Leveraging Targeted Regions of Interest by Analyzing Code Comprehension With AI-Enabled Eye-Tracking

Md Shakil Hossain

LEVERAGING TARGETED REGIONS OF INTEREST BY ANALYZING CODE COMPREHENSION WITH AI-ENABLED EYE-TRACKING

by

MD SHAKIL HOSSAIN

(Under the Direction of Andrew A. Allen)

ABSTRACT

Code comprehension studies techniques for extracting information that give insights on how code is understood. For educators teaching programming courses, this is an important but often difficult task, especially given the challenges of large class sizes, limited time, and grading resources. By analyzing where a student looks during a code comprehension task, instructors can gain insights into what information the student deems important and assess whether they are looking in the right areas of the code. The proportion of time spent viewing a part of the code is also a useful indicator of the student's decision-making process. The goal of this research is to analyze the differences in how students' eyes traverse across code during coding comprehension activities and to offer a systematic way for distinguishing students with a solid understanding of the exercise from those who require further assistance. The study uses coding exercises seeded with errors, measured fixation counts, and average fixation durations of the students' eyes within targeted regions of interest (TROI) using an AI-Enabled Eye-Tracking System (NiCATS). The results of the study showed that students' grades (as a proxy for understanding of the code's context and their decision-making skills) were positively correlated with a higher ratio of the number of fixations in the TROI.

INDEX WORDS: Code comprehension, Eye tracking, Cognition, Targeted regions of interest (TROI)

LEVERAGING TARGETED REGIONS OF INTEREST BY ANALYZING CODE COMPREHENSION WITH AI-ENABLED EYE-TRACKING

by

MD SHAKIL HOSSAIN

B.S., Bangladesh University of Engineering & Technology, 2011

A Thesis Submitted to the Graduate Faculty of Georgia Southern University

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

LEVERAGING TARGETED REGIONS OF INTEREST BY ANALYZING CODE COMPREHENSION WITH AI-ENABLED EYE-TRACKING

by

MD SHAKIL HOSSAIN

Major Professor:	Andrew A. Allen
Committee:	Lixin Li
	Ryan Florin

Electronic Version Approved:
May 2023

DEDICATION

I dedicate this thesis to my family, friends, my beautiful wife, and my adorable daughter, who kept me motivated and provided constant inspiration throughout my research.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Allen for advising me throughout the entire research. Dr. Allen's invaluable guidance and support were instrumental in helping me successfully complete this project. I would also like to extend my heartfelt thanks to Dr. Li and Dr. Florin for being a part of my thesis committee. Their insightful comments and feedback greatly improved the quality of my work.

I am forever grateful to Dr. Walia for providing me with the opportunity to come to GSU and introducing me to the incredible NiCATS team. His mentorship and encouragement have been a source of inspiration for me. Finally, I would like to thank Andrew and Brad for their previous work, which laid the foundation for my research. I cannot thank them enough for their contributions, which were instrumental in the success of my project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the realm of Computer Science education, an ongoing and complex research issue involves gauging student comprehension in real-time during classroom instruction. While this dilemma applies to all types of classes, it is especially problematic in Computer Science classrooms, where students often spend a significant amount of time working behind computer monitors. This setup limits visual interaction between students and instructors, making it more challenging for instructors to get a real-time sense of their students' understanding of the material. As a result, traditional assessments of student comprehension are often conducted much later, relying on methods such as reviewing test and quiz results, collecting feedback about specific topics, or other activities. While these evaluations can provide some insight into whether a student has grasped the material, they frequently fail to reveal the cognitive process that led to the student's solutions. This makes it difficult for educators to identify the root causes of any misunderstandings or difficulties that students may be experiencing. Without the ability to identify these issues early on, instructors are often unable to adjust the pace of their lectures or provide tailored assistance for struggling students unless those students seek out the necessary information on their own. This lack of timely intervention can lead to students falling behind, becoming disengaged from the material, or feeling discouraged about their ability to succeed in the course. Therefore, developing effective methods for real-time assessment of student comprehension is an ongoing challenge in the field of Computer Science education.

Previous research has explored the use of machine learning and computer vision techniques to automatically determine a student's level of attentiveness, albeit with limited success (Tabassum et al. 2020, 127–134; Whitehill et al. 2014, 86–98). Average fixation duration is one eye tracking metric that has been identified as offering insights into the cognitive processes involved in perception and processing of presented information (Sharafi et al. 2015, 96–103; Veliyath et al. 2019, 2–9). Facial expressions can give some idea of how attentive a student seems to be, but eye tracking metrics can give more information about the student's visual focus and how well they are learning the material. We think that instructors would benefit a lot from having access to a variety of tools that help them find eye metrics data that is highly related to how well students understand. The fundamental benefit of this method is that it provides accuracy that is comparable to that of humans, who are domain experts, while eliminating the observational impact that can result from human observation.

This thesis assesses the various ways that instructors might use the information obtained from the Non-Intrusive Classroom Attention Tracking System (NiCATS) to analyze student comprehension (Sanders et al., 2021). An AI-enabled data collection tool called NiCATS was developed by Georgia Southern University students. When the program was first developed, its main objective was to use webcam images and gaze data collected from eye trackers to predict student attentiveness. Nonetheless, the main goal of this thesis is to use NiCATS to analyze student comprehension and expand the system's functionality to help instructors enhance the learning environment for their students. Boswell et al. conducted a study in which they used NiCATS to track students' eye movements and analyze their cognitive processes in an introductory computer programming course (Boswell et al., 2022). The researchers utilized a multilayered approach for calculating gaze metrics against automatically defined regions of interest (ROIs), providing instructors with valuable insights into students' comprehension patterns and identifying hard-to-comprehend content topics. However, this work was focused on assessing a single student. A part of this work extends that to multiple students at multiple levels of successful code comprehension.

The regions of interest, from the students' perspective, include the areas where they should fixate the most in the presented region. The fixation patterns indicate what areas the student considered important. It is important to distinguish between regions where students fixated frequently versus regions where they did not fixate much, even if the investigator wanted them to focus on those areas. In this thesis, we have introduced another term, “Targeted Region of Interest”(TROIs). Targeted regions of interest (TROIs) can enhance the accessibility of eye tracking research for various types of instructors. Instead of considering lines of code, TROIs can be defined by (x,y) coordinates, allowing investigators to analyze specific regions and gain insights into what happens in those areas. The targeted regions of interest allow investigators to observe the student's behavior and analyze their engagement with those areas. By analyzing the fixation patterns in the targeted regions of interest, investigators can gain insight into how the students interacted with those areas. Also, using fixation ratios instead of time intervals makes it possible to get a more accurate picture of how much a student is paying attention to a certain area. Researchers can measure a student's relative attention and focus on a given location by comparing the ratio of fixations on that area to the total number of fixations. This method provides for a more sophisticated study and can be used to compare the performance of students or groups with varying reading rates or approaches to the subject matter. This method gives a more complete picture of a student's performance.

We understand that students process information at different speeds. However, a slower processing speed does not necessarily indicate a lesser understanding of the material. For example, if one student takes five minutes to understand a concept and another takes 15 minutes, but they both arrive at the same understanding and achieve the same grade, then we can compare their performance without explicitly measuring their processing speed. An approach to provide a reasoned comparison of time spent on specific parts of the code is needed to provide a scalable analysis for all students in the class, categorize and derive insights.

We carried out and reported the results of three distinct experiments to assess the possible applicability of NiCATS. To gain insight into the connections between eye measurements and the students' comprehension processes, the experiment data was assessed in three different contexts (a PowerPoint presentation regarding human error, source code review, and a timed testing environment). This thesis will only cover the source code review experiment, and via the analysis of that experiment, we will show how instructors can use NiCATS to learn more about students' comprehension patterns than they would be able to in a conventional classroom. The results of these source code trials showed that NiCATS's eye metrics and screenshot data could give instructors useful information about how well their students understand what they are reading. One observable pattern was that struggling students tended to repeatedly retrace their steps to and from areas that held data relevant to the experiment's objective (e.g., seeded errors in source code, the correct answer on an exam).

This thesis' background section details eye tracking research on student understanding and attentiveness as well as how these eye measures might be utilized to forecast students' attention spans and other cognitive processes. The suggested method discusses the high-level architecture of NiCATS and how the information gathered by the system might be used for comprehension-related evaluations. The source code review experiments’ designs are covered in the Experiment Design. The analysis and results are then given separately for each of the seeded questions in the experiment analysis section.

CHAPTER 2

BACKGROUND

The background discusses relevant research on eye tracking studies relating to knowledge comprehension as well as the shortcomings of previous investigations.

Eye trackers are commonly used by researchers to examine the cognitive processes associated with comprehension from the subject's perspective. Eye tracking is a research field that is growing quickly, and it can be used to measure things like attention, emotion, and cognitive ability.
In this thesis, 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 (Sharafi et al. 2015, 96–103).
- Saccades: The brief (usually 50 ms) and continuous eye movements between fixations (Sharafi et al. 2015, 96–103).
- 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.

A general ROI covers a larger area of the scene and is used to study broad patterns of eye movement or attentional engagement. These can include the entire visual scene or specific regions of interest that are not narrowly defined. General ROIs are useful for exploring initial patterns of eye movement and attention, but they may not provide as much detailed information about specific aspects of visual perception. On the other hand, a TROI is a specific area or areas of the visual scene that the researcher is interested in studying in more detail. These areas are typically selected based on a specific research question or hypothesis, and they may be smaller and more narrowly defined than a general ROI. Targeted ROIs are useful when the researcher has a specific hypothesis or question about a particular aspect of visual perception that they want to investigate. By selecting a specific ROI, the researcher can control for extraneous variables and focus on the specific stimuli or features that are of interest. Typically, an eye tracker is used for each of them. Eye-attached (like a contact lens), optical (reflected infrared), or electric potential are the three different types of eye trackers (electrodes placed around the eyes). The majority of consumer eye trackers use optical tracking and can be worn on the head or linked to a computer monitor.

Hijazi et al. used an eye tracker on a desktop and a non-intrusive Heart Rate Variability (HRV) monitor to predict which code reviews would be good and which would be bad (Hijazi et al., 2021). They collected biometric data while the subjects looked over code samples and measured the quality of their review based on how many bugs they missed in the code they were given. First results showed that their tool could predict bad reviews of medium and complex programs with 75%–87% accuracy.

Rodeghero et al. did an experiment to find out how people's eyes moved when they were summarizing source code (Rodeghero et al. 2015, 1–10). The study analyzed the eye movements of experienced programmers evaluating source code and discovered that all 10 participants had eye movements that were remarkably similar to those of readers of natural language. The study's qualitative findings revealed that programmers tended to skim source code rather than read it thoroughly and had a propensity to scan sections rather than read it from left to right or top to bottom (transitions between two fixation coordinates are usually within two lines of the previously fixated line).

Using data from an eye tracker, an electrodermal activity sensor, and an electroencephalography sensor, Fritz et al. conducted an experiment with 15 experienced programmers to determine whether they would find a work challenging or not (Fritz et al. 2014, 402-413). With 64.99% precision and 64.58%

recall, their classifier was able to estimate whether a task would be simple or complex for a novice developer.

Veliyath et al. utilized data acquired from self-reporting and a Tobii Eye Tracker 4c as a non-intrusive method for predicting student attention during the duration of a class by using self-reported data and a Tobii Eye Tracker 4c (Veliyath et al. 2019, 2–9). The researchers set eye trackers on the computer monitors in a computer lab and had students occasionally rate the lecturer's engagement on a Likert Scale ranging from 1 (not engaging) to 10 (extremely engaging). This information, together with the gaze data gathered from the eye tracker, enabled the researchers to develop a machine learning model that could predict the level of student engagement throughout the presentation of the material.

Rosengrant et al. used Tobii Glasses to detect the eye movements of students throughout a presentation and figure in order to discover sources of inattention (Rosengrant et al. 2012, 323–326). During a presentation on physical science, they utilized eight participants and recorded where the students were gazing. Students rarely paid attention to the professor unless he or she was displaying emotion, drawing something, being humorous, or making comparisons that were not on the presentation slides, according to their findings. Fresh slides tend to keep or distract students' focus on the board. They preferred viewing illustrations or diagrams over reading text. According to the study, students who had written notes prior to class tended to pay less attention in class.

Zhu et al. employed sensors on wearable computers to capture both hand movements and heart activity in order to predict "interest level" and "judgment of difficulty" (Zhu et al. 2017, 13-18). During two lectures with fourteen topic sessions, 30 volunteers were asked to wear Moto 360 clocks. They collected motion data at 25 Hz and PPG(Photoplethysmography) data at 12.5 Hz, as well as survey data regarding students' level of interest and sense of difficulty in each lecture topic. The researchers discovered that using wrist-worn smartwatches for attention monitoring yields good accuracy, and that other physiological sensors might potentially be employed to further improve accuracy.

Dewan et al. address attention detection approaches and claim that they may be generally grouped into three groups — automatic, semi-automatic, and manual (depending on strategy and type of user engagement) — and each of the referenced works employs at least one of them to evaluate attentiveness (Dewan et al., 2019). See Figure 1 for a categorization of the three groups. Automatic can be separated into computer vision-based techniques (facial expressions, gestures, postures, and eye movements), sensor data analysis, and log file analysis. Engagement tracing, which uses the time and precision of student responses to measure attentiveness, is one of the semi-automatic methods. Observational checklists and self-reporting are manual attention detection techniques. Each of these strategies has its drawbacks. Automated approaches require additional development before they can be utilized in a typical educational setting. Semi-automatic techniques necessitate probabilistic reasoning, and there are few applications upon which to build. Due to the presence of the self-reporting or observer, manual procedures are more subject to bias and the Hawthorne effect and take much more time and effort from both observers and pupils.

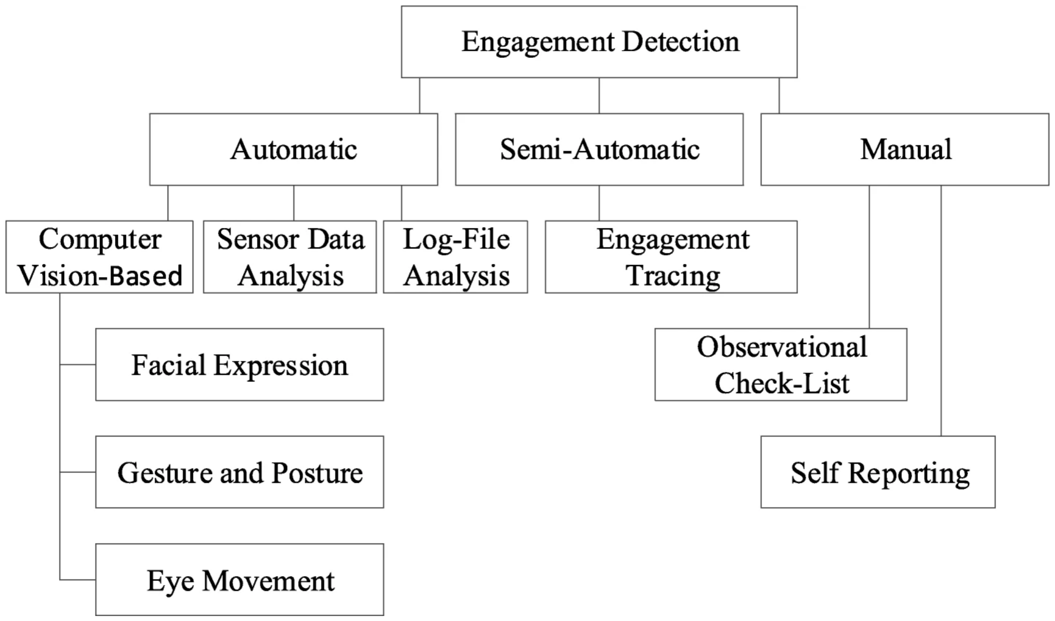


Fig. 1 Taxonomy of learners' engagement detection methods (Dewan et al., 2019)

Tabassum et al. employed a deep learning convolutional neural network and the outputs of a cloud-based emotion detection service (Amazon Rekognition) to find correlations between emotions and attentiveness (Tabassum et al. 2020, 127–134). They collected data by capturing webcam images of students during class lectures. Faces were extracted using a motion detection approach when video motion was detected. The BERI method was used to classify facial photographs as attentive or inattentive. Using these tagged facial images, a convolutional neural network was trained. Then, Amazon's Rekognition technology was utilized to extract the facial emotions from the photographs. The possible emotions were happiness, sadness, anger, confusion, disgust, fear, and surprise, and each was assigned a score between 0 and 100. Each emotion was statistically significant in a regression study between the emotion and attentiveness, demonstrating that facial emotions can be employed to improve the accuracy of attention detection algorithms. Researchers were able to identify inattentive kids with 93 percent accuracy using these methods.

CHAPTER 3

PROPOSED APPROACH: NiCATS

Sanders et al. used NiCATS for collecting the necessary data for the studies conducted in this thesis (Sanders et al., 2021). In this section, only the parts of NiCATS that relate to this thesis are described. The NiCATS system is used for collecting data and designing experiments. Our hypothesis is that the results of these analyses will lead us toward adding automatic comprehension analysis to the NiCATS application in the future.

The high-level architecture of the NiCATS application is shown in Figure 2, which provides a high-level overview of NiCATS in terms of data collection, preprocessing, and analysis. During student sessions, the NiCATS system is designed to collect three distinct types of data about the student from the student's machine. The first type of data is facial images, which are captured periodically by a webcam mounted on the student's computer monitor. These images are sent to the server at a 5-second interval and are used to determine the student's attentiveness.

The second type of data is screen captures, which are triggered whenever the student interacts with their machine via keyboard/mouse input or after a pre-defined interval of 15 seconds. These screen captures are used to determine the context of the student's behavior and help identify periods of distraction or disengagement.

The third type of data is continuous eye movement, which is captured throughout the lecture using an eye tracker. The gaze points of the student are continuously recorded and sent to the server for pre-processing and subsequent analysis.

In addition to the three core data metrics, NiCATS has the ability to integrate EEG (Electroencephalography) signals. The NeuroSky MindSet system is used to record the EEG signals, which measure attention, meditation, and eye blinks, and various types of EEG signals generated by the NeuroSky ThinkGear Protocols (Hossain et al., 2022). The NiCATS system played a crucial role in collecting data and designing experiments for this thesis, providing real-time feedback on a student's attentiveness. It is a powerful tool for institutions to monitor and improve student engagement during lectures.

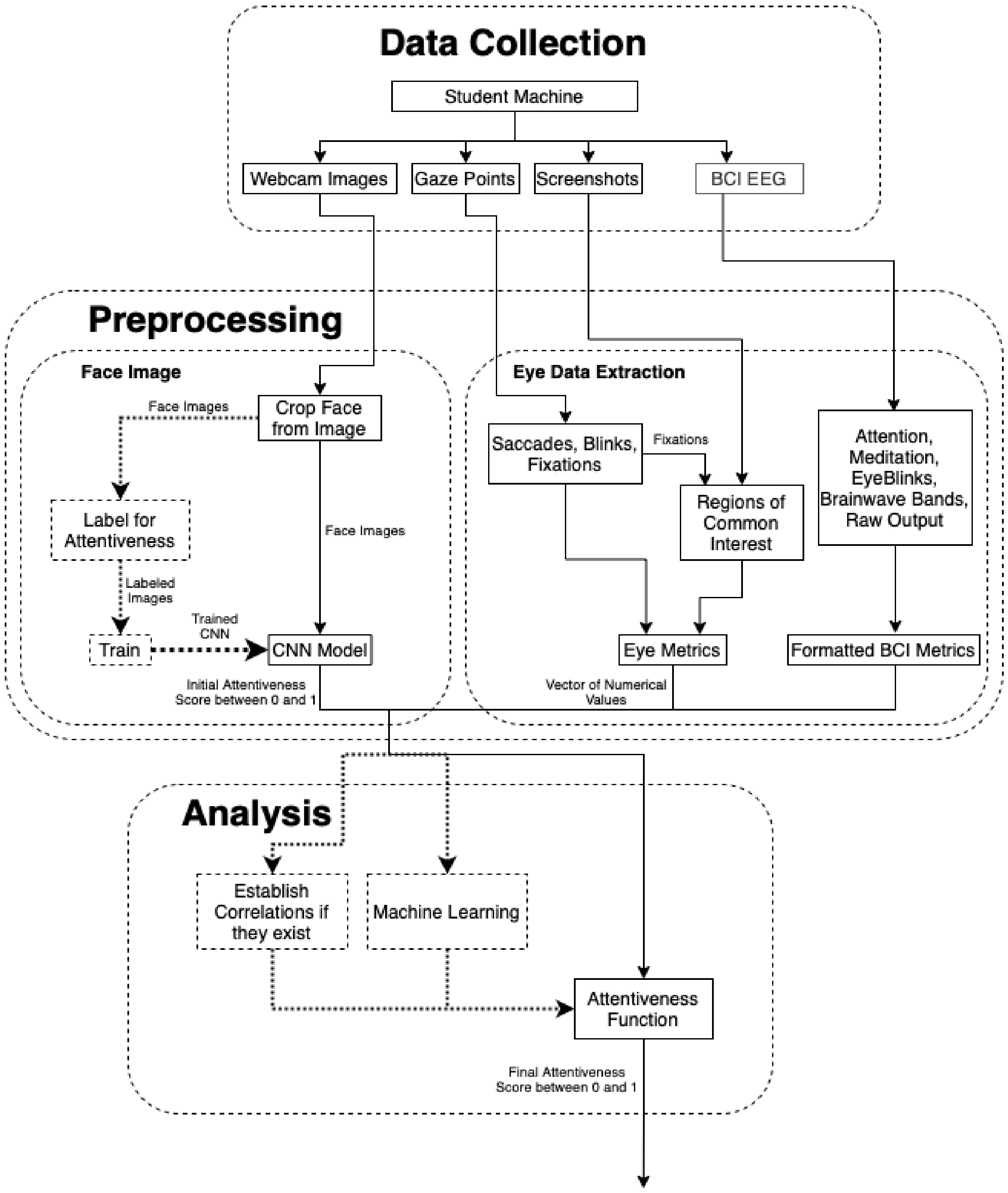


Fig. 2 High-Level Design of NiCATS

In this thesis, only the screenshots and gaze data gathered by NiCATS 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.

In order to a) extract eye metrics from the raw gaze data and b) produce useful visual aids of the post-processed data to be used for comprehension analysis, data-processing steps were carried out after each trial. These data-processing procedures varied based on the experiment's context, and they are each specifically covered in the sections on experiment design. A high-level overview of the data-processing workflow is shown in Figure 3.

From Figure 3, we can see that the machine we used to capture screenshots of the displayed code and raw gaze data, which were then stored on a server and associated with each student. This allowed us to identify specific screenshots that belonged to a particular student. Since the screenshots were of textual code, we used a system that did character recognition to identify blocks of text; in our study, we specifically used Amazon Textract.

Once we obtained the necessary information, we created ROI maps. This helped us identify areas on the screen where specific pieces of text resided. Based on these maps, we were able to tag specific regions as TROI areas, giving us a better understanding of where the students' focus was while they looked at the code. After we created the maps and identified the TROIs, we were able to do the necessary calculations for saccades and fixations. This helped us get a more detailed understanding of how the students were processing the code. By analyzing fixation metrics, we were able to determine how much time students spent looking at specific areas of the code and how frequently they moved their gaze across the screen. Overall, this data collection process provided us with valuable insights into the students' thought processes and helped us better understand how they approached code comprehension.

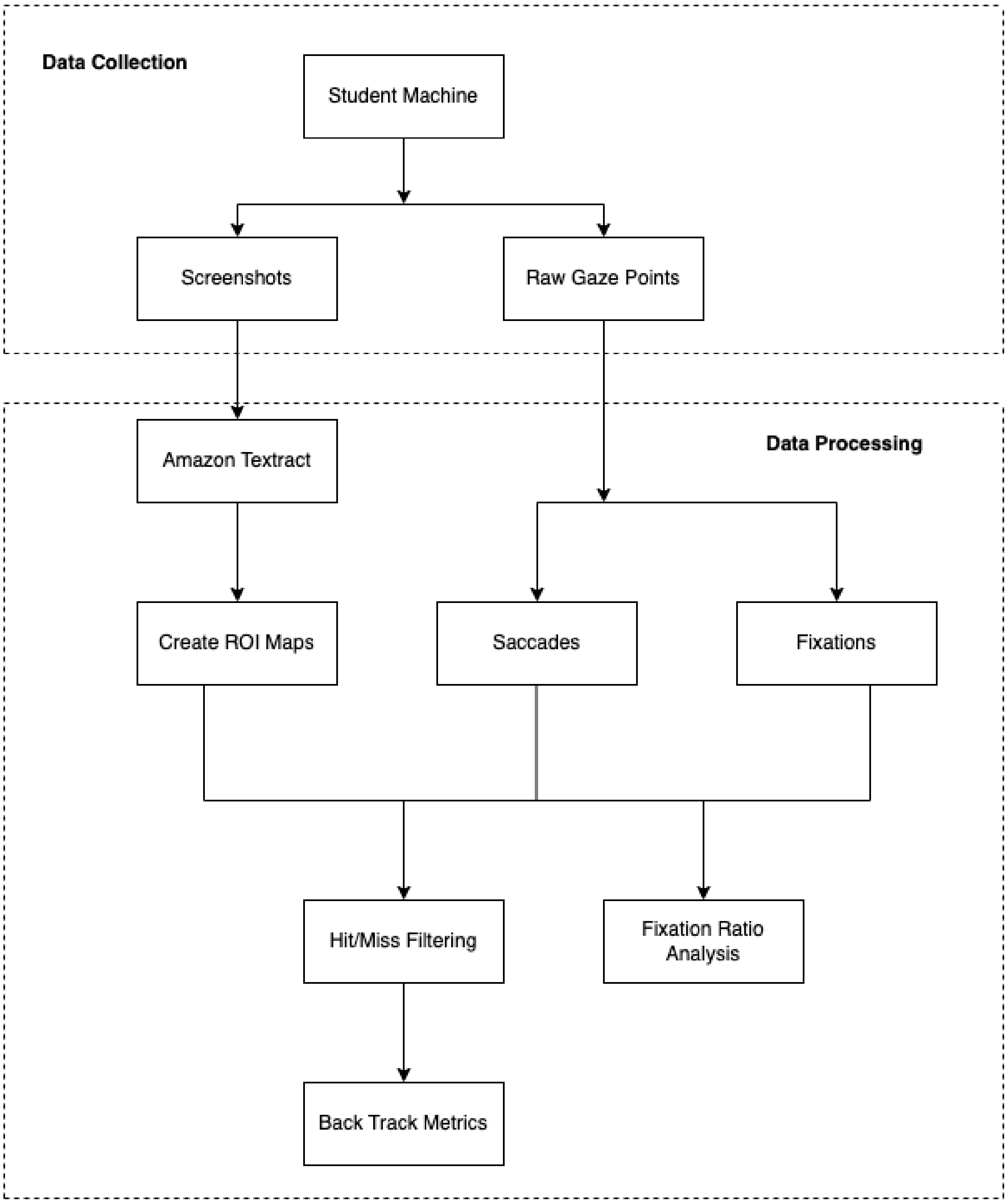


Fig. 3 High-level overview of the data processing workflow

To summarize the workflow, our experiment involves collecting screenshots and raw gaze points. After the data is gathered, the screenshots are processed in post-hoc operations using Amazon Textract to define the boundaries of ROIs. Meanwhile, the raw gaze points are converted into meaningful eye metrics such as fixations and saccades. Once we have the eye metrics, we use them to generate transitions data between ROIs while filtering out non-ROI regions. Additionally, we calculate the fixation ratio and other relevant parameters to analyze participant behavior from a group perspective.

In this study, we will use the Fixation ratio, specifically the Average Fixation Ratio within TROI by Group. This metric is calculated by taking the average within the TROI of the fixation ratio (fixation count in a certain region/total fixation count) of a group.

The purpose of using a ratio instead of specifying time intervals in seconds or milliseconds is to gain insights into how well a student is focusing on a particular area, rather than just measuring the absolute time spent on that area. By calculating the ratio of fixations in a particular ROI to the total number of fixations across all ROIs, we can determine the relative attention and focus of the student on that area compared to other areas. This allows for a more nuanced analysis of the student's performance and understanding of the code.

Additionally, using ratios enables us to compare the performance of different students or groups, even if they have different reading speeds or approach the code in different ways. For example, a novice student may spend more time on a particular area of the code than an experienced student, but by calculating the fixation ratio, we can compare their relative focus and attention on that area. Overall, using ratios gives a fuller picture of a student's performance and makes it easier to draw conclusions that are useful for research.

CHAPTER 4

EXPERIMENT DESIGN

4.1 FRAMEWORK OF RESEARCH STUDIES

The aim of this research is to assess the effectiveness of the AI-enabled NiCATS software in aiding instructors to analyze students' knowledge comprehension patterns in a variety of common classroom and laboratory settings in the field of Computer Science education. To achieve this goal, three experiments were meticulously designed and executed in different scenarios, including students watching pre-recorded videos, students undertaking a hands-on programming exercise, and students taking a timed exam. Each experiment had the same overarching goal of evaluating the potential of the NiCATS software to support instructors in gauging students' understanding of complex concepts and identifying common areas of difficulty.

Of the three experiments, the hands-on programming exercise produced the most comprehensive and statistically useful data set for analyzing students' comprehension errors in code. To facilitate this analysis, we identified seeded error regions as TROIs (Target Regions of Interest) and analyzed how fixation statistics such as count, average, ratio, and others provided meaningful insights for answering our research questions. We utilized the NiCATS software to collect students' gaze and screenshot data through eye tracking.

This section of the thesis offers an overview of the experiment, including the experimental design and data collection methods used during the code comprehension exercise. The data collected through this experiment serves as a basis for evaluating the efficacy of the NiCATS software in facilitating instructors' ability to assess student comprehension and identifying areas where students may be struggling

4.2 EXPERIMENT

This section provides an overview of the experiment's design, including the experiment's purpose, setting, artifacts, setup, data collection, and data analysis for the hands-on programming exercise experiment.

4.2.1 RESEARCH GOAL

The goal of this research was to explore how instructors can utilize NiCATS data (such as gaze metrics and screenshots) to gain insights into the thought 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. The research question investigated in this study was:

RQ1: How can the use of targeted regions of interest (TROI) aid in the analysis of code comprehension by students?

RQ2: Are there any patterns in eye tracking data that can be used to identify successful or unsuccessful students in understanding code?

To address RQ1, we will examine the fixation scan path towards the TROI, as well as the fixation ratio metrics, as a means of analyzing code comprehension. Conversely, RQ2 will involve an analysis of participants' derived eye-tracking metrics, including fixations.

4.2.2 EXPERIMENT SETTING

In our study, we conducted an experiment with volunteer undergraduate Computer Science students who were enrolled in the second course of an introductory programming course. To set up the experiment, we installed the necessary hardware and the NiCATS software on each participant's computer. We equipped the students' computers with a Tobii Eye Tracker 4c, which was mounted to the bottom-center region of the computer monitor. We conducted experiments on different lab computers and even with some basic configuration of i7-4790 CPU and 4 GB of RAM, we found that it was sufficient for running the NiCATS client program. Figure 4 illustrates the experimental setup used by the researchers.

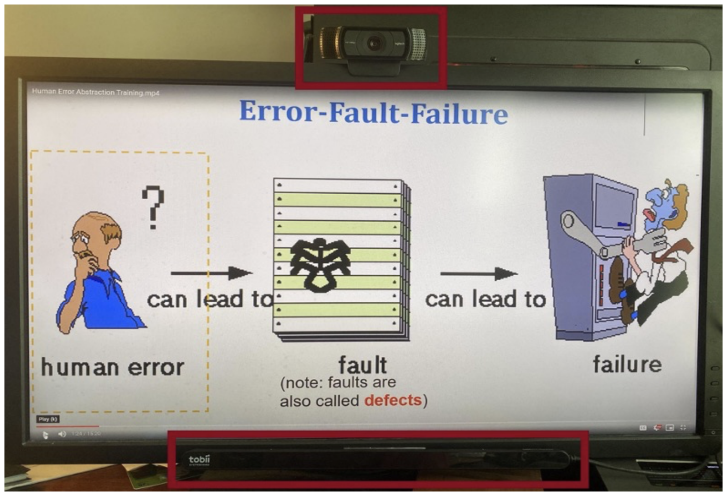
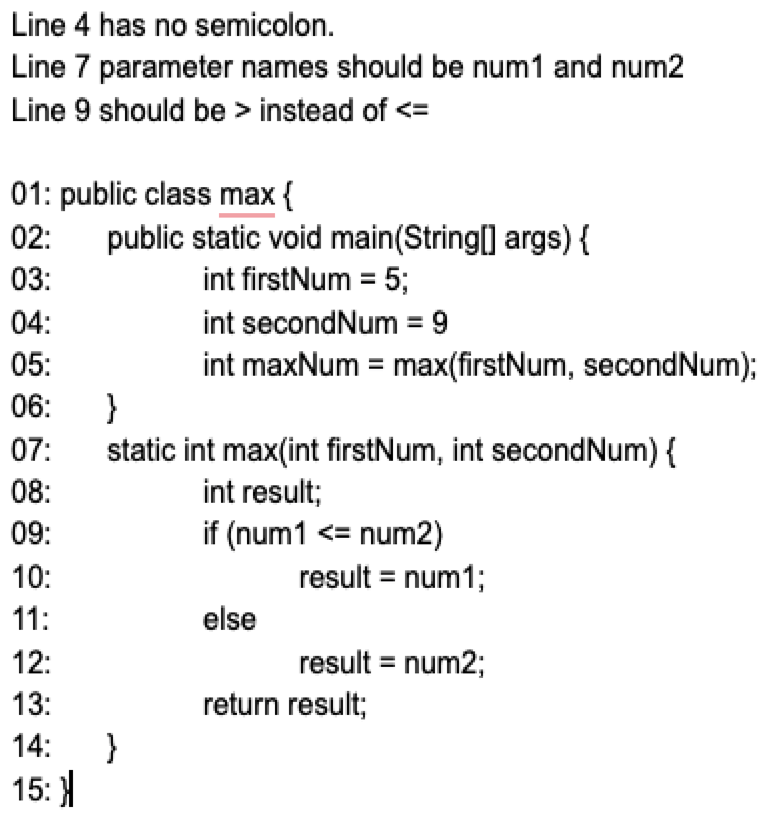

Human Error Abstraction Training.mp4
Error-Fault-Failure
can lead to
can lead to
human error
fault
(note: faults are also called defects)
failure
tobii

Fig. 4 Webcam and Eye Tracker mounted.

During the experiment, the participants were presented with four separate Java programs and asked to identify the line numbers that contained errors, as well as provide explanations for why those lines had errors. This allowed us to collect responses and associate the lines that the participants perceived as containing errors with their calculated eye metrics for those lines. Unfortunately, due to technical issues, we were only able to analyze the data from the first three code review tasks. By analyzing this data, we aimed to gain insights into how the use of eye tracking and the NiCATS software can help in understanding code comprehension patterns among students.

4.2.3 ARTIFACTS

In our study, we recruited participants with varied degrees of programming experience, ranging from six months to one year of Java programming experience. Each participant was instructed to evaluate four distinct Java programs using various complexity metrics, the number of lines, and the number of seeded flaws. We chose Java as the experiment's programming language because it was the most familiar to the participants. The code examples contained errors of varying degrees of difficulty and relevance to previously taught course material. Simple syntax problems, for instance, could easily be detected by reviewing a single line of code and were extremely simple for even a beginner programmer to identify. In contrast, identifying errors linked to object inheritance and data structures usually required novice programmers to examine numerous lines of code and proved more challenging. By picking programs with diverse levels of complexity and seeded errors, we wanted to examine NiCATS's capacity to assist instructors in measuring students' comprehension patterns across a variety of error types and code structures. Figure 5 represents the first code review task with the solution.



Line 4 has no semicolon.
Line 7 parameter names should be num1 and num2
Line 9 should be > instead of <=

```
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. 5 Java code sample with errors

4.2.4 EXPERIMENT STEPS

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

Step 1 - *Calibration*: In order to collect accurate and precise eye-tracking data during the experiment, all participants were instructed to calibrate the mounted eye tracker using the Tobii Eye Tracking Core Software. This software allowed each participant to save a calibration profile that was specific to their own eyes, ensuring that the collected gaze points were as accurate as possible. This calibration step was crucial for calculating eye metrics against regions of interest with tighter boundaries, as it improved the precision of the collected data. After logging in to their machine, each participant was prompted to begin the calibration process, which involved following a series of on-screen instructions while the software tracked their eye movements. Once the calibration was complete, the software saved the participant's calibration profile, which could be used throughout the experiment to collect accurate and precise eye-tracking data.

Step 2 - *NiCATS Client*: The participants in the experiment were given clear instructions to download and install the NiCATS client software on their machines. Once installed, they were asked to launch the software, which opened a pop-up window providing a description of the personal data that will be collected during the experiment. The participants were given the option to "opt-in" or "opt-out" of the experiment, with full transparency about the data that would be collected and how it would be used. This step was essential to ensure that the participants were fully aware of the nature of the study and could make an informed decision about their participation. By providing clear and transparent information about the data collection process, the researchers ensured that the participants could trust the experiment and were willing to participate.

Step 3 - *Initialize Data Collection*: In order to collect data for the experiment, the researchers initiated a recording session through the NiCATS web client. This action triggered all the machines in the room to start collecting data, provided that the participants had opted-in for the study. During the recording session, various raw data points such as gaze-point coordinates and screenshots were continuously collected and stored on the server for further analysis and post-processing. These data points are critical for analyzing the participants' eye movements and the areas of the screen that they focused on during the experiment. The post-processing of this data allowed the researchers to calculate various eye metrics, such as fixation counts and durations, that were then used to draw insights about the participants' code comprehension abilities. By collecting and analyzing this data, the researchers were able to gain a deeper understanding of the participants' behavior and thought processes while reviewing Java code.

Step 4 - *Program Presentation*: The participants were given five minutes to review each Java program. During the five-minute review period for each Java program, participants were instructed to identify the line number(s) they believed contained errors and provide a brief explanation for their choice. Participants were also allowed to navigate through the code using the scrollbar or by using the arrow keys. The fullscreen mode was enforced to avoid any other visual distractions that may impact the participants' gaze patterns. This also ensured that the region of interest (ROI) layouts defined by the researchers were consistent across all participants. This consistency is essential for accurate analysis of the eye-tracking data and for comparing the gaze patterns of different participants. By using the same ROI layouts, the researchers were able to determine which lines of code were fixated on the most, and whether there were any common patterns in the participants' gaze behavior when reviewing the code.

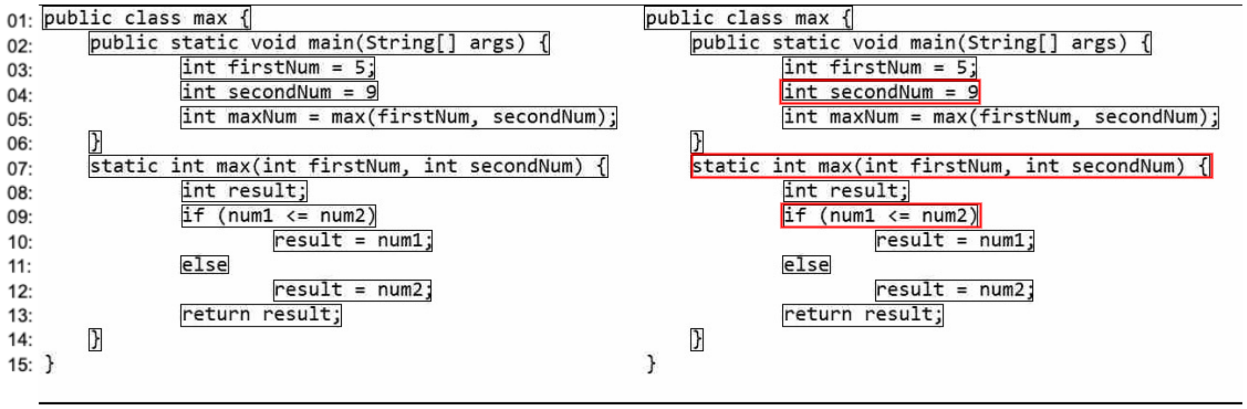
Step 5 - *Gathering Responses*: After the participants completed reviewing each of the four Java programs, the data was processed using the NiCATS software. The software extracted the fixation count, duration, and average fixation duration for each ROI defined by the researchers. The ROI for each line of code was defined as a rectangular area surrounding the line number and text of the line. The software was also able to identify the location of the participants’ gaze points within each ROI and calculate the ratio of time spent fixating within each ROI compared to the total time spent viewing the code. This data was then analyzed to identify patterns in the participants’ gaze and fixation behavior that could be associated with code comprehension and error detection.

Step 6 - *Concluding Data Collection*: The researchers stopped the recording session using the NiCATS web client after instructing the students to close the NiCATS software after reviewing each of the four code samples.

4.2.5 DATA PROCESSING

Simply collecting raw gaze points and screenshots during an experiment is not enough, as they hold little meaning on their own. In order 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.

1. *Defining ROI and TROI Boundaries*: Due to the uniformity of the experimental setting, where each participant reviewed the programs in fullscreen mode with the same resolution, a singular screenshot could be selected for each question and assigned region of interest boundaries. These region of interest maps were manually defined for each question with three different granularities, namely complexity-based, line-based, and lexer-based. The seeded error lines were labeled as TROIs during post-hoc analysis to analyze fixation ratio and backtrack patterns. An example of a line-based ROI map with TROI map can be found in Figure 6, where the specified TROIs are indicated with red markers on the right part of the image. Figure 6 provides a comparison representation of the visualizations used.



```
public class max {
    public static void main(String[] args) {
        int firstNum = 5;
        int secondNum = 9
        int maxNum = max(firstNum, secondNum);
    }
    static int max(int firstNum, int secondNum) {
        int result;
        if (num1 <= num2)
            result = num1;
        else
            result = num2;
        return result;
    }
}
```

```
public class max {
    public static void main(String[] args) {
        int firstNum = 5;
        int secondNum = 9
        int maxNum = max(firstNum, secondNum);
    }
    static int max(int firstNum, int secondNum) {
        int result;
        if (num1 <= num2)
            result = num1;
        else
            result = num2;
        return result;
    }
}
```

Fig. 6 A line based ROI and TROI map for analysis

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/Saccades*: In the context of ROI (Region of Interest) 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*: The calculation of fixations from raw gaze data was performed utilizing the I-DT (Identification by Dispersion-Threshold) algorithm (Salvucci et al. 2000, 71-78), with a maximum dispersion of 1 degree and a duration threshold of 300 milliseconds or more. The fixation coordinates obtained were subjected to boundary checks against the pre-established ROI maps to determine whether they were hits or misses. In this context, a hit corresponds to a fixation that occurred within the boundaries of any ROI in the predefined map, while a miss indicates a fixation outside the boundaries of all regions in the ROI map. The processed fixations were thereby determined, and include information such as the (x, y) coordinates, start and end times, the duration of the fixation, and the specific region in which the fixation occurred.
   b. *Saccades*: In order to define a saccade, we used a criterion based on two consecutive raw gaze points, satisfying the following conditions: the Euclidean distance between the two points is greater than 0.2 degrees, and the velocity exceeds 25 degrees per second. Each saccade was then characterized by its start and end times, duration, distance, velocity, (x, y) start coordinate, and (x, y) end coordinate. To ensure accuracy, the boundary checking procedure used for fixations was also applied to the start and end coordinates of each saccade. Specifically, a hit was defined as a saccade with both start and end coordinates lying within the boundaries of an ROI in the predefined map, whereas a miss corresponded to a saccade with at least one coordinate lying outside the

ROI boundaries. Saccades do not significantly contribute to this thesis since we have concentrated on the TROIs that we had previously described in the post hoc study and are interested in the effects that fixation parameters are having there.

4. *Additional Metrics*: Based on the results obtained from the hit/miss boundary detection, several essential metrics were calculated for use in our analysis. Specifically, at the participant-to-question level, we calculated the following metrics with respect to each ROI:
   a. *Average Fixation Duration*: This metric was determined by dividing the total fixation time for an ROI by the number of fixations in that ROI.
   b. *Number of Fixations*: The total number of times a participant fixated on an ROI was calculated.
   c. *Number of Backtracks*: The total number of times a participant revisited the ROI was determined.
   d. *Average Fixation Count ratio in TROI:* The Avg. Fixation Count Ratio (%) in the TROI exhibits the average of fixations percentage in that TROI of all the subjects. This metric was calculated as the average percentage of the fixation count in the specific ROI, providing an indication of how much focus the participant had placed on the TROI.
   e. *Average Fixation Ratio within TROI by Group*: Here, the term "group" refers to participants who achieved the same grade or result. The Average Fixation Ratio within TROI by Group indicates the percentage of focus in the TROI based on the result group. To calculate this metric for a group, the fixation ratio (i.e., the count of fixations within a certain region divided by the total fixation count) is averaged across the TROI. This metric has particular significance in distinguishing between good and novice participants.

CHAPTER 5

RESULTS AND ANALYSIS

We performed qualitative analyses of the results obtained in the context of individual students and across the entire population. Notably, this analysis excluded the raw gaze points and only explored the findings related to the fixation and saccade metrics within the context of multiple ROI maps. To ensure the accuracy and reliability of the results, all data points identified as a miss were removed from the analysis. This was based on the reasoning that such data points did not provide any significant benefit to the research goal, as they were not related to the reading and comprehension of source code contained in the regions defined by the ROI map. Examples of such data points include fixations detected as misses that occur when a student is looking off-screen or, in some instances, when the clock in the bottom right of the computer monitor is being viewed. Consequently, the exclusion of such data points from the analysis was deemed appropriate and necessary.

To answer the research questions for each code review exercise, we will first define the TROIs for each question and the category or group of participants based on grade. As mentioned before, in the code review exercises, the seeded error regions will be the TROIs in this study. In this study, each exercise will be evaluated based on the fixation parameters in the TROI and the top fixated regions of the subjects.

We focused our analysis on comparing the performance of two groups: the Participant Group with the Highest Grade and the Participant Group with the Lowest Grade. To aid in our analysis, we generated the fixation-based scan paths for each group, respectively. The TROIs are marked with red markers. We analyzed the scan paths of both groups to determine how they traversed through the code sample and to gauge their level of code comprehension. For each code review exercise, we compared one representative sample(single student) from top and bottom scoring groups in visualization perspective, the analysis includes the metrics from whole group. Fixation scan path visualization as group representation are provided in the Appendix section for reference.

The Average Fixation Count Ratio (%) in the TROI indicates the average of fixations of all the subjects within that TROI. In this research, we will be analyzing the subject group, not the individual. This metric helps the study aggregate the assumptions and speculation for the subject group's perspective towards the TROIs. We have also included another metric, the Avg. Fixation Ratio within TROI By Group, to indicate any hindsight in fixation metrics as a whole in all TROIs for all participants in the subject group. This metric is calculated by taking the average within the TROI of the fixation ratio (fixation count in a certain region/total fixation count) of a subject group.

Both fixation scan paths and ratio metrics will contribute towards answering RQ1. For RQ2, in the following section, this study will consider the top 10 backtracking transitions between two ROIs of the subjects from Group A (highest grade) and Group D(lowest grade). This behavior of backtracking transitions will be explained using three parameters: Start ROI, End ROI, and Frequency. The first two, Start and End ROI, lead to two possible scenarios:

a. *Start and End ROI are the same:* If both ROI are same (i.e., 7—>7 in Q1) in any backtracking transition, it could indicate the following points:
    - The subject fixated multiple times consecutively in that region, for example, two fixations happened in the same region consecutively.
    - The subject fixated in the region, left the region and fixated in whitespace/ceiling and immediately fixated back to that region.
b. *Start and End ROIs are different:* On the other hand, if both ROI are different (i.e 7—>6 in Q1), this could mean the following:
    - 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 indicates how many times the subject fixated on that region two times in a row. If frequency is n, then this also means that the subject focused on that region for the duration of two fixations for n times.

The reasoning behind choosing the top 10 backtracking transitions is that the top 10 represent the most frequent transitions between the ROI pairs and thus help to analyze patterns in comprehending the coding exercise. Table 1 below represents the significance of the top 10 transitions. In Q1 and Q3, the top 10 fell within the range of 33-60% of total frequency, while Q2 ranges from 24-30%. In this research, we aimed to identify significant fixation points from a line or region perspective. We selected the top 10 backtracking transitions, which proved to be a good representative sample. In question one, which consisted of 15 lines, the top 10 transitions covered 52% of the regions, representing 8 regions. Similarly, in question two, which had 57 regions, the top 10 transitions covered over 27%, indicating the discovery of 16 regions. Finally, in question three, which had 37 lines, the top 10 transitions covered 42%, representing almost 16 regions.

Questions	Participant Group	Subjects	Total frequency of Backtracking Transition Pairs	Total frequency of Top 10 Backtracking Transition 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

Table 1. Significance of Top 10 Backtracking Transition

This study will omit the fourth coding exercise from analysis due to errors in the eye tracker's data collection and low participation rates. The majority of participants did not focus on the monitor long enough, which left insufficient information for analysis.

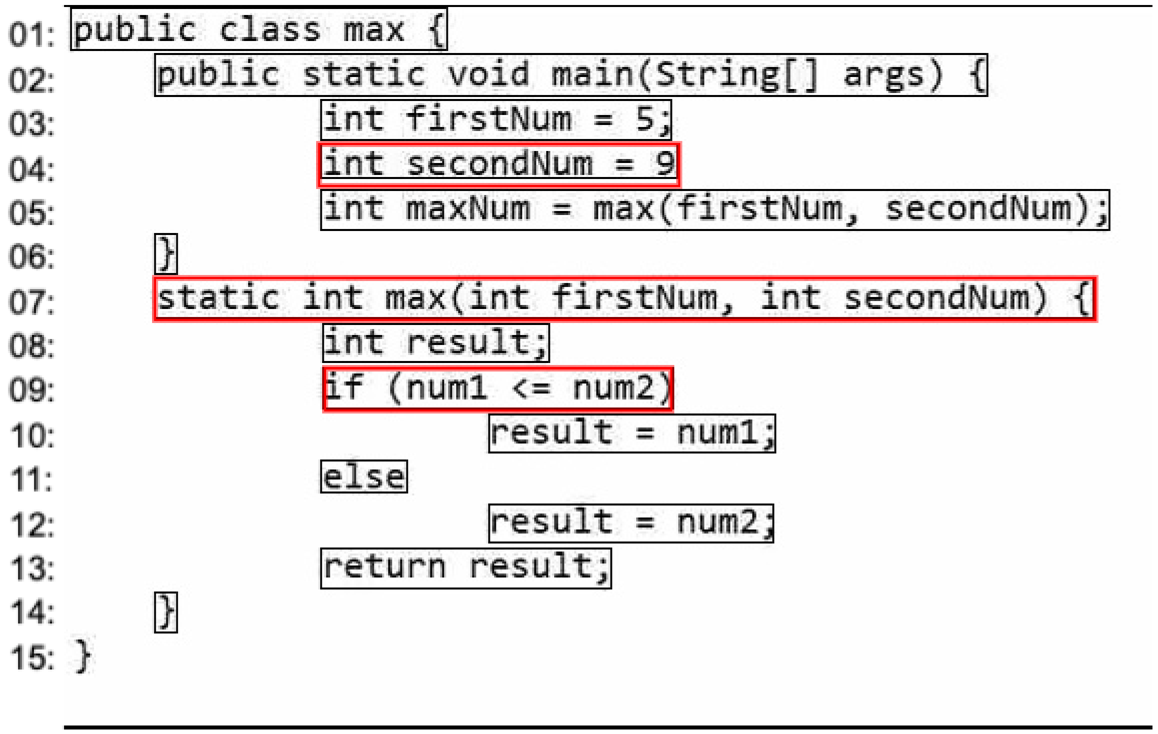
5.1 EXPERIMENT: CODE REVIEW EXERCISE

In the sections that follow, we will look at the data from each of the code reviews from the experiment.

5.1.1 CODE REVIEW EXERCISE: QUESTION1(Q1)

RQ1: How can the use of targeted regions of interest (TROIs) aid in the analysis of code comprehension by students?

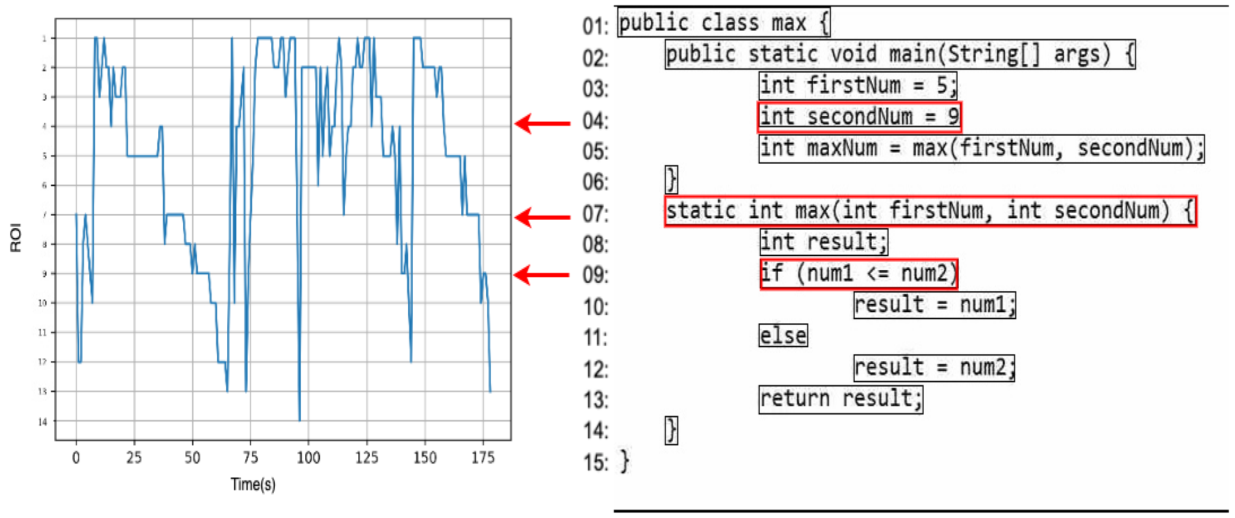
In this example, the red rectangle marked in Figure 7 indicates the TROIs in this Q1.



```
public class max {
    public static void main(String[] args) {
        int firstNum = 5;
        int secondNum = 9
        int maxNum = max(firstNum, secondNum);
    }
    static int max(int firstNum, int secondNum) {
        int result;
        if (num1 <= num2)
            result = num1;
        else
            result = num2;
        return result;
    }
}
```

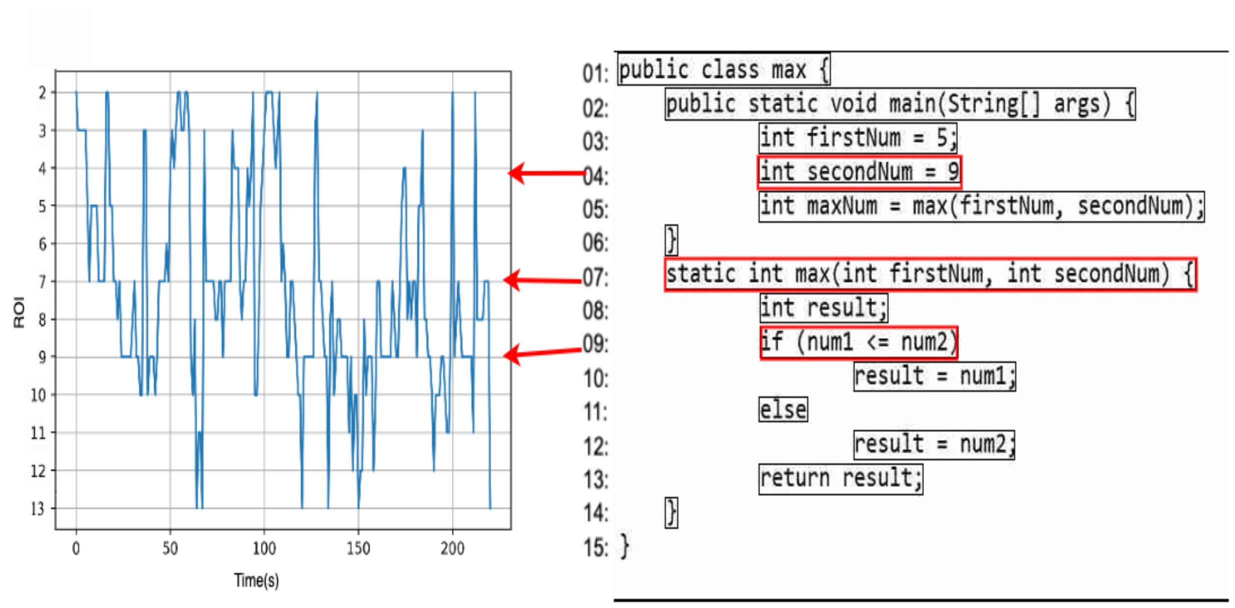
Fig. 7 Q1 with TROIs

Table 1 indicates how the fixation scan path is relevant to the TROIs. Figure 7 represents all the ROIs along with the additional indication of the TROIs. The fixation scan path of a subject of Group D(0%) is shown in Figure 8. On the other hand, Figure 9 illustrates the relevant details of Group A(100%).



```
public class max {
    public static void main(String[] args) {
        int firstNum = 5;
        int secondNum = 9
        int maxNum = max(firstNum, secondNum);
    }
    static int max(int firstNum, int secondNum) {
        int result;
        if (num1 <= num2)
            result = num1;
        else
            result = num2;
        return result;
    }
}
```

Fig. 8 Fixation scan path of a student from Group D (grade: 0%) in Q1



```
public class max {
    public static void main(String[] args) {
        int firstNum = 5;
        int secondNum = 9
        int maxNum = max(firstNum, secondNum);
    }
    static int max(int firstNum, int secondNum) {
        int result;
        if (num1 <= num2)
            result = num1;
        else
            result = num2;
        return result;
    }
}
```

Fig. 9 Fixation scan path of a student from Group A (Grade: 100%) in Q1

Student Group	Participant count	Grade	Avg. Fixation Count (%) ratio in TROI		
			Line 4	Line 7	Line 9
A	1	100%	4.98	20.81	24.43
B	3	66%	7.03	20.08	16.00
C	6	33%	6.71	18.66	12.94
D	1	0%	7.26	11.17	7.26

Table 2. Q1 parameters for determining relation of TROIs and participant groups.

Our analysis revealed that both groups tended to fixate for shorter durations at the beginning and end of the code sample. From the code perspective, Line 4 contains a syntax error, and Line 7 and 9 represent logical errors.

We can speculate the following from these data points in Table 2:

1. Students with higher grades tend to fixate more than students with lower grades. From Table 2, in the TROIs, if we consider the total of the ratio, Group A: 50.22, Group B: 43.11, Group C: 38.31, Group D: 25.69. Groups with higher grade fixated more in the TROI to understand the logic while the fixation ratio in those regions have a gradual decrease in fixation by the groups with less grade.
2. If we consider specific areas among the TROI for analysis, we can point out that in Line 7 and 9 which are logical error regions, subjects with higher grade tend to fixate more than the lower grade. In Line 7 the ratio value of Group A is 20.81 and in Line 9 it is 24.43, and it gradually decreases as the grade decreases. Line 4 represents a syntax error, and here we see a different perspective, increase in the ratio values with slight deviation. This could imply the possibility that higher grade subjects fixate less in the syntax error regions. But as the participant count in highest

and lowest graded subjects groups are just 1, we could not put much emphasis on this progression of analysis. Error Wise fixation behavior in the TROIs could be investigated in the future research.

3. Another observational point is the Average Fixation Ratio within TROI by Group. This gives an idea of the whole picture of the code comprehension analysis. By analyzing the Average Fixation Ratio data presented in Table 3, it becomes apparent that proficient students exhibit a higher level of attentiveness towards the TROI when solving the given exercise when the code sample is small with less complexity.

Participant Group	Avg. Fixation Ratio within TROI by Group
Group A (100%)	16.740
Group B (66%)	14.373
Group C (33%)	12.772
Group D (0%)	8.563

Table 3. Average Fixation Ratio within TROI by participant group in Q1

RQ2: Are there any patterns in eye tracking data that can be used to identify successful or unsuccessful students in understanding code?

Table 4 and 5 represent the top 10 sequences of transitions between two Subject Groups based on grading.

Subject 1			
ROI	ROI	Transitions	Frequency
5	5	int maxNum = nax(firstNum, secondNum); --> int maxNum = nax(firstNum, secondNum);	22
2	2	public static void main(String[] args) { --> public static void main(String[] args) {	18
1	1	public class max { --> public class max {	15
7	7	static int max(int firstNum, int secondNum) { --> static int max(int firstNum, int secondNum) {	12
2	1	public static void main(String[] args) { --> public class max {	7
9	9	if (num1 <= num2) --> if (num1 <= num2)	7
3	2	int firstNum = 5; --> public static void main(String[] args) {	6
1	2	public class max { --> public static void main(String[] args) {	5
12	12	result = num2; --> result = num2;	4
5	4	int maxNum = nax(firstNum, secondNum); --> int secondNum = 9	4

Table 4. Top 10 sequence of backtracking transitions of the Group D student in Q1

Subject 1			
ROI	ROI	Transitions	Frequency
7	7	static int max(int firstNum, int secondNum) { --> static int max(int firstNum, int secondNum) {	22
8	7	int result; --> static int max(int firstNum, int secondNum) {	9
7	8	static int max(int firstNum, int secondNum) { --> int result;	8
9	10	if (num1 <= num2) --> result = num1;	6
8	9	int result; --> if (num1 <= num2)	6
8	8	int result; --> int result;	6
7	9	static int max(int firstNum, int secondNum) { --> if (num1 <= num2)	6
10	10	result = num1; --> result = num1;	6
2	2	public static void main(String[] args) { --> public static void main(String[] args) {	6

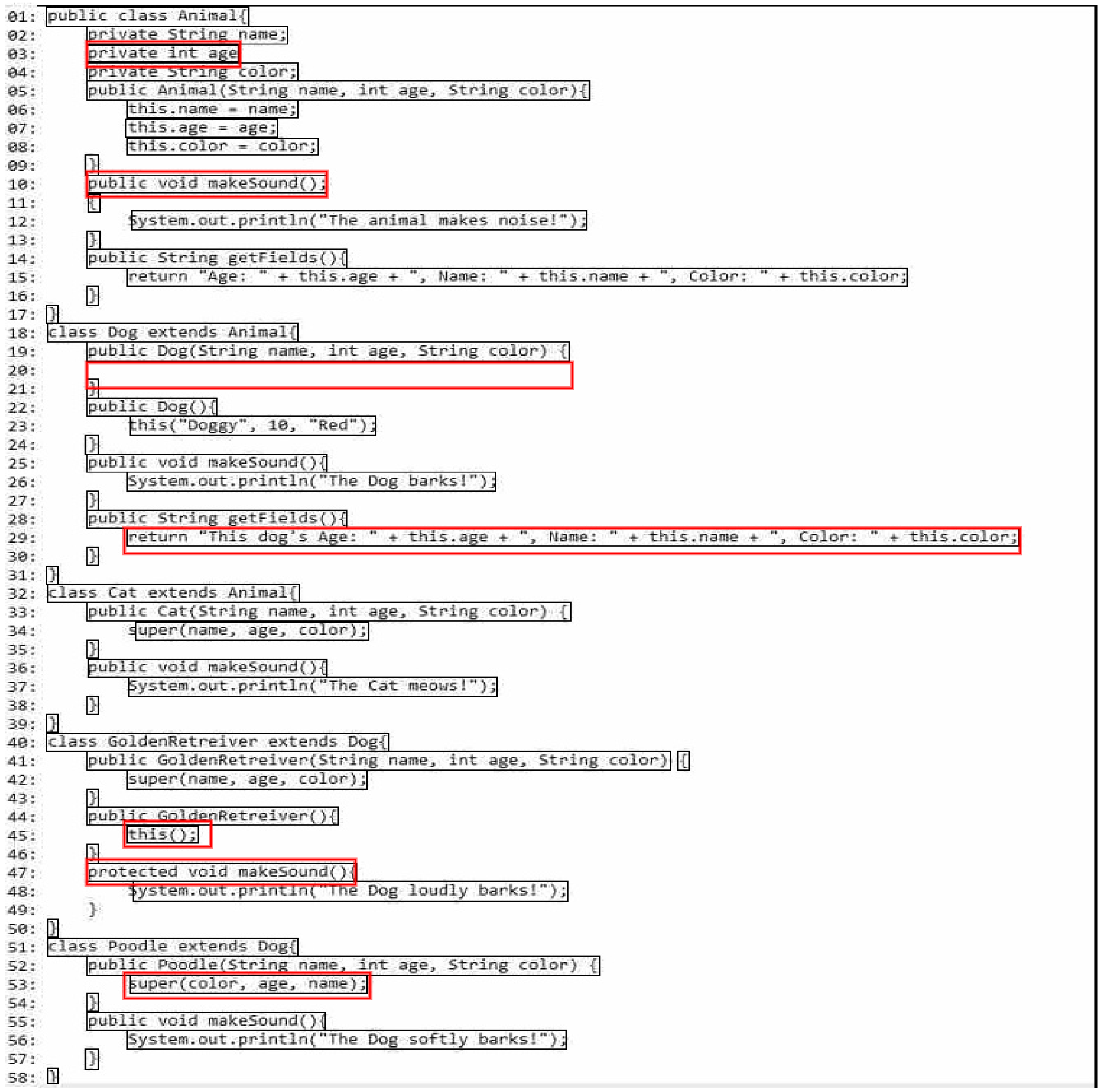
Table 5. Top 10 sequence of backtracking transitions of the Group A student in Q1

The compilation of the top 10 most frequently used backtracking transitions among student groups demonstrates their progression in understanding the code. Students with high scores devote more time to comprehending the logical aspect of the code. Group A backtracked more in the TROI regions and successfully figured out the answer. In Q1, lines are pretty few, and the errors are seeded closer, so with fewer fixation transitions, Group A was able to determine 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 4 and 5, 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.

5.1.2 CODE REVIEW EXERCISE: QUESTION2(Q2)

RQ1: How can the use of targeted regions of interest (TROIs) aid in the analysis of code comprehension by students?

Figure 10 represents the Q2, along with the TROI in the Q2.



```
public class Animal{
    private String name;
    private int age
    private String color;
    public Animal(String name, int age, String color){
        this.name = name;
        this.age = age;
        this.color = color;
    }
    public void makeSound();
    {
        System.out.println("The animal makes noise!");
    }
    public String getFields(){
        return "Age: " + this.age + ", Name: " + this.name + ", Color: " + this.color;
    }
}
class Dog extends Animal{
    public Dog(String name, int age, String color) {

    }
    public Dog(){
        this("Doggy", 10, "Red");
    }
    public void makeSound(){
        System.out.println("The Dog barks!");
    }
    public String getFields(){
        return "This dog's Age: " + this.age + ", Name: " + this.name + ", Color: " + this.color;
    }
}
class Cat extends Animal{
    public Cat(String name, int age, String color) {
        super(name, age, color);
    }
    public void makeSound(){
        System.out.println("The Cat meows!");
    }
}
class GoldenRetreiver extends Dog{
    public GoldenRetreiver(String name, int age, String color) {
        super(name, age, color);
    }
    public GoldenRetreiver(){
        this();
    }
    protected void makeSound(){
        System.out.println("The Dog loudly barks!");
    }
}
class Poodle extends Dog{
    public Poodle(String name, int age, String color) {
        super(color, age, name);
    }
    public void makeSound(){
        System.out.println("The Dog softly barks!");
    }
}
```

Fig. 10 Q2 with TROIs

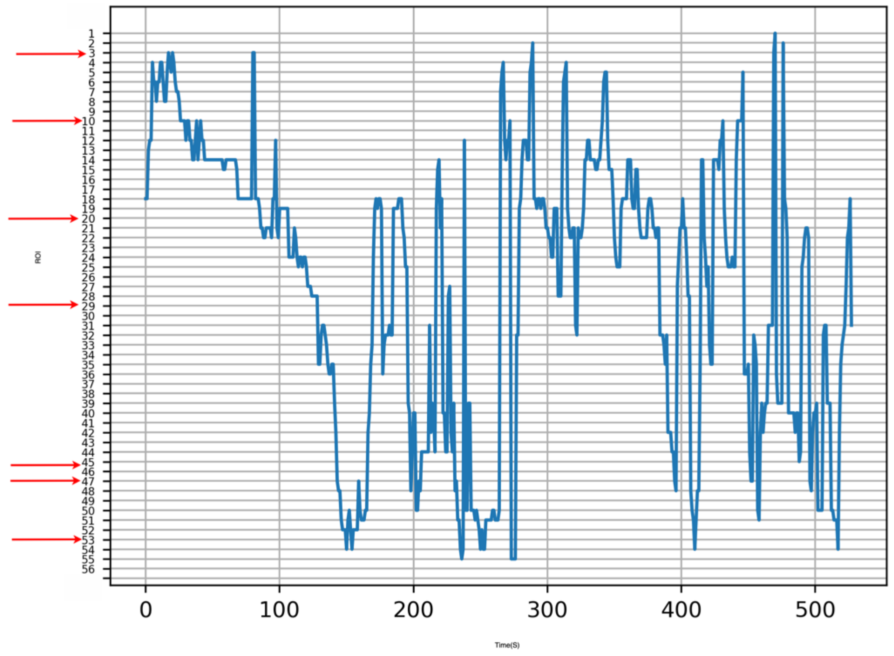


Fig. 11 Fixation scan path of a student from Group D (grade: 14%) in Q2

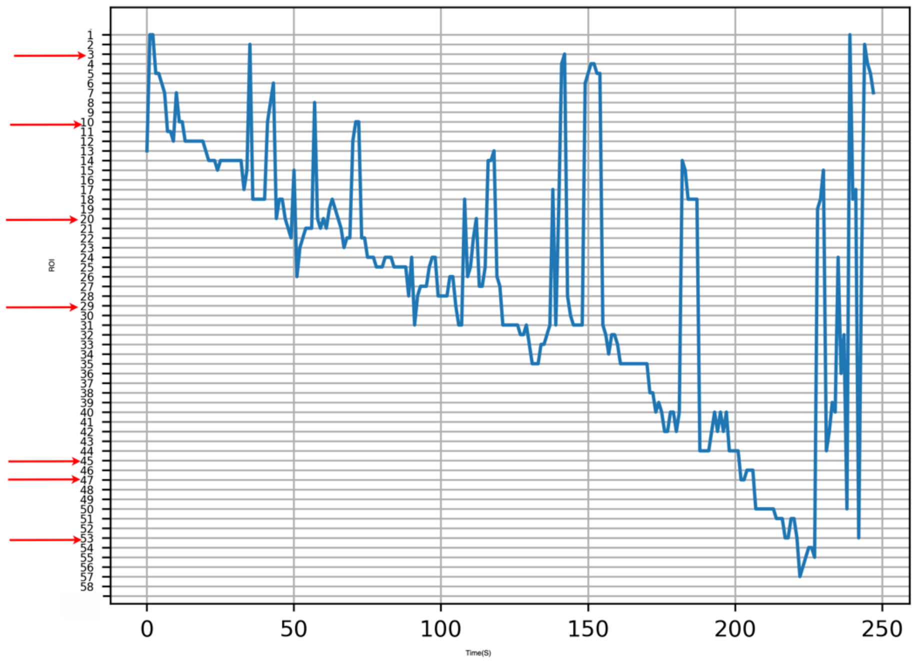


Fig. 12 Fixation scan path of a student from Group A (grade: 72%) in Q2

Student Group	Participant count	Grade	Avg. Fixation Count (%) ratio in TROI						
			Line 3	Line 10	Line 20	Line 29	Line 45	Line 47	Line 53
A	2	72%	0.865	3.19	1.21	0.2	0.67	1.92	0.9
B	1	58%	0	4.77	0.43	0	0.22	1.95	0.22
C	3	28%	3.36	2.57	0.51	0.07	0.37	1.18	0.37
D	3	14%	1.12	4.15	0	0	0.28	2.84	0

Table 6. Q2 parameters for determining relation of TROIs and participant groups

Figures 11 and 12 represent the fixation scan paths of one representable sample from Group D (14%) and Group A (72%) respectively. From the figures, it is evident that low-scoring subjects skimmed the full code more often than the high-scoring subjects.

Table 6 presents the average fixation count (%) ratio, where a value less than 1 indicates that the subject skipped the lines without visiting the region; this behavior could indicate a lack of understanding. Notably, Group D tends to fixate more frequently on the TROIs located at the beginning of the code sample, indicating a preference for top-down processing. In contrast, Group A appears to navigate the code linearly, from top to bottom, to gain a better understanding of the material. This behavior is further illustrated in Figures 11 and 12

We can make the following assumptions based on these data points:

1. Students with higher grades are more fixated on TROIs than students with lower grades. Group A has the total ratio of 8.96 whereas Group D has 8.39.
2. The average fixation ratio within TROI by Group (Table 7) property yields a similar result to that of Q1, albeit with slight deviation from Group B which had only one participant. However, the pattern remains consistent, with Group A exhibiting a higher level of attentiveness towards the

TROI when solving the given exercise, especially in cases where the code sample is large and more complex. Group B can be considered insignificant here as only one subject belongs to this group.

Participant Group	Avg. Fixation Ratio within TROI by Group
Group A (72%)	1.279
Group B (58%)	1.084
Group C (28%)	1.205
Group D (14%)	1.201

Table 7. Average Fixation Ratio within TROI by participant group in Q2

RQ2: Are there any patterns in eye tracking data that can be used to identify successful or unsuccessful students in understanding code?

In order to visualize and understand better, we will categorize the regions into four separate categories of regions marked with separate colors, i.e., 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

Subject 1			Subject 2			Subject 3		
ROI	ROI	**Frequency**	ROI	ROI	**Frequency**	ROI	ROI	**Frequency**
12	12	**21**	14	14	**33**	32	32	**17**
39	39	**19**	18	18	**24**	1	1	**16**
14	14	**15**	19	19	**12**	28	28	**14**
18	18	**14**	21	22	**11**	50	50	**12**
28	28	**11**	21	21	**11**	15	15	**12**
27	27	**10**	51	51	**11**	31	31	**12**
47	47	**10**	50	50	**10**	12	12	**11**
24	24	**9**	18	19	**9**	4	4	**9**
35	35	**9**	12	12	**8**	47	47	**9**
44	44	**9**	22	22	**8**	10	10	**8**

Table 8. Top 10 sequence of backtracking transitions of the Group D(14% grade) subjects in Q2

Subject 1			Subject 2		
ROI	ROI	**Frequency**	ROI	ROI	**Frequency**
35	35	**11**	14	14	**23**
14	14	**10**	12	12	**16**
31	31	**9**	44	44	**16**
18	18	**8**	15	15	**14**
12	12	**6**	10	10	**13**
50	50	**6**	21	21	**11**
44	44	**6**	12	14	**9**
25	25	**6**	42	40	**8**
24	24	**5**	14	12	**8**
42	40	**5**	28	28	**8**

Table 9. Top 10 sequence of backtracking transitions of the Group A(72% grade) subjects in Q2

If we consider Table 8 and 9 and calculate the average backtracking transition frequency of Subjects in each group towards the four categorical regions, we can have the following metrics:

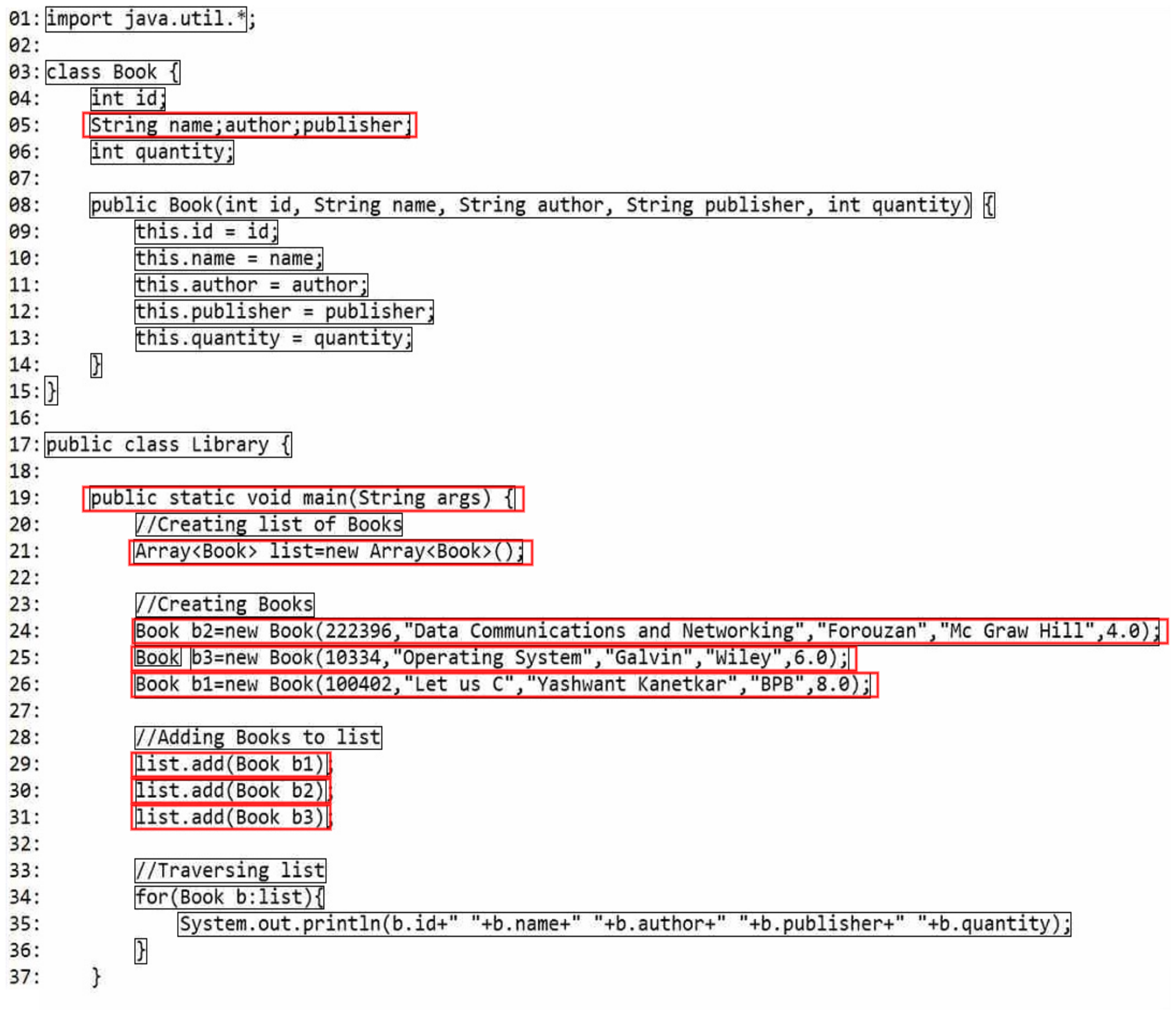
1. Method Declaration Regions:
    a. Group D: 46.66
    b. Group A: 49.50
2. Class Declaration Regions:
    a. Group D: 25.00
    b. Group A: 10.50
3. Return statement or Print Statement Regions:
    a. Group D: 17.33
    b. Group A: 26.50
4. End Curly Brace Regions of Method or Class:
    a. Group D: 34.33
    b. Group A: 21.00

From these metrics, we can speculate on the cognitive processes of subjects in comprehending different aspects of the code. The Q2 coding exercise contains more regions and methods than other exercises, and the analysis illustrates that Group A backtracked more on the return or print statement (26.50) than Group D (17.33) where Group D backtracked more to the end of the method (34.33), which is higher than Group A (21.00). This could imply that Group A intends to understand what is happening in the method body, while Group D may be skimming the code with less understanding. Additionally, Group A transitioned their fixation more (49.50) than Group D in the method declaration regions (46.66), but less (10.50) than Group D (25.00) in the class declaration regions. This could imply that high-scoring subjects tend to understand the class declarations quickly and less fixated on class declaration but more fixated on method declaration.

5.1.3 CODE REVIEW EXERCISE: QUESTION3(Q3)

RQ1: How can the use of targeted regions of interest (TROIs) aid in the analysis of code comprehension by students?

Q3 is a more difficult coding exercise than the previous two. Subjects needed to have a better understanding of Java to answer them correctly.



```java
import java.util.*;

class Book {
    int id;
    String name;author;publisher;
    int quantity;

    public Book(int id, String name, String author, String publisher, int quantity) {
        this.id = id;
        this.name = name;
        this.author = author;
        this.publisher = publisher;
        this.quantity = quantity;
    }
}

public class Library {

    public static void main(String args) {
        //Creating list of Books
        Array<Book> list=new Array<Book>();

        //Creating Books
        Book b2=new Book(222396,"Data Communications and Networking","Forouzan","Mc Graw Hill",4.0);
        Book b3=new Book(10334,"Operating System","Galvin","Wiley",6.0);
        Book b1=new Book(100402,"Let us C","Yashwant Kanetkar","BPB",8.0);

        //Adding Books to list
        list.add(Book b1);
        list.add(Book b2);
        list.add(Book b3);

        //Traversing list
        for(Book b:list){
            System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
        }
    }
```

Fig. 13 Q3 with TROIs

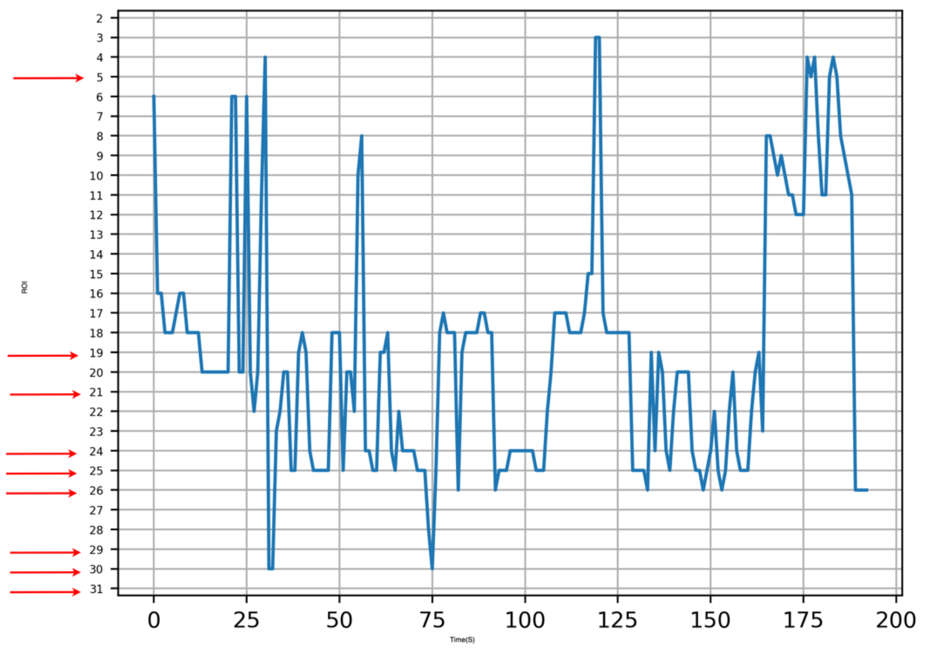


Fig. 14 Fixation scan path of a student from Group D (grade: 20%) in Q3

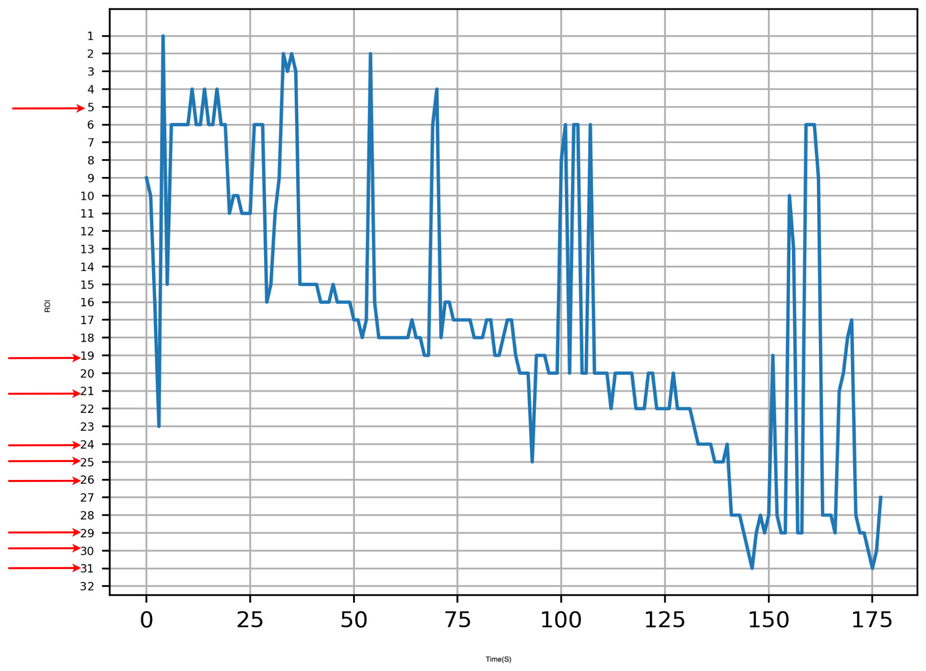


Fig. 15 Fixation scan path of a student from Group A (grade: 80%) in Q3

Student Group	Participant count	Grade	Avg. Fixation Count (%) ratio in TROI								
			Line 5	Line 19	Line 21	Line 24	Line 25	Line 26	Line 29	Line 30	Line 31
A	1	80%	0	5.06	0.56	**2.81**	**2.25**	**0**	5.62	1.69	1.12
B	2	60%	1.19	4.205	0.15	**2.46**	**2.83**	**1.315**	2.195	3.99	0
C	7	40%	2.35	4.62	0.26	**4.86**	**3.65**	**1.36**	1.02	2.27	0.16
D	2	20%	1.68	2.52	0	**6.53**	**10.16**	**3.23**	0	2.57	0

Table 10. Q3 parameters for determining relation of TROIs and participant groups.

Table 10 in Q3 reveals a specific pattern in code comprehension when highly compact and condensed TROIs are involved. Figure 13 demonstrates that TROIs 24, 25, and 26 are closely located, and based on the programming language concept perspective, these regions represent complex code lines. To evaluate the cognitive process involved in these regions, let us consider the TROIs in this specific area:

24: Book b2=new Book(222396,"Data Communications and Networking","Forouzan","Mc Graw Hill", 4.0);

25: Book b3=new Book(10334,"Operating System","Galvin","Wiley",6.0);

26: Book b1=new Book(100402,"Let us C","Yashwant Kanetkar","BPB",8.0);

Region 24 requires subjects to process several different elements, including understanding that it takes three strings, an integer, and another integer as parameters, and working with five data types, potentially causing speculation about a syntax error. Group D fixated heavily on regions 24 and 25, attempting to comprehend their meaning, as shown in Table 10. In contrast, the ratio values for ROI 24, 25, and 26 were 2.81, 2.25, and 0 for Group A, 2.46, 2.83, and 1.315 for Group B, and 4.86, 3.65, and

1.36 for Group C, indicating that higher-grade subjects spent more time on the first region before quickly moving through the next regions, using their knowledge from the initial fixation to reduce processing time in later fixations. Conversely, the ratio values for Group D were 6.53, 10.16, and 3.23, indicating that subjects with lower grades were unable to comprehend the first region and became increasingly confused and fixated more on later regions. This behavior can also be observed in Figure 14 and 15 where we can see the low scoring student fixated a lot on the condensed and closed regions than the high scoring student.

The subjects in this study were expected to demonstrate a pattern of identifying the problem and using contextual information to process later closed and identical regions with fewer fixations. Groups A, B, and C showed this pattern, but Group D, as the lowest-scoring group, exhibited a different behavior. This cognitive approach toward closed and condensed regions revealed a different pattern in the Average Fixation Ratio within the TROI by Group. As shown in Table 11, Group A fixated less than Group D in TROIs where the progression is in ascending order, while in Q1 and Q2, it was the opposite. This explanation confirms the reasoning behind this different order in this metric. Table 10 reveals that, for consecutive lines with errors (i.e., 24-26, 29-30), Group A tended to fixate less on the later lines and, on average, fixated less to detect the significance. Conversely, subjects from Group D either fixated a lot for unsuccessful code comprehension or never answered those questions because they did not fixate at all.

Participant Group	Avg. Fixation Ratio within TROI by Group
Group A (80%)	2.123
Group B (60%)	2.038
Group C (40%)	2.298
Group D (20%)	2.966

Table 11. Average Fixation Ratio within TROI by participant group in Q3

The findings suggest that while some subjects demonstrated an ability to process information quickly and effectively, others struggled to comprehend the code, requiring more fixations and taking longer to identify the problem. Additionally, the study shows that the pattern of fixations varied among different groups, with higher-performing groups exhibiting a more efficient pattern of fixations, whereas lower-performing groups displayed a less efficient pattern. These results emphasize the significance of differences in understanding between various subject areas and how they impact the ability to comprehend complex code.

RQ2: Are there any patterns in eye tracking data that can be used to identify successful or unsuccessful students in understanding code?

Subject 1			Subject 2		
ROI	ROI	**Frequency**	ROI	ROI	**Frequency**
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	**8**
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 12. Top 10 sequence of backtracking transitions of the Group D(20% grade) subjects in Q3

Subjects from Group A had the tendency to briefly skim the code from top to bottom first, and in general, if the subject did fixate on a particular region during the initial scan, these fixations were less frequent. Q3 contains some helpful comment lines (Line # 20, 23,28,33) and if we consider the backtracking transitions towards those comment lines from Table 12 and 13, we see that, on average, Subjects in Group D navigated to the comment regions less than Group A. Group D had average 16 backtracking transitions towards the comment regions while Group A had 22 . Group A fixated more on the comment regions, probably to figure out the answer. Group D, on the other hand, was less focused on comment lines, indicating less understanding than Group A.

Subject 1		
ROI	ROI	**Frequency**
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**

Table 13. Top 10 sequence of backtracking transitions of the Group A(80% grade) subjects in Q3

CHAPTER 6

DISCUSSION OF RESULTS

6.1 SUMMARY OF RESULTS

The study examined code review exercises, and the results of this experiment gave a more in-depth understanding of the cognitive processes that students undertake when studying source code. We prefer to evaluate in the context of lines rather than granular metrics such as words or tokens since lines provide a greater variety of areas for study. By extracting the eye metrics using hit-or-miss ROI border detection, we can determine which regions students concentrate on to comprehend the code sample. This knowledge is valuable for both teaching students how to evaluate the code of others and teaching students how to write easily understood source code.

1. In summary we can say that the high scoring subjects fixated more in 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 the Group A.
2. The Average Fixation Ratio within TROI by group represents a more detailed picture of the code comprehension in this study. Compared with the whole region analysis, this filtering enabled us to determine how fixations in certain regions impact the comprehension by different groups. In code review exercises Q1 and Q2, the metric represents that high scoring subjects have fixated more in the TROIs and thus their score is higher. Q3 represented a different scenario of condensed and closed TROIs case, where the high-scoring subjects fixated less and quickly solved the problem statements.
3. Rather than analyzing the transition patterns of the full code, our study focused solely on the top 10 transitions or backtracking in order to gain insight into the more fixated regions and their relevance to understanding the code. The analysis revealed that the fixation pattern provides meaningful insights into the code comprehension of different subject groups. High-scoring groups tend to backtrack to the method definition frequently for analyzing the logical comprehension of

the code. In contrast, the fixation pattern of low-scoring groups indicates abrupt skimming of the code, moving back and forth to the end of the methods indicating failed comprehension analysis.

CHAPTER 7

CONCLUSION

Overall, the hands-on programming exercise experiment was carefully planned and carried out so that we could collect data that would help us figure out how well the NiCATS software helps Computer Science instructors figure out how much their students understand. To process the data, we identified seeded error regions as the TROIs and analyzed fixation statistics (count, average, ratio, etc.) patterns to gain meaningful insights into students' comprehension of errors in code. Processing code at different speeds does not necessarily mean that one person understands the code less than another. By using ratio properties, we have compared data without solely relying on time. It is important to identify the specific regions within the code that are important, and by analyzing how individuals perceive and understand these regions, we have gained valuable insights. Presenting visual representations of eye metric data, as reported in this study, can be valuable for researchers investigating the cognitive processes utilized by students to comprehend information in a laboratory context.

This study will help the educators define the region beforehand and analyze the important regions only as per the requirements. The subject's cognitive load could spread out over the whole region, even in the unnecessary white or blank regions, but determining the attentiveness for solving problems in a certain regional context can reduce the overhead of additional complicated analysis. However, it's important to note that fixation counts and durations alone cannot provide a complete picture of code comprehension. Other factors, such as the complexity of the code, the student's prior knowledge and cognitive load, can also influence code comprehension. So, fixation data should be looked at along with other measures, like self-reported understanding, performance on related tasks, and interviews with the students, to get a fuller picture of code comprehension.

CHAPTER 8

FUTURE WORK

In future work, a significant improvement to the current system could be achieved through the automation of certain aspects of the data processing pipeline. Eye metrics calculation (fixations, saccades) and ROI labeling are done in the post-hoc process, automating them will enable seamless integration into any platform. In the post-hoc process, the ROI labeling is done using Amazon TextTract, but this could be replaceable with an OCR (Optical Character Recognition) machine if we want to generate the ROI labeling while collecting the data from the student machine.

Based on the findings of this study, future work can focus on investigating the differences in the amount of time that students with higher grades spend on fixing syntax errors versus logic errors. The current study suggests that these students spend more time fixated in areas where errors exist, but it is unclear if this fixation is consistent across different types of errors. Further investigation into this aspect could provide a better understanding of the cognitive processes involved in error correction and potentially inform the development of more effective educational interventions.

The TROI concept could extend the student comprehension process not only in the coding environments, but in other classroom environments as well. Educators can predefine the regions on which students should focus during a certain period of time in a presentation or specific class, and during post-hoc or runtime, they can visualize if the students are paying attention to those certain important regions. This could add diversity to NiCATS.

NiCATS can act as a base platform for eye tracking studies. One collaborative study has been done finding correlation between eye tracking metrics with EEG signals data. That showed some promising results in aggregating other inputs with eye tracking metrics. The System could be enhanced by adding external system inputs for extending its research capabilities.

REFERENCES

Boswell, Bradley, A. Sanders, A. Allen, G. Singh Walia and M. Shakil Hossain, "Using AI-based NiCATS System to Evaluate Student Comprehension in Introductory Computer Programming Courses," 2022 IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden, 2022, pp. 1-9, doi: 10.1109/FIE56618.2022.9962681.

Sanders, Andrew, Bradley Boswell, Gursimran Singh Walia, and Andrew Allen. "Non-Intrusive Classroom Attention Tracking System (Nicats)" 2021 IEEE Frontiers in Education Conference (FIE) (October 13, 2021). https://doi.org/10.1109/FIE49875.2021.9637411

Fritz, Thomas, Andrew Begel, Sebastian C. Müller, Serap Yigit-Elliott, and Manuela Züger. "Using Psycho-Physiological Measures to Assess Task Difficulty in Software Development." Proceedings of the 36th International Conference on Software Engineering (May 31, 2014). https://doi.org/10.1145/2568225.2568266

Hijazi, Haytham, Jose Cruz, Joao Castelhano, Ricardo Couceiro, Miguel Castelo-Branco, Paulo de Carvalho, and Henrique Madeira. "IReview: An Intelligent Code Review Evaluation Tool Using Biofeedback." 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE) (February 11, 2022). https://doi.org/10.1109/ISSRE52982.2021.00056

Rodeghero, Paige, and Collin McMillan. "An Empirical Study on the Patterns of Eye Movement during Summarization Tasks." 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (November 9, 2015). https://doi.org/10.1109/ESEM.2015.7321188

Rosengrant, David, Doug Hearrington, Kerriann Alvarado, Danielle Keeble, N. Sanjay Rebello, Paula V. Engelhardt, and Chandralekha Singh. "Following Student Gaze Patterns in Physical Science Lectures." AIP Conference Proceedings (February 9, 2012). https://doi.org/10.1063/1.3680060

Salvucci, Dario D., and Joseph H. Goldberg. “Identifying Fixations and Saccades in Eye-Tracking Protocols.” Proceedings of the symposium on Eye tracking research & applications - ETRA '00 (November 2000). https://doi.org/10.1145/355017.355028

Dewan, M. A. A., Murshed, M., & Lin, F. (2019). “Engagement detection in online learning: a review.”, Smart Learning Environments, 6(1), 1-11. doi: 10.1186/s40561-018-0080-z.

Sharafi, Zohreh, Timothy Shaffer, Bonita Sharif, and Yann-Gael Gueheneuc. “Eye-Tracking Metrics in Software Engineering.” 2015 Asia-Pacific Software Engineering Conference (APSEC) (May 12, 2016). https://doi.org/10.1109/APSEC.2015.53

Tabassum, Tasnia, Andrew A. Allen, and Pradipta De. “Non-Intrusive Identification of Student Attentiveness and Finding Their Correlation with Detectable Facial Emotions.” Proceedings of the 2020 ACM Southeast Conference (2020). https://doi.org/10.1145/3374135.3385263

Veliyath, Narayanan, Pradipta De, Andrew A. Allen, Charles B. Hodges, and Aniruddha Mitra. “Modeling Students' Attention in the Classroom Using Eyetrackers.” Proceedings of the 2019 ACM Southeast Conference (April 18, 2019). https://doi.org/10.1145/3299815.3314424

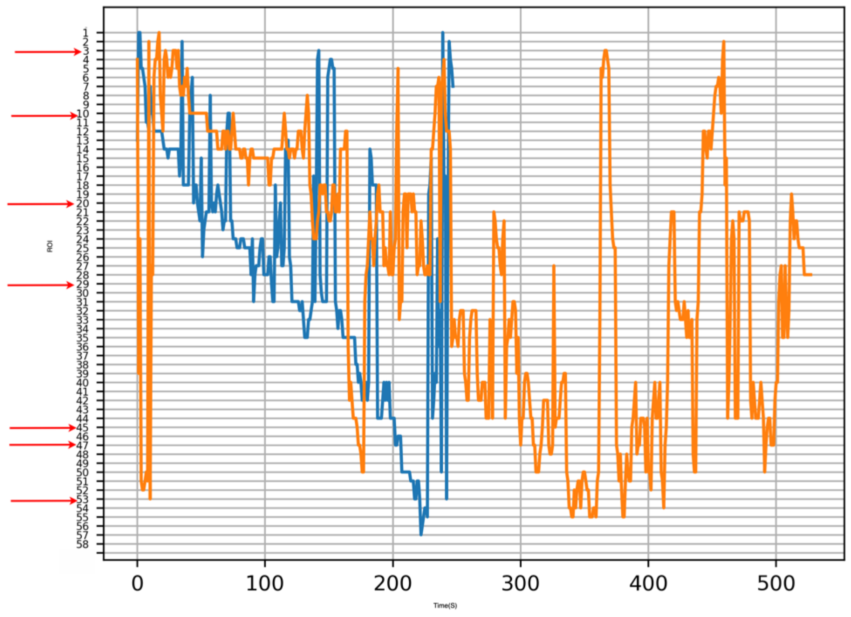
Hossain, M. S., Pandya, D., Allen, A., & Hamza-Lup, F. G. (2023). “Learner Attention Quantification Using Eye Tracking and EEG Signals.” In K. Arai (Ed.), Proceedings of the Future Technologies Conference (FTC) 2022, Volume 2 (pp. 1-15). Lecture Notes in Networks and Systems, vol 560. Springer. doi: https://doi.org/10.1007/978-3-031-18458-1_57.

Whitehill, Jacob, Zewelanji Serpell, Yi-Ching Lin, Aysha Foster, and Javier R. Movellan. "The faces of engagement: Automatic recognition of student engagement from facial expressions." IEEE Transactions on Affective Computing 5, no. 1 (2014): 86-98.

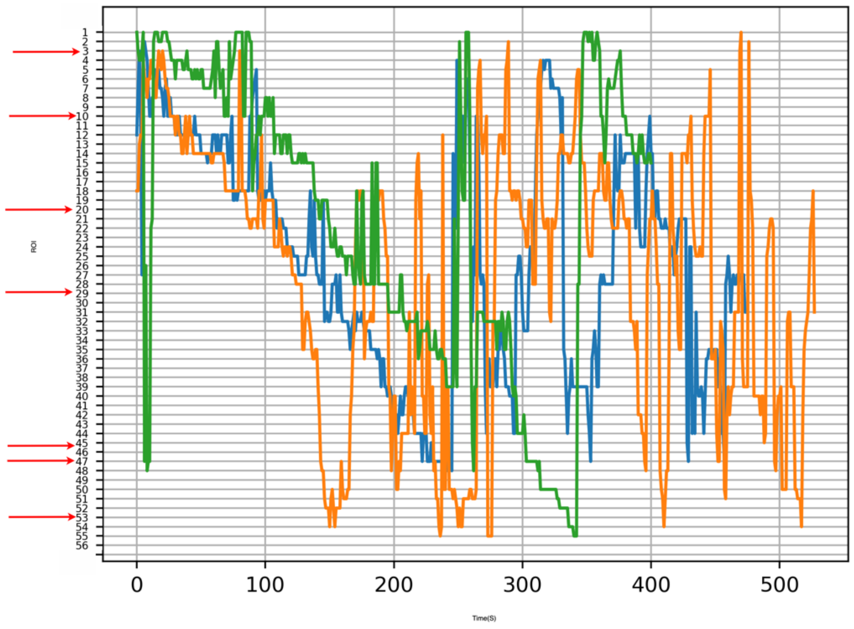
Zhu, Ziwei, Sebastian Ober, and Roozbeh Jafari. “Modeling and Detecting Student Attention and Interest Level Using Wearable Computers.” 2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN) (June 1, 2017). https://doi.org/10.1109/BSN.2017.7935996.

APPENDIX

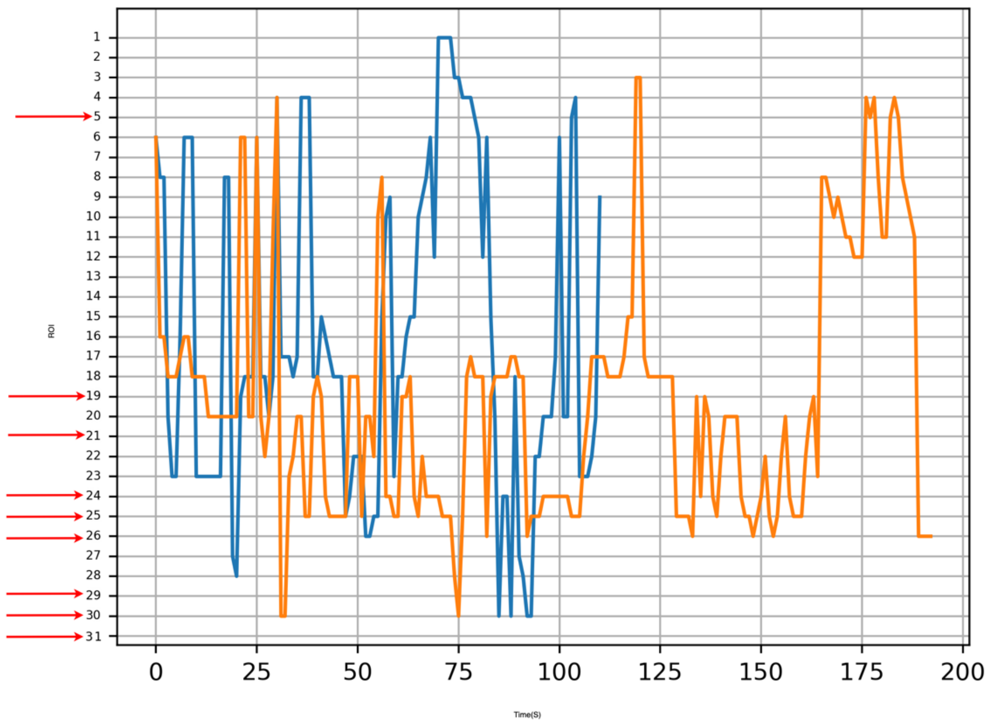
FIXATION SCAN PATH OF STUDENTS FROM TOP & BOTTOM GROUP IN CODE SAMPLE WITH TROI INDICATION



Fixation scan path of students from Group A (72%) in Code Sample 2



Fixation scan path of students from Group D (14%) in Code Sample 2



Fixation scan path of students from Group D (20%) in Code Sample 3