

[59] N. Tang et al., "An empirical study of developer behaviors for validating and repairing ai-generated code," in *Proc. 13th Workshop Intersect. HCI PL (PLATEAU)*, 2023, pp. 1–15.

[60] H. Ahmad, Z. Karas, K. Diaz, A. Kamil, J.-B. Jeannin, and W. Weimer, "How do we read formal claims? Eye-tracking and the cognition of proofs about algorithms," in *Proc. IEEE/ACM 45th Int. Conf. Softw. Eng. (ICSE)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 208–220.

[61] M. Borys and M. Plechawska-Wójcik, "Eye-tracking metrics in perception and visual attention research," *EJMT*, vol. 3, no. 16, pp. 11–23, 2017.

[62] B. Birawo and P. Kasprowski, "Review and evaluation of eye movement event detection algorithms," *Sensors*, vol. 22, no. 22, p. 8810, 2022.

[63] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 522–532, May 1998.

[64] J. Ramos et al., "Using TF-IDF to determine word relevance in document queries," in *Proc. 1st Instructional Conf. Mach. Learn.*, vol. 242, no. 1, pp. 29–48, 2003.

[65] Z. Sharafi, T. Shaffer, B. Sharif, and Y.-G. Guéhéneuc, "Eye-tracking metrics in software engineering," in *Proc. Asia-Pacific Softw. Eng. Conf. (APSEC)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 96–103.

[66] U. Obaidellah and M. A. Haek, "Evaluating gender difference on algorithmic problems using eye-tracker," in *Proc. ACM Symp. Eye Tracking Res. & Appl.,* 2018, pp. 1–8.

[67] J. C. Romano Bergstrom, E. L. Olmsted-Hawala, and M. E. Jans, "Age-related differences in eye tracking and usability performance: website usability for older adults," *Int. J. Human-Comput. Interact.*, vol. 29, no. 8, pp. 541–548, 2013.

[68] J. K. Bertrand, "The utility of webcam eye-tracking: Lessons learned and practical applications in decision-making and digital human-computer interactions," 2023.