Leveraging Fixation Transition Patterns and Targeted Regions of Interest for Analyzing Code Comprehension

Md Shakil Hossain
Computer Science
Georgia Southern University
Statesboro, USA
mh34922@georgiasouthern.edu

Noushin Gauhar
Computer Science
Georgia Southern University
Statesboro, USA
ng06503@georgiasouthern.edu

Rushmila Shabneen
Computer Science
Georgia Southern University
Statesboro, USA
rs24222@georgiasouthern.edu

Andrew Allen
Computer Science
Georgia Southern University
Statesboro, USA
andrewallen@georgiasouthern.edu

***Abstract*—This research-to-practice full paper describes a study on specific eye metrics data and the possible correlations with students' comprehension level. There is a consensus understanding in academia that providing feedback early and often significantly helps students as problems are identified and remedied early. However, as class sizes increase, this becomes more difficult to achieve. Code comprehension techniques can be helpful for educators by identifying students needing assistance and guide them toward academic success and mastery of coding concepts. We hypothesize that by analyzing students' gaze patterns in certain regions of the code, educators can assess and address students' coding challenges. The study distinguishes between frequently and infrequently fixated code regions, using eye metrics to assess students' error perception and challenges. We examined eye-tracking during coding comprehension exercises and proposed a systematic method to identify students needing help based on fixation patterns within Targeted Regions of Interest (TROIs). The methodology uses coding exercises seeded with errors, grades on the tasks, student feedback, measured fixation counts, and average fixation durations of the students' eyes within TROIs of the code compared to the rest code. We conducted preliminary experiments using Java code with 10 students in an introductory programming course. Our initial findings from the collected eye-tracking data showed that the high-scoring students fixated more on the TROIs, frequently backtracked to the method definition, and skimmed the code from top to bottom. The absence of these patterns could be early indicators of code comprehension challenges.**

***Index Terms*—Eye Tracking, Region of Interest, Student Comprehension, Classroom Observational Study, Amazon Textract**

I. Introduction

Assessing student comprehension poses a persistent challenge, especially in classrooms where students spend significant time behind monitors. This task is even more challenging due to factors like large classes, limited time, and grading resources, hindering early identification of students who may need help. Analyzing where a student looks during a code comprehension task offers insights into what information they find important, allowing the instructor to assess if the student is focusing on the right areas. We investigated differences in how a student's eyes traverse code during a coding comprehension exercise and propose a systematic method to distinguish between students with a good understanding of the exercise and those who need additional help. We introduce "Targeted Regions of Interest" (TROIs) in eye tracking, allowing precise analysis of specific code areas defined by (x, y) coordinates, and a ratio comparison of TROIs and non-TROIs to provide insights into student understanding and facilitating comparisons among students or groups with varying reading pattern. Researchers have investigated the application of machine learning and computer vision techniques to assess students' attentiveness as a proxy for understanding, with acknowledged limitations in its efficacy [1], [2]. Researchers have also investigated eye-tracking metrics, particularly average fixation duration, in understanding the cognitive processes associated with perceiving and processing information [3], [4]. This study advocates for tools that offer teachers access to eye metrics data correlated with students' comprehension levels. To that end, this work builds on prior research [5], [6] to extend to predictive analyze of student comprehension focusing on improving the earning in introductory programming. This study utilized tracked eye movements, post analysis of code, and a multilayered approach to calculate gaze metrics against TROIs, to gain insights into students' comprehension patterns and identifying challenging content topics. We observed that the ratio of fixations in TROIs vs non-TROIs were higher for higher scoring students.

II. Background

Eye trackers are commonly used by researchers to examine the cognitive processes associated with comprehension from the subject's perspective. In this research, we will use the following terms in the context of comprehension studies that make use of eye tracking hardware:

- Eye gaze data: The immediate direction of a person's eyes when staring at a computer monitor is converted to (x,y) coordinates.
- Fixations: The stability of the eye for a period of time on a specific portion of a stimulus, which typically lasts between 200 and 300 ms [3].
- Saccades: The brief (usually 50 ms) and continuous eye movements between fixations [3].
- Region of Interest (ROI) : A particular area or region of the computer screen that has been designated for any reason. These areas are commonly described as an (x,y) coordinate pair in comprehension studies.
- Targeted region of interest (TROI): TROI refers to a specific area or areas of a visual scene that a researcher or analyst deliberately selects as the focus of their study. This is in contrast to a broader, more general ROI that may cover a larger area of the scene.

Previous studies employed diverse methodologies to gather data and gain insights into participant's code comprehension. Researchers utilized an eye tracker and a non-intrusive Heart Rate Variability (HRV) monitor to predict the outcomes of code reviews [7]. By gathering biometric data during participants' code review sessions, they assessed the quality of the reviews based on bug identification. Their findings demonstrated the tool's ability to predict bad reviews for medium and complex programs with an accuracy range of 75%–87%. In another study, the eye movements of experienced programmers while summarizing source code were explored [8]. Analyzing the eye movements of participants, the researchers discovered that programmers tended to skim source code rather than read it thoroughly. The eye movement patterns were remarkably similar to those of natural language readers, and programmers often scanned sections rather than following a traditional left-to-right or top-to-bottom reading pattern. During an experiment with 15 experienced programmers, [9] used data from eye tracking, electrodermal activity, and electroencephalography sensors to predict the perceived challenge of a task. Their classifier achieved a precision of 64.99% and a recall of 64.58%, successfully estimating whether a task would be perceived as simple or complex for novice developers. Inspired by these code review settings this study involves students in a programming course and their code review sessions from eye movements perspective.

Similar to this study's experimental setup, [4] used Tobii Eye Tracker 4c and employed self-reporting to predict student attention during class. Placing eye trackers on computer monitors in a computer lab, they had students occasionally rate the lecturer's engagement. By combining self-reported data and gaze data, the researchers developed a machine-learning model capable of predicting the level of student engagement throughout the presentation. Tobii Glasses were also used to study students' eye movements during a presentation on physical science [10]. Their findings indicated that students paid attention when the professor displayed emotion, drew something, used humor, or made comparisons not on presentation slides. The study highlighted the impact of visual elements, such as illustrations or diagrams, in maintaining or diverting students' focus. Tobii Eye Trackers have been successful in gaining eye movements and our data collection methodology heavily involves Tobii Eye Tracker Instruments.

In a prior study, [11] employed sensors on wearable computers to capture hand movements and heart activity, predicting "interest level" and "judgment of difficulty." Through two lectures with fourteen topic sessions, the study revealed that wrist-worn smartwatches provided good accuracy for attention monitoring. The study suggested that other physiological sensors could potentially enhance accuracy further. Addressing attention detection approaches and categorizing them into automatic, semi-automatic, and manual strategies, [12] highlighted the drawbacks of each approach, with automated methods requiring further development, semi-automatic techniques relying on probabilistic reasoning, and manual procedures being more prone to bias and time-consuming.

Tabassum, Allen, and De [1] employed a deep learning convolutional neural network and a cloud-based emotion detection service to find correlations between emotions and attentiveness. Using webcam images of students during class lectures, they trained a convolutional neural network to identify facial expressions and emotions. The study demonstrated that facial emotions can enhance the accuracy of attention detection algorithms, identifying inattentive students with 93% accuracy.

This research utilizes real-time eye-tracking data from an actual classroom setting. The study conducts post-hoc analysis of the collected data to extract transition patterns, facilitating comparisons between participants with varying grades. These transition patterns offer valuable insights into code comprehension.

III. Methodology and framework

NiCATS collects the necessary data for the studies conducted in this research. The NiCATS system is especially important to this research because it is the experimental setup for collecting data and designing experiments.

A. Data Collection High Level Design

The high-level architecture of the NiCATS application is shown in Fig. 1. During student sessions, the system collects data, such as webcam images, the gaze points of eye movements, screenshots, and EEG signals. The webcam captures images every 5 seconds, and gaze points are grouped into blocks depending on user input. Screen captures are taken each time the grouped gaze points are sent, and EEG signals are recorded using the NeuroSky MindSet system, which measures attention, meditation, and eye blinks, among other things [13].

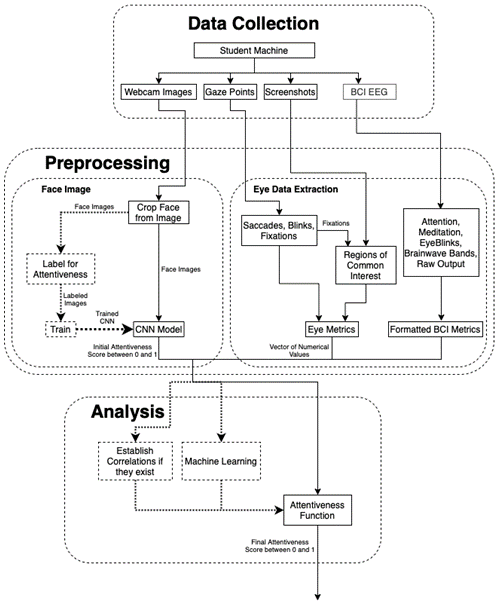

Fig. 1: High-Level Design of NiCATS

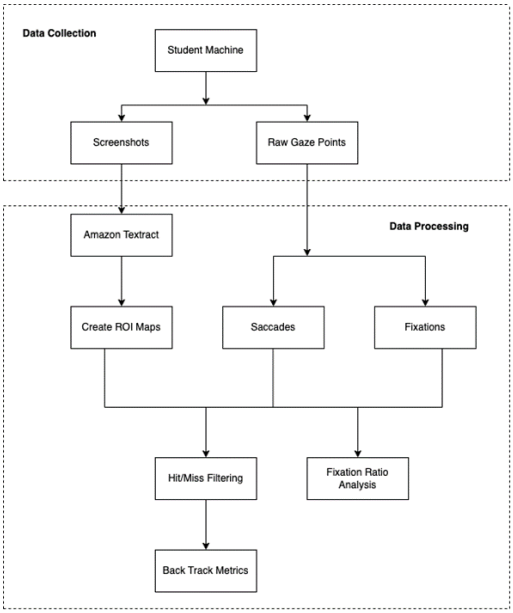

Fig. 2: High-level overview of the post-processing workflow

In this research, only the screenshots and gaze data gathered by the system were used for the analysis to provide details about the students' comprehension patterns:

- Screenshots: For each code review exercise, a screenshot of the student's computer screen was taken every 15 seconds or whenever the student altered the screen's content (e.g., mouse click, scroll wheel). A minimum threshold of 1 second was used in order to avoid the sending of too many screenshots.
- Gaze Data: The raw (x,y) coordinates of where a student is looking as measured by an eye tracker. The inclusion of a timestamp in each gaze coordinate allows for synchronization of the gaze points with each screenshot.

B. Post Processing Workflow

Post-processing steps were conducted after each trial to a) extract eye metrics from raw gaze data and b) generate useful visual aids for comprehension analysis. A high-level overview is presented in Fig. 2, with more details in the experiment-specific sections. The workflow involves collecting screenshots and raw gaze points, followed by post-hoc processing using Amazon Textract to define ROI boundaries in screenshots. Simultaneously, raw gaze points are transformed into meaningful eye metrics like fixations and saccades. Utilizing these metrics, we generate transition data between ROIs, analyzing fixation patterns and computing backtracking transition metrics for nuanced insights into students' code comprehension.

Furthermore, employing backtracking transitions allows for performance comparisons across students or groups with varying reading speeds or coding approaches vs timed metrics. For instance, a novice student might spend more time on a specific code area than an experienced one, but analyzing fixation transition patterns enables the comparison of their relative focus and attention on that area. Incorporating backtracking transition properties offers a comprehensive understanding of a student's performance, enabling valuable research conclusions.

The experiments were designed for hands-on programming exercises, with the overarching goal of assessing the potential to support instructors in gauging students' understanding and identifying common difficulties. Hands-on programming exercises were chosen for their comprehensive and statistically useful data set. Seeded error regions were identified as TROIs, and fixation statistics(count, average, and ratio) were analyzed using the NiCATS software to provide meaningful insights for answering research questions.

IV. Research Goal & Experimental Setup

The goal of this research was to explore how teachers can utilize gaze metrics and screenshots to gain insights into the comprehension processes of students during code review tasks. The research aimed to understand how the acquisition of knowledge by students can be assessed using eye-tracking and screenshots of the information presented to them. This study investigates the following research question:

RQ1: What, if any, patterns in eye tracking data can be used to identify successful or unsuccessful students in code comprehension exercises?

To address this question, we will involve an analysis of participants' derived eye-tracking metrics, including fixations.

A. Experiment Setting

In an experiment with volunteer undergraduate computer science students, enrolled in the second course of an introductory programming course, each participant's computer was equipped with the necessary hardware and NiCATS software.

The task involved presenting participants with three Java programs, requiring them to identify error-containing line numbers and provide explanations. The aim was to gain insights into how eye tracking and NiCATS software contribute to understanding code comprehension patterns among students.

B. Artifacts

In our study, participants with varying degrees of programming experience (six months to one year of Java programming) assessed three Java programs using complexity metrics, line numbers, and seeded flaws. Java was chosen for its familiarity to participants. The code examples had errors of different difficulty levels and relevance to the course material. The goal was to evaluate NiCATS's ability to help instructors measure comprehension patterns across diverse error types and code structures.

C. Experiment Setup

The following section outlines the technique for setting up the experiment.

1) *Step 1 - Calibration and NiCATS Client Setup:* Participants calibrated the eye tracker using Tobii Eye Tracking Core Software, ensuring precise eye-tracking data. Following on-screen instructions upon login, the software saved the calibration profile for consistent and precise eye-tracking throughout the experiment. After that, participants downloaded, installed, and launched the NiCATS client software. During this step, a transparent pop-up window provided the option to "opt-in" or "opt-out," ensuring full awareness, informed decision-making, trust, and voluntary participation.
2) *Step 2 - Initialize Data Collection:* Experiment data was gathered by initiating a NiCATS web client recording session. This activated data collection on all participant machines for those who opted in, continuously storing raw data, including gaze-point coordinates and screenshots, on the server.
3) *Step 3 - Program Presentation:* Participants spent five minutes reviewing each Java program, identifying error-containing line numbers, and providing explanations. They navigated using the scrollbar or arrow keys, with enforced fullscreen mode to minimize distractions and maintain consistent ROI layouts.
4) *Step 4 - Gathering Responses and Data Collection:* After reviewing each Java program, NiCATS software processed the data, extracting fixation metrics for each ROI around line numbers and text, and the recording session concluded with students instructed to close the software after reviewing three code samples.

D. Data Processing

Simply collecting raw gaze points and screenshots during an experiment is not enough, as they hold little meaning on their own. To extract valuable insights and draw meaningful conclusions, the collected data must undergo post-hoc processing. This is especially true for both the screenshots and the raw gaze points, which require specific techniques and analyses to reveal important patterns and behaviors.

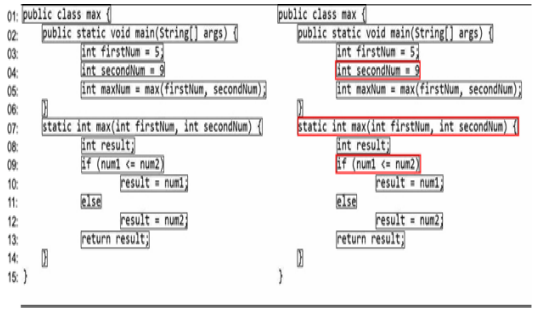

Fig. 3: A line based ROI and TROI map for analysis

1) *Defining ROI and TROI Boundaries:* In the uniform experimental setting, where participants reviewed programs in fullscreen mode with the same resolution, a singular screenshot was chosen for each question, and ROI boundaries were assigned. ROIs were manually defined for each question with three granularities: complexity-based, line-based, and lexer-based. Seeded error lines were labeled as TROIs during post-hoc analysis for fixation ratio and backtrack pattern analysis. Fig. 3 illustrates a line-based ROI map with TROI markers indicated by red markers on the right side for comparison visualizations.
2) *Number of Tokens:* For each ROI we calculate the number of tokens contained within. The tokens are collected using a lexical analyzer which classifies each token by token type (e.g, Keyword, Operator, etc.).
3) *Gaze Data to Fixations:* In the context of ROI maps, extracting meaningful eye metrics from raw gaze data requires examining and quantifying various components of eye movement patterns. Based on the study topic and the particular ROI map being used, it will be possible to determine the precise metrics that are relevant.
    a) *Fixations:* Using the I-DT algorithm [14] with a maximum dispersion of 1 degree and a duration threshold of 300 milliseconds or more, fixations were calculated from raw gaze data. The resulting fixation coordinates underwent boundary checks against pre-established ROI maps to distinguish hits (within ROI boundaries) from misses (outside all ROI boundaries). Processed fixations included information like (x, y) coordinates, start and end times, fixation duration, and the specific region in which the fixation occurred.
4) *Additional Metrics:* From the hit/miss boundary detection results, we calculated essential metrics at the participant-to-question level for our analysis, including metrics for each ROI: a) *Number of Fixations:* The total number of times a participant fixated on an ROI, b) *Number of Backtracks:* The total number of times a participant revisited the ROI.

V. Results and Analysis

In our analysis, we focused on fixation metrics, excluding off-screen and clock-related data for accuracy. TROIs, defined based on participant grade categories with seeded error regions, served as key points. Comparing the Highest and Lowest Grade groups, we used fixation-based scan paths to gauge code comprehension.

To answer RQ1, the top 10 backtracking transitions between two ROIs of the subjects from Group A (highest grade) and Group D (lowest grade) are being considered. The behavior of backtracking transitions are being explained using three parameters: Start ROI (SROI), End ROI (EROI) and Frequency (Freq.). Two scenarios can be observed using EROI and SROI.

1) **Start and End ROI are the same:** If both ROI are same (i.e 1 –> 1 in Question 1, Q1) in any backtracking transition, it could indicate the following points: a) The subject fixated multiple times consecutively in that region, for example, two fixation happened in same region consecutively, b) The subject fixated in the region, left the region and fixated in whitespace/ceiling and immediately fixated back to that region.
2) **Start and End ROIs are different:** On the other hand if both ROI are different (i.e 3 –> 2 in Q1), this means the subject transitioned their fixation from the start ROI to the end ROI, indicating a change in focus between the two regions, as evidenced by two consecutive fixations in different regions.

The frequency parameter reflects how often the subject fixated on a region two times consecutively. If the frequency is n, it indicates the subject focused on that region for two fixations n times. In the analysis of code review exercises, the SROI and EROI with transitions and frequency are presented in Table I for a student of a participant group.

TABLE I: Sample sequence of backtracking transitions of a student in Q1

Subject 1			
SROI	EROI	Transitions	Freq.
1	1	**int maxNum = nax(firstNum, secondNum); –> int maxNum = nax(firstNum, secondNum);**	15
3	2	**public static void main(String[] args) { –> public static void main(String[]args) {**	6

The top 10 backtracking transitions were chosen based on their representation of the most frequent transitions between TROI pairs, aiding in analyzing comprehension patterns in coding exercises. Table II illustrates the significance of these top 10 transitions, ranging from 24-60% of the total frequency across Q1, Q2, and Q3. The study aimed to identify significant fixation points from a line or region perspective, with the top 10 transitions serving as a representative sample. For Q1 (15 lines), these transitions covered 52% of the regions (8 regions), for Q2 (57 regions) over 27% (16 regions), and for Q3 (37 lines) 42% (almost 16 regions).

TABLE II: Significance of Top 10 Backtracking Transition (BT)

Questions	Participant Group	Subjects	Total freq. of BT Pairs	Total freq. of Top 10 BT Pairs	Ratio(%)
Q1	A	Subject 1	220	108	49.091
	D	Subject 2	178	100	56.180
Q2	A	Subject 1	247	72	29.150
		Subject 2	527	126	23.909
	D	Subject 1	475	127	26.737
		Subject 2	527	137	25.996
		Subject 3	400	120	30.000
Q3	A	Subject 1	177	74	41.808
	D	Subject 1	110	37	33.636
		Subject 2	192	94	48.958

```
01: public class max {
02:     public static void main(String[] args) {
03:             int firstNum = 5;
04:             int secondNum = 9
05:             int maxNum = max(firstNum, secondNum);
06:     }
07:     static int max(int firstNum, int secondNum) {
08:             int result;
09:             if (num1 <= num2)
10:                     result = num1;
11:             else
12:                     result = num2;
13:             return result;
14:     }
15: }
```

Fig. 4: Code Review Exercise Questions 1 (Q1) with TROI

VI. Experiment: Code Review Exercise

The data from each of code reviews from the experiments are discussed in the following sections.

A. Code Review Exercise: Question 1 (Q1)

In Fig. 4, the red rectangle denotes the TROIs for Q1. Table III presents the Top 10 sequences of backtracking transitions between two Subject Groups, categorized by grading in Q1.

The top 10 most frequently used backtracking transitions among student groups highlight their progression in understanding the code, with higher-scoring students devoting more time to comprehending the logical aspect. In Q1, where lines

TABLE III: Top 10 sequence of backtracking transitions of the Group D and Group A students for Subject 1 in Q1

Group D(0% grade)			Group A(100% grade)		
Subject 1			Subject 1		
SROI	EROI	Freq.	SROI	EROI	Freq.
5	5	**22**	7	7	**22**
2	2	**18**	8	7	**9**
1	1	**15**	7	8	**8**
7	7	**12**	9	10	**6**
2	1	**7**	8	9	**6**
9	9	**7**	8	8	**6**
3	2	**6**	7	9	**6**
1	2	**5**	10	10	**6**
12	12	**4**	2	2	**6**
5	4	**4**			

TABLE IV: Top 10 sequence of backtracking transitions of the Group D (14% grade) and Group A (72% grade) subjects in Q2

Group D (14% grade)									Group A (72% grade)					
Subject 1			Subject 2			Subject 3			Subject 1			Subject 2		
SROI	EROI	Freq.	SROI	EROI	Freq.	SROI	EROI	Freq.	SROI	EROI	Freq.	SROI	EROI	Freq.
12	12	**21**	14	14	**33**	32	32	**17**	35	35	**11**	14	14	**23**
39	39	**19**	18	18	**24**	1	1	**16**	14	14	**10**	12	12	**16**
14	14	**15**	19	19	**12**	28	28	**14**	31	31	**9**	44	44	**16**
18	18	**14**	21	22	**11**	50	50	**12**	18	18	**8**	15	15	**14**
28	28	**11**	21	21	**11**	15	15	**12**	12	12	**6**	10	10	**13**
27	27	**10**	51	51	**11**	31	31	**12**	50	50	**6**	21	21	**11**
47	47	**10**	50	50	**10**	12	12	**11**	44	44	**6**	12	14	**9**
24	24	**9**	18	19	**9**	4	4	**9**	25	25	**6**	42	40	**8**
35	35	**9**	12	12	**8**	47	47	**9**	24	24	**5**	14	12	**8**
44	44	**9**	22	22	**8**	10	10	**8**	42	40	**5**	28	28	**8**

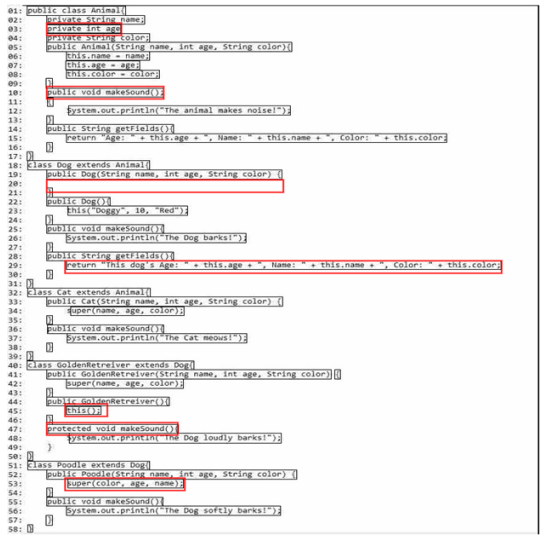

Fig. 5: Code Review Exercise Questions 2 (Q2) with TROI

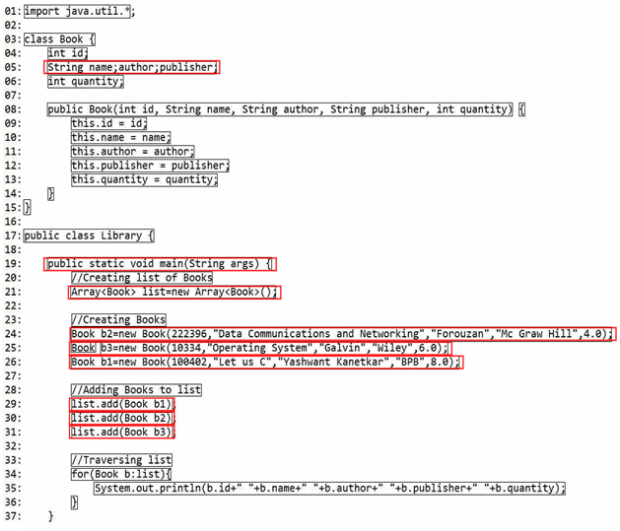

Fig. 6: Code Review Exercise Questions 3 (Q3) with TROI

TABLE V: Avg. backtracking transition frequency of subjects in groups in Q2

Regions	Group D	Group A
Method Declaration Regions	46.66	49.50
Class Declaration Regions	25.00	10.50
Return statement or Print Statement Regions	17.33	26.50
End Curly Brace Regions of Method or Class	34.33	21.00

are few and errors are seeded closer, Group A had fewer fixation transitions but successfully determined the answer. Group D has top transitions are **5-5, 2-2, 1-1, 7-7**, on the other hand Group A has **7-7, 8-7, 7-8, 9-10**. From Table III, it can be seen that 7 of the top 10 transition ROIs are TROIs in Group A, while in group D only 3. This implies how significantly Group A is fixated in the TROIs than Group D.

B. Code Review Exercise: Question 2 (Q2)

Fig. 5 illustrates Q2 along with the TROI in Q2. To enhance visualization and understanding, we'll categorize the regions into four color-coded categories: blue, green, magenta, and purple.

1. **Method Declaration**: Line # 10, 14, 19, 22, 25, 28, 33, 36, 41, 44, 47, 52, 55
2. **Class Declaration**: Line # 18, 32, 40, 51
3. **Return Statement or Print Statement**: Line # 12, 15, 26, 29, 37, 48, 56
4. **End Curly Brace of Method or Class**: Line # 9, 13, 16, 17, 21, 24, 27, 30, 31, 35, 38, 39, 43, 46, 49, 50, 54, 57, 58

From Table IV, the average backtracking transition frequency of Subjects in each group can be calculated for the four categorical regions as shown in Table V. From these metrics, we can speculate on subjects' cognitive processes in comprehending different aspects of the code. In the Q2 coding exercise, which contains more regions and methods, Group A showed more backtracking on the return or print statement (26.50) than Group D (17.33), suggesting an intention to understand the method body. In contrast, Group D backtracked more to the end of the method (34.33), indicating potential skimming with less understanding. Furthermore, Group A transitioned their fixation more (49.50) than Group D in method declaration regions (46.66) but less (10.50) than Group D (25.00) in class declaration regions. This suggests that high-

scoring subjects may quickly understand class declarations, focusing less on them and more on method declarations.

C. Code Review Exercise: Question 3 (Q3)

Q3 is a more challenging coding exercise than the previous two (Fig. 6), requiring a deeper understanding of Java for correct answers. Subjects from Group A tended to briefly skim the code from top to bottom during the initial scan, with less frequent fixations on specific regions. On average, Group D subjects navigated to comment regions (Line # 20, 23, 28, 33) less frequently than Group A, with Group D having 16 backtracking transitions compared to Group A's 22 (Table VI and VII). Group A fixated more on comment regions, potentially to understand the answer, while Group D showed less focus on comment lines, suggesting a lower level of comprehension than Group A.

TABLE VI: Top 10 sequence of backtracking transitions of the Group D (14% grade) subjects in Q3

Group D (20% grade)					
Subject 1			Subject 2		
SROI	EROI	Freq.	SROI	EROI	Freq.
23	23	**9**	18	18	**22**
18	18	**7**	25	25	**18**
4	4	**4**	20	20	**13**
1	1	**3**	24	24	**10**
22	22	**3**	24	25	**88**
20	20	**3**	22	20	**6**
6	6	**2**	25	22	**5**
15	10	**2**	17	17	**4**
8	8	**2**	17	18	**4**
22	20	**2**	18	17	**4**

TABLE VII: Top 10 sequence of backtracking transitions of the Group A (80% grade) subjects in Q3

Group A (80% grade)		
Subject 1		
SROI	EROI	Freq.
20	20	**13**
6	6	**12**
18	18	**10**
22	22	**8**
17	17	**7**
16	16	**6**
18	17	**5**
28	29	**5**
20	22	**4**
15	15	**4**

VII. Discussion

Through source code examination, this study gained insight into students' cognitive processes. The focus was on evaluating in the context of lines rather than granular metrics like words or tokens, providing a broader scope for analysis. Extracting eye metrics through hit-or-miss ROI border detection enables the identification of regions where students focus on comprehending the code, offering valuable knowledge for teaching students how to assess others' code and write easily understood source code. With respect to the research question and based on the results from the Java coding exercises, specific patterns in the eye-tracking data, combined with TROIs, indicate the success or struggles of students in code comprehension exercises.

1) In summary, the high-scoring subjects fixated more on the TROI regions, hence the subjects were able to comprehend the meaning of the region. On the other hand, lower-grade subjects tend to fixate less than Group A.
2) Instead of analyzing the full code's transition patterns, we focused on the top 10 transitions or backtracking to gain insights into the more fixated regions and their relevance to understanding the code. The analysis revealed that fixation patterns offer meaningful insights into the code comprehension of different subject groups. High-scoring groups frequently backtrack to the method definition and skim the code from top to bottom, while low-scoring groups exhibit an abrupt skimming pattern rather than analyzing the logical comprehension of the code.

VIII. Conclusion and Future Work

In our carefully designed programming exercise, we aimed to evaluate the NiCATS software's effectiveness for computer science teachers in assessing their students' comprehension. Using seeded error regions as TROIs, we analyzed fixation transitions to understand code comprehension patterns. We recognize varying student processing speeds do not necessarily indicate different levels of understanding, and therefore leveraged transition properties to aid effective data comparison. Identifying crucial code regions and analyzing individual perceptions provided valuable insights, with visual representations of eye metric data serving as a tool for investigating cognitive processes. Educators can benefit by defining and analyzing specific targeted regions, reducing unnecessary analysis overhead. However, fixation data alone may not offer a complete picture of code comprehension, necessitating consideration of factors like code complexity, prior knowledge, and cognitive load for a comprehensive understanding. In the future, we plan on enhancing the current system to automate eye metrics calculation (fixations, saccades) and ROI labeling processes for seamless integration into any platform. The current post-hoc process utilizes Amazon Textract for ROI labeling, in future we can consider Optical Character Recognition (OCR) for data collection. Future investigations could examine differences between syntax errors versus logic errors, providing further insights into cognitive processes in error correction.

References

[1] T. Tabassum, A. A. Allen, and P. De, "Non-intrusive identification of student attentiveness and finding their correlation with detectable facial emotions," in *Proceedings of the 2020 ACM Southeast Conference*, ser. ACM SE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 127–134. [Online]. Available: https://doi.org/10.1145/3374135.3385263

[2] J. Whitehill, Z. Serpell, Y.-C. Lin, A. Foster, and J. R. Movellan, "The faces of engagement: Automatic recognition of student engagementfrom

facial expressions," *IEEE Transactions on Affective Computing*, vol. 5, no. 1, pp. 86–98, 2014.

[3] Z. Sharafi, T. Shaffer, B. Sharif, and Y.-G. Guéhéneuc, "Eye-tracking metrics in software engineering," in *2015 Asia-Pacific Software Engineering Conference (APSEC)*, 2015, pp. 96–103.

[4] N. Veliyath, P. De, A. A. Allen, C. B. Hodges, and A. Mitra, "Modeling students' attention in the classroom using eyetrackers," in *Proceedings of the 2019 ACM Southeast Conference*, ser. ACM SE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2–9. [Online]. Available: https://doi.org/10.1145/3299815.3314424

[5] A. Sanders, B. Boswell, A. Allen, G. S. Walia, and M. S. Hossain, "Development and field-testing of a non-intrusive classroom attention tracking system (nicats) for tracking student attention in cs classrooms," in *2022 IEEE Frontiers in Education Conference (FIE)*, 2022, pp. 1–9.

[6] B. Boswell, A. Sanders, A. Allen, G. S. Walia, and M. S. Hossain, "Using ai-based nicats system to evaluate student comprehension in introductory computer programming courses," in *2022 IEEE Frontiers in Education Conference (FIE)*, 2022, pp. 1–9.

[7] H. Hijazi, J. Cruz, J. Castelhano, R. Couceiro, M. Castelo-Branco, P. de Carvalho, and H. Madeira, "ireview: an intelligent code review evaluation tool using biofeedback," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, pp. 476–485.

[8] P. Rodeghero and C. McMillan, "An empirical study on the patterns of eye movement during summarization tasks," in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2015, pp. 1–10.

[9] T. Fritz, A. Begel, S. C. Müller, S. Yigit-Elliott, and M. Züger, "Using psycho-physiological measures to assess task difficulty in software development," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 402–413. [Online]. Available: https://doi.org/10.1145/2568225.2568266

[10] D. Rosengrant, D. Hearrington, K. Alvarado, and D. Keeble, "Following student gaze patterns in physical science lectures," *AIP Conference Proceedings*, vol. 1413, no. 1, pp. 323–326, 02 2012. [Online]. Available: https://doi.org/10.1063/1.3680060

[11] Z. Zhu, S. Ober, and R. Jafari, "Modeling and detecting student attention and interest level using wearable computers," in *2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 2017, pp. 13–18.

[12] M. A. A. Dewan, M. Murshed, and F. Lin, "Engagement detection in online learning: a review," *Smart Learning Environments*, vol. 6, no. 1, p. 1, Jan 2019. [Online]. Available: https://doi.org/10.1186/s40561-018-0080-z

[13] M. S. Hossain, D. Pandya, A. Allen, and F. G. Hamza-Lup, "Learner attention quantification using eye tracking and eeg signals," in *Proceedings of the Future Technologies Conference (FTC) 2022, Volume 2*, K. Arai, Ed. Cham: Springer International Publishing, 2023, pp. 836–847.

[14] D. D. Salvucci and J. H. Goldberg, "Identifying fixations and saccades in eye-tracking protocols," in *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, ser. ETRA '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 71–78. [Online]. Available: https://doi.org/10.1145/355017.355028