# On the Suitability of Real-Time Assessment of Programming Proficiency using Gaze Properties

**Calvin Liang**[1], **Jakob Karolus**[2], **Thomas Kosch**[2], **Albrecht Schmidt**[2]
[1]Tufts University, Medford, United States; [2]LMU Munich, Munich, Germany
[1]calvinalanliang@gmail.com, [2]{jakob.karolus, thomas.kosch, albrecht.schmidt}@ifi.lmu.de

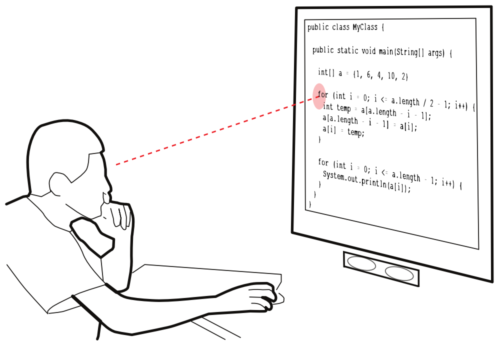
## ABSTRACT
Evaluating programming proficiency has become more relevant as the demand for coding skills has increased. Current methods, such as questionnaires or interviews, are methods that lack intuition, flexibility, and real-time capabilities. In our work, we investigate eye gaze behavior as an estimate for skill assessment. Specifically, we conducted a study (N=14) using an eye tracker to analyze the participants' abilities to understand source code by presenting them with a series of programs. We evaluated their eye movements based on common eye tracking metrics and identified mutual task-solving strategies among the participants. While we cannot relate these indicators to programming proficiency directly, this study serves as an evaluation of real-time methods for evaluating programming proficiency.

## ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

## Author Keywords
Eye Tracking; Programming Proficiency; Behavioral Patterns.

## INTRODUCTION
Due to an increasing demand for coding skills in academia and industry, evaluating the programming proficiency of a possible employee or student is a vital step towards acceptance. Yet, objective models to evaluate programming proficiency have been scarcely explored. Gaze properties have been shown to be influenced by the representation form of text such as its language and difficulty [8]. Eye tracking has come a long way and affordable consumer products provide decent accuracy for eye gaze interaction. Hence, eye tracking is already available for everyone to be used. Previous research showcased, that it is possible to assess cognitive processes by analyzing eye gaze [6], which impact text understanding. Post hoc and real-time evaluation language proficiency using machine learning has already been shown to be feasible [5, 7].

*PerDis '18* June 6–8, 2018, Munich, Germany
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5765-4/18/06.
DOI: https://doi.org/10.1145/3205873.3210702



```
public class MyClass {

 public static void main(String[] args) {

  int[] a = {1, 6, 4, 10, 2}

  for (int i = 0; i <= a.length / 2 - 1; i++) {
   int temp = a[a.length - i - 1];
   a[a.length - i - 1] = a[i];
   a[i] = temp;
  }

  for (int i = 0; i <= a.length - 1; i++) {
   System.out.println(a[i]);
  }
 }
}
```

**Figure 1. A user's gaze while analyzing program source code on a desktop PC.**

The application of using eye tracking to evaluate programming proficiency (cf. Figure 1) in school settings has broad implications. Through this, teachers can develop an insightful understanding of each student's progress. This enables educators to foster specific learning modules, to provide additional support, or to modify teaching plans according to the overall progress. Continuous retrieval and processing of eye gaze enables frequent evaluation during exercises rather than having single exams at the end of courses. Furthermore, user interfaces can provide implicit assistance by using proficiency as a metric. Certain abilities can then be specifically fostered to expand the skill set of the user.

In our work, we investigate source code as an intermediate textual representation. Research has identified self-reported scores [3] and past performance [2] as the best indicators of programming proficiency. We introduce eye movements as an additional factor. So far researchers have leveraged eye tracking in combination with customized source code visualizations to test programming proficiency [1]. Contrarily, we evaluate the gathered gaze data without showing any supplemental code visualizations to the participants.

We present a study which investigates the feasibility of eye gaze analysis to estimate programming language proficiency and discuss further implications in examination and education settings. Our contribution is two-fold: We (1) present a study in which we correlate eye gaze behavior with several program comprehension tasks of different complexities and (2) conclude with a discussion about future implications of using eye tracking as a modality for proficiency estimation.

## METHODOLOGY

For our study, we adapted twelve programs (mainly algorithms and string/array manipulation) from online beginner courses and lectures[1], relying on Java as a programming language[2]. To determine the difficulty of our programs, five experts with an academic background in computer science[3] rated each program on a three-point scale. The average score determined the program's difficulty. The lowest was 1.2 for *faculty calculation*, while the highest was 2.6 for *exponental calcuation* ($\bar{x} = 1.7, SD = 0.44$).

We recruited 20 participants from the University of Stuttgart through mailing lists for our study. All participants were students and reported different programming abilities. The data of 14 participants (4 female) was submitted for further analysis. All participants were ages between 18 and 25 years ($\bar{x} = 21.4, SD = 1.93$).

Participants first signed a consent form and provided their demographic data as well as rate their programming ability on a 10-point scale. After calibrating the eye tracker, participants were shown a simple training task. The actual experiment consisted of the twelve programs in randomized order with a break in the middle. Participants were asked to enter the behavior of each program on the following screen. The sequence for each trial was a blank screen, the program itself, and a text entry box. Each study lasted a maximum of 60 minutes. Participants were compensated with 10 Euro.

## RESULTS AND DISCUSSION

Participants had self-reported programming proficiency scores of 3 to 7 ($\bar{x} = 5.5, SD = 1.30$). Additionally, we measured the assessed performance by grading the participants' answers by three experts[3] using a three-point scale. When comparing self-reported scores with the assessed performance, no correlation between the two variables was found. We analyzed eye movements related to fixations and saccades described in past work [4]. However, we were unable to relate these measurements to programming proficiency. Neither self-reported nor assessed performance showed significant differences given program difficulty.

We examined several areas of interest (AOIs) such as method declarations, loops, main classes, and method bodies. No significant relationship between our eye tracking metrics and the respective AOIs were found. We also reviewed participants' strategies when looking through each program. However, there was no distinct strategy when solving the task for highly proficient participants. Participants tend to either use a holistic approach by getting an overview first and then looking at individual methods and code fragments. The other main strategy that we identified revolves around on-demand gazes at specific methods without scanning the whole program first. These patterns are not correlated to proficiency.

[1]www.infosun.fim.uni-passau.de/spl/janet/fMRI/index.php - last access 2018-04-13

[2]www.tiobe.com/tiobe-index/ - last access 2018-04-13

[3]Academic researchers holding a degree in computer science.

Our results show that common eye tracking metrics are not suitable to reliably identify a person's programming proficiency. The same strategy used for analyzing reading patterns is not suitable for evaluating more complex text visualization, such as structured source code. Moreover, we could not identify distinct problem-solving strategies for either proficient nor less proficient participants. The employed strategy is not driven by one's proficiency but rather by personal preferences and experience. However, we believe this provides a vital aspect of education. By visualizing solving strategies, educators are able to identify mistakes more easily and provide tailored feedback. Even during the instruction process, optimal strategies can be visualized and conveyed. To foster further research, we have decided to make our dataset publicly available.

## CONCLUSION

Fast and accurate assessment of programming proficiency is vital as current methods rely on post hoc tests preventing real-time evaluations. In this paper, we have evaluated a method to address this problem by investigating eye movements. Common eye tracking metrics do not contain enough information to discern different proficiency levels. Our findings prove the connection to programming proficiency to be more complex, hence, difficult to grasp using a linear relationship. However, we identified mutual strategies amongst the participants, leading us to conclude that analyzing source code is mainly driven by personal preference and experience.

## REFERENCES

1. R. Bednarik and M. Tukiainen. 2006. An eye-tracking methodology for characterizing program comprehension processes *(ETRA '06)*. ACM, 125–132.
2. D. F. Butcher and W. A. Muth. 1985. Predicting Performance in an Introductory Computer Science Course. *Commun. ACM* (1985).
3. J. Feigenspan, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg. 2012. Measuring programming experience *(ICPC '12)*. IEEE, 73–82.
4. R. J.K. Jacob and K. S. Karn. 2003. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research* (2003).
5. J. Karolus, P. W. Wozniak, L. L. Chuang, and A. Schmidt. 2017. Robust Gaze Features for Enabling Language Proficiency Awareness *(CHI '17)*. ACM, 2998–3010.
6. T. Kosch, M. Hassib, P. Wozniak, D. Buschek, and F. Alt. 2018. Your Eyes Tell: Leveraging Smooth Pursuit for Assessing Cognitive Workload *(CHI '18)*. ACM.
7. P. Martínez-Gómez and A. Aizawa. 2014. Recognition of Understanding Level and Language Skill Using Measurements of Reading Behavior *(IUI '14)*. ACM.
8. K. Rayner. 1998. Eye Movements in Reading and Information Processing: 20 Years of Research. *Psychol Bull* 124, 3 (Nov. 1998), 372–422.