Predicting Student Performance Based on Eye Gaze During Collaborative Problem Solving

Mehmet Celepkolu Kristy Elizabeth Boyer
Computer & Information Science & Engineering
University of Florida
mckolu@ufl.edu, keboyer@ufl.edu

ABSTRACT

Eye-gaze activity provides rich information about individuals' engagement in social interactions. Gaze is one of the strongest visual cues in face-to-face interaction. Previous studies have examined how eye gaze can be used to coordinate social interactions such as turn taking and identifying the focus of attention. In this study, we investigate the role of gaze during collaborative problem-solving tasks, specifically how individuals perform different gazing activities when holding different team roles in pair programming, and also whether the differences in eye gaze (if any) provide predictive insight into learning outcomes. We analyzed 40 students' eye-gaze activities, which were annotated for each second during a collaborative problem-solving task (~50 min on average). The results show that students' roles in the collaborative task have a significant relationship with eye-gaze activities. Moreover, participants' gaze activities can provide predictive insight into their post-test scores. These findings suggest that simple activity measures such as relative frequency of eye-gaze activities can be very useful in understanding the collaborative process.

CCS CONCEPTS

• Applied Computing • Education • Collaborative Learning

KEYWORDS

Eye-Gaze Activity, Collaborative Problem Solving, Team Role

1. INTRODUCTION

Humans use more than words to express themselves. Facial expressions, body orientation, gestures, and other visual cues often serve as a signaling mechanism and provide rich information to the recipient during social interactions. Eye-gaze activity has extensively been studied for understanding the coordination of social interactions [10,19,21], turn-taking behaviors [1,17], and identifying the focus of attention of a speaker [20].

Previous studies show that factors such as conversational setting, individuals' role in the conversation and context of the interaction can have important impacts on individuals' gazing activities. For example, Foddy [12] found that people look at their teammates more in cooperative teams compared to competitive teams. The context of the interaction can also affect the eye gaze activity: when under high cognitive load, people tend to avert gaze away from their partner [9]. Moreover, there is increasing literature on how findings from studies conducted in artificial lab environments differ from natural, real world situations [13,14,31]. Studies conducted in real life environments [15,23] suggest that participants usually do not engage in direct eye contact with the person who they talk to, while participants tend to have direct eye contact in artificial lab environments [33,35].

In this study, we analyze participants' gazing activity in a natural collaborative educational context: pair programming (Figure 1). In pair programming, two students collaboratively construct code by taking turns in different roles: the *driver* is responsible for writing the code and implementing the solution, while the *navigator* helps in catching mistakes and providing feedback on the in-progress solution [38]. In pair programming activities, pairs are encouraged to change roles on a frequent basis [2]. This pair programming approach has long been shown to be effective for teaching programming in introductory programming courses [25,26]. On the other hand, challenges such as conflicts between partners [37] or inequity arising within the discussion [11,24] have been reported in pair programming. There has been criticism on the division of the roles as navigator and driver [3]. In pair programming activities, the navigator may lose track of the activity [18] and have difficulties with following the driver's actions [30] due to not having direct control of the activity. Moreover, some programmers reported feeling more passive and disengaged when holding the navigator role [5,30] and more engaged in the task when holding the driver role [6]. In contrast, some studies suggest that even if students hold the navigator role, they still approach the problem tasks similar to the driver [3,6].

Article Title Footnote needs to be captured as Title Note
†Author Footnote to be captured as Author Note

WOODSTOCK'18, June, 2018, El Paso, Texas USA
© 2018 Copyright held by the owner/author(s). 978-1-4503-0000-0/18/06...$15.00
https://doi.org/10.1145/1234567890

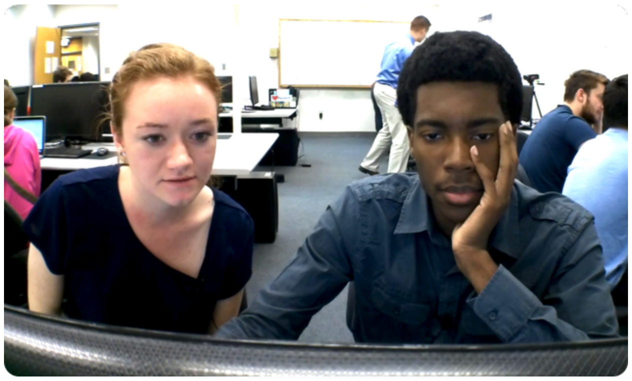

Figure 1. Introductory computer science students engaged in pair programming

This problem highlights an important open question: having navigator and driver roles in pair programming is one of the core differences between pair programming and other collaborative problem-solving methods, but how do students' behaviors differ between these roles? Can their gaze behavior differences (if any) provide predictive insight into understanding the learning outcomes? This paper reports on a study to investigate these questions. We have examined 40 students' gazing activities for each second while they were working on a programming problem with a partner in a pair programming activity (~50 min on average). The gazing activities were manually tagged by three researchers based on five different gaze targets. The relative frequency of each gaze target in each pair programming session was calculated for each participant. We hypothesized that students' eye-gaze activities would be different depending on their roles, and in addition, the differences in gaze activities would be predictors for learning outcomes. This analysis focuses on two research questions:

Research Question 1: What are the gaze distribution differences when students have different roles in pair programming?

Research Question 2: Can gaze distributions be used to predict learning outcomes?

We found that students gazed at six different targets during pair programming: the programming screen, the instruction screen, their partner, their notes, and "other places." The results show that that students tend to look at other places more while holding the navigator role. Moreover, the results show that the relative frequency of eye-gaze activities while holding the driver role can be a predictor of learning outcomes measured at the end of the task. This early work contributes to our understanding of how gazing activities can be used as predictors for learning outcomes.

2. RELATED WORK

There has been a dramatic increase in utilizing eye-tracking techniques in research. Eye tracking provides useful insight into individuals' visual exploration behaviors with minimal bias, since the measures are based on individuals' physical behaviors. This analysis technique has been used successfully in usability studies of single-user interactive systems [36]. Using eye tracking technology in programming education was first introduced by Crosby and Stelovsky [7] in 1990, and its popularity has drastically increased in recent years [22,29]. However, the number of studies on collaborative programming environments, especially on pair programming, is still relatively very small compared to the studies conducted on topics like tractability and debugging [29]. One reason is the difficulty of utilizing eye-tracking technologies for collaborative tasks in which participants do not only gaze at the screen, but rather, at each other and at shared materials.

Stein and Brennan [34] conducted one of the first eye-gaze tracking studies in pair programming and investigated whether seeing a partner's visual focus of attention during debugging can be helpful to programmers. They found that this system helped programmers find programming bugs more quickly. In a recent study, D'Angelo and Begel [8] designed a similar gaze visualization system based on the locations the partner is currently looking at. They found that students spent most of their time looking at where their partner was looking. The results also showed that they could respond to references made by the partner more quickly. In another dual eye-tracking activity, Sharma et. al [32] showed that pairs with better understanding focus on reading/tracing the code flow instead of doing systematic execution of the code. These findings inspired us to further investigate gaze during pair programming in this study which takes role into account.

The impact of programming experience level on gaze activity is another factor that has been studied. Crosby and Stelovsky [7] observed that while novice students focused on comments in the code, expert students focused on meaningful and complex areas. Nivala et. al [27] suggest that while novice students spend more time on attempting to understand the code rather than writing the code, expert students start writing the code sooner. Moreover, expert students had faster eye movement between fixation points. Busjahn et. al [4] showed that novices and experts read the code in different orders: experts read code in a more random manner than the linear form that novices stick to. Based on this prior work, we included students' prior experience as a covariate alongside gaze features in the regression models reported here.

Finally, ways in which eye-gaze patterns during collaboration can provide predictive information about the collaboration have recently been suggested in several studies. For example, Nussli [28] approached the interpretation of dual tracking in two ways. First, they conducted classical statistical analyses to gain insight on how gaze features and some aspects of collaboration are related. Second, they made predictions based on eye-movement patterns about some aspects of collaboration. They suggested that being able to make automatic predictions about the ongoing collaborative process can inform real-time feedback to the collaborators to enhance their interaction. Our study has a similar motivation, as our goal was to explore whether we can make predictions based on eye-movement patterns. However, we differ in methodological approach, as we designed our study by exploring whether we can make predictions on the learning outcomes based on simple relative frequency measures in a collaborative problem-solving task.

Grover et. al [16] have suggested using several different multimodal analytics tools to analyze the pair programming activity of four pairs. In their study, they used a simple annotation scheme for observable activities, which consist of screen pointing, leaning forward, joint attention (looking at screen), taking the mouse, and synchrony in body position. We use a similar approach but take the additional step of building predictive models of learning gain based on the labeled gaze activities.

3. METHODS

3.1 Participants

As part of a large introductory computer science class at a research university in Spring 2017, more than 300 students attended a weekly programming lab. We randomly selected 20 pairs of students who had consented to data collection and collected video of their collaboration process while working on the programming lab. Consent to the data collection was voluntary. Students did not receive any incentive for consenting to data collection, and there was no penalty for declining data collection. Of the 40 students, there were 28 (70%) male students and 12 (30%) female students. The group was balanced in terms of prior experience as 21 (55%) students had no programming experience and 19 (45%) students had some programming experience. The pairs consists of 10 male-male, 2 female-female and 8 male-female student combinations.

3.2 Data Collection Procedure

Students attended class lectures three times per week in a large lecture hall. In addition to lectures, each student attended a mandatory two-hour lab each week facilitated by an undergraduate teaching assistant. In the lab, students were asked to work on a programming task with an assigned partner. After students completed the lab assignment, post-quiz, and post-survey, they were free to leave. In the first week of the semester, students learned about the course logistics and completed surveys on demographics, prior experience in programming, motivation, and personality. Students were also informed about the motivation for, and implementation of, pair programming in the first week. In the second week, students did a programming activity in the computer lab for the first time as part of the lab assignment, and the video data used in this study was collected during that week. At the beginning of the lab session, students were provided with a description of a problem to solve with their partner. The lab instructions were accompanied by instructions on pair programming:

"PAIR PROGRAMMING: You will complete this lab exercise using pair programming. Your TA will assign you to pairs. You must work with the partner your TA assigns. In pair programming, both programmers use the same computer. They take turns in two roles: driver and navigator. The driver controls the keyboard and mouse, and the navigator watches for errors and thinks about the "big picture" goal. Both driver and navigator should actively be talking to each other while they work, asking questions and saying what they are thinking. You should switch the roles of driver and navigator approximately every 15 minutes."

The objective of the lab was for students to learn mathematical operators in the Java programming language. The programming task for that week was to write a program with the following four parts:

Part 1: Write a program to calculate the area of a rectangle (after asking the user to input the length and width as integers).

Part 2: Similar to Part 1, write a program to calculate the area of a triangle (after asking for the user to input the base and height as integers).

Part 3: Write a program that will do division between two integers, and gives you an exact answer with the remainder. (Your program must work for whatever combination of length and width the user enters. If the user enters zero as the divisor, the program will crash. This is fine).

Part 4: Given a 4-digit number (from 1000-9999), write a program that will print out all of the digits individually by using mathematical operators.

Sample outputs and some other instructions were provided to students for each part.

3.3 Video Annotation Procedure

The goal of the data annotation was to label the gaze target of students throughout their collaborative process. We did not initially know the set of gaze targets needing to be labeled, so we first reviewed a subset of the data to determine this set. As shown in Figure 2, students gazed at four distinct targets: programming screen, instruction screen, notes, and partner. We also included a tag for gazing at other places. Finally, we included a label for instances in which the student's face was not visible on camera and these were excluded from the predictive models built for that student.

1: Student looks at the programming screen. When the student's gaze was focused on the programming screen. When in the role as the driver, some students needed to look at the keyboard while typing; thus, this behavior was also counted as part of looking at the programming screen.

2: Student looks at the instruction screen. Most pairs opted to view the instructions on one screen and conduct the programming task on the screen next to it. In the small number of cases where this was not the case, we applied a separate tag (One-Screen) but omitted these instances from further analysis due to their sparsity.

3: Student checks notes or looks at partner keyboard while typing. When the student used his/her own laptop, phone, or notes to check syntax or other related things to solve the problem. Additionally, while holding the navigator role, some students looked at the keyboard when the driver was typing.

4: Student looks at partner. When the student looked at his/her partner during the activity.

5: Student looks at other places. When the student looked at any other places (e.g., the ceiling or at other students in the room).

6. Face not visible: When the student's face was not visible due to camera angle or other reasons.

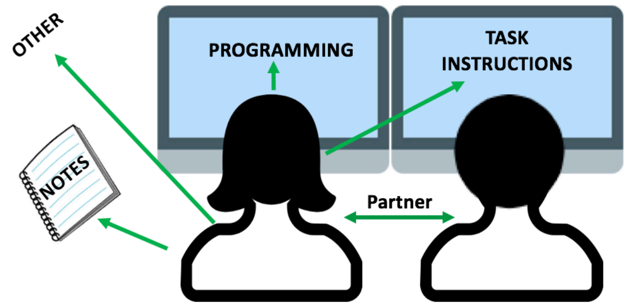

Figure 2. Pair programming setting. Students look in different directions during the session. Students take turns as driver and navigator.

During the annotator training phase, which included finalizing the set of tags listed above, the annotators first annotated a small portion of a video based on a draft annotation scheme, then met to discuss conflicts, resolve them, and refine the set of labels. After several such meetings, the annotators proceeded to tag independently. In this early work, quantitative measures of inter-rater reliability (such as Kappa) have not yet been conducted.

Twenty different pair programming videos with different pairs of students were annotated with the schema described above, assigning a label to each second of video for each student in the pair. This study was conducted in an actual CS1 lab session. Along with that rich context came numerous complicating factors that had to be dealt with in the study. For example, in this naturalistic classroom environment, some students interacted with the teaching assistant and sought help from them. We did not intervene in these cases but excluded those parts of the video from our analysis. Also, some students preferred physically switching seats while switching roles, which might have caused a temporary distraction from the task. These portions of the video were also excluded from our analysis. Four of the twenty groups used one screen for task instructions and programming, and this gaze target received the One-Screen tag as mentioned above. Due to it sparsity, this gaze target label was not included as a feature within the models reported here.

After excluding the aforementioned segments of annotated video, the total annotated video data analyzed was approximately 33 hours (118494 seconds). The session durations ranged from 1216 seconds to 4494 seconds and the average duration was 2962 seconds (~50 min). Because different pairs' collaboration lasted for different durations, we chose not to use absolute frequency of gaze labels, but rather relative frequency. Each relative frequency feature was computed as the number of seconds a student held each gaze target, divided by the duration of that student's collaborative session in seconds.

For the learning outcomes, we used the post-test scores that were collected after students completed the assignment or after the allocated time (two hours) had elapsed. The post-test consisted of eight multiple choice questions. It was constructed by the teaching staff and reviewed by the professor of the course. The questions consisted of both conceptual and programming questions.

4. RESULTS

RQ1: To investigate the first research question, which is to examine whether there are gaze distribution differences between navigator and driver, we compared each students' navigator gaze distribution frequency with his/her own driver gaze distribution frequency. Several paired-samples *t*-tests were conducted to compare the amount of time spent looking at *Programming_Screen*, *Instructions_Screen, Checking_Notes, Other_Places, and Partner* when the student was the navigator versus when they were the driver. We found no differences for the gaze distributions of *Programming_Screen, Instructions_Screen, Checking_Notes and Partner;* however, there was a significant difference in the gaze distributions of *Other_Places* between the Navigator (*M*=0.026, *SD*=0.02) and Driver roles (*M* = 0.018, *SD*=0.02); *t*(39) = 2.161, *p*=0.04, *d*=0.341. When students hold the navigator role, they tend look at *Other Places* more often than when they hold the driver role.

RQ2: To investigate the second research question, which is whether gaze distributions can be predictors of learning outcomes, we built multiple regression models to investigate whether eye-gaze activities and prior experience could significantly predict participants' post-test scores. The dependent (outcome) variable was the post-test score and the independent variables were prior experience and the frequency factors that we included in the eye-gaze activity annotation scheme. In total, six variables were provided to the model as predictors: (a) Prior experience, (b) Programming_Screen, (c) Instructions_Screen, (d) Checking_Notes (e) Other_Places, and (f) Partner. We included prior experience because previous literature shows that student with different background knowledge perform different eye-gaze behaviors [4,27]. The prior experience factor was a nominal variable, since students reported their prior programming experience as either not having any prior programming experience or having some prior programming experience. Other variables were continuous values based on the relative frequency of each gaze type in each pair programming session for each individual student.

In the linear regression analysis, we omitted the data from the four groups that worked on one computer because they would be counted as a completely new category. Therefore, the models were created based on the data from 32 students. Based on the variables described above, we created two linear regression models. These two models were created based on the distribution of eye-gaze behaviors when the students hold different roles, with the goal of predicting post-test score. The model created based on navigator gaze behaviors resulted in no statistically significant independent variables. However, the multiple regression model created based on the driver' eye-gaze activity with prior experience as a covariate indicated that the model explained 43.5% of the variance and that the model was a significant predictor of post-test score, *F*(6,25)=3.21, *p*=0.019.

Table 1. Regression analysis for predicting post-test score based on relative frequency features of driver gaze, with prior experience included as a control variable.

Variables	B	SE B	β	t	p
(Constant)	5.763	.899		6.409	.000
Prior Experience	.654	.504	.202	1.299	.206
Programming Screen	-4.712	2.287	-.321	-2.060	.049*
Instructions Screen	14.343	5.332	.458	2.690	.013*
Other Places	-7.297	11.456	-.101	-.637	.530
Checking Notes	10.762	3.623	.484	2.971	.006*
Partner	24.287	18.750	.205	1.295	.207

As can be seen in Table 1, the *Instructions_Screen and Checking_Notes* features had significant positive regression coefficients, which indicate that students with higher relative frequency values were expected to have a higher learning outcome after controlling for the other variables in the model. The *Programming_Screen* variable has a significant negative weight, which indicates that after accounting for *Instructions_Screen and Checking_Notes* values, students with higher relative frequency of *Programming_Screen* gaze were expected to have lower learning outcomes. *Other_Places, Partner and Prior_Experience* did not contribute to the multiple regression model significantly.

5. DISCUSSION

The goal of the study reported here was to explore students perform different eye-gaze activities when they take on different roles during pair programming, and also whether these differences can provide predictive insight into learning outcomes. The results suggest that students' gaze activities indeed differ depending on their roles in collaboration and that gaze activities of students when they hold the driver role are predictive of learning outcomes.

RQ1: Differences in Eye-gaze Activity between Navigator and Driver

Perhaps unsurprisingly, the results showed that students looked at *Other Places* more while holding the navigator role compared to the driver role. This result aligned with previous research suggesting that the navigator may lose the track of the activity [18] and have difficulty following the driver's actions [30]. When two programmers work on one computer with a shared keyboard and mouse, only one programmer takes control of the keyboard and mouse, which can leave the other programmer in the role of "backseat driver." While this role is intended to be highly active in terms of cognitive contribution, it leaves open the possibility for the navigator to become distracted from the task more often. Additionally, the navigator may feel more free to look elsewhere while contemplating feedback on the solution.

RQ2: Predicting Learning Outcomes from Eye-Gaze Distributions

The results showed that the eye-gaze features are predictive of learning outcomes when students are in the driver role. The model indicates that while students with higher relative frequency of *Instructions_Screen and Checking_Notes* were expected to have higher learning outcomes, students with higher relative frequency of *Programming_Screen* were expected to have lower learning outcomes. It is interesting that higher relative frequency of looking at the programming screen for driver had a negative weight because the main role of the driver is to write the code and implement the solution, which requires spending most of their time looking at the programming screen. However, the results suggest that when the driver is also engaged with the instructions screen and checks their notes more, their scores increase. This finding may indicate the importance of balancing the gaze targets.

Time	Partner	Gaze Focus	Dialogue Transcript
29:04	Nate	Programming	So we're gonna have to use mod, it is called division
29:17	David	Programming	Okay so…
29:23	Nate	Instructions then Programming	It's division, and we're inputting a dividend and a divisor
29:31	Nate	Instructions then Programming	So input, just enter the dividend. Say enter dividend
30:06	Nate	Programming	And enter the divisor
30:29	Nate	Instructions then Programming	And we're gonna need two more, one for quotient and one for remainder
30:42	David	Other then Programming	Okay, how would we do this
30:47	Nate	Instructions then Programming	Just do dividend divided by divisor because it rounds it
30:49	David	Programming	Wait what's the command for dividing
30:51	Nate	Programming	Just (that)
31:02	David	Instructions then Programming	Syntax error?
31:06	Nate	Programming	You need to have the colon, or the semicolon. Whatever
31:13	Nate	Programming	And then the remainder
31:17	David	Programming	Oh yeah

Figure 3. Sample excerpt illustrating when the Driver mostly focuses on the Programming Screen

Analysis of Excerpts

To explore how these gaze behaviors played a role in these results, we now consider two excerpts from a pair under different conditions, in which they have different roles. Table X and Y shows a conversation between two students, whom we call David and Nate. Both students did not have any prior programming experience. On the post-test, David scored two and Nate scored six out of eight. The following excerpts show the differences in their behaviors when they have different roles.

In the first Excerpt (Figure 3), David holds the driver role and Nate holds the navigator role. At 29:04, Nate made a suggestion and waited for David to implement it. However, David did not know how to do so, and continued looking at the programming screen without checking any notes or instructions. After they both stayed silent for a while, David, still looking at the programming screen, responded with "*Okay, so…*", which indicated that he expected Nate to give him instructions. Nate looked at the instructions screen, then the programming screen, and explained the solution step by step. During this time, David looked at the programming screen without making any comments and implemented the solution. At 30:47, Nates asked David to use the "dividend command" to do division, but then David asked, "*Wait, what's the command for dividing?*" and glanced at the instructions screen for two seconds. Nate answered the question by showing it on the programming screen immediately. David implemented what Nate showed him and did not check the instructions screen or the notes. At 31:02, David ran into a syntax error but again directly asked Nate instead of trying to find the answer by himself. During this interaction, David implemented the solution but whenever he an into a problem, he continued gazing at the programming screen instead of looking at the instructions screen or checking notes.

In this sample excerpt, David, the driver, did not use other information sources, such as the instructions screen, and only focused on the programming screen. However, checking the notes or looking at the instructions screen may have helped David to find the answer after a brief exploration, which could have made him more active in the process. However, he instead, passively waited for the navigator to tell him the answers.

In the second excerpt (Figure 4), David and Nate have changed roles. In the second excerpt taken from earlier in the session, David is now the Navigator and Nate is now the driver. At 06:33, Nate gazed at the programming screen, then read the instructions and said, "Um let's see." He learned the steps that he needed to implement and started looking at the programming screen. David checked the notes and asked whether they "needed a variable for each" and David immediately gave the answer, which was actually explained in the lab instructions. At 06:55, Nate had a question while checking the notes, but he found the answer by himself. At 07:01 David had a question but neither of them knew the answer and they both said, "*I don't know.*" However, Nate referred to his notes and said, "*I don't think you have to have it, that's just what I put in my notes*" which indicates that he was using his notes as an information source in the problem-solving process. As the excerpt shows, Nate did not spend the entire time on the programming screen; instead, he often switched between gazing at the programming screen, instructions screen, and checking notes while solving the problem.

In this sample excerpt, when Nate, the driver, got stuck on coding part, he actively explored for alternative ways to solve the problem by checking notes and looking at the instructions screen. This exploration phase helped him construct different ideas and solutions, which may have positively affected his learning. Nate scored 6 out of 8 on the post-test, in contrast to David's 2 out of 8.

Time	Partner	Gaze Focus	Dialogue Transcript
06:33	Nate	Programming then Instructions	Um let's see
06:36	David	Notes	And then don't we need a variable for each?
06:45	Nate	Programming then Notes	I didn't make the double, I made them length and width.
06:46	David	Programming	So like length times width.
06:52	Nate	Programming	Oh we need to ask them to enter it.
06:53	David	Notes then Programming	Oh true, true
06:55	Nate	Notes	What is it? Input dot next double.
07:01	David	Programming	And what's the purpose of the initial ones if we're gonna add something?
07:04	Nate	Notes then Programming	I don't know.
07:04	David	Programming	I don't know either, that's why.
07:07	Nate	Programming	I don't think you have to have it, that's just what I put in my notes.
07:08	David	Programming	Okay, we'll just keep it.

Figure 4. Sample interaction illustrating when the Driver engages in different activities

6. LIMITATIONS and THREATS TO VALIDITY

This study was conducted at the beginning of the semester when students were just exposed to the pair programming methodology; thus, it is important to interpret these results in light of the fact that the students were just learning how to collaborate under driver-navigator paradigm. Additionally, in this early work, inter-rater reliability for the gaze tagging has not yet been conducted, though extensive discussions between annotators were utilized in the training phase for annotation. Finally, the natural environment in which this study was conducted led to numerous challenges with video annotation including students moving out of the camera frame and using different collaborative setups in terms of number of screens. As previously described, these data were omitted from further analysis but the two-screen versus one-screen setup likely warrants further exploration.

7. CONCLUSIONS and FUTURE WORK

Investigating students' eye-gaze activities during collaborative problem-solving tasks can provide useful insight for understanding the team dynamic. In this study, we examined how students perform different eye gaze activities when holding different team roles in pair programming for computer science learning. We also examined whether the differences in eye-gaze activities can provide useful information for predicting learning outcomes. We found that students' roles in the pair programming collaboration have a significant relationship with eye-gaze activities: When students hold the navigator role, they tend look at *Other Places* more often than when they hold the driver role. Even though the role of navigator is to check the driver's programming actions and provide feedback, not having direct control over the programming task provides the navigator with fewer constraints on gaze. Our observations suggest that navigators were more prone to disengagement, though productive gaze outside of the workspace is also possible.

Next, we found participants' roles and their gaze activities can provide predictive insight into their post-test scores. While holding the driver role, students with higher relative frequency of *Instructions_Screen and Checking_Notes* were expected to have higher learning outcomes, while students with higher frequency values of *Programming_Screen* were expected to have lower learning outcomes. Moreover, examination of the excerpts showed some of the mechanisms by which checking notes and gazing at the instructions screen may have facilitated student learning. In contrast, spending a large majority of the time on the programming screen may have hindered learning. This study provides even more evidence that simple activity measures such as frequency of eye-gaze activities can be useful in understanding the collaboration process.

There are many promising research areas for the future. First, some specific eye-gaze activity patterns might have relationships with some aspects of the collaboration process, and a deeper analysis that identifies the patterns emerging from the combination of gazing behaviors and students' dialogues can shed more light on understanding team dynamics. Also, investigating impacts of other factors such as personality and gender jointly with gaze is an important research direction. By better understanding teamwork for computer science learning, the community can move toward adaptive support tailored toward teams' needs.

8. ACKNOWLEDGEMENTS

The authors wish to thank the members of the LearnDialogue group at the University of Florida for their helpful input, especially Erin O'Halloran and Timothy Brown for their invaluable help which made this work possible. This work is supported in part by Google through a CS Capacity Research Award and by the National Science Foundation through grant CNS-1622438. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the authors, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

REFERENCES

[1] Peter Auer. 2017. Gaze, addressee selection and turn-taking in three-party interaction. Journal of InLiSt - Interaction and Linguistic Structures 60.

[2] Kent Beck. 2000. *Extreme programming eXplained: embrace change*. Addison-Wesley.

[3] Sallyann Bryant and Benedict du Boulay. 2008. Pair programming and the mysterious role of the navigator. *International Journal of Human-Computer Studies* 66, 7: 519–529. https://doi.org/10.1016/J.IJHCS.2007.03.005

[4] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye Movements in Code Reading: Relaxing the Linear Order. In *2015 IEEE 23rd International Conference on Program Comprehension*, 255–265. https://doi.org/10.1109/ICPC.2015.36

[5] Mehmet Celepkolu and Kristy Elizabeth Boyer. 2018. Thematic Analysis of Students' Reflections on Pair Programming in CS1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE '18*, 771–776. https://doi.org/10.1145/3159450.3159516

[6] Jan Chong and Tom Hurlbutt. 2007. The Social Dynamics of Pair Programming. In *29th International Conference on Software Engineering (ICSE'07)*, 354–363. https://doi.org/10.1109/ICSE.2007.87

[7] Martha E. Crosby and Jan Stelovsky. 1990. How do we read algorithms? A case study. *Computer* 23, 1: 25–35. https://doi.org/10.1109/2.48797

[8] Sarah D'Angelo and Andrew Begel. 2017. Improving Communication Between Pair Programmers Using Shared Gaze Awareness. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, 6245–6290. https://doi.org/10.1145/3025453.3025573

[9] Gwyneth Doherty-Sneddon and Fiona G. Phelps. 2005. Gaze aversion: A response to cognitive or social difficulty? *Memory & Cognition* 33, 4: 727–733. https://doi.org/10.3758/BF03195338

[10] Starkey Duncan and Donald W. Fiske. 2015. *Face-to-Face Interaction*. Routledge. https://doi.org/10.4324/9781315660998

[11] Randi A. Engle, Jennifer M. Langer-Osuna, and Maxine McKinney de Royston. 2014. Toward a Model of Influence in Persuasive Discussions: Negotiating Quality, Authority, Privilege, and Access Within a Student-Led Argument. *Journal of the Learning Sciences* 23, 2: 245–268. https://doi.org/10.1080/10508406.2014.883979

[12] Margaret Foddy. 1978. Patterns of Gaze in Cooperative and Competitive Negotiation. *Human Relations* 31, 11: 925–938. https://doi.org/10.1177/001872677803101101

[13] Tom Foulsham, Esther Walker, and Alan Kingstone. 2011. The where, what and when of gaze allocation in the lab and the natural environment. *Vision Research* 51, 17: 1920–1931. https://doi.org/10.1016/J.VISRES.2011.07.002

[14] Megan Freeth, Tom Foulsham, and Alan Kingstone. 2013. What Affects Social Attention? Social Presence, Eye Contact and Autistic Traits. *PLoS ONE* 8(1): e53286. https://doi.org/10.1371/journal.pone.0053286

[15] Andrew C Gallup, Joseph J Hale, David J T Sumpter, Simon Garnier, Alex Kacelnik, John R Krebs, and Iain D Couzin. 2012. Visual attention and the acquisition of information in human crowds. *In Proceedings of the*

National Academy of Sciences, 109, 19: 7245–7250. https://doi.org/10.1073/pnas.1116141109

[16] Shuchi Grover, Marie Bienkowski, Amir Tamrakar, Behjat Siddiquie, David Salter, and Ajay Divakaran. 2016. Multimodal analytics to study collaborative problem solving in pair programming. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK '16*, 516–517. https://doi.org/10.1145/2883851.2883877

[17] Simon Ho, Tom Foulsham, and Alan Kingstone. 2015. Speaking and Listening with the Eyes: Gaze Signaling during Dyadic Interactions. *PLoS ONE* 10(8): e0136905. https://doi.org/10.1371/journal.pone.0136905

[18] Danielle L. Jones and Scott D. Fleming. 2013. What use is a backseat driver? A qualitative investigation of pair programming. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, 103–110. https://doi.org/10.1109/VLHCC.2013.6645252

[19] Adam Kendon. 1967. Some functions of gaze-direction in social interaction. *Acta Psychologica* 26: 22–63. https://doi.org/10.1016/0001-6918(67)90005-4

[20] Sonu Chopra Khullar and Norman I. Badler. 2001. Where to Look? Automating Attending Behaviors of Virtual Human Characters. *Autonomous Agents and Multi-Agent Systems* 4, 1/2: 9–23. https://doi.org/10.1023/A:1010010528443

[21] Chris L. Kleinke. 1986. Gaze and Eye Contact: A Research Review. *Psychological Bulletin* 100, 1: 78–100.

[22] Meng-Lung Lai, Meng-Jung Tsai, Fang-Ying Yang, Chung-Yuan Hsu, Tzu-Chien Liu, Silvia Wen-Yu Lee, Min-Hsien Lee, Guo-Li Chiou, Jyh-Chong Liang, and Chin-Chung Tsai. 2013. A review of using eye-tracking technology in exploring learning from 2000 to 2012. *Educational Research Review* 10: 90–115. https://doi.org/10.1016/J.EDUREV.2013.10.001

[23] Kaitlin E. W Laidlaw, Tom Foulsham, Gustav Kuhn, and Alan Kingstone. 2011. Potential social interactions are important to social attention. *Proceedings of the National Academy of Sciences* 108, 14: 5548–53. https://doi.org/10.1073/pnas.1017022108

[24] Colleen M. Lewis and Niral Shah. 2015. How Equity and Inequity Can Emerge in Pair Programming. In *Proceedings of the Eleventh annual International Conference on International Computing Education Research - ICER '15*, 41–50. https://doi.org/10.1145/2787622.2787716

[25] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin* 34, 1: 38. https://doi.org/10.1145/563517.563353

[26] Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, and Suzanne Balik. 2003. Improving the CS1 experience with pair programming. *ACM SIGCSE Bulletin* 35, 1: 359. https://doi.org/10.1145/792548.612006

[27] Markus Nivala, Florian Hauser, Jurgen Mottok, and Hans Gruber. 2016. Developing visual expertise in software engineering: An eye tracking study. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, 613–620. https://doi.org/10.1109/EDUCON.2016.7474614

[28] Marc-Antoine Nüssli. 2011. Dual Eye-Tracking Methods for the Study of Remote Collaborative Problem Solving. PhD thesis, Ecole Polytechnique Federale de Lausanne.

[29] Unaizah Obaidellah, Mohammed Al Haek, and Peter C. H. Cheng. 2018. A Survey on the Usage of Eye-Tracking in Computer Programming. *ACM Computing Surveys* 51, 1: 1–58. https://doi.org/10.1145/3145904

[30] Laura Plonka, Helen Sharp, and Janet Van Der Linden. 2012. Disengagement in Pair Programming: Does It Matter? In *Proceedings of the 34th International Conference on Software Engineering*, 496–506.

[31] Evan F. Risko, Kaitlin Laidlaw, Megan Freeth, Tom Foulsham, and Alan Kingstone. 2012. Social attention with real versus reel stimuli: toward an empirical approach to concerns about ecological validity. *Frontiers in Human Neuroscience* 6: 143. https://doi.org/10.3389/fnhum.2012.00143

[32] Kshitij Sharma, Patrick Jermann, Marc-Antoine Nüssli, and Pierre Dillenbourg. 2012. Gaze Evidence for Different Activities in Program Understanding. In *Proceedings of 24th Annual conference of Psychology of Programming Interest Group.*

[33] Tim J. Smith and Parag K. Mital. 2013. Attentional synchrony and the influence of viewing task on gaze behavior in static and dynamic scenes. *Journal of Vision* 13, 8: 16–16. https://doi.org/10.1167/13.8.16

[34] Randy Stein and Susan E. Brennan. 2004. Another person's eye gaze as a cue in solving programming problems. In *Proceedings of the Sixth International Conference on Multimodal interfaces - ICMI '04*, 9. https://doi.org/10.1145/1027933.1027936

[35] Melissa L. H. Vo, Tim J. Smith, Parag K. Mital, and John M. Henderson. 2012. Do the eyes really have it? Dynamic allocation of attention when viewing moving faces. *Journal of Vision* 12, 13: 3–3. https://doi.org/10.1167/12.13.3

[36] Jiahui Wang, Pavlo Antonenko, Mehmet Celepkolu, Yerika Jimenez, Ethan Fieldman, and Ashley Fieldman. 2018. Exploring Relationships Between Eye Tracking and Traditional Usability Testing Data. *International Journal of Human–Computer Interaction*: 1–12. https://doi.org/10.1080/10447318.2018.1464776

[37] Armin Weinberger, Karsten Stegmann, and Frank Fischer. 2010. Learning to argue online: Scripted groups surpass individuals (unscripted groups do not). *Computers in Human Behavior* 26, 4: 506–515. https://doi.org/10.1016/J.CHB.2009.08.007

[38] Laurie Williams, Eric Wiebe, Kai Yang, Miriam Ferzli, and Carol Miller. 2002. In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education* 12, 3: 197–212. https://doi.org/10.1076/csed.12.3.197.8618